aws

Guía del usuario

Amazon Neptune

Amazon Neptune: Guía del usuario

Table of Contents

¿Qué es Amazon Neptune?

Amazon Neptune es un servicio de base de datos de gráficos rápido, fiable y completamente administrado que le permite crear y ejecutar fácilmente aplicaciones que funcionen con conjuntos de datos altamente conectados. El componente principal de Neptune es un motor de base de datos de gráficos de alto rendimiento y personalizado. Este motor está optimizado para almacenar miles de millones de relaciones y consultar el gráfico con una latencia de milisegundos. Neptune es compatible con los conocidos lenguajes de consulta de gráficos de propiedades Apache, TinkerPop, Gremlin y openCypher de Neo4j, así como con el lenguaje de consulta RDF de W3C, SPARQL. Esto le permite crear consultas que naveguen de manera eficaz por conjuntos de datos altamente conectados. Neptune es la solución ideal para casos de uso de gráficos como, por ejemplo, motores de recomendaciones, detección de fraudes, gráficos de conocimiento, descubrimiento de fármacos y seguridad de red.

La base de datos de Neptune ofrece alta disponibilidad, con réplicas de lectura, recuperación a un momento dado, copia de seguridad continua en Amazon S3 y replicación entre zonas de disponibilidad. Neptune ofrece características de seguridad de datos y admite el cifrado en reposo y en tránsito. Neptune es un servicio totalmente administrado, por lo que ya no tendrá que preocuparse de las tareas de administración de base de datos, como el aprovisionamiento de hardware, los parches de software, la instalación, la configuración o las copias de seguridad.

Neptune Analytics: es un motor de base de datos de análisis que complementa la base de datos de Neptune y que puede analizar rápidamente grandes cantidades de datos de gráficos en la memoria para obtener información y encontrar tendencias. Neptune Analytics es una solución para analizar rápidamente las bases de datos de gráficos existentes o los conjuntos de datos de gráficos almacenados en un lago de datos. Utiliza algoritmos de análisis de gráficos populares y consultas analíticas con baja latencia.

Para obtener información sobre cómo utilizar Amazon Neptune, le recomendamos que comience por las siguientes secciones:

- Introducción a Amazon Neptune
- Información general de las características de Amazon Neptune

Si utiliza por primera vez los gráficos o aún no está preparado para invertir en un entorno de producción completo de Neptune, visite nuestro tema Introducción para descubrir cómo utilizar los cuadernos de Jupyter de Neptune para aprender y desarrollar sin incurrir en costos.

Además, antes de comenzar el diseño de una base de datos, le recomendamos que consulte el repositorio de GitHub Arquitecturas de referencia de AWS para utilizar bases de datos de gráficos, donde puede informar de sus opciones sobre los modelos de datos de los gráficos y lenguajes de consulta, y examinar los ejemplos de arquitecturas de implementación de referencia.

Componentes de servicio clave

- Instancia de base de datos principal: admite operaciones de lectura y escritura y realiza todas las modificaciones de los datos en el volumen de clúster. Cada clúster de base de datos de Neptune cuenta con una instancia de base de datos principal responsable de escribir (es decir, cargar o modificar) el contenido de bases de datos de gráficos.
- Réplica de Neptune: se conecta con el mismo volumen de almacenamiento que la instancia de base de datos principal y solo admite operaciones de lectura. Cada clúster de base de datos de Neptune puede tener hasta 15 réplicas de Neptune, además de la instancia de base de datos principal. Esto ofrece alta disponibilidad al localizar réplicas de Neptune en distintas zonas de disponibilidad y carga de distribución de la lectura de clientes.
- Volumen de clúster: los datos de Neptune se almacenan en el volumen del clúster, diseñado para ofrecer fiabilidad y alta disponibilidad. Un volumen de clúster se compone de copias de los datos repartidas entre varias zonas de disponibilidad de una sola región de AWS. Como sus datos se replican automáticamente entre las distintas zonas de disponibilidad, tienen una larga duración y es poco probable que se pierdan datos.

Compatibilidad con API de Open Graph

Amazon Neptune admite las API de gráficos abiertos para gráficos de propiedades (Gremlin y openCypher) y gráficos RDF (SPARQL). Proporciona un gran rendimiento en ambos modelos de gráficos y en sus respectivos lenguajes de consulta. Puede elegir el modelo del gráfico de propiedad (PG) y acceder al mismo gráfico con el lenguaje de consulta de openCypher o el lenguaje de consulta de Gremlin. Si utiliza el modelo estándar del marco de descripción de recursos (RDF) de W3C, puede acceder al gráfico con el lenguaje de consulta de SPARQL estándar.

Alto nivel de seguridad

Neptune ofrece varios niveles de seguridad para las bases de datos. Entre las características de seguridad se incluyen el aislamiento de redes mediante Amazon VPC y el cifrado en reposo mediante claves que usted puede crear y controlar a través de AWS Key Management Service (AWS KMS). En una instancia cifrada de Neptune, los datos del almacenamiento subyacente

están cifrados, al igual que las copias de seguridad, las instantáneas y las replicaciones que se automatizan en el mismo clúster.

Totalmente administrado

Con Amazon Neptune, ya no tiene que preocuparse de las tareas de administración de base de datos, como el aprovisionamiento de hardware, la aplicación de parches de software, la instalación, la configuración o las copias de seguridad.

Puede utilizar Neptune para crear aplicaciones de gráficos interactivas y sofisticadas que pueden consultar miles de millones de relaciones en cuestión de milisegundos. Las consultas SQL para datos altamente conectados son complejas y difíciles de ajustar para lograr un buen nivel de desempeño. Con Neptune, podrá usar los conocidos lenguajes de consulta de gráficos (Gremlin, openCypher y SPARQL) para ejecutar consultas eficientes que son fáciles de escribir y funcionan correctamente en los datos conectados. Esta capacidad reduce de manera significativa la complejidad del código y le permite crear aplicaciones que procesen las relaciones con mayor rapidez.

Neptune se ha diseñado para ofrecer una disponibilidad superior al 99,99 %. Aumenta el rendimiento y la disponibilidad de las bases de datos gracias a la estrecha integración del motor de base de datos con una capa de almacenamiento virtualizada basada en SSD, diseñada para las cargas de trabajo de base de datos. Almacenamiento de Neptune cuenta con recuperación automática y tolerancia a errores. Los errores del disco se reparan en segundo plano sin perder la disponibilidad de la base de datos. Neptune detecta automáticamente los bloqueos de las bases de datos y se reinicia sin necesidad de realizar recuperaciones tras bloqueos ni de recompilar la caché de la base de datos. Si se produce un error en toda la instancia, Neptune realizará una conmutación por error automática a una de las 15 réplicas de lectura.

Cambios y actualizaciones de Amazon Neptune

En la siguiente tabla se describen los cambios importantes en Amazon Neptune.

Cambio	Descripción	Fecha
Versión del motor 1.3.2.1	A partir del 20 de junio de 2020, la versión 1.3.2.1 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.3.2.1.	20 de junio de 2024
Versión del motor 1.3.2.0	A partir del 10 de junio de 2022, la versión 1.3.2.0 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.3.2.0.	10 de junio de 2024
Versión del motor 1.2.1.1	A partir del 11 de marzo de 2020, la versión 1.2.1.1 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las	11 de marzo de 2024

	regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.1.	
Versión del motor 1.3.1.0	A partir del 3 de marzo de 2022, la versión 1.3.1.0 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.3.1.0.	6 de marzo de 2024
Actualización de los permisos de política gestionados AWS	Las políticas `NeptuneFullAccess` gestionadas `NeptuneReadOnlyAcess` y las políticas ahora incluyen `Sid` (ID de declaración) como identificador en la declaración de política.	22 de enero de 2024
Neptune ahora ofrece almacenamiento con optimización de E/S	Con el almacenamiento optimizado para E/S, paga por el almacenamiento y las instancias que utiliza. Los costos de almacenamiento son más elevados que los del almacenamiento estándar, pero no paga nada por la E/S que utilice.	13 de diciembre de 2023

Cambios en la política administrada de IAM para Neptune	Se ha actualizado la política gestionada de NeptuneConsoleFullAccessIAM para conceder los permisos necesarios para interactuar con los gráficos de Neptune Analytics, se ha añadido una NeptuneGraphReadOnlynueva política de acceso para proporcionar acceso de solo lectura a los recursos de gráficos de Neptune Analytics y se ha añadido `AWSServiceRoleForNeptuneGraphPolicy` una nueva política para permitir que los gráficos CloudWatch de Neptune Analytics publiquen métricas y registros operativos y de uso.	29 de noviembre de 2023
Versión del motor 1.3.0.0	A partir del 15 de noviembre de 2023, se implementará de forma general la versión 1.3.0.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.3.0.0.	15 de noviembre de 2023

Neptune se lanzó en la región de Israel (Tel Aviv)	Amazon Neptune ya está disponible en la región de Israel (Tel Aviv) (`il-central-1`).	13 de noviembre de 2023
Entrada de blog sobre la implementación de time-to-live gráficos de propiedades en Neptune	Consulte Implement Time to Live in Amazon Neptune, Part 1: Property Graph, de Melissa Kwok, Mike Havey y Kevin Phillips.	27 de octubre de 2023
Versión del motor 1.2.1.0.R7	A partir del 6 de octubre de 2023, se implementará de forma general la versión 1.2.1.0.R7 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.0.R7.	6 de octubre de 2023
Versión del motor 1.2.0.0.R4	A partir del 29 de septiembre de 2023, se implementará de forma general la versión 1.2.0.0.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.0.R4.	29 de septiembre de 2023

Versión del motor 1.2.0.1.R3	A partir del 27 de septiembre de 2023, se implementará de forma general la versión 1.2.0.1.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.1.R3.	27 de septiembre de 2023
Versión del motor 1.2.1.0.R6	A partir del 12 de septiembre de 2023, se implementará de forma general la versión 1.2.1.0.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.0.R6.	12 de septiembre de 2023
Versión del motor 1.2.0.2.R6	A partir del 12 de septiembre de 2023, se implementará de forma general la versión 1.2.0.2.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.2.R6.	12 de septiembre de 2023

Entrada de blog sobre el uso de una estrategia de implementación azul/verde para las actualizaciones del motor de Neptune	Consulte Improve availability of Amazon Neptune during engine upgrade using blue/green deployment, de Ankit Gupta y Abhishek Mishra.	11 de septiembre de 2023
Versión del motor 1.2.1.0.R5	A partir del 2 de septiembre de 2023, se implementará de forma general la versión 1.2.1.0.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.0.R5.	2 de septiembre de 2023
Versión del motor 1.2.0.2.R5	A partir del 16 de agosto de 2023, se implementará de forma general la versión 1.2.0.2.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.2.R5.	16 de agosto de 2023

Versión del motor 1.2.1.0.R4	A partir del 10 de agosto de 2023, se implementará de forma general la versión 1.2.1.0.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.0.R4.	10 de agosto de 2023
Entrada de blog sobre la versión 1.2.1.0 del motor de Neptune	Consulte Exploring the feature packed 1,2.1.0 release for Amazon Neptune, de Joy Wang, Kevin Phillips, Andrea Nassisi y Navtanay Sinha.	4 de agosto de 2023
Entrada de blog sobre la creación de una solución de base de datos multimodal con Neptune	Consulte Design a use case-driven, highly scalable multimodel database solution using Amazon Neptune, de Mike Havey.	18 de julio de 2023
Entrada de blog sobre casos de uso y prácticas recomendadas de Neptune sin servidor	Consulte Use cases and best practices to optimize cost and performance with Amazon Neptune Serverless, de Kevin Phillips y Ankit Gupta.	28 de junio de 2023

Versión del motor 1.2.1.0.R3	A partir del 13 de junio de 2023, se implementará de forma general la versión 1.2.1.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.0.R3.	13 de junio de 2023
Entrada de blog sobre cómo generar sugerencias de ocio en tiempo real	Consulte Generate suggestions for leisure activities in real time with Amazon Neptune, de Michael Meidlinger y Nils Müller.	6 de junio de 2023
Entrada de blog sobre modelado molecular con Neptune y RDKit	Consulte Model molecular SMILES data with Amazon Neptune and RDKit, de Graham Kutchek.	1 de junio de 2023
Entrada de blog sobre el uso del almacenamiento en caché para acelerar el rendimiento de Neptune (parte 3)	Consulte Acelere el rendimiento de las consultas gráficas con el almacenamiento en caché en Amazon Neptune, Parte 3: Arquitecturas de almacenamiento en caché para todo el clúster de Neptuno con ElastiCache Amazon, de Taylor Riggan, Abhishek Mishra, Melissa Kwok y Kelvin Lawrence.	26 de mayo de 2023

Entrada de blog sobre el uso del almacenamiento en caché para acelerar el rendimiento de Neptune (parte 2)	Consulte Accelerate graph query performance with caching in Amazon Neptune, Part 2: Additional Neptune caching features, de Taylor Riggan, Abhishek Mishra, Melissa Kwok y Kelvin Lawrence.	26 de mayo de 2023
Entrada de blog sobre el uso del almacenamiento en caché para acelerar el rendimiento de Neptune (parte 1)	Consulte Accelerate graph query performance with caching in Amazon Neptune, Part 1: Queries and buffer pool caching, de Taylor Riggan, Abhishek Mishra, Melissa Kwok y Kelvin Lawrence.	26 de mayo de 2023
Entrada de blog sobre el análisis de la cadena de suministro con Neptune	Consulte Supply chain data analysis and visualization using Amazon Neptune and the Neptune workbench, de Dhiraj Thakur y Rajdip Chaudhur.	10 de mayo de 2023
Versión del motor 1.2.0.2.R4	A partir del 8 de mayo de 2023, se implementará de forma general la versión 1.2.0.2.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.2.R4.	8 de mayo de 2023

Se ha lanzado Neptune en la región de Oriente Medio (UAE)	Amazon Neptune está disponible ahora en la región de Oriente Medio (EAU) (`me-central-1`).	2 de mayo de 2023
Versión del motor 1.2.1.0.R2	A partir del 2 de mayo de 2023, se implementará de forma general la versión 1.2.1.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.0.R2.	2 de mayo de 2023
Entrada de blog sobre la creación de un gráfico de conocimiento en Neptune con análisis en vídeo basado en IA utilizando Media2Cloud	Consulte Build a knowledge graph on Amazon Neptune with AI-powered video analysis using Media2Cloud, de Mike Havey.	2 de mayo de 2023
Entrada de blog sobre cómo DevOcean se creó una plataforma de remediación de vulnerabilidades con Neptune	Consulte Cómo DevOcean se creó una plataforma de administración de remediación de vulnerabilidades para aplicaciones nativas de la nube con Amazon Neptune, de Gil Makmel y Charles Ivie.	25 de abril de 2023

Entrada de blog sobre cómo Getir creó un sistema de detección de fraudes con Neptune	Consulte How Getir build a comprehensive fraud detection system using Amazon Neptune and Amazon DynamoDB, de Berkay Berkman, Mahmut Turan, Mutlu Polatcan, Umut Cemal Kıraç, Yağız Yanıkoğlu y Esra Kayabali.	6 de abril de 2023
Entrada de blog sobre cómo Wiz reinventa la seguridad en la nube con Neptune	Consulte The World is a graph: How Wiz reimagines cloud security using a graph in Amazon Neptune, de Ami Luttwak y Brad Bebee.	31 de marzo de 2023
Versión del motor 1.2.0.2.R3	A partir del 27 de marzo de 2023, se implementará de forma general la versión 1.2.0.2.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.2.R3.	27 de marzo de 2023
Entrada de blog sobre cómo CSC Generation impulsa el descubrimiento de productos con Neptune	Consulte How CSC Generation powers product discovery with knowledge graphs using Amazon Neptune, de Bobber Cheng, Ronit Rudra y Melissa Kwok.	21 de marzo de 2023

Versión del motor 1.2.1.0	A partir del 8 de marzo de 2023, se implementará de forma general la versión 1.2.1.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.1.0.	8 de marzo de 2023
Entrada de blog sobre la exploración de las nuevas funciones de la versión TinkerPop 3.6.x en Neptune	Consulte Exploración de las nuevas características de Apache TinkerPop 3.6.x en Amazon Neptune de Stephen Mallette.	8 de marzo de 2023
Entrada de blog sobre el uso del razonamiento semántico para inferir nuevos datos a partir de un gráfico RDF	Consulte Use semantic reasoning to infer new facts from your RDF graph by integrating RDFox with Amazon Neptune, de Charles Ivie y Diana Marks.	20 de febrero de 2023
Entrada de blog sobre el análisis de los datos FHIR de asistencia sanitaria con Neptune	Consulte Analyze healthcare FHIR data with Amazon Neptune, de Alena Schmickl.	13 de febrero de 2023
Entrada de blog sobre la creación de una solución de detección de fraudes en tiempo real con Neptune ML	Consulte Build a real-time fraud detection solution using Amazon Neptune ML, de Hua Shu y Soji Adeshina.	8 de febrero de 2023

Versión del motor 1.1.1.0.R7	A partir del 23 de enero de 2023, se implementará de forma general la versión 1.1.1.0.R7 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.1.0.R7.	23 de enero de 2023
Lanzamiento de Graph-explorer	Graph-explorer es una herramienta de aplicación web frontend de código abierto para visualizar datos de gráficos. Consulte https://github.com/aws/graph-explorer.	3 de enero de 2023
Versión de mantenimiento 1.1.0.0.R3	A partir del 23 de diciembre de 2022, se implementará de forma general la versión de mantenimiento 1.1.0.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.0.0.R3.	23 de diciembre de 2022

Neptune Workbench ahora funciona en Amazon Linux 2 y 3. JupyterLab	Los cuadernos gráficos de Neptune ahora se ejecutan en un entorno Amazon Linux 2 con 3. JupyterLab Consulte Migración de sus cuadernos Neptune de Jupyter JupyterLab a 3 para obtener información sobre cómo migrar a este nuevo entorno.	21 de diciembre de 2022
Entrada de blog sobre recomendaciones de potencia y búsqueda mediante un gráfico de conocimiento de IMDb (parte 3)	Consulte Power recommendations and search using an IMDb knowledge graph – Part 3, de Divya Bhargavi, Soji Adeshina, Gaurav Rele, Karan Sindwani, Vidya Sagar Ravipati y Matthew Rhodes.	20 de diciembre de 2022
Entrada de blog sobre recomendaciones de potencia y búsqueda mediante un gráfico de conocimiento de IMDb (parte 2)	Consulte Power recommendations and search using an IMDb knowledge graph – Part 2, de Matthew Rhodes, Soji Adeshina, Divya Bhargavi, Gaurav Rele, Karan Sindwani y Vidya Sagar Ravipati.	20 de diciembre de 2022
Entrada de blog sobre recomendaciones de potencia y búsqueda mediante un gráfico de conocimiento de IMDb (parte 1)	Consulte Power recommendations and search using an IMDb knowledge graph – Part 1, de Gaurav Rele, Soji Adeshina, Divya Bhargavi, Karan Sindwani, Vidya Sagar Ravipati y Matthew Rhodes.	20 de diciembre de 2022

Neptune Serverless ya está disponible en nuevas regiones AWS	El 16 de diciembre de 2022, Neptune sin servidor se lanzó en las siguientes nuevas regiones de AWS : Canadá (centro), Europa (Estocolmo), Europa (Fráncfort), Asia-Pacífico (Singapur) y Asia-Pacífico (Sídney). Consulte las restricciones de Amazon Neptune sin servidor para todas las regiones en las que está disponible Neptune sin servidor.	16 de diciembre de 2022
Versión del motor 1.2.0.2.R2	A partir del 15 de diciembre de 2022, se implementará de forma general la versión 1.2.0.2.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.2.R2.	15 de diciembre de 2022

Versión del motor 1.2.0.0.R3	A partir del 15 de diciembre de 2022, se implementará de forma general la versión 1.2.0.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.0.R3.	15 de diciembre de 2022
Versión del motor 1.2.0.1.R2	A partir del 13 de diciembre de 2022, se implementará de forma general la versión 1.2.0.1.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.1.R2.	13 de diciembre de 2022
Entrada de blog sobre el diseño de una arquitectura educativa de análisis de macrodatos con AWS	Consulte Designing an educational big data analysis architecture with AWS, de Lavanya Sood.	13 de diciembre de 2022
Entrada de blog sobre cómo las bases de datos de gráficos pueden mejorar el aprendizaje	Consulte How graph databases can enhance learning, de Lavanya Sood.	8 de diciembre de 2022
Entrada de blog sobre la carga de datos RDF en Neptune con Glue AWS	Consulte Cargar datos RDF en Amazon Neptune with AWS Glue, de Mike Havey y Fabrizio Napolitano.	23 de noviembre de 2022

Versión del motor 1.2.0.2	A partir del 20 de noviembre de 2022, se implementará de forma general la versión 1.2.0.2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.2.	20 de noviembre de 2022
Versión del motor 1.2.0.1	A partir del 26 de octubre de 2022, se implementará de forma general la versión 1.2.0.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.1.	26 de octubre de 2022
Entrada de blog sobre cómo la detección de fraudes con Neptune	Consulte Empowering fraud detection at Delivery Hero with Amazon Neptune, de Wilson Tang, Amr Elnaggar, Matias Pons, Mohammad Azzam, Saurabh Deshpande y Luis Rodrigues Soares.	26 de octubre de 2022
Entrada de blog sobre Neptune sin servidor	Consulte Introducing Amazon Neptune Serverless – A Fully Managed Graph Database that Adjusts Capacity for Your Workloads, de Danilo Poccia.	26 de octubre de 2022

Entrada de blog sobre las importaciones de RDF basadas en eventos a Neptune mediante Lambda y SPARQL UPDATE LOAD	Vea cómo NXP realiza importaciones de RDF basadas en eventos a Amazon Neptune mediante AWS Lambda y SPARQL UPDATE LOAD de John Walker, Onno Buijs, Charles Ivie y Javy de Koning.	20 de octubre de 2022
Versión del motor 1.2.0.0.R2	A partir del 14 de octubre de 2022, se implementará de forma general la versión 1.2.0.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.0.R2.	14 de octubre de 2022
Entrada de blog sobre la codificación de propiedades de texto multilingüe en Neptune	Consulte Encode multi-lingual text properties in Amazon Neptune to train predictive models, de Jiani Zhang.	14 de octubre de 2022
Entrada de blog sobre las pruebas automatizadas del acceso a los datos de Neptune	Consulte Pruebas automatizadas del acceso a los datos de Amazon Neptune con Apache TinkerPop Gremlin, de Greg Biegel.	28 de septiembre de 2022

Versión del motor 1.1.1.0.R6	A partir del 23 de septiembre de 2022, se implementará de forma general la versión 1.1.1.0.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.1.0.R6.	23 de septiembre de 2022
Entrada de blog sobre cómo Informatica® utiliza Neptune	Consulte How Informatica® Cloud Data Governance and Catalog uses Amazon Neptune for knowledge graphs, de Tiju Titus John, Deepak Ram y Farooq Ashraf.	20 de septiembre de 2022
Entrada de blog sobre el uso de las perspectivas de Neptune y Tom Sawyer para detectar fraudes financieros	Consulte Uncover financial fraud with Amazon Neptune and Tom Sawyer Perspectives, de Janet M. Six, director de producto sénior en Tom Sawyer Software.	30 de agosto de 2022
Entrada de blog sobre la creación de una solución de detección de fraudes en tiempo real basada en GNN con SageMaker Neptune y DGL	Consulte Cree una solución de detección de fraudes en tiempo real basada en GNN con Amazon SageMaker, Amazon Neptune y la biblioteca Deep Graph de Jian Zhang, Haozhu Wang y Mengxin Zhu.	11 de agosto de 2022

Entrada de blog sobre el uso de etiquetas de recursos para detener e iniciar los recursos del entorno de Neptune	Consulte Automate the stopping and starting of Amazon Neptune environment resources using resource tags, de Kevin Phillips.	1 de agosto de 2022
Entrada de blog sobre un artista que colabora con Apache TinkerPop	Véase Beyond Code: The Artist Who Contributions to Apache, TinkerPop de Stephen Mallette y Ketrina Thompson.	1 de agosto de 2022
Entrada de blog sobre control de acceso detallado para las acciones del plano de datos de Neptune	Consulte Fine Grained Access Control for Amazon Neptune data plane actions, de Abhishek Mishra y Ankit Gupta.	29 de julio de 2022
Entrada de blog sobre las bases de datos globales de Neptune	Consulte Introducing Amazon Neptune Global Database, de Navtanay Sinha.	27 de julio de 2022
Versión del motor 1.2.0.0	A partir del 21 de julio de 2022, se implementará de forma general la versión 1.2.0.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.2.0.0.	21 de julio de 2022

Versión del motor 1.1.1.0.R5	A partir del 21 de julio de 2022, se implementará de forma general la versión 1.1.1.0.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.1.0.R5.	21 de julio de 2022
Versión del motor 1.1.1.0.R4	A partir del 23 de junio de 2022, se implementará de forma general la versión 1.1.1.0.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.1.0.R4.	23 de junio de 2022
Flujos de trabajo de machine learning y análisis de gráficos simplificados con la integración de Python	Ahora puede ejecutar tareas de análisis de gráficos y machine learning en datos de gráficos almacenados en Amazon Neptune mediante una integración de código abierto de Python que simplifica los flujos de trabajo de ciencia de datos y ML. Consulte la documentación de AWS Data Wrangler para Neptune.	7 de junio de 2022

Versión del motor 1.1.1.0.R3	A partir del 7 de junio de 2022, se implementará de forma general la versión del motor 1.1.1.0.R3. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.1.0.R3.	7 de junio de 2022
Entrada de blog sobre la detección de noticias falsas	Consulte Detect social media fake news using graph machine learning with Amazon Neptune ML, de Hasan Shojaei y Sarita Joshi.	19 de mayo de 2022
Entrada de blog sobre el uso de SQL Server Integration Services (SSIS) con Neptune	Consulte Discover new insights from your data using SQL Server Integration Services (SSIS) and Amazon Neptune, de Mesgana Gormley y Melissa Kwok.	18 de mayo de 2022
Versión de mantenimiento 1.1.1.0.R2	A partir del 16 de mayo de 2022, se implementará de forma general la versión de mantenimiento 1.1.1.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibl es en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.1.0.R2.	16 de mayo de 2022

Versión de mantenimiento 1.1.0.0.R2	A partir del 16 de mayo de 2022, se implementará de forma general la versión de mantenimiento 1.1.0.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibl es en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.0.0.R2.	16 de mayo de 2022
Versión de mantenimiento 1.0.5.1.R4	A partir del 16 de mayo de 2022, se implementará de forma general la versión de mantenimiento 1.0.5.1.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibl es en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.1.R4.	16 de mayo de 2022

Versión de mantenimiento 1.0.5.0.R5	A partir del 16 de mayo de 2022, se implementará de forma general la versión de mantenimiento 1.0.5.0.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibl es en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.0.R5.	16 de mayo de 2022
Entrada de blog sobre la disponibilidad general de OpenCypher en Neptune	Consulte Announcing the General Availability of openCypher support for Amazon Neptune, de Navtanay Sinha y Dave Bechberger.	22 de abril de 2022
Versión del motor 1.1.1.0	A partir del 19 de abril de 2022, se implementará de forma general la versión 1.1.1.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.1.0.	19 de abril de 2022

Entrada de blog sobre el linaje de datos	Consulte Cree un linaje de datos para lagos de datos con AWS Glue, Amazon Neptune y Spline, de Khoa Nguyen, Krithivasan Balasubramaniyan y Rahul Shaurya.	1 de abril de 2022
Se han vuelto a habilitar las actualizaciones de la versión 1.1.0.0 del motor	A partir del 21 de febrero de 2022, las actualizaciones a la versión 1.1.0.0 del motor se deshabilitaron temporalmente. Se han vuelto a habilitar.	22 de marzo de 2022
Entrada de blog sobre la fiabilidad de los servicios de ingeniería	Consulte Using cloud-based, data-informed, power system models to engineer utility reliability, de Abhineet Parchure.	22 de marzo de 2022
Neptune se ha lanzado en África (Ciudad del Cabo)	Amazon Neptune ya está disponible en la región de África (Ciudad del Cabo) (`af-south-1`). Sin embargo, la compatibilidad con los bloc de notas del entorno de trabajo de Neptune está deshabilitada temporalmente en la consola de Neptune de esa región.	24 de febrero de 2022
Entrada de blog sobre gráficos basados en modelos que utilizan OWL	Consulte Model-driven graphs using OWL in Amazon Neptune, de Mike Havey.	23 de febrero de 2022

Entrada de blog sobre cómo explorar los gráficos de conocimiento semántico con Rhizomer	Consulte Explore the semantic knowledge graphs without SPARQL using Amazon Neptune with Rhizomer, de Roberto García.	22 de febrero de 2022
Entrada de blog sobre la representación gráfica de la red eléctrica	Véase AWS Graphing the utilities grid on, de Bobby Wilson y Joseph Beer.	18 de febrero de 2022
Nuevas opciones de codificación de características de texto de Neptune ML	Neptune ahora admite FastText la codificación de texto Sentence BERT para el entrenamiento. Consulte FastTextlas características de Neptune ML y las funciones Sentence BERT de Neptune ML.	15 de febrero de 2022
Entrada de blog sobre consultas geoespaciales que se utilizan OpenSearch con Neptune	Consulte Combine Amazon Neptune y Amazon OpenSearch Service para consultas geoespaciales, de Ross Gabay y Abhilash Vinod.	1 de febrero de 2022
Entrada de blog sobre la detección de delitos financieros con Amazon EKS y Neptune	Consulte Financial Crime Discovery using Amazon EKS and Graph Databases, de Severin Gassauer-Fleissner y Zahi Ben Shabat.	1 de febrero de 2022

Ahora un volumen de clúster de Neptune puede llegar a un tamaño de 128 tebibytes (TiB)	En todas las regiones compatibles, excepto en China GovCloud, el límite de tamaño del volumen de un cúmulo de Neptune ahora ha aumentado de 64 TiB a 128 TiB. Esto se aplica a todas las versiones del motor a partir de la versión 1.0.2.2. Consulte la página de almacenamiento de Amazon Neptune.	1 de febrero de 2022
La búsqueda de texto completo de Neptune ahora se integra con todas las versiones de. OpenSearch	Consulte Búsqueda de texto completo en Amazon Neptune mediante Amazon OpenSearch Service.	28 de enero de 2022
Entrada de blog sobre el uso de contenedores Docker para implementar cuadernos de gráficos	Consulte Uso de contenedores Docker para implementar Graph Notebooks en AWS, de Ganesh Sawhney y Qiang Zhang.	22 de enero de 2022
Versión del motor 1.0.5.1.R3	A partir del 13 de enero de 2022, se implementará de forma general la versión 1.0.5.1.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.1.R3.	13 de enero de 2022

Entrada de blog sobre un sistema de recomendaciones basado en gráficos que utiliza Neptune ML	Consulte Graph-based recommendation system with Neptune ML: An illustration on social network link prediction challenges, de Yanwei Cui y Will Badr.	12 de enero de 2022
Entrada de blog sobre el escalado automático de Neptune	Consulte Auto scale your Amazon Neptune database to meet workload demands, de Navtanay Sinha y Sudhanshu Gupta.	29 de noviembre de 2021
Entrada de blog sobre visualizaciones y análisis de datos de gráficos interactivos	Consulte Cree visualizaciones y análisis de datos gráficos interactivos con Amazon Neptune, Amazon Athena Federated Query y Amazon, de Sandeep Veldi y QuickSight Abhishek Mishra.	24 de noviembre de 2021
Entrada de blog sobre la detección de fraudes de identidad mediante el aprendizaje profundo basado en gráficos	Consulte How Careem is detecting identity fraud using graph-based deep learning and Amazon Neptune, de Kevin O'Brien, Kamran Habib y Will Badr.	23 de noviembre de 2021

Versión del motor 1.1.0.0	A partir del 19 de noviembre de 2021, se implementará de forma general la versión 1.1.0.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.1.0.0.	19 de noviembre de 2021
Entrada de blog sobre la centralización de la protección de datos y el cumplimiento	Consulte Centralización de la protección y el cumplimiento de datos en Amazon Neptune with AWS Backup, de Brian O'Keefe.	8 de noviembre de 2021
Entrada de blog sobre la lucha contra el fraude y los pagos indebidos	Consulte Fighting fraud and improper payments in real-time at the scale of federal expenditures, de Vladi Royzman and Spencer Smith.	2 de noviembre de 2021
Entrada de blog sobre la prevención de registros de cuentas falsas	Consulte Prevent fake account sign-ups in real time with AI using Amazon Fraud Detector, de Anjan Biswas.	29 de octubre de 2021

Versión del motor 1.0.5.1.R2	A partir del 26 de octubre de 2021, se implementará de forma general la versión del motor 1.0.5.1.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.1.R2.	26 de octubre de 2021
Entrada de blog sobre cómo predecir HawkEye 360 grados el riesgo de los buques mediante la biblioteca Deep Graph	See HawkEye 360 predice el riesgo de las embarcaciones utilizando la biblioteca Deep Graph y Amazon Neptune de Tim Pavlick, Ian Avilez, Dan Ford y Gaurav Rele.	15 de octubre de 2021
Versión del motor 1.0.5.1	A partir del 1 de octubre de 2021, se implementará de forma general la versión 1.0.5.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.1.	1 de octubre de 2021

Entrada de blog sobre por qué a los desarrolladores les gusta TinkerPop	Consulte Por qué a los desarrolladores les gusta Apache TinkerPop, un marco de código abierto para la computación gráfica, de Brad Bebee, Kelvin Lawrence y Stephen Mallette.	27 de septiembre de 2021
Versión del motor 1.0.5.0.R3	A partir del 15 de septiembre de 2021, se implementará de forma general la versión 1.0.5.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.0.R3.	15 de septiembre de 2021
Entrada de blog sobre la creación de una solución de detección de datos con Amundsen y Neptune	Consulte Building a data discovery solution with Amundsen and Amazon Neptune, de Peter Hanssens y Don Simpson.	8 de septiembre de 2021
Neptune ha actualizado el sondeador de flujos para admitir consultas de búsqueda de texto completo sin cadenas	En esta versión, se incluyen muchas mejoras en la búsqueda de texto completo, incluida la compatibilidad con la indexación de valores de propiedades que no sean cadenas. Consulte OpenSearch Indexación sin cadenas en Amazon Neptune.	23 de agosto de 2021

Versión del motor 1.0.5.0.R2	A partir del 16 de agosto de 2021, se implementará de forma general la versión del motor 1.0.5.0.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.0.R2.	16 de agosto de 2021
Versión del motor 1.0.4.2.R5	A partir del 16 de agosto de 2021, se implementará de forma general la versión 1.0.4.2.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.2.R5.	16 de agosto de 2021
Entrada de blog sobre la compatibilidad con el protocolo Graph Store en Neptune	Consulte Introducing Graph Store Protocol support for Amazon Neptune, de Chris Smith.	2 de agosto de 2021
Entrada de blog sobre las nuevas características de Neptune ML	Consulte Discover more insights in your graphs with new features from Amazon Neptune ML, de Soji Adeshina.	30 de julio de 2021

Entrada de blog sobre cómo obtener predicciones más rápido con Neptune ML	Consulte Get predictions for evolving graph data faster with Amazon Neptune ML, de Soji Adeshina.	30 de julio de 2021
Entrada de blog sobre un machine learning de gráficos más fácil y rápido con Neptune ML	Consulte Easier and faster graph machine learning with Amazon Neptune ML, de Soji Adeshina.	30 de julio de 2021
Entrada de blog sobre la compatibilidad de Neptune OpenCypher	Consulte Announcing openCypher for Amazon Neptune: Building better graph applications with openCypher and Gremlin together, de Brad Bebee.	29 de julio de 2021
Versión del motor 1.0.5.0	A partir del 27 de julio de 2021, se implementará de forma general la versión 1.0.5.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.5.0.	27 de julio de 2021

Versión del motor 1.0.4.2.R4	A partir del 23 de julio de 2021, se implementará de forma general la versión 1.0.4.2.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.2.R4.	23 de julio de 2021
Se ha lanzado Neptune en China (Pekín)	Amazon Neptune ya está disponible en China (Pekín) (`cn-north-1`).	21 de julio de 2021
Versión del motor 1.0.4.2.R3	A partir del 28 de junio de 2021, se implementará de forma general la versión 1.0.4.2.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.2.R3.	28 de junio de 2021
Entrada de blog sobre cómo Dream11 escaló su red social usando Neptune	Descubre cómo Dream11, la plataforma de deportes de fantasía más grande del mundo, amplía su red social con Amazon Neptune y Amazon. ElastiCache	25 de junio de 2021

Entrada de blog sobre la transformación de datos en conocimiento con Neptune y Semantic Suite PoolParty	Consulte Transforma los datos en conocimiento con PoolParty Semantic Suite y Amazon Neptune, de Ioanna Lytra y Albin Ahmeti.	16 de junio de 2021
Entrada de blog sobre el uso de Neptune para explorar la base de conocimientos UniProt	Consulte Explorando la base de conocimientos sobre UniProt proteínas con AWS Open Data y Amazon Neptune, de Eric Greene, Rafa Xu y Yuan Shi.	10 de junio de 2021
Entrada de blog sobre el uso de Neptune para el análisis de riesgos basado en datos	Consulte Notas de campo: Análisis de riesgos basado en datos con Amazon Neptune y OpenSearch Amazon Service, de Adriaan de Jonge y Rohit Satyanarayana.	10 de junio de 2021
Versión del motor 1.0.4.2.R2	A partir del 1 de junio de 2021, se implementará de forma general la versión 1.0.4.2.R 2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.2.R2.	1 de junio de 2021
Entrada de blog sobre el uso de Neptune para visualizar su infraestructura AWS	Consulte Visualice su AWS infraestructura con Amazon Neptune y AWS Config, de Rohan Raizada y Amey Dhavle.	25 de mayo de 2021

Entrada de blog sobre la configuración para usar Data Lens con Neptune	Consulte Configurar AWS servicios para crear un gráfico de conocimiento en Amazon Neptune mediante Data Lens, de Russell Waterson.	5 de mayo de 2021
Entrada de blog sobre la creación de un gráfico de conocimiento en Neptune con Data Lens	Consulte Build a knowledge graph in Amazon Neptune using Data Lens, de Russell Waterson.	5 de mayo de 2021
Las versiones del motor 1.0.1.0, 1.0.1.1 y 1.0.1.2 ya están en desuso	A partir de ahora, no se creará ninguna nueva instancia de base de datos con ninguna de estas versiones del motor ni con ningún parche relacionado con ellas.	26 de abril de 2021
Traducción al inglés del estudio de caso sobre el uso de Neptune en el Ministerio de Economía, Comercio e Industria de Japón	Consulte Japan's Ministry of Economy, Trade and Industry Powers gBizINFO Corporate Information Search Database with AWS.	31 de marzo de 2021
Entrada de blog sobre el uso de Neptune con Amazon Comprehend y Lex	Consulte Supercharge your knowledge graph using Amazon Neptune, Amazon Comprehend, and Amazon Lex, de Dave Bechberger.	31 de marzo de 2021
Entrada de blog sobre el uso de funciones de Lambda con Neptune	Consulte Uso de funciones de AWS Lambda con Amazon Neptune, de Ian Robinson.	26 de marzo de 2021

Versión del motor 1.0.4.1.R1.1	A partir del 22 de marzo de 2021, se implementará de forma general la versión 1.0.4.1.R1.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.1.R1.1.	22 de marzo de 2021
Versión del motor 1.0.4.1.R2.1	A partir del 11 de marzo de 2021, se implementará de forma general la versión 1.0.4.1.R2.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.1.R2.1.	11 de marzo de 2021
Entrada de blog sobre el uso del cuaderno de gráficos de código abierto de Neptune para la visualización de gráficos	Consulte Getting started with open source graph notebook for graph visualization, de Joy Wang, Ora Lassila y Stephen Mallette.	10 de marzo de 2021
Tutorial sobre la integración de Neptune con el motor de detección de datos y metadatos de Amundsen	Consulte How to use Amundsen with Amazon Neptune, de Andrew Ciambrone.	2 de marzo de 2021

Versión del motor 1.0.4.1.R2	A partir del 24 de febrero de 2021, se implementará de forma general la versión del motor 1.0.4.1.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.1.R2.	24 de febrero de 2021
Versión del motor 1.0.4.0.R2	A partir del 24 de febrero de 2021, se implementará de forma general la versión del motor 1.0.4.0.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.0.R2.	24 de febrero de 2021
Versión del motor 1.0.3.0.R3	A partir del 19 de febrero de 2021, se implementará de forma general la versión 1.0.3.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.3.0.R3.	19 de febrero de 2021

Versión del motor 1.0.2.2.R6	A partir del 19 de febrero de 2021, se implementará de forma general la versión 1.0.2.2.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.2.2.R6.	19 de febrero de 2021
Entrada de blog sobre la creación de un gráfico de conocimientos mediante eventos de Amazon Comprehend	Consulte Building a knowledge graph in Amazon Neptune using Amazon Comprehend Events, de Brian O'Keefe, Graham Horwood y Navtanay Sinha.	19 de enero de 2021
Entrada de blog sobre cómo habilitar aplicaciones de datos de gráficos de poco código	Consulte Enabling low code graph data apps with Amazon Neptune and Graphistry, de Leo Meyerovich, Dave Bechberger y Taylor Riggan.	18 de enero de 2021
Se ha añadido documentación de cuaderno sobre cómo comenzar a utilizar los datos de gráficos.	Se ha añadido una sección que se integra con el entorno de trabajo de Neptune que le ayuda a comenzar a crear datos de gráficos y desarrollar aplicaciones gráficas sin tener que poner en marcha un clúster de Neptune hasta que esté listo.	15 de enero de 2021

Entrada de blog sobre cómo restablecer los datos de su gráfico de Neptune en cuestión de segundos	Consulte Resetting your graph data in Amazon Neptune in seconds, de Niraj Jetly y Navtanay Sinha.	17 de diciembre de 2020
Entrada de blog sobre cómo Novartis AG utiliza SageMaker Neptuno con BERT	Vea cómo Novartis AG utiliza Amazon SageMaker y Amazon Neptune para crear y enriquecer un gráfico de conocimiento con BERT, de Othmane Hamzaoui, Fatema Alkhanaizi y Viktor Malesevic.	14 de diciembre de 2020
Entrada de blog sobre la creación de un gráfico de conocimientos con redes temáticas	Consulte Building a knowledge graph with topic networks in Amazon Neptune, de Edward Brown, director de proyectos de IA, Eduardo Piairo, Architect, Marcia Oliveira, científica de datos jefe, y Jack Hampson, director general de Deeper Insights.	14 de diciembre de 2020
Versión del motor 1.0.4.1	A partir del 8 de diciembre de 2020, se implementará de forma general la versión 1.0.4.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.1.	8 de diciembre de 2020

Entrada de blog sobre cómo empezar a usar Neptune ML	Consulte How to get started with Neptune ML, de George Karypis, Dave Bechberger y Karthik Bharathy.	8 de diciembre de 2020
Neptune ahora tiene una API de restablecimiento rápido	Con la API de restablecimiento rápido, puede eliminar rápida y fácilmente todos los datos de un clúster de base de datos. Consulte Fast reset API.	4 de diciembre de 2020
Entrada de blog sobre la creación de un gráfico de conocimientos biológicos en Pendulum	Consulte Building a biological knowledge graph at Pendulum using Amazon Neptune, de Connor Skennerton.	26 de noviembre de 2020
Entrada de blog sobre las nuevas funciones de la versión TinkerPop 3.4.8 en Neptune	Consulte Explorando las nuevas características de Apache TinkerPop 3.4.8 en Amazon Neptune, de Stephen Mallette.	18 de noviembre de 2020
Entrada de blog sobre el uso del servicio de búsqueda de Amazon Kendra con Neptune	Consulte Incorporating your enterprise knowledge graph into Amazon Kendra, de Yazdan Shirvany, Mohit Mehta y Dipto Chakravarty.	17 de noviembre de 2020
Las notificaciones de eventos ya están disponibles	Neptune ahora admite notificaciones de eventos que puede usar para monitorizar más fácilmente los clústeres de bases de datos. Consulte Using Neptune Event Notification, de	29 de octubre de 2020

Ya están disponibles los puntos de conexión personalizados	Neptune ahora admite puntos de conexión personalizados para disponer de un mayor control al conectarse a las instancias de base de datos. Consulte Connecting to Amazon Neptune Endpoints.	29 de octubre de 2020
Entrada de blog sobre el uso del AWS Database Migration Service (DMS) para rellenar el gráfico de Neptune	Consulte Rellenar un gráfico en Amazon Neptune a partir de una base de datos relacional mediante AWS Database Migration Service (DMS), parte 4: Cómo unirlo todo, de Chris Smith.	22 de octubre de 2020
Entrada de blog sobre el uso del AWS Database Migration Service (DMS) para rellenar el gráfico de Neptune	Consulte Rellenar un gráfico en Amazon Neptune a partir de una base de datos relacional mediante AWS Database Migration Service (DMS), parte 3: Diseño del modelo RDF, de Chris Smith.	22 de octubre de 2020
Entrada de blog sobre el uso del AWS Database Migration Service (DMS) para rellenar el gráfico de Neptune	Consulte Rellenar un gráfico en Amazon Neptune a partir de una base de datos relacional mediante Database AWS Migration Service (DMS), parte 2: Diseñar el modelo de gráficos de propiedades, de Chris Smith.	22 de octubre de 2020

Entrada de blog sobre el uso del AWS Database Migration Service (DMS) para rellenar el gráfico de Neptune	Consulte Rellenar un gráfico en Amazon Neptune a partir de una base de datos relacional mediante AWS Database Migration Service (DMS), parte 1: Preparando el escenario, de Chris Smith.	22 de octubre de 2020
Versión del motor 1.0.4.0	A partir del 12 de octubre de 2020, se implementará de forma general la versión 1.0.4.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.4.0.	12 de octubre de 2020
Versión del motor 1.0.3.0.R2	A partir del 12 de octubre de 2020, se implementará de forma general la versión 1.0.3.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.3.0.R2.	12 de octubre de 2020

Versión del motor 1.0.2.2.R5	A partir del 12 de octubre de 2020, se implementará de forma general la versión 1.0.2.2.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.2.2.R5.	12 de octubre de 2020
Entrada de blog sobre la configuración de su VPC para consultas federadas de SPARQL	Consulte Configure Amazon VPC for SPARQL 1,1 Federated Query with Amazon Neptune, de Charles Ivie.	12 de octubre de 2020
Entrada de blog sobre cómo crear una eliminación en cascada de SPARQL	Consulte Write a cascading delete in SPARQL, de Ora Lassila.	5 de octubre de 2020
Entrada de blog sobre la representación gráfica de AWS recursos con Neptuno	Consulte Gráfica tus AWS recursos con Amazon Neptune, de Dave Bechberge r.	28 de septiembre de 2020
Entrada de blog sobre la creación de un mapa de terminología de MedDRA para la notificación de eventos adversos y farmacovigilancia mediante Neptune	Consulte Building Amazon Neptune based MedDRA terminology mapping for pharmacovigilance and adverse event reporting, de Vaijayanti Joshi, Deven Atnoor, Ph.D, y Sudhanshu Malhotra.	24 de septiembre de 2020

Entrada de blog sobre la creación de un gráfico de conocimientos a partir de un almacenamiento de datos utilizando Neptune para complementar la inteligencia comercial	Consulte Complement Commercial Intelligence by Building a Knowledge Graph out of a Data Warehouse with Amazon Neptune, de Shahria Hossain y Mikael Graindorge.	23 de septiembre de 2020
Entrada de blog sobre el equilibrio de carga utilizando el cliente Neptune Gremlin	Consulte Load balance graph queries using the Amazon Neptune Gremlin Client, de Ian Robinson.	16 de septiembre de 2020
Entrada de blog sobre la personalización digital mediante un gráfico de identidades en Cox Automotive	Consulte Cox Automotive scales digital personalization using an identity graph powered by Amazon Neptune, de Carlos Rendon y Niraj Jetly.	16 de septiembre de 2020
Entrada de blog sobre el filtrado colaborativo de datos de Yelp	Consulte Using collaborative filtering on Yelp data to build a recommendation system in Amazon Neptune, de Chad Tindel.	8 de septiembre de 2020
Entrada de blog sobre la visualización de resultados de consultas en Amazon Neptune	Consulte Visualize query results using the Amazon Neptune workbench, de Kelvin Lawrence.	2 de septiembre de 2020

Neptune ha lanzado la visualización de gráficos	Amazon Neptune ahora ofrece amplias capacidades de visualización de gráficos en los cuadernos de Jupyter del entorno de trabajo de Neptune, además de una serie de nuevas características que facilitan el uso de los cuadernos. Consulte Graph visualization.	12 de agosto de 2020
Neptune se ha lanzado en América del Sur (São Paulo)	Amazon Neptune ya está disponible en América del Sur (São Paulo) (`sa-east-1`).	6 de agosto de 2020
Neptune se ha lanzado en Asia-Pacífico (Hong Kong)	Amazon Neptune ya está disponible en Asia-Pacífico (Hong Kong) (`ap-east-1`).	6 de agosto de 2020
Versión del motor 1.0.3.0	A partir del 3 de agosto de 2020, se implementará de forma general la versión 1.0.3.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.3.0.	3 de agosto de 2020

Versión del motor 1.0.2.2.R4	A partir del 23 de julio de 2020, se implementará de forma general la versión 1.0.2.2.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.2.2.R4.	23 de julio de 2020
Entrada de blog sobre el rastreo de contactos automatizado de Zerobase mediante Amazon Neptune	Consulte Zerobase creates private, secure, and automated contact tracing using Amazon Neptune, de David Harris y Aron Szanto.	13 de julio de 2020
Neptune se ha lanzado en el Oeste de EE. UU. (Norte de California)	Amazon Neptune ya está disponible en el Oeste de EE. UU. (Norte de California) (`us-west-1`).	9 de julio de 2020
Amazon Neptune admite el control de acceso basado en etiquetas	Ahora puede usar AWS etiquetas en las políticas de IAM para controlar el acceso a la base de datos de Neptune. Consulte Tag-based access control in Amazon Neptune.	7 de julio de 2020

Ya está disponible un sondeador de flujos de Java	Amazon Neptune ahora admite una versión Java del sondeador de flujos de Lambda para los flujos de Neptune, así como de Python. Consulte Add details about the Neptune streams consumer stack you are creating.	6 de julio de 2020
Entrada de blog sobre el gráfico de conocimiento sobre la AWS COVID-19	Consulte Cómo crear y consultar el gráfico de conocimientos sobre la AWS COVID-19, de Ninad Kulkarni, Colby Wise, George Price y Miguel Romero.	1 de julio de 2020
Versión del motor 1.0.1.1	A partir del 26 de junio de 2020, se implementará de forma general la versión 1.0.1.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.1.1.	26 de junio de 2020
Entrada de blog sobre la migración de Blazegraph a Amazon Neptune	Consulte Trasladarse a la nube: Migrar Blazegraph a Amazon Neptune, de Dave Bechberger.	25 de junio de 2020

Entrada de blog sobre cómo cambiar la captura de datos de Neo4j a Amazon Neptune	Consulte Cambiar captura de datos de Neo4j a Amazon Neptune mediante Amazon Managed Streaming for Apache Kafka, de Sanjeet Sahay.	22 de junio de 2020
Entrada de blog sobre cómo Waves está usando Amazon Neptune	Consulte Cómo Waves ejecuta las consultas y recomendaciones de los usuarios a escala con Amazon Neptune, de Pavel Vasilyev.	16 de junio de 2020
Versión del motor 1.0.1.2	A partir del 10 de junio de 2020, se implementará de forma general la versión 1.0.1.2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.1.2.	10 de junio de 2020
Entrada de blog sobre la creación de un repositorio de conocimiento del cliente	Consulte Creación de un repositorio de conocimientos 360 para clientes con Amazon Neptune y Amazon Redshift, de Ram Bhandarkar.	9 de junio de 2020
Entrada de blog sobre cómo Gunosy está usando Amazon Neptune	Consulte Cómo Gunosy creó una función de comentarios en News Pass usando Amazon Neptune de Yosuke Uchiyama.	8 de junio de 2020

Entrada de blog sobre el gráfico de conocimientos sobre la COVID-19 AWS	Consulte Cómo crear y consultar el gráfico de conocimientos sobre la AWS COVID-19 de Ninad Kulkarni, Colby Wise, George Price y Miguel Romero.	2 de junio de 2020
Entrada de blog sobre la exploración de la investigación sobre el COVID-19 usando Amazon Neptune	Consulte Exploring scientific research on COVID-19 with Amazon Neptune, Amazon Comprehend Medical, and the Tom Sawyer Graph Database Browser por George Price, Colby Wise, Miguel Romero y Ninad Kulkarni.	2 de junio de 2020
Ahora puede cargar datos en Neptune usando AWS DMS	Consulte Uso AWS de Database Migration Service para cargar datos en Amazon Neptune desde un almacén de datos diferente.	1 de junio de 2020
La versión 1.0.2.0 del motor está en desuso	La versión 1.0.2.0 del motor de Amazon Neptune se ha dejado de usar. Los clústeres que se ejecutan en esta versión del motor se actualizarán automáticamente a la versión 1.0.2.1 durante el primer período de mantenimiento posterior al 1 de junio de 2020.	19 de mayo de 2020
Entrada de blog sobre la creación de un gráfico de identidad de clientes usando Neptune	Consulte Building a customer identity graph with Amazon Neptune por Rajesh Wunnava y Taylor Riggan.	12 de mayo de 2020

Versión del motor 1.0.2.0.R3	A partir del 5 de mayo de 2020, se implementará de forma general la versión 1.0.2.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.2.0.R3.	5 de mayo de 2020
Versión del motor 1.0.2.1.R6	A partir del 22 de abril de 2020, se implementará de forma general la versión 1.0.2.1.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.2.1.R6.	22 de abril de 2020
Entrada de blog sobre la migración de datos de Neo4j a Neptune	Consulte Migración de una base de datos de gráficos Neo4j a Amazon Neptune con una utilidad totalmente automatizada de Sanjeet Sahay.	13 de abril de 2020
Entrada de blog sobre la reducción del costo de la construcción de aplicaciones gráficas con Neptune	Consulte Reducir el coste de creación de aplicaciones gráficas hasta en un 76 % con instancias de Amazon Neptune T3 de Karthik Bharathy y Brad Bebee.	9 de abril de 2020

Neptune ofrece una clase de instancia de ráfagas T3	Ahora puede crear una instancia de ráfagas T3de Amazon Neptune para realizar desarrollos y pruebas rentables. Consulte Clase de instancia de ráfaga Neptune T3.	8 de abril de 2020
Versión del motor 1.0.2.2.R2	A partir del 2 de abril de 2020, se implementará de forma general la versión 1.0.2.2.R 2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.2.2.R2.	2 de abril de 2020
Entrada de blog sobre el gráfico de la dependencia de la inversión en EDGAR	Consulte Gráficos de dependencia de inversión con Amazon Neptune por Lawrence Verdi.	17 de marzo de 2020
Neptune se ha lanzado en Europa (París)	Amazon Neptune ya está disponible en Europa (París) (eu-west-3).	11 de marzo de 2020

Versión del motor 1.0.2.2	A partir del 9 de marzo de 2020, se implementará de forma general la versión 1.0.2.2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones. Para obtener más información sobre esta versión del motor, consulte Neptune Engine Release 1.0.2.2.	9 de marzo de 2020
Detener y reiniciar un clúster de base de datos	Ahora puede detener un clúster de base de datos durante siete días utilizando la consola de Neptune y reiniciarlo más adelante, cuando lo necesite de nuevo. Mientras el clúster de base de datos esté detenido, solo se le cobrará por el almacenamiento del clúster, las instantáneas manuales y el almacenamiento de las copias de seguridad automatizadas, pero no por las horas de instancia de la base de datos. Consulte Detención e inicio de un clúster de base de datos de Amazon Neptune.	19 de febrero de 2020

Vídeo sobre un gráfico social en Nike	Escuche cómo Todd Escalona AWS conversa con Marc Wangenheim, gerente sénior de ingeniería de Nike, sobre cómo la empresa potencia una serie de aplicaciones a través de un gráfico social creado en Amazon Neptune. Ver Nike: Un gráfico social a escala con Amazon Neptune.	11 de febrero de 2020
Ahora, los clústeres de Neptune pueden configurarse para que requieran conexiones SSL	En las regiones que todavía admiten conexiones HTTP, SSL se activa de forma predeterminada en todos los nuevos grupos de parámetros. No hay cambios en los grupos de parámetros existentes, pero puede exigir que los clientes usen SSL cambiando el parámetro `neptune_enforce_ssl` a 1. Consulte Cifrado en tránsito: conexión con Neptune mediante SSL/HTTPS para obtener información acerca de cómo habilitar conexiones HTTP en un clúster de una región que todavía admita este tipo de conexiones. Consulte Parámetros que puede utilizar para configurar Amazon Neptune para ver una descripción de los parámetros del clúster y la instancia.	10 de febrero de 2020

Ahora puede especificar la versión del motor y la protección contra eliminaciones en la plantilla de Neptune CloudFormation	Amazon Neptune ha actualizado su CloudFormation plantilla para incluir un `AWS::Neptune::DBCluster.EngineVersion` parámetro que le permite especificar una versión de motor concreta para el nuevo clúster de base de datos y un `AWS::Neptune::DBCluster.DeletionProtection` parámetro que le permite activar la protección contra la eliminación de dicho clúster.	9 de febrero de 2020
Protección contra eliminación	Amazon Neptune ha proporcionado protección de eliminación para clústeres e instancias de base de datos. Si la protección de eliminación está habilitada en un clúster o instancia de base de datos, no podrá eliminarla. Consulte No se puede eliminar una instancia de base de datos si la protección de eliminación está habilitada.	20 de enero de 2020
Se ha lanzado Neptune en China (Ningxia)	Amazon Neptune ya está disponible en China (Ningxia) (cn-northwest-1).	15 de enero de 2020

Versión del motor 1.0.2.1.R4	El parche R4 para la versión 1.0.2.1 del motor está disponible con carácter general. Para obtener más información, consulte Neptune Engine Release 1.0.2.1.R4.	20 de diciembre de 2019
Versión del motor 1.0.2.1.R3	El parche R3 para la versión 1.0.2.1 del motor está disponible con carácter general. Para obtener más información, consulte Neptune Engine Release 1.0.2.1.R3.	12 de diciembre de 2019
Entrada de blog sobre el uso de Neptune para analizar las fuentes de las redes sociales	Consulte Análisis de fuentes de redes sociales con Amazon Neptune.	27 de noviembre de 2019
Versión del motor 1.0.2.1.R2	El parche R2 para la versión 1.0.2.1 del motor está disponible con carácter general. Para obtener más información, consulte Neptune Engine Release 1.0.2.1.R2.	25 de noviembre de 2019
Versión del motor 1.0.2.1.R1	La versión 1.0.2.1.R1 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte Neptune Engine Release 1.0.2.1.	22 de noviembre de 2019

Versión del motor 1.0.2.0.R2	El parche R2 para la versión 1.0.2.0 del motor está disponible con carácter general. Para obtener más información, consulte Neptune Engine Release 1.0.2.0.R2.	21 de noviembre de 2019
Entrada de blog sobre sesiones y talleres de Neptune en re:Invent 2019	Consulta tu guía de sesiones, talleres y charlas sobre Amazon Neptune en AWS re:Invent 2019.	20 de noviembre de 2019
Versión del motor 1.0.2.0.R1	La versión 1.0.2.0.R1 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte Neptune Engine Release 1.0.2.0.	8 de noviembre de 2019
Entrada de blog sobre la captura de cambios de gráfico mediante Neptune Streams	Consulte Capturar cambios de gráficos mediante Neptune Streams.	6 de noviembre de 2019
Versión del motor 1.0.1.0.200502.0	La versión 1.0.1.0.200502.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200502.0.	31 de octubre de 2019
Se ha lanzado Neptune en Medio Oriente (Baréin)	Amazon Neptune ya está disponible en Medio Oriente (Baréin) (me-south-1).	30 de octubre de 2019
Se ha lanzado Neptune en Canadá (centro)	Amazon Neptune ya está disponible en Canadá (centro) (ca-central-1).	30 de octubre de 2019

Entrada de blog sobre la nueva función SPARQL Streams de Neptune y la compatibilidad de consultas federadas de SPARQL	Consulte esta publicación sobre los siguientes temas Amazon Neptune lanza Streams, consulta federada de gráficos de SPARQL y mucho más.	17 de octubre de 2019
Versión del motor 1.0.1.0.200463.0	La versión 1.0.1.0.200463.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200463.0.	15 de octubre de 2019
Versión del motor 1.0.1.0.200457.0	La versión 1.0.1.0.200457.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200457.0.	19 de septiembre de 2019
Entrada de blog sobre la nueva característica SPARQL Explain de Neptune	Consulte Uso de SPARQL Explain para comprender la ejecución de consultas en Amazon Neptune.	17 de septiembre de 2019
Entrada de blog sobre la compatibilidad de Neptune con 3.4 TinkerPop	Consulte Amazon Neptune ahora es compatible con las funciones TinkerPop 3.4.	6 de septiembre de 2019
Entrada de blog sobre el uso de Neptune en Amazon PyTorch SageMaker	Vea una experiencia personalizada de «compra por estilo» en PyTorch Amazon y Amazon SageMaker Neptune.	22 de agosto de 2019

Entrada de blog sobre el uso de Neptune con AWS AppSync Amazon Elasticache	Consulte Integrar fuentes de datos alternativas con AWS AppSync: Amazon Neptune y Amazon. ElastiCache	22 de agosto de 2019
Neptune fue lanzado en AWS GovCloud (este de EE. UU.)	Amazon Neptune ya está disponible en AWS GovCloud (US-East) (us-gov-east-1).	21 de agosto de 2019
Neptune se lanzó en AWS GovCloud (EE. UU., oeste)	Amazon Neptune ya está disponible en AWS GovCloud (US-West) (us-gov-west-1).	14 de agosto de 2019
Versión del motor 1.0.1.0.200369.0	La versión 1.0.1.0.200369.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200369.0.	13 de agosto de 2019
Versión del motor 1.0.1.0.200366.0	La versión 1.0.1.0.200366.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200366.0.	26 de julio de 2019
Entrada de blog sobre el uso de Neptune en Amazon PyTorch SageMaker	Vea una experiencia personalizada de «compra por estilo» en PyTorch Amazon y Amazon SageMaker Neptune.	3 de julio de 2019

Versión del motor 1.0.1.0.200348.0	La versión 1.0.1.0.200348.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200348.0.	2 de julio de 2019
Neptune se ha lanzado en Europa (Estocolmo)	Amazon Neptune ya está disponible en Europa (Estocolmo) (eu-north-1).	27 de junio de 2019
Neptune ahora puede publicar registros de auditoría en Logs CloudWatch	Para obtener más información, consulte Publicar registros de Neptune en Amazon CloudWatch Logs.	18 de junio de 2019
Versión del motor 1.0.1.0.200310.0	La versión 1.0.1.0.200310.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200310.0.	12 de junio de 2019
Entrada de blog sobre LifeOmic JupiterOne	Descubra cómo JupiterOne simplifica LifeOmic las operaciones de seguridad y conformidad con Amazon Neptune.	2 de mayo de 2019
Neptune se ha lanzado en Asia-Pacífico (Seúl)	Amazon Neptune ya está disponible en Asia Pacífico (Seúl) (ap-northeast-2).	1 de mayo de 2019

Versión del motor 1.0.1.0.200296.0	La versión 1.0.1.0.200296.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200296.0.	1 de mayo de 2019
Neptune se ha lanzado en Asia-Pacífico (Bombay)	Amazon Neptune ya está disponible en Asia Pacífico (Bombay) (ap-south-1).	6 de marzo de 2019
Entrada de blog sobre sugerencias de consultas de Gremlin	Consulte la presentación de las sugerencias de consulta de Gremlin para Amazon Neptune.	26 de febrero de 2019
Neptune se ha lanzado en Asia-Pacífico (Tokio)	Amazon Neptune ya está disponible en Asia Pacífico (Tokio) (ap-northeast-1).	23 de enero de 2019
AWS CloudFormation plantilla para crear una AWS Lambda función para acceder a Neptune	Se actualizó la sección de introducción y se agregó una AWS CloudFormation plantilla para crear una función Lambda para usarla con Neptune. Para obtener más información, consulte Introducción a Amazon Neptune.	23 de enero de 2019

Versión del motor 1.0.1.0.200267.0	La versión 1.0.1.0.200267.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200267.0.	21 de enero de 2019
Neptune se ha lanzado en Asia-Pacífico (Sídney)	Amazon Neptune ya está disponible en Asia Pacífico (Sídney) (ap-southeast-1).	9 de enero de 2019
Entrada de blog sobre el uso de Metaphactory	Consulte la exploración de gráficos de conocimiento en Amazon Neptune mediante Metaphactory.	9 de enero de 2019
Neptune se ha lanzado en Asia-Pacífico (Singapur)	Amazon Neptune ya está disponible en Asia Pacífico (Singapur) (ap-southeast-1).	13 de diciembre de 2018
Versión del motor 1.0.1.0.200264.0	La versión 1.0.1.0.200264.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200264.0.	19 de noviembre de 2018
Compatibilidad con SSL de Amazon Neptune	Neptune ahora admite conexiones SSL.	19 de noviembre de 2018
Temas de errores consolidados	Todos los mensajes de error y la información de código ahora están en un solo tema.	15 de noviembre de 2018

Tema de introducción actualizado	Se ha actualizado el tema de introducción con enlaces adicionales y documentación reorganizada.	14 de noviembre de 2018
Versión del motor 1.0.1.0.200258.0	La versión 1.0.1.0.200258.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200258.0.	8 de noviembre de 2018
Neptune se ha lanzado en Europa (Fráncfort)	Amazon Neptune ya está disponible en Europa (Fráncfort) (eu-central-1).	7 de noviembre de 2018
Entrada de blog n.º 1 de una serie	Consulte la publicación sobre Lo explicaré de forma gráfica - Parte 1 - Rutas aéreas.	7 de noviembre de 2018
Entrada de blog sobre el uso de Amazon SageMaker Jupyter Notebooks	Consulte Análisis de gráficos de Amazon Neptuno con Amazon SageMaker Jupyter Notebooks.	1 de noviembre de 2018
Versión del motor 1.0.1.0.200255.0	La versión 1.0.1.0.200255.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200255.0.	29 de octubre de 2018
Neptune se ha lanzado en Europa (Londres)	Amazon Neptune ya está disponible en Europa (Londres) (eu-west-2).	3 de octubre de 2018

Versión del motor 1.0.1.0.200237.0	La versión 1.0.1.0.200237.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200237.0.	6 de septiembre de 2018
Versión del motor 1.0.1.0.200236.0	La versión 1.0.1.0.200236.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200236.0.	24 de julio de 2018
Versión del motor 1.0.1.0.200233.0	La versión 1.0.1.0.200233.0 del motor de Amazon Neptune está disponible con carácter general. Para obtener más información, consulte la actualización 1.0.1.0.200233.0.	22 de junio de 2018
Nuevo inicio rápido de Neptune	Se actualizó el tutorial de inicio rápido AWS CloudFormation y el tutorial de la consola Gremlin. Para obtener más información, consulte Amazon Neptune Quick Start Using AWS CloudFormation.	19 de junio de 2018

Versión inicial de Amazon Neptune	Esta es la versión inicial de la Guía del usuario de Neptune. Consulte también la entrada de blog del lanzamiento sobre Amazon Neptune disponible en general.	30 de mayo de 2018
Entrada de blog de introducción de Neptune	Consulte Amazon Neptune: un servicio de base de datos de gráficos completamente administrado.	29 de noviembre de 2017

Introducción a Amazon Neptune

Amazon Neptune es un servicio de base de datos de gráficos totalmente administrado que se escala para gestionar miles de millones de relaciones y le permite consultarlas con una latencia de milisegundos, a un bajo costo para ese tipo de capacidad.

Si quiere obtener información más detallada sobre Neptune, consulte Información general de las características de Amazon Neptune.

Si ya conoce los gráficos, continúe directamente con Uso de los cuadernos de gráficos. O bien, si desea crear una base de datos de Neptune de inmediato, consulte Uso de una AWS CloudFormation pila para crear un clúster de base de datos de Neptune.

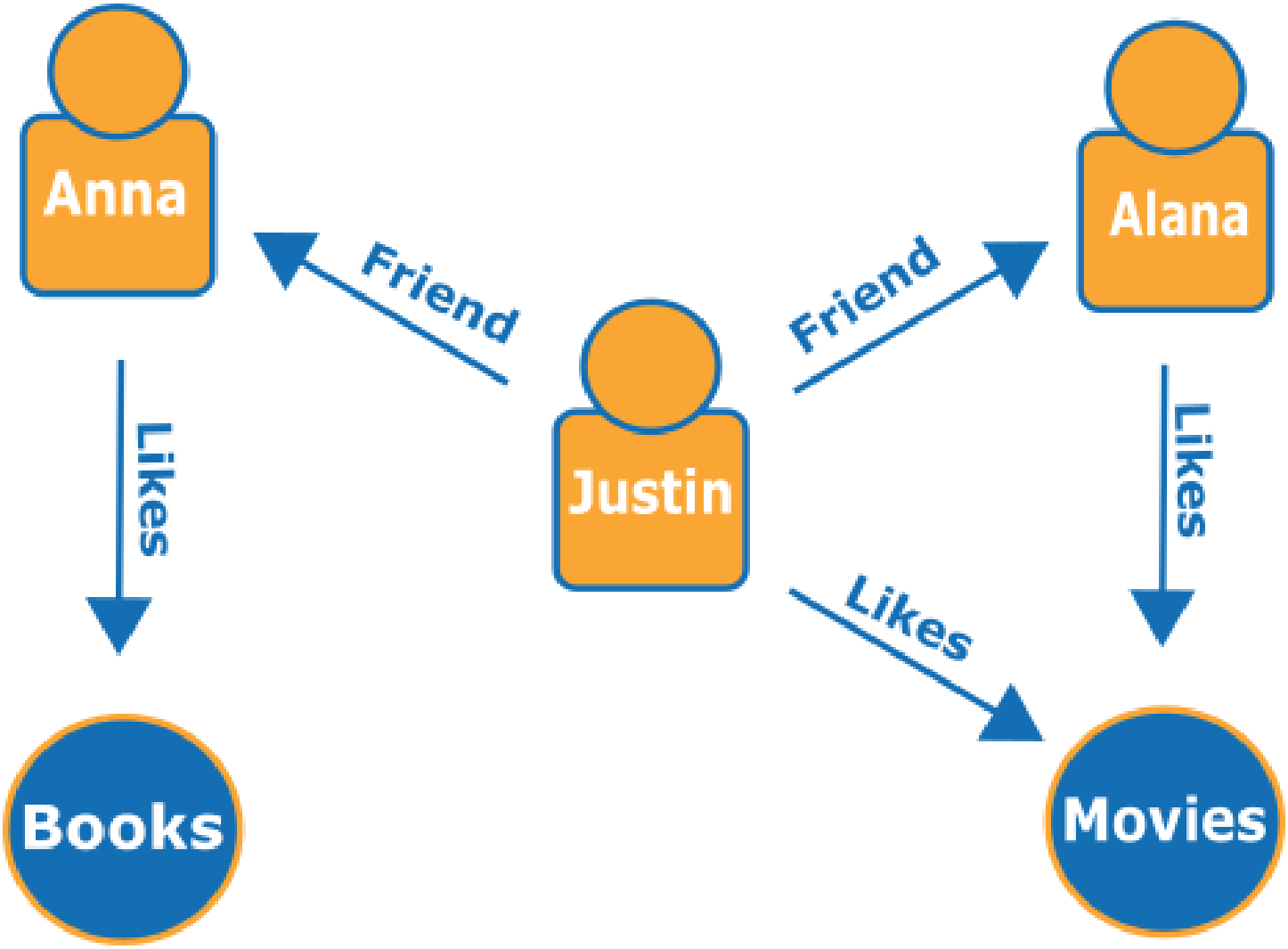
De lo contrario, puede que desee saber un poco más sobre las bases de datos de gráficos antes de empezar.

¿Qué es exactamente una base de datos de gráficos?

Las bases de datos de gráficos están optimizadas para almacenar y consultar las relaciones entre los elementos de datos.

Almacenan los propios elementos de datos como vértices del gráfico y las relaciones entre ellos como bordes. Cada borde tiene un tipo y se dirige de un vértice (el inicio) a otro (el final). Las relaciones pueden denominarse predicados, así como bordes, y, en ocasiones, los vértices también se denominan nodos. En los denominados gráficos de propiedades, tanto los vértices como los bordes también pueden tener propiedades adicionales asociadas.

A continuación, se muestra un pequeño gráfico que representa amigos y aficiones en una red social:



Los bordes se muestran con flechas con nombres y los vértices representan personas y aficiones específicas que conectan.

En un recorrido simple de este gráfico se puede saber qué les gusta a los amigos de Justin.

¿Por qué usar una base de datos de gráficos?

Cuando las conexiones o relaciones entre entidades forman parte fundamental de los datos que intenta modelar, una base de datos de gráficos es la elección más obvia.

Por un lado, es fácil modelar las interconexiones de datos como un gráfico y, después, escribir consultas complejas que extraigan información del mundo real del gráfico.

Para crear una aplicación equivalente mediante una base de datos relacional, es necesario crear muchas tablas con varias claves externas y, a continuación, escribir consultas SQL anidadas y

combinaciones complejas. Este enfoque no solo se vuelve difícil de gestionar desde el punto de vista de la codificación, sino que su rendimiento se degrada rápidamente a medida que aumenta la cantidad de datos.

Por el contrario, una base de datos de gráficos como Neptune puede consultar relaciones entre miles de millones de vértices sin bloquearse.

¿Qué se puede hacer con una base de datos de gráficos?

Los gráficos pueden representar las interrelaciones de las entidades del mundo real de muchas maneras, en términos de acciones, propiedad, parentesco, opciones de compra, conexiones personales, vínculos familiares, etc.

A continuación, se muestran algunas de las áreas más comunes en las que se utilizan las bases de datos de gráficos:

- Gráficos de conocimiento: los gráficos de conocimiento le permiten organizar y consultar todo tipo de información conectada para responder a preguntas generales. Con un gráfico de conocimiento, puede añadir información sobre un tema a los catálogos de productos y modelar información diversa, como la que se encuentra en Wikidata.

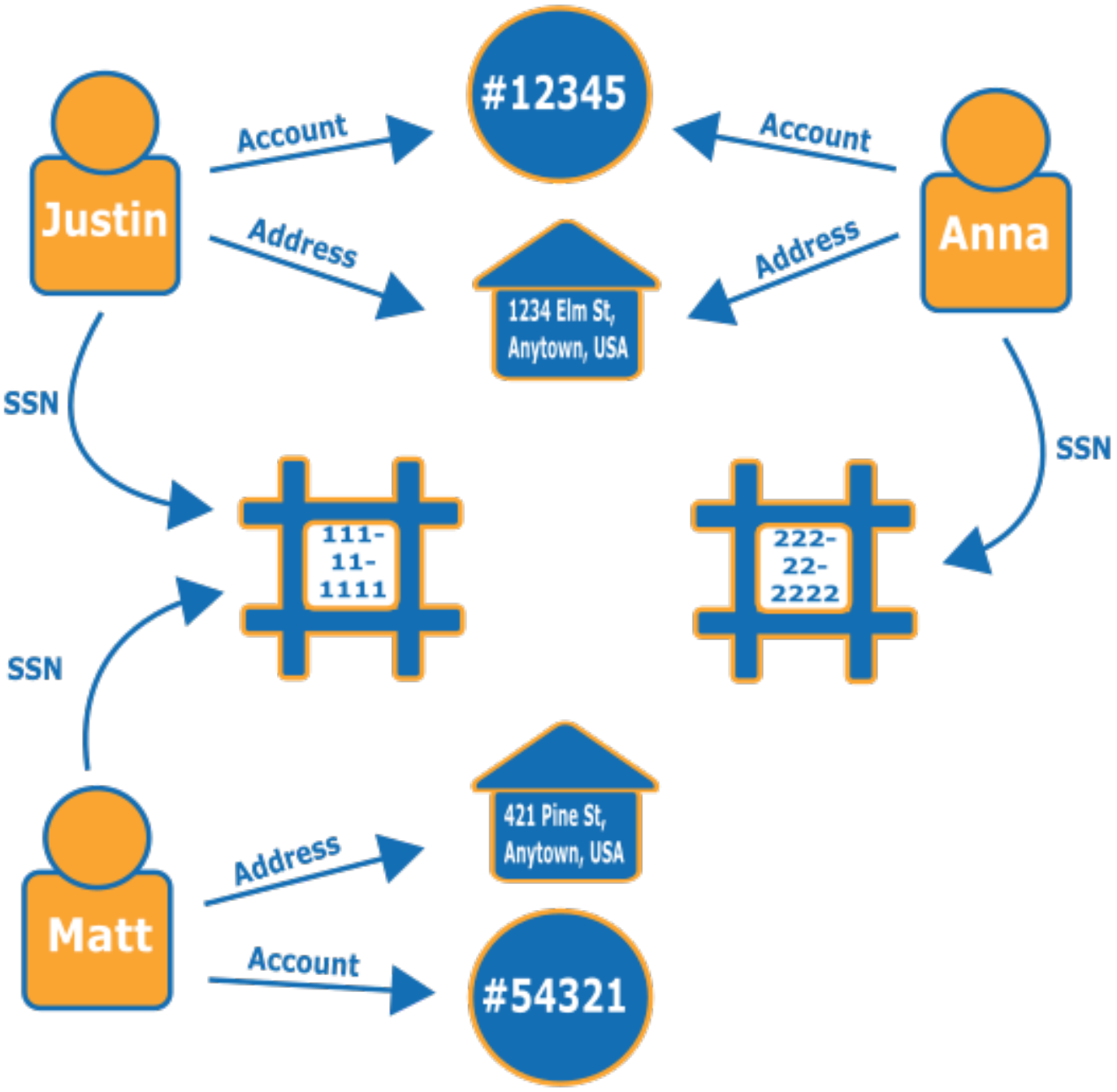
  Para obtener más información sobre cómo funcionan los gráficos de conocimiento y dónde se utilizan, consulte Gráficos de conocimiento en AWS.
- Gráficos de identidad: en una base de datos de gráficos, puede almacenar las relaciones entre categorías de información, como los intereses de los clientes, los amigos y el historial de compras, y luego consultar esos datos para hacer recomendaciones personalizadas y relevantes.

  Por ejemplo, puede utilizar una base de datos de gráficos para recomendar productos a un usuario a partir de los productos que han comprado otros usuarios que siguen el mismo deporte o que presentan un historial de compras similar. También puede identificar a las personas que tienen un amigo en común, pero que todavía no se conocen, y enviarles una recomendación de amistad.

  Los gráficos de este tipo se conocen como gráficos de identidad y se utilizan habitualmente para personalizar las interacciones con los usuarios. Para obtener más información, consulte Gráficos de identidad en AWS. Para crear su propio gráfico de identidad, puede empezar con el ejemplo Gráfico de identidad mediante Amazon Neptune.
- Gráficos de fraude: se utilizan habitualmente en las bases de datos de gráficos. Pueden ayudarle a realizar un seguimiento de las compras con tarjeta de crédito y los lugares de compra para detectar usos poco habituales o detectar que un comprador está intentando utilizar la misma

dirección de correo electrónico y la misma tarjeta de crédito que las utilizadas en un caso de fraude conocido. Le permiten comprobar si hay varias personas asociadas a una dirección de correo electrónico personal o si hay varias personas en diferentes ubicaciones físicas que comparten la misma dirección IP.

Tenga en cuenta el siguiente gráfico. Muestra la relación de tres personas y la información relacionada con su identidad. Cada persona tiene una dirección, una cuenta bancaria y un número de la seguridad social. Sin embargo, vemos que Matt y Justin comparten el mismo número de la seguridad social, lo cual no es normal e indica un posible fraude por parte de uno de ellos. Una consulta a un gráfico de fraude puede revelar conexiones de este tipo que pueden revisarse.



Para obtener más información sobre cómo los gráficos de fraude y dónde se utilizan, consulte Gráficos de fraude en AWS.

- Redes sociales: una de las primeras y más comunes áreas en las que se utilizan las bases de datos de gráficos es en las aplicaciones de redes sociales.

Por ejemplo, supongamos que desea crear una fuente social en un sitio web. Puede utilizar fácilmente una base de datos de gráficos en el backend para ofrecer a los usuarios resultados que reflejen las últimas actualizaciones de sus familiares, sus amigos, las personas cuyas actualizaciones les “gustan” y las personas que viven cerca de ellos.

- Indicaciones: un gráfico puede ayudar a encontrar la mejor ruta desde un punto de partida hasta un destino, teniendo en cuenta el tráfico actual y los patrones de tráfico habituales.
- Logística: los gráficos pueden ayudar a identificar la forma más eficaz de utilizar los recursos de envío y distribución disponibles para cumplir con los requisitos de los clientes.
- Diagnósticos: los gráficos pueden representar árboles de diagnóstico complejos que se pueden consultar para identificar el origen de los problemas y errores observados.
- Investigación científica: con una base de datos de gráficos, puede crear aplicaciones que almacenen y naveguen por datos científicos e incluso información médica confidencial mediante el cifrado en reposo. Por ejemplo, puede almacenar modelos de interacciones de enfermedades y genes. Puede buscar patrones de gráficos dentro de las cadenas de proteínas para encontrar otros genes que podrían estar relacionados con una enfermedad. Puede modelar compuestos químicos como un gráfico y consultar patrones en las estructuras moleculares. Puede correlacionar los datos de los pacientes con los registros médicos en diferentes sistemas. Puede organizar las publicaciones de investigación por temas para encontrar información pertinente con rapidez.
- Reglas normativas. puede almacenar requisitos normativos complejos en forma de gráficos y consultarlos para detectar situaciones en las que podrían aplicarse a sus operaciones empresariales diarias.
- Topología y eventos de la red: una base de datos de gráficos puede ayudarlo a administrar y proteger una red de TI. Al almacenar la topología de la red como un gráfico, también puede almacenar y procesar muchos tipos diferentes de eventos en la red. Puede responder a preguntas como cuántos hosts ejecutan una aplicación determinada. Puede buscar patrones que puedan indicar que un determinado host se ha visto afectado por un programa malintencionado y consultar datos de conexión que puedan ayudar a realizar un seguimiento del programa hasta el host original que lo descargó.

¿Cómo se consulta un gráfico?

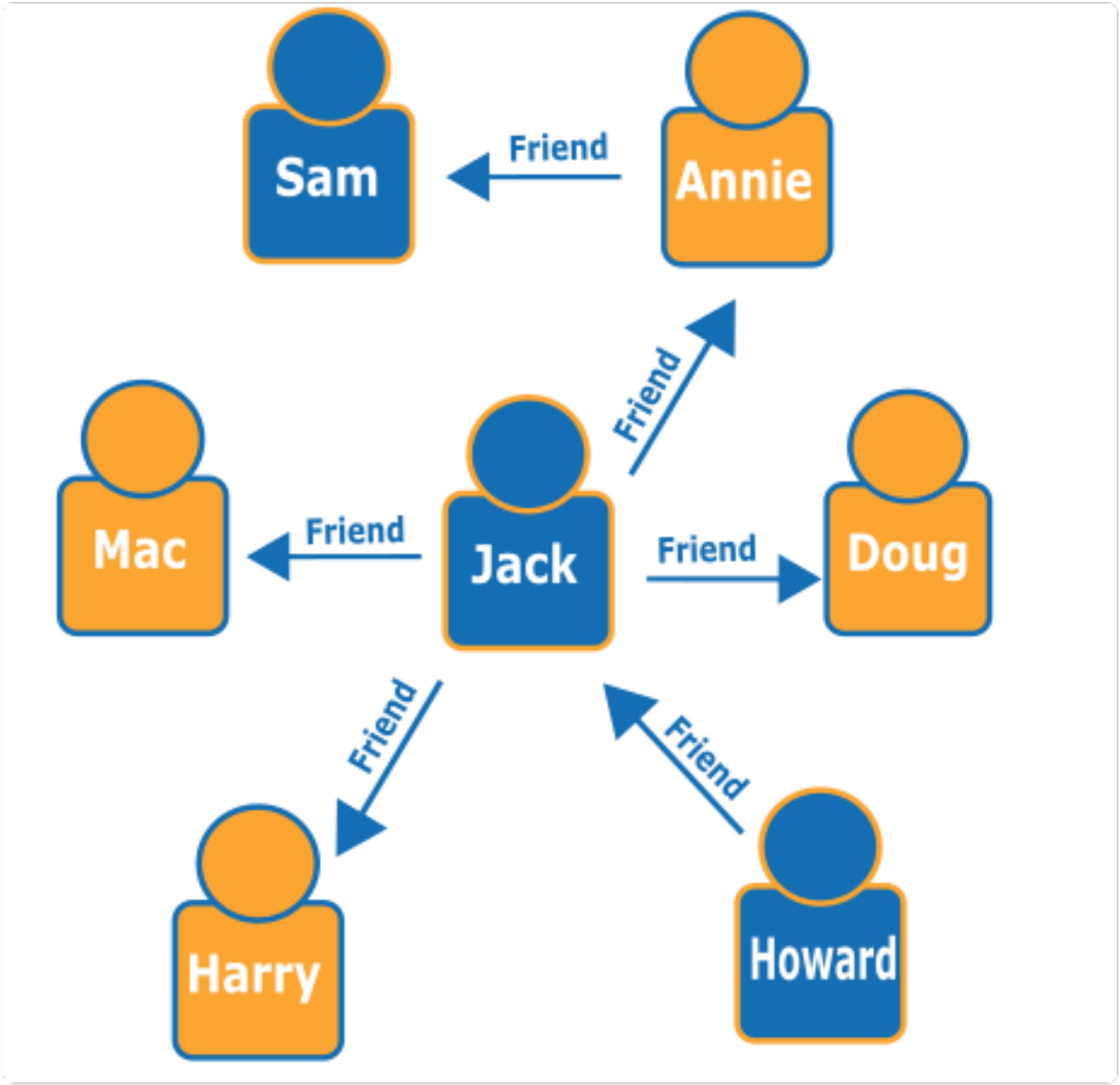
Neptune admite tres lenguajes de consulta especiales diseñados para consultar datos de gráficos de diferentes tipos. Puede usar estos lenguajes para añadir, modificar, eliminar y consultar datos en una base de datos de gráficos de Neptune:

- Gremlin es un lenguaje de recorrido de gráficos para gráficos de propiedades. En Gremlin, una consulta es un recorrido compuesto por pasos discretos, cada uno de los cuales sigue un borde hasta un nodo. Consulte la documentación de Gremlin en Apache TinkerPop3 para obtener más información.

  La implementación de Gremlin en Neptune tiene algunas diferencias con respecto a otras implementaciones, especialmente cuando se utiliza Gremlin-Groovy (consultas de Gremlin enviadas como texto serializado). Para obtener más información, consulte Conformidad con los estándares de Gremlin en Amazon Neptune.
- openCypher: openCypher es un lenguaje de consulta declarativo para gráficos de propiedades que desarrolló originalmente Neo4j, luego de código abierto en 2015, y que contribuyó al proyecto openCypher en virtud de una licencia de código abierto Apache 2. Consulte la Referencia del lenguaje de consultas de Cypher (versión 9) para la especificación del lenguaje, así como la Guía de estilo de Cypher para obtener información adicional.
- SPARQL es un lenguaje de consulta declarativo para datos RDF, que se basa en la coincidencia de patrones gráficos que se ajusta al estándar World Wide Web Consortium (W3C) y se describe en SPARQL 1.1 Overview y en la especificación de SPARQL 1.1 Query Language. Consulte Conformidad con los estándares de SPARQL en Amazon Neptune para obtener información específica sobre la implementación de SPARQL en Neptune.

Ejemplos de consultas coincidentes de Gremlin y SPARQL

En el siguiente gráfico de personas (nodos) y sus relaciones (bordes), puede encontrar quiénes son los “amigos de amigos” de una persona determinada, por ejemplo, los amigos de los amigos de Howard.



Al examinar el gráfico, puede ver que Howard tiene un amigo, Jack, que a su vez tiene cuatro amigos: Annie, Harry, Doug y Mac. Este es un ejemplo sencillo con un gráfico simple, pero la complejidad y el tamaño del conjunto de datos y de los resultados de estos tipos de consultas pueden aumentar considerablemente.

A continuación, se muestra una consulta de recorrido Gremlin que devuelve los nombres de los amigos de los amigos de Howard:

```
g.V().has('name', 'Howard').out('friend').out('friend').values('name')
```

A continuación, se muestra una consulta de SPARQL que devuelve los nombres de los amigos de amigos de Howard:

```
prefix : <#>

  select ?names where {
    ?howard :name "Howard" .
    ?howard :friend/:friend/:name ?names .
  }
```

> **Note**
>
> Cada una de las partes de un triple del marco de descripción de recursos (RDF) tiene un URI asociado. En el ejemplo anterior, el prefijo del URI es corto a propósito.

Realice un curso en línea sobre el uso de Amazon Neptune

Si le gusta aprender con vídeos, AWS ofrece cursos en línea en las Charlas de tecnología en línea de AWS para ayudarle a ponerse manos a la obra. El curso que trata de las bases de datos de gráficos es:

Introducción, análisis detallado y demostración de bases de datos de gráficos con Amazon Neptune.

Profundizar más en la arquitectura de referencia de gráficos

Al pensar en los problemas que podría resolver una base de datos de gráficos y en cómo abordarlos, uno de los mejores puntos de partida es el proyecto GitHub de arquitecturas de referencia de gráficos de Neptune.

Allí encontrará descripciones detalladas de los tipos de carga de trabajo de gráficos y tres secciones que le ayudarán a diseñar una base de datos de gráficos eficaz:

- Modelos de datos y lenguajes de consulta: en esta sección se explican las diferencias entre Gremlin y SPARQL y cómo elegir entre una de estas opciones.

- Modelado de datos gráficos: se trata de una explicación más detallada sobre cómo tomar decisiones de modelado de datos gráficos, que incluye tutoriales detallados sobre el modelado de gráficos de propiedades con Gremlin y el modelado RDF con SPARQL.
- Conversión de otros modelos de datos en un modelo de gráfico: aquí encontrará información sobre cómo traducir un modelo de datos relacional en un modelo de gráfico.

También hay tres secciones que explican los pasos específicos para usar Neptune:

- Conexión a Amazon Neptune desde clientes ajenos a la VPC de Neptune: en esta sección se muestran varias opciones para conectarse a Neptune desde fuera de la VPC en la que se encuentra el clúster de base de datos.
- Acceso a Amazon Neptune desde funciones de AWS Lambda: aquí encontrará información sobre cómo conectarse de forma fiable a Neptune desde funciones de Lambda.
- Escribir en Amazon Neptune desde una transmisión de Amazon Kinesis Data Streams: en esta sección puede obtener información sobre cómo gestionar escenarios de alto rendimiento de escritura con Neptune.

Uso de los cuadernos de gráficos de Neptune para empezar rápidamente

No tiene que usar cuadernos de gráficos de Neptune para trabajar con un gráfico de Neptune, así que si lo desea, puede continuar y crear una nueva base de datos de Neptune mediante una plantilla de AWS CloudFormation.

Al mismo tiempo, tanto si es nuevo en el mundo de los gráficos y si desea aprender y experimentar, como si ya tiene experiencia y desea afinar sus consultas, el entorno de trabajo de Neptune ofrece un entorno de desarrollo interactivo (IDE) que puede aumentar su productividad al crear aplicaciones gráficas.

Neptune proporciona Jupyter y JupyterLabcuadernos en el proyecto de cuaderno gráfico Neptuno de código abierto y en la mesa de trabajo de Neptune. GitHub Estos cuadernos ofrecen ejemplos de tutoriales de aplicaciones y fragmentos de código en un entorno de codificación interactivo en el que puede aprender sobre la tecnología de gráficos y Neptune. Puede utilizarlos para preparar, configurar, rellenar y consultar gráficos mediante distintos lenguajes de consulta, distintos conjuntos de datos e incluso distintas bases de datos en el backend.

Puede alojar estos cuadernos de varias maneras diferentes:

- El entorno de trabajo Neptune le permite ejecutar los cuadernos de Jupyter en un entorno totalmente gestionado, alojado en Amazon SageMaker, y carga automáticamente la última versión del proyecto de bloc de notas de gráficos Neptune. Es fácil configurar el entorno de trabajo en la consola de Neptune al crear una nueva base de datos de Neptune.

  > **Note**
  >
  > Al crear una instancia de bloc de notas de Neptune, se te proporcionan dos opciones de acceso a la red: acceso directo a través de Amazon SageMaker (la opción predeterminada) y acceso a través de una VPC. En cualquiera de las dos opciones, el portátil requiere acceso a Internet para recuperar las dependencias de los paquetes para instalar el entorno de trabajo de Neptune. La falta de acceso a Internet provocará un error en la creación de una instancia de bloc de notas de Neptune.

- También puede instalar Jupyter de forma local. Esto le permite ejecutar los cuadernos desde su portátil, conectado a Neptune o a una instancia local de una de las bases de datos de gráficos de código abierto. En este último caso, puede experimentar con la tecnología de gráficos todo lo que desee sin gastar ni un céntimo. Luego, cuando esté listo, podrá pasar sin problemas al entorno de producción administrado que ofrece Neptune.

Uso del entorno de trabajo de Neptune para alojar los cuadernos de Neptune

Neptune ofrece los tipos de instancia T3 y T4g con los que puede empezar por menos de 0,10 USD por hora. Los recursos de Workbench se facturan a través de Amazon SageMaker, aparte de la facturación de Neptune. Consulte la página de precios de Neptune. Tanto Jupyter como JupyterLab los cuadernos creados en el entorno de trabajo de Neptune utilizan un entorno Amazon Linux 2 y 3. JupyterLab Para obtener más información sobre la compatibilidad con JupyterLab ordenadores portátiles, consulta la SageMakerdocumentación de Amazon.

Puede crear un Jupyter o un JupyterLab cuaderno con el banco de trabajo Neptune de dos maneras AWS Management Console :

- Utilice el menú de Configuración de cuaderno al crear un nuevo clúster de base de datos de Neptune. Para ello, siga los pasos que se indican en Lanzamiento de un clúster de base de datos de Neptune mediante la AWS Management Console.

- Utilice el menú Cuadernos del panel de navegación izquierdo cuando ya se haya creado el clúster de base de datos. Para ello, siga estos pasos.

Para crear un Jupyter o un bloc de notas mediante el menú Cuadernos JupyterLab

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación de la izquierda, seleccione Notebooks (Bloc de notas).
3. Elija Crear cuaderno.
4. En la lista de clústeres, elija el clúster de base de datos de Neptune. Si todavía no tiene ningún clúster de base de datos, seleccione Create cluster (Crear clúster) para crear uno.
5. Seleccione un tipo de instancia de cuaderno.
6. Asigne un nombre al bloc de notas y, opcionalmente, una descripción.
7. A menos que ya haya creado un rol AWS Identity and Access Management (de IAM) para sus cuadernos, elija Crear un rol de IAM e introduzca un nombre de rol de IAM.

   > **Note**
   >
   > Si decide volver a utilizar un rol de IAM, creado para un cuaderno anterior, la política de roles debe incluir los permisos correctos para acceder al clúster de base de datos de Neptune que esté utilizando. Para comprobarlo, verifique que los componentes del ARN del recurso de la acción `neptune-db:*` coincidan con ese clúster. Los permisos configurados de forma incorrecta provocan errores de conexión al intentar ejecutar los comandos mágicos del cuaderno.

8. Elija Crear cuaderno. El proceso de creación puede tardar entre 5 y 10 minutos antes de que todo esté listo.
9. Una vez creado el bloc de notas, selecciónelo y, a continuación, elija Abrir Jupyter o Abrir. JupyterLab

La consola puede crear un rol de AWS Identity and Access Management (IAM) para sus cuadernos, o puede crear uno usted mismo. La política para este rol debe incluir lo siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::aws-neptune-notebook-(AWS region)",
        "arn:aws:s3:::aws-neptune-notebook-(AWS region)/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": [
        "arn:aws:neptune-db:(AWS region):(AWS account ID):(Neptune resource ID)/*"
      ]
    }
  ]
}
```

Tenga en cuenta que la segunda instrucción de la política anterior enumera uno o varios identificadores de recursos del clúster de Neptune.

Además, el rol debe establecer la siguiente relación de confianza:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Una vez más, preparar todo para que funcione puede llevar de 5 a 10 minutos.

Puede configurar el nuevo cuaderno para que funcione con Neptune ML, tal y como se explica en Configuración manual de un cuaderno de Neptune para Neptune ML.

Uso de Python para conectar un SageMaker cuaderno genérico a Neptuno

Conectar una computadora portátil a Neptuno es fácil si tiene instalada la magia de Neptuno, pero también es posible conectar una computadora portátil a SageMaker Neptuno usando Python, incluso si no está usando una computadora portátil Neptuno.

Pasos a seguir para conectarse a Neptune en una celda portátil SageMaker

1. Instale el cliente Python de Gremlin:

   ```
   !pip install gremlinpython
   ```

   Las libretas Neptune instalan el cliente Python Gremlin por ti, por lo que este paso solo es necesario si utilizas una libreta normal. SageMaker
2. Escriba un código como el siguiente para conectarse y emitir una consulta de Gremlin:

   ```
   from gremlin_python import statics
   from gremlin_python.structure.graph import Graph
   from gremlin_python.process.graph_traversal import __
   from gremlin_python.process.strategies import *
   from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
   from gremlin_python.driver.aiohttp.transport import AiohttpTransport
   from gremlin_python.process.traversal import *
   import os

   port = 8182
   server = '(your server endpoint)'

   endpoint = f'wss://{server}:{port}/gremlin'

   graph=Graph()

   connection = DriverRemoteConnection(endpoint,'g',

    transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))

   g = graph.traversal().withRemote(connection)

   results = (g.V().hasLabel('airport')
                   .sample(10)
                   .order()
                   .by('code')
   ```

```
                .local(__.values('code','city').fold())
                .toList())

# Print the results in a tabular form with a row index
for i,c in enumerate(results,1):
    print("%3d %4s %s" % (i,c[0],c[1]))

connection.close()
```

> **Note**
>
> Si está usando una versión del cliente Python de Gremlin anterior a la 3.5.0, esta línea:
>
> ```
> connection = DriverRemoteConnection(endpoint,'g',
>
>  transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))
> ```
>
> Simplemente sería:
>
> ```
> connection = DriverRemoteConnection(endpoint,'g')
> ```

Habilitación CloudWatch de registros en los cuadernos Neptune

CloudWatch Los registros ahora están habilitados de forma predeterminada para los cuadernos Neptune. Si tiene un portátil antiguo que no produce CloudWatch registros, siga estos pasos para habilitarlos manualmente:

1. Inicie sesión en la SageMaker consola AWS Management Console y ábrala.
2. En el panel de navegación de la izquierda, seleccione Cuaderno y, a continuación, Instancias de cuaderno. Busque el nombre del cuaderno de Neptune para el que desea habilitar los registros.
3. Seleccione el nombre de esa instancia de cuaderno para ir a la página de detalles.
4. Si la instancia del cuaderno se está ejecutando, seleccione el botón Detener, situado en la parte superior derecha de la página de detalles del cuaderno.
5. En Permisos y cifrado hay un campo para el ARN del rol de IAM. Seleccione el enlace de este campo para ir al rol de IAM con el que se ejecuta esta instancia de cuaderno.
6. Cree la política siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Guarde esta nueva política y adjúntela al rol de IAM que se encuentra en el paso 4.
8. Haz clic en Iniciar en la parte superior derecha de la página de detalles de la instancia del SageMaker bloc de notas.
9. Cuando los registros comiencen a fluir, debería ver el enlace Ver registros debajo del campo denominado Configuración del ciclo de vida, cerca de la parte inferior izquierda de la sección Configuración de instancias de cuaderno de la página de detalles.

Si un bloc de notas no se inicia, aparecerá un mensaje en la página de detalles del bloc de notas de la SageMaker consola en el que se indica que la instancia del bloc de notas ha tardado más de 5 minutos en iniciarse. CloudWatch Los registros relacionados con este problema se encuentran con este nombre:

```
(your-notebook-name)/LifecycleConfigOnStart
```

Configuración de cuadernos de gráficos en una máquina local

El proyecto de cuaderno de gráficos incluye instrucciones para configurar los cuadernos de Neptune en una máquina local:

- Requisitos previos
- Jupyter e instalación JupyterLab
- Conexión a una base de datos de gráficos

Puede conectar sus cuadernos locales a un clúster de base de datos de Neptune o a una instancia local o remota de una base de datos de gráficos de código abierto.

Uso de cuadernos de Neptune con clústeres de Neptune

Si te estás conectando a un clúster de Neptune en el back-end, es posible que desees ejecutar los cuadernos en Amazon. SageMaker Conectarse a Neptune desde SageMaker puede resultar más cómodo que desde una instalación local de los portátiles, y le permitirá trabajar más fácilmente con Neptune ML.

Para obtener instrucciones sobre cómo configurar las libretas en SageMaker, consulta Cómo lanzar graph-notebook con Amazon. SageMaker

Para obtener instrucciones sobre cómo configurar Neptune, consulte Configuración de Neptune.

También puede conectar una instalación local de los cuadernos de Neptune a un clúster de base de datos de Neptune. Esto puede resultar algo más complicado porque los clústeres de bases de datos de Amazon Neptune solo se pueden crear en una nube Amazon Virtual Private Cloud (VPC), que, por diseño, está aislada del mundo exterior. Hay varias formas de conectarse a una VPC desde fuera de ella. Una de ellas es usar un equilibrador de carga. Otra opción es utilizar la interconexión con VPC (consulte la Guía de interconexión de Amazon Virtual Private Cloud).

Sin embargo, la forma más práctica para la mayoría de las personas es conectarse para configurar un servidor proxy Amazon EC2 dentro de la VPC y, a continuación, utilizar la tunelización de SSH (también denominada reenvío de puertos) para conectarse a él. Encontrará instrucciones sobre cómo configurarlo en Conectar un cuaderno de gráficos localmente a Amazon Neptune, en la `additional-databases/neptune` carpeta del proyecto de cuaderno de gráficos GitHub .

Uso de cuadernos de Neptune con bases de datos de gráficos de código abierto

Para empezar a utilizar la tecnología de gráficos sin costo alguno, también puede utilizar los cuadernos de Neptune con varias bases de datos de código abierto en el backend. Algunos ejemplos son el servidor TinkerPop Gremlin y la base de datos Blazegraph.

Para usar el servidor de Gremlin como base de datos backend, siga las instrucciones que se encuentran en:

- La carpeta Conectar un cuaderno gráfico a un servidor Gremlin. GitHub
- La carpeta de configuración Gremlin del cuaderno gráfico. GitHub

Para usar una instancia local de Blazegraph como base de datos backend, siga estas instrucciones:

- Instrucciones de inicio rápido de Blazegraph
- La carpeta de configuración Blazegraph del cuaderno gráfico. GitHub

Migración de tus cuadernos Neptune de Jupyter a 3 JupyterLab

Los cuadernos de Neptune creados antes del 21 de diciembre de 2022 utilizan el entorno de Amazon Linux 1. Puede migrar cuadernos Jupyter antiguos creados antes de esa fecha al nuevo entorno Amazon Linux 2 con JupyterLab 3 siguiendo los pasos descritos en esta entrada de AWS blog: Migrar su trabajo a una instancia de Amazon SageMaker Notebook con Amazon Linux 2.

Además, también hay algunos pasos más que se aplican específicamente a la migración de los cuadernos de Neptune al nuevo entorno:

Requisitos previos específicos de Neptune

En el rol de IAM del cuaderno de Neptune de origen, añada todos los permisos siguientes:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:ListBucket",
    "s3:CreateBucket",
    "s3:PutObject"
  ],
```

```
  "Resource": [
    "arn:aws:s3:::(your ebs backup bucket name)",
    "arn:aws:s3:::(your ebs backup bucket name)/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "sagemaker:ListTags"
  ],
  "Resource": [
    "*"
  ]
}
```

Asegúrese de especificar el ARN correcto para el bucket S3 que utilizará para realizar la copia de seguridad.

Configuración del ciclo de vida específica de Neptune

Al crear el segundo script de configuración del ciclo de vida para restaurar la copia de seguridad (desde `on-create.sh`), tal y como se describe en la publicación del blog, el nombre del ciclo de vida debe seguir el mismo formato de `aws-neptune-*`, por ejemplo `aws-neptune-sync-from-s3`. Esto garantiza que se pueda seleccionar la LCC durante la creación del cuaderno en la consola de Neptune.

Sincronización específica de Neptune a partir de una instantánea a una nueva instancia

En los pasos descritos en la publicación del blog para la sincronización de una instantánea a una nueva instancia, puede encontrar los cambios específicos de Neptune:

- En el paso 4, seleccione notebook-al2-v2.
- En el paso 5, vuelva a utilizar el rol de IAM del cuaderno de Neptune de origen.
- Entre los pasos 7 y 8:
  - En Configuración de instancias de cuaderno, establezca un nombre que utilice el formato de `aws-neptune-*`.
  - Abra el acordeón de configuración de Red y seleccione la misma VPC, subred y grupo de seguridad que en el cuaderno de origen.

Pasos específicos de Neptune tras la creación del nuevo cuaderno

1. Seleccione el botón Abrir Jupyter para el cuaderno. Una vez que el archivo `SYNC_COMPLETE` aparezca en el directorio principal, continúe con el siguiente paso.
2. Ve a la página de instancias de bloc de notas de la consola. SageMaker
3. Detenga el bloc de notas.
4. Seleccione Editar.
5. En la configuración de la instancia del cuaderno, edite el campo Configuración del ciclo de vida. Para ello, seleccione el ciclo de vida original del cuaderno de Neptune de origen. Tenga en cuenta que este no es el ciclo de vida de las copias de seguridad de EBS.
6. Seleccione Actualizar la configuración del cuaderno.
7. Vuelva a iniciar el cuaderno.

Con las modificaciones que se describen aquí en los pasos descritos en la entrada del blog, sus cuadernos de gráficos ahora deberían migrarse a una nueva instancia de bloc de notas de Neptune que utilice el entorno Amazon Linux 2 JupyterLab y 3. Aparecerán para su acceso y administración en la página de Neptune del AWS Management Console, y ahora puedes continuar tu trabajo desde donde lo dejaste seleccionando Abrir Jupyter o Abrir. JupyterLab

Uso de los comandos mágicos del entorno de trabajo de Neptune en sus cuadernos

El entorno de trabajo de Neptune incluye una serie de los denominados comandos mágicos en los cuadernos que ahorran una gran cantidad de tiempo y esfuerzo. Se dividen en dos categorías: comandos mágicos de línea y comandos mágicos de celda.

Los comandos mágicos de línea son comandos precedidos por un único signo de porcentaje (`%`). Solo reciben entradas de línea, no entradas del resto del cuerpo de la celda. El entorno de trabajo de Neptune ofrece los siguientes comandos mágicos de línea:

- %seed
- %load
- %load_ids
- %load_status
- %cancel_load

- %status
- %gremlin_status
- %opencypher_status u %oc_status
- `%stream_viewer`
- %sparql_status
- %graph_notebook_config
- %graph_notebook_host
- %graph_notebook_version
- %graph_notebook_vis_options
- %statistics
- %summary

Los comandos mágicos de celda están precedidos por dos signos de porcentaje (%%) en lugar de uno, y utilizan el contenido de la celda como entrada, aunque también pueden tomar el contenido de la línea como entrada. El entorno de trabajo de Neptune ofrece los siguientes comandos mágicos de celda:

- %%sparql
- %%gremlin
- %%opencypher u %%oc
- %%graph_notebook_config
- %%graph_notebook_vis_options

También hay dos comandos mágicos, uno de línea y otro de celda, con los que trabajar con Machine learning de Neptune:

- %neptune_ml
- %%neptune_ml

> **Note**
>
> Cuando se trabaja con los comandos mágicos de Neptune, por lo general, se puede obtener texto de ayuda mediante un parámetro `--help` o `-h`. Con un comando mágico de celda, el

cuerpo no puede estar vacío, así que cuando solicite ayuda, coloque texto de relleno, aunque sea un solo carácter, en el cuerpo. Por ejemplo:

```
%%gremlin --help
  x
```

Inyección de variables en comandos mágicos de celda o línea

Se puede hacer referencia a las variables definidas en un cuaderno dentro de cualquier comando mágico de celda o línea del cuaderno mediante el formato: `${VAR_NAME}`.

Por ejemplo, supongamos que define estas variables:

```
c = 'code'
my_edge_labels = '{"route":"dist"}'
```

Luego, esta consulta de Gremlin en un comando mágico de celda:

```
%%gremlin -de $my_edge_labels
g.V().has('${c}','SAF').out('route').values('${c}')
```

Equivale a:

```
%%gremlin -de {"route":"dist"}
g.V().has('code','SAF').out('route').values('code')
```

Argumentos de consulta que funcionan con todos los lenguajes de consulta

Los siguientes argumentos de consulta funcionan con los comandos mágicos `%%gremlin`, `%%opencypher` y `%%sparql` en el entorno de trabajo de Neptune:

Argumentos comunes de consulta

- **`--store-to`** (o **`-s`**): especifica el nombre de una variable en la que se van a almacenar los resultados de la consulta.
- **`--silent`**: si está presente, no se muestra ningún resultado una vez finalizada la consulta.

- **`--group-by`** (o **`-g`**): especifica la propiedad utilizada para agrupar los nodos (como, por ejemplo, `code` o`T.region`). Los vértices se colorean en función del grupo asignado.
- **`--ignore-groups`**: si están presentes, se ignoran todas las opciones de agrupación.
- **`--display-property`** (o **`-d`**): especifica la propiedad cuyo valor debe mostrarse para cada vértice.

  El valor predeterminado de cada lenguaje de consulta es el siguiente:

  - Para Gremlin: `T.label`.
  - Para openCypher: `~labels`.
  - Para SPARQL: `type`.
- **`--edge-display-property`** (o **`-t`**): especifica la propiedad cuyo valor debe mostrarse para cada borde.

  El valor predeterminado de cada lenguaje de consulta es el siguiente:

  - Para Gremlin: `T.label`.
  - Para openCypher: `~labels`.
  - Para SPARQL: `type`.
- **`--tooltip-property`** (o **`-de`**): especifica la propiedad cuyo valor debe mostrarse como ayuda contextual para cada nodo.

  El valor predeterminado de cada lenguaje de consulta es el siguiente:

  - Para Gremlin: `T.label`.
  - Para openCypher: `~labels`.
  - Para SPARQL: `type`.
- **`--edge-tooltip-property`** (o **`-te`**): especifica la propiedad cuyo valor debe mostrarse como ayuda contextual para cada borde.

  El valor predeterminado de cada lenguaje de consulta es el siguiente:

  - Para Gremlin: `T.label`.
  - Para openCypher: `~labels`.
  - Para SPARQL: `type`.
- **`--label-max-length`** (o **`-l`**): especifica la longitud máxima de caracteres de cualquier etiqueta de vértice. El valor predeterminado es 10.

- **`--edge-label-max-length`** (o **`-le`**): especifica la longitud máxima de caracteres de cualquier etiqueta de borde. El valor predeterminado es 10.

  Solo en el caso de openCypher, es `--rel-label-max-length` o, en su lugar, `-rel`.
- **`--simulation-duration`** (o **`-sd`**): especifica la duración máxima de la simulación física de visualización. El valor predeterminado es 1500 ms.
- **`--stop-physics`** (o **`-sp`**): deshabilita la simulación física de visualización una vez que la simulación inicial se haya estabilizado.

Los valores de propiedad de estos argumentos pueden constar de una única clave de propiedad o de una cadena JSON que puede especificar una propiedad diferente para cada tipo de etiqueta. Una cadena JSON solo se puede especificar mediante la inyección de variables.

El comando mágico de línea **`%seed`**

El comando mágico de línea `%seed` es una forma práctica de añadir datos al punto de conexión de Neptune que puede utilizar para explorar y experimentar con consultas de Gremlin, openCypher o SPARQL. Proporciona un formulario en el que puede seleccionar el modelo de datos que desea explorar (gráfico de propiedades o RDF) y, a continuación, elegir entre varios conjuntos de datos de muestra diferentes que proporciona Neptune.

El comando mágico de línea **`%load`**

El comando mágico de línea `%load` genera un formulario que puede utilizar para enviar una solicitud de carga masiva a Neptune (consulte Comando del programa de carga de Neptune). El origen debe ser una ruta de Amazon S3 en la misma región que el clúster de Neptune.

El comando mágico de línea **`%load_ids`**

El comando mágico de línea `%load_ids` recupera los identificadores de carga que se enviaron al punto de conexión del host del cuaderno (consulte Parámetros de las solicitudes de obtención de estado del programa de carga de Neptune). La solicitud se realiza de la siguiente forma:

```
GET https://your-neptune-endpoint:port/loader
```

El comando mágico de línea `%load_status`

El comando mágico de línea `%load_status` recupera el estado de carga de un determinado trabajo de carga que se ha enviado al punto de conexión del host del cuaderno, especificado en la entrada de línea (consulte Parámetros de las solicitudes de obtención de estado del programa de carga de Neptune). La solicitud se realiza de la siguiente forma:

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

El comando mágico de línea tiene un aspecto similar al siguiente:

```
%load_status load id
```

El comando mágico de línea `%cancel_load`

El comando mágico de línea `%cancel_load` cancela un determinado trabajo de carga (consulte Cancelación de trabajo del programa de carga de Neptune). La solicitud se realiza de la siguiente forma:

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

El comando mágico de línea tiene un aspecto similar al siguiente:

```
%cancel_load load id
```

El comando mágico de línea `%status`

Recupera la información de estado del punto de conexión del host del cuaderno (%graph_notebook_config muestra el punto de conexión del host).

El comando mágico de línea `%gremlin_status`

Recupera información de estado de consulta de Gremlin.

El comando mágico de línea `%opencypher_status` (también `%oc_status`)

Recupera el estado de una consulta de openCypher. Este comando mágico de línea adopta los siguientes argumentos opcionales:

- **`--queryId`** o **`-q`**: especifica el ID de una consulta en ejecución específica cuyo estado debe mostrarse.
- **`--cancel_query`** o **`-c`**: cancela una consulta en ejecución. No toma ningún valor.
- **`--silent`** o **`-s`**: si `--silent` se establece en `true` al cancelar una consulta, la consulta en ejecución se cancela con un código de respuesta HTTP de `200`. De lo contrario, el código de respuesta HTTP sería `500`.
- **`--store-to`**: especifica el nombre de una variable en la que se van a almacenar los resultados de la consulta.

El comando mágico de línea **`%sparql_status`**

Recupera la información sobre el estado de la consulta de SPARQL.

El comando mágico de línea **`%stream_viewer`**

El comando mágico de línea `%stream_viewer` muestra una interfaz que permite explorar de forma interactiva las entradas registradas en las transmisiones de Neptune, si estas están habilitadas en el clúster de Neptune. Acepta los siguientes argumentos opcionales:

- **`language`**: el lenguaje de consulta de los datos de la transmisión: `gremlin` o `sparql`. Si no indica este argumento, el valor predeterminado es `gremlin`.
- **`--limit`**: especifica el número máximo de entradas de transmisión que se van a mostrar por página. Si no indica este argumento, el valor predeterminado es `10`.

> **Note**
>
> El comando mágico de línea `%stream_viewer` solo es totalmente compatible con las versiones de motor 1.0.5.1 y anteriores.

El comando mágico de línea **`%graph_notebook_config`**

Este comando mágico de línea muestra un objeto JSON que incluye la configuración que utiliza el cuaderno para comunicarse con Neptune. La configuración incluye:

- `host`: el punto de conexión al que conectarse y emitir comandos.

- `port`: el puerto que se utiliza al emitir comandos a Neptune. El valor predeterminado es `8182`.
- `auth_mode`: el modo de autenticación que se utilizará al emitir comandos a Neptune. Debe ser `IAM` si se conecta a un clúster que tenga habilitada la autenticación de IAM, o de lo contrario, `DEFAULT`.
- `load_from_s3_arn`: especifica un ARN de Amazon S3 para que lo utilice el comando mágico `%load`. Si este valor está vacío, se debe especificar el ARN en el comando `%load`.
- `ssl`: un valor booleano que indica si se debe conectar o no a Neptune mediante TLS. El valor predeterminado es `true`.
- `aws_region`: la región en la que está implementado este cuaderno. Esta información se utiliza para la autenticación de IAM y para las solicitudes de `%load`.

Para cambiar la configuración, copie el resultado de `%graph_notebook_config` en una celda nueva y modifíquelo ahí. Luego, al ejecutar el comando mágico de celda `%%graph_notebook_config` en la nueva celda, la configuración cambiará en consecuencia.

El comando mágico de línea **`%graph_notebook_host`**

Establece la entrada de línea como host del cuaderno.

El comando mágico de línea **`%graph_notebook_version`**

El comando mágico de línea `%graph_notebook_version` devuelve el número de versión del cuaderno del entorno de trabajo de Neptune. Por ejemplo, la visualización de gráficos se introdujo en la versión `1.27`.

El comando mágico de línea **`%graph_notebook_vis_options`**

El comando mágico de línea `%graph_notebook_vis_options` muestra la configuración de visualización actual que está utilizando el cuaderno. Estas opciones se explican en la documentación de vis.js.

Para modificar esta configuración, copie el resultado en una celda nueva, realice los cambios que desee y, a continuación, ejecute el comando mágico de celda `%%graph_notebook_vis_options` en la celda.

Para restaurar la configuración de visualización a sus valores predeterminados, puede ejecutar el comando mágico de línea `%graph_notebook_vis_options` con un parámetro `reset`. Esto restablece todos los ajustes de visualización:

```
%graph_notebook_vis_options reset
```

El comando mágico de línea **`%statistics`**

El comando mágico de línea `%statistics` se utiliza para recuperar o administrar las estadísticas del motor DFE (consulte Administración de las estadísticas para que las utilice el DFE de Neptune). Este comando mágico también se puede utilizar para recuperar un resumen de gráficos.

Acepta los siguientes parámetros:

- **`--language`**: el lenguaje de consulta del punto de conexión de las estadísticas: `propertygraph` (o `pg`) o `rdf`.

  Si no se indica, el valor predeterminado es `propertygraph`.
- **`--mode`** (o **`-m`**): especifica el tipo de solicitud o acción que se va a enviar. Una de las siguientes opciones: `status`, `disableAutoCompute`, `enableAutoCompute`, `refresh`, `delete`, `detailed` o `basic`.

  Si no se indica, el valor predeterminado `status`, a no ser que se especifique `--summary`, en cuyo caso el valor predeterminado es `basic`.
- **`--summary`**: recupera el resumen de gráficos del punto de conexión del resumen de estadísticas del lenguaje seleccionado.
- **`--silent`**: si está presente, no se muestra ningún resultado una vez finalizada la consulta.
- **`--store-to`**: se utiliza para especificar una variable en la que almacenar los resultados de la consulta.

El comando mágico de línea **`%summary`**

El comando mágico de línea `%summary` se utiliza para recuperar la información del resumen de gráficos. Está disponible a partir de la versión del motor de Neptune `1.2.1.0`.

Acepta los siguientes parámetros:

- **`--language`**: el lenguaje de consulta del punto de conexión de las estadísticas: `propertygraph` (o `pg`) o `rdf`.

  Si no se indica, el valor predeterminado es `propertygraph`.

- **`--detailed`**: habilita o deshabilita la visualización de los campos de estructuras en el resultado.

  Si no se indica, el valor predeterminado es el modo de visualización del resumen `basic`.
- **`--silent`**: si está presente, no se muestra ningún resultado una vez finalizada la consulta.
- **`--store-to`**: se utiliza para especificar una variable en la que almacenar los resultados de la consulta.

El comando mágico de celda **`%%graph_notebook_config`**

El comando mágico de celda `%%graph_notebook_config` utiliza un objeto JSON que incluye información de configuración para modificar los ajustes que utiliza el cuaderno para comunicarse con Neptune, si es posible. La configuración adopta la misma forma devuelta por el comando mágico de línea %graph_notebook_config.

Por ejemplo:

```
%%graph_notebook_config
{
  "host": "my-new-cluster-endpoint.amazon.com",
  "port": 8182,
  "auth_mode": "DEFAULT",
  "load_from_s3_arn": "",
  "ssl": true,
  "aws_region": "us-east-1"
}
```

El comando mágico de celda **`%%sparql`**

El comando mágico de celda `%%sparql` emite una consulta de SPARQL al punto de conexión de Neptune. Acepta la siguiente entrada de línea opcional:

- **`-h`** o **`--help`**: devuelve texto de ayuda sobre estos parámetros.
- **`--path`**: prefija una ruta al punto de conexión de SPARQL. Por ejemplo, si especifica `--path "abc/def"`, el punto de conexión llamado sería *`host`*`:`*`port`*`/abc/def`.
- **`--expand-all`**: se trata de una sugerencia de visualización de consultas que indica al visualizador que incluya todos los resultados `?s ?p ?o` en el diagrama del gráfico, independientemente del tipo de enlace.

De forma predeterminada, una visualización de SPARQL solo incluye patrones triples en los que o? es `uri` o `bnode` (nodo en blanco). Todos los demás tipos de enlaces `?o`, como las cadenas literales o los números enteros, se tratan como propiedades del nodo `?s` y se pueden ver mediante el panel detalles de la pestaña Gráfico.

Utilice la sugerencia de consulta `--expand-all` cuando desee incluir valores literales como vértices en la visualización.

No combine esta sugerencia de visualización con los parámetros de explicación, ya que las consultas de explicación no se visualizan.

- **`--explain-type`**: se utiliza para especificar el modo de explicación que se va a utilizar (una de las siguientes opciones: `dynamic`, `static` o `details`).

- **`--explain-format`**: se utiliza para especificar el formato de respuesta de una consulta Explain (una de las siguientes opciones: `text/csv` o `text/html`).
- --store-to: se utiliza para especificar una variable en la que almacenar los resultados de la consulta.

Ejemplo de una consulta de `explain`:

```
%%sparql explain

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

Ejemplo de una consulta de visualización con un parámetro de sugerencia de visualización `--expand-all` (consulte Visualización de SPARQL):

```
%%sparql --expand-all

SELECT * WHERE {?s ?p ?o} LIMIT 10
```

El comando mágico de celda **`%%gremlin`**

La magia `%%gremlin` celular envía una consulta de Gremlin al punto final de Neptuno utilizando. WebSocket Acepta una entrada de línea opcional para alternar entre el modo /> `explain` de Gremlin o API `profile` de Gremlin, y una entrada de sugerencia de visualización opcional

independiente para modificar el comportamiento de la salida de la visualización (consulte Visualización de Gremlin).

Ejemplo de una consulta de `explain`:

```
%%gremlin explain

g.V().limit(10)
```

Ejemplo de una consulta de `profile`:

```
%%gremlin profile

g.V().limit(10)
```

Ejemplo de una consulta de visualización con una sugerencia de consulta de visualización:

```
%%gremlin -p v,outv

g.V().out().limit(10)
```

Parámetros opcionales para consultas de **`%%gremlin profile`**

- **`--chop`**: especifica la longitud máxima de la cadena de resultados del perfil. Si no indica este argumento, el valor predeterminado es 250.
- **`--serializer`**: especifica el serializador que se utilizará para los resultados. Los valores permitidos son cualquiera de los valores de enumeración válidos del tipo MIME o de los « TinkerPop serializadores» del controlador. Si no indica este argumento, el valor predeterminado es `application.json`.
- **`--no-results`**: muestra solo el recuento de resultados. Si no se utiliza, todos los resultados de la consulta se muestran de forma predeterminada en el informe de perfil.
- **`--indexOps`**: muestra un informe detallado de todas las operaciones de índice.

El comando mágico de celda **`%%opencypher`** (también **`%%oc`**)

El comando mágico de celda `%%opencypher` (que también tiene la forma abreviada `%%oc`) emite una consulta openCypher al punto de conexión de Neptune. Acepta los siguientes argumentos de entrada de línea opcionales:

- modo: el modo de consulta (`query` o `bolt`). Si no indica este argumento, el valor predeterminado es `query`.
- **`--group-by`** o **`-g`**: especifica la propiedad utilizada para agrupar los nodos. Por ejemplo, `code, ~id`. Si no indica este argumento, el valor predeterminado es `~labels`.
- **`--ignore-groups`**: si están presentes, se ignoran todas las opciones de agrupación.
- **`--display-propery`** o **`-d`**: especifica la propiedad cuyo valor debe mostrarse para cada vértice. Si no indica este argumento, el valor predeterminado es `~labels`.
- **`--edge-display-propery`** o **`-de`**: especifica la propiedad cuyo valor debe mostrarse para cada borde. Si no indica este argumento, el valor predeterminado es `~labels`.
- **`--label-max-length`** o **`-l`**: especifica el número máximo de caracteres que debe mostrar una etiqueta de vértice. Si no indica este argumento, el valor predeterminado es `10`.
- **`--store-to`** o **`-s`**: especifica el nombre de una variable en la que se van a almacenar los resultados de la consulta.
- **`--plan-cache`** o **`-pc`**: especifica el modo de caché del plan que se va a utilizar. El valor predeterminado es. `auto` (*plan-cache solo está disponible para Neptune Analytics)
- **`--query-timeout`** o **`-qt`**: especifica el tiempo de espera máximo de la consulta en milisegundos. El valor predeterminado es `1800000`.
- **`--query-parameters`** o **`qp`**: definiciones de parámetros que se van a aplicar a la consulta. Esta opción puede aceptar un único nombre de variable o una representación en cadena de la asignación.

  Ejemplo de uso de **`--query-parameters`**

  1. Defina un mapa de los parámetros de openCypher en una celda del cuaderno.

     ```
     params = '''{
      "name":"john",
      "age": 20,
     }'''
     ```

  2. Pase los parámetros de `--query-parameters` a otra celda con `%%oc`.

     ```
     %%oc --query-parameters params

     MATCH (n {name: $name, age: $age})
     RETURN n
     ```

- --explain-type: se utiliza para especificar el modo de explicación que se va a utilizar (uno de los siguientes: dinámico, estático o detallado).

El comando mágico de celda **%%graph_notebook_vis_options**

El comando mágico de celda `%%graph_notebook_vis_options` le permite configurar las opciones de visualización del cuaderno. Puede copiar los ajustes devueltos por el comando mágico de línea `%graph-notebook-vis-options` en una nueva celda, realizar cambios en ellos y usar el comando mágico de celda `%%graph_notebook_vis_options` para establecer los nuevos valores.

Estas opciones se explican en la documentación de vis.js.

Para restaurar la configuración de visualización a sus valores predeterminados, puede ejecutar el comando mágico de línea `%graph_notebook_vis_options` con un parámetro `reset`. Esto restablece todos los ajustes de visualización:

```
%graph_notebook_vis_options reset
```

El comando mágico de línea **%neptune_ml**

Puede utilizar el comando mágico de línea `%neptune_ml` para iniciar y administrar diversas operaciones de Neptune ML.

> **Note**
>
> También puede iniciar y administrar algunas operaciones de Neptune ML con el comando mágico de celda %%neptune_ml.

- **`%neptune_ml export start`**: inicia un nuevo trabajo de exportación.

  Parámetros

  - **`--export-url`** *`exporter-endpoint`*: (opcional) el punto de conexión de Amazon API Gateway donde se puede llamar al exportador.
  - **`--export-iam`**: (opcional) indica que las solicitudes a la URL de exportación deben firmarse con SigV4.
  - **`--export-no-ssl`**: (opcional) indica que SSL no debe utilizarse al conectarse al exportador.

  - **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado la exportación.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Establece el tiempo, en segundos, entre las comprobaciones del estado de la exportación (predeterminado: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de exportación antes de devolver el estado más reciente (valor predeterminado: 3600).
  - **`--store-to`*`location-to-store-result`*** — (opcional) La variable en la que se almacena el resultado de la exportación. Si se especifica `--wait`, el estado final se almacenará allí.
- **`%neptune_ml export status`**: recupera el estado de un trabajo de exportación.

  Parámetros

  - **`--job-id`** *`ID del trabajo de exportación`*: el ID del trabajo de exportación del que se va a recuperar el estado.
  - **`--export-url`** *`exporter-endpoint`*: (opcional) el punto de conexión de Amazon API Gateway donde se puede llamar al exportador.
  - **`--export-iam`**: (opcional) indica que las solicitudes a la URL de exportación deben firmarse con SigV4.
  - **`--export-no-ssl`**: (opcional) indica que SSL no debe utilizarse al conectarse al exportador.
  - **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado la exportación.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Establece el tiempo, en segundos, entre las comprobaciones del estado de la exportación (predeterminado: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de exportación antes de devolver el estado más reciente (valor predeterminado: 3600).
  - **`--store-to`*`location-to-store-result`*** — (opcional) La variable en la que se almacena el resultado de la exportación. Si se especifica `--wait`, el estado final se almacenará allí.
- **`%neptune_ml dataprocessing start`**: inicia el paso de procesamiento de datos de Neptune ML.

  Parámetros

  - **`--job-id`** *`Identificador para este trabajo`*: (opcional) identificador para asignar a este trabajo.

- **`--s3-input-uri`** *`URI de S3`*: (opcional) el URI de S3 en el que se encuentra la entrada de este trabajo de procesamiento de datos.
- **`--config-file-name`** *`nombre de archivo`*: (opcional) nombre del archivo de configuración de este trabajo de procesamiento de datos.
- **`--store-to`*`location-to-store-result`*— (opcional) La variable en la que se almacenará el resultado del procesamiento de datos.
- **`--instance-type`** *`(tipo de instancia)`*: (opcional) el tamaño de la instancia que se utilizará para este trabajo de procesamiento de datos.
- **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado el procesamiento de datos.
- **`--wait-interval`*`interval-to-wait`*— (opcional) Establece el tiempo, en segundos, entre las comprobaciones del estado del procesamiento de datos (predeterminado: 60).
- **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de procesamiento de datos antes de devolver el estado más reciente (valor predeterminado: 3600).

- **`%neptune_ml dataprocessing status`**: recupera el estado de un trabajo de procesamiento de datos.

  Parámetros

  - **`--job-id`** *`ID del trabajo`*: el ID del trabajo del que se va a recuperar el estado.
  - **`--store-to`** *`tipo de instancia`*: (opcional) la variable en la que se almacena el resultado del entrenamiento de modelos.
  - **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado el entrenamiento de modelos.
  - **`--wait-interval`*`interval-to-wait`*— (opcional) Establece el tiempo, en segundos, entre las comprobaciones de estado del entrenamiento del modelo (predeterminado: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de procesamiento de datos antes de devolver el estado más reciente (valor predeterminado: 3600).
- **`%neptune_ml training start`**: inicia el proceso de entrenamiento de modelos de Neptune ML.

Parámetros

- **`--job-id`** *`Identificador para este trabajo`*: (opcional) identificador para asignar a este trabajo.
- **`--data-processing-id`** *`ID del trabajo de procesamiento de datos`*: (opcional) ID del trabajo de procesamiento de datos que creó los artefactos para usarlos en el entrenamiento.
- **`--s3-output-uri`** *`URI de S3`*: (opcional) el URI de S3 en el que se almacena el resultado de este trabajo de entrenamiento de modelos.
- **`--instance-type`** *`(tipo de instancia)`*: (opcional) el tamaño de la instancia que se utilizará para este trabajo de entrenamiento de modelos.
- **`--store-to`*`location-to-store-result`*** — (opcional) La variable en la que se almacena el resultado del entrenamiento del modelo.
- **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado el entrenamiento de modelos.
- **`--wait-interval`*`interval-to-wait`*** — (opcional) Establece el tiempo, en segundos, entre las comprobaciones del estado del entrenamiento del modelo (predeterminado: 60).
- **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de entrenamiento de modelos antes de devolver el estado más reciente (valor predeterminado: 3600).

- **`%neptune_ml training status`**: recupera el estado de un trabajo de entrenamiento de modelos de Neptune ML.

  Parámetros

  - **`--job-id`** *`ID del trabajo`*: el ID del trabajo del que se va a recuperar el estado.
  - **`--store-to`** *`tipo de instancia`*: (opcional) la variable en la que se almacena el resultado del estado.
  - **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado el entrenamiento de modelos.
  - **`--wait-interval`*`interval-to-wait`*** — (opcional) Establece el tiempo, en segundos, entre las comprobaciones del estado del entrenamiento del modelo (predeterminado: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de procesamiento de datos antes de devolver el estado más reciente (valor predeterminado: 3600).

- **`%neptune_ml endpoint create`**: crea un punto de conexión de consulta de un modelo de Neptune ML.

  Parámetros

  - **`--job-id`** *`Identificador para este trabajo`*: (opcional) identificador para asignar a este trabajo.
  - **`--model-job-id`** *`ID del trabajo de entrenamiento de modelos`*: (opcional) ID del trabajo de entrenamiento de modelos para el que se creará un punto de conexión de consulta.
  - **`--instance-type`** *`(tipo de instancia)`*: (opcional) el tamaño de la instancia que se utilizará para el punto de conexión de consulta.
  - **`--store-to`*`location-to-store-result`*— (opcional) La variable en la que se almacena el resultado de la creación del punto final.
  - **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado la creación del punto de conexión.
  - **`--wait-interval`*`interval-to-wait`*— (opcional) Establece el tiempo, en segundos, entre las comprobaciones de estado (predeterminado: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de creación de punto de conexión antes de devolver el estado más reciente (valor predeterminado: 3600).
- **`%neptune_ml endpoint status`**: recupera el estado de un punto de conexión de consulta de Neptune ML.

  Parámetros

  - **`--job-id`** *`ID de creación de punto de conexión`*: (opcional) ID de un trabajo de creación de punto de conexión para notificar el estado.
  - **`--store-to`*`location-to-store-result`*— (opcional) La variable en la que se almacena el resultado del estado.
  - **`--wait`**: (opcional) indica que la operación debe esperar hasta que se haya completado la creación del punto de conexión.
  - **`--wait-interval`*`interval-to-wait`*— (opcional) Establece el tiempo, en segundos, entre las comprobaciones de estado (predeterminado: 60).
  - **`--wait-timeout`** *`timeout-seconds`*: (opcional) establece el tiempo, en segundos, que se debe esperar a que se complete el trabajo de creación de punto de conexión antes de devolver el estado más reciente (valor predeterminado: 3600).

El comando mágico de celda `%%neptune_ml`

El comando mágico de celda `%%neptune_ml` ignora las entradas de línea como `--job-id` o `--export-url`. En cambio, le permite proporcionar esas entradas y otras dentro del cuerpo de la celda.

También puede guardar estas entradas en otra celda, asignarlas a una variable de Jupyter y luego inyectarlas en el cuerpo de la celda con esa variable. De esta forma, puede utilizar dichas entradas una y otra vez sin tener que volver a introducirlas cada vez.

Esto solo funciona si la variable de inyección es el único contenido de la celda. No puede utilizar varias variables en una celda ni una combinación de texto y una variable.

Por ejemplo, el comando mágico de celda `%%neptune_ml export start` puede consumir un documento JSON en el cuerpo de la celda que incluya todos los parámetros descritos en Parámetros utilizados para controlar el proceso de exportación de Neptune.

En el cuaderno Neptune-ML-01-Introduction-to-Node-Classification-Gremlin, en Configuración de características de la sección Exportación de la configuración de modelos y datos, puede ver cómo la siguiente celda incluye los parámetros de exportación en un documento asignado a una variable de Jupyter denominada `export-params`:

```
export_params = {
  "command": "export-pg",
  "params": {
    "endpoint": neptune_ml.get_host(),
    "profile": "neptune_ml",
    "useIamAuth": neptune_ml.get_iam(),
    "cloneCluster": False
   },
  "outputS3Path": f'{s3_bucket_uri}/neptune-export',
  "additionalParams": {
    "neptune_ml": {
      "targets": [
        {
          "node": "movie",
          "property": "genre"
        }
      ],
      "features": [
        {
```

```
          "node": "movie",
          "property": "title",
          "type": "word2vec"
        },
        {
          "node": "user",
          "property": "age",
          "type": "bucket_numerical",
          "range" : [1, 100],
          "num_buckets": 10
        }
      ]
    }
  },
  "jobSize": "medium"}
```

Al ejecutar esta celda, Jupyter guarda el documento de parámetros con ese nombre. A continuación, puede usar `${export_params}` para inyectar el documento JSON en el cuerpo de una `%%neptune_ml export start cell`, de la siguiente manera:

```
%%neptune_ml export start --export-url {neptune_ml.get_export_service_host()} --export-
iam --wait --store-to export_results

${export_params}
```

Formas disponibles del comando mágico de celda **%%neptune_ml**

El comando mágico de celda `%%neptune_ml` se puede utilizar de las siguientes formas:

- **`%%neptune_ml export start`**: inicia un proceso de exportación de Neptune ML.
- **`%%neptune_ml dataprocessing start`**: inicia un trabajo de procesamiento de datos de Neptune ML.
- **`%%neptune_ml training start`**: inicia un trabajo de entrenamiento de modelos de Neptune ML.
- **`%%neptune_ml endpoint create`**: crea un punto de conexión de consulta de Neptune ML para un modelo.

Visualización de gráficos en el entorno de trabajo de Neptune

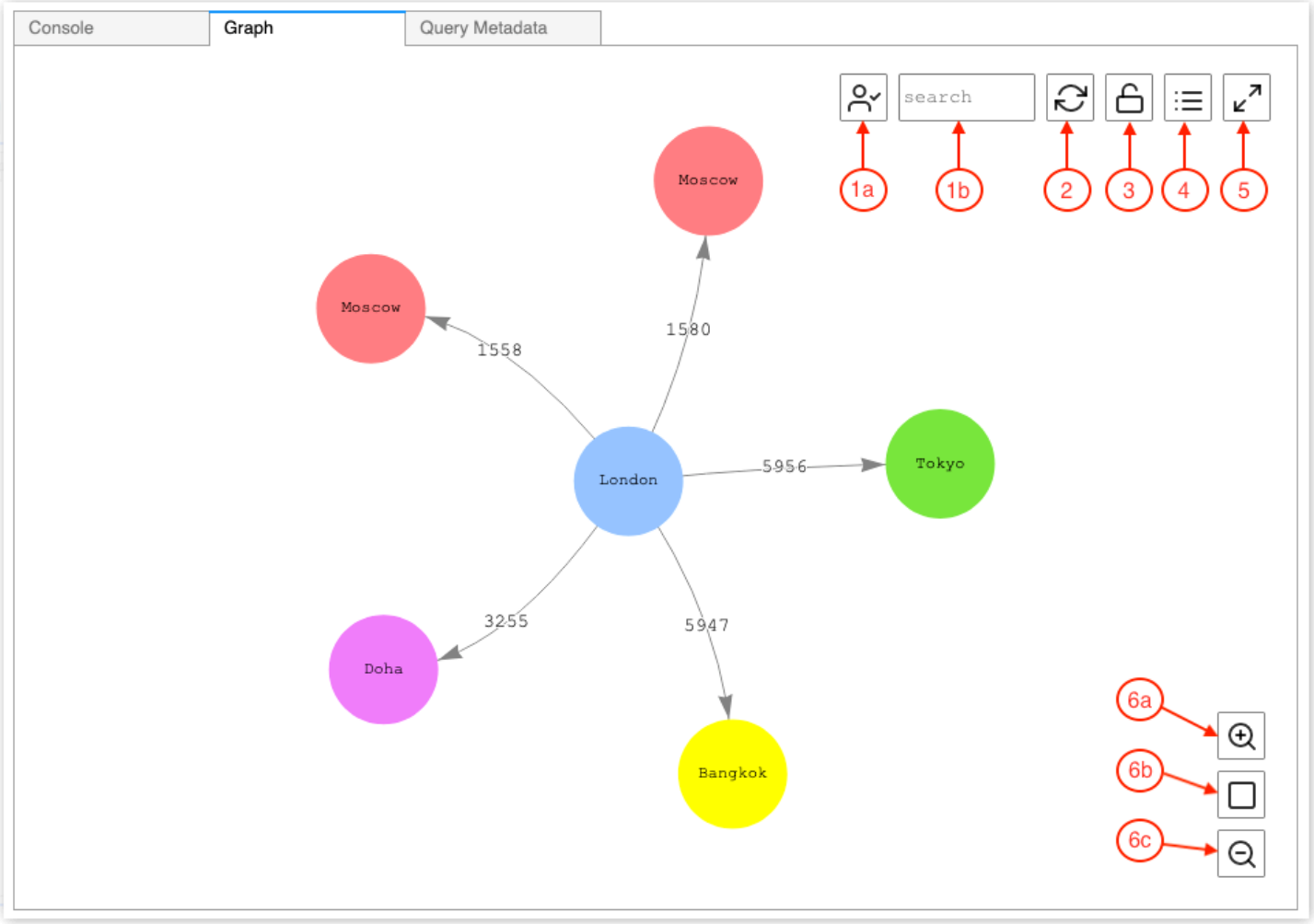
En muchos casos, el entorno de trabajo de Neptune puede crear un diagrama visual de los resultados de la consulta y devolverlos en formato de tabla. La visualización gráfica está disponible en la pestaña Gráfico de los resultados de la consulta siempre que sea posible visualizarla.

Además de las funciones de visualización integradas que se describen aquí, también puede utilizar herramientas de visualización más avanzadas con los cuadernos de gráficos de Neptune.

> **Note**
>
> Para acceder a las funciones y correcciones que se han añadido recientemente en los cuadernos que ya esté utilizando, primero detenga la instancia de cuaderno y, a continuación, reiníciela.

Información general sobre la interfaz de la pestaña Gráfico

Este diagrama identifica los elementos de la interfaz de usuario presentes en la pestaña Gráfico:



1. Búsqueda de gráficos
   a. Alternar el UUID: cambia la inclusión de los valores de las propiedades del ID en la búsqueda de gráficos. De forma predeterminada, la inclusión de ID está habilitada. Si está deshabilitada, las coincidencias en las propiedades de los identificadores, incluidas las propiedades de borde que hacen referencia a los identificadores de los nodos, no aparecerán resaltadas.
   b. Campo de texto de búsqueda: resalta todos los valores de las propiedades de vértice y borde que incluyen la cadena de texto que especifique aquí.
2. Restablecimiento de gráfico: vuelve a ejecutar la simulación física del gráfico y establece el zoom para que el gráfico quepa en la ventana.
3. Alternar la física de los gráficos: cambia el funcionamiento de la simulación física del gráfico. La simulación física está habilitada de forma predeterminada, lo que permite que el gráfico cambie de forma dinámica. Si está deshabilitada, los vértices permanecen bloqueados en su posición cuando se mueven otros vértices.

4. Vista de detalles: cuando se selecciona un nodo o un borde, se muestra una lista de las claves y valores de las propiedades del elemento, si están disponibles en los resultados de la consulta.
5. Vista de pantalla completa: expande la ventana de la pestaña Gráfico para adaptarla a la pantalla. Al hacer clic de nuevo, la pestaña se minimizará.
6. Opciones de zoom
   a. Ampliar
   b. Restablecimiento del zoom: establece el zoom para que quepan todos los vértices en la ventana de la pestaña Gráfico.
   c. Alejar

Visualización de resultados de las consultas Gremlin

El entorno de trabajo de Neptune crea una visualización de los resultados de cualquier consulta de Gremlin que devuelva una `path`. Para ver la visualización, seleccione la pestaña Gráfico situada a la derecha de la pestaña Consola, debajo de la consulta, después de ejecutarla.

Puede utilizar las sugerencias de visualización de consultas para controlar la forma en que el visualizador representa el resultado de las consultas. Estas sugerencias siguen los comandos mágicos de celda `%%gremlin` y van precedidas del nombre del parámetro `--path-pattern` (o su forma abreviada, `-p`):

```
%%gremlin -p comma-separated hints
```

También puede usar el indicador `--group-by` (o `-g`) para especificar una propiedad de los vértices por la que agruparlos. Esto permite especificar un color o un icono para diferentes grupos de vértices.

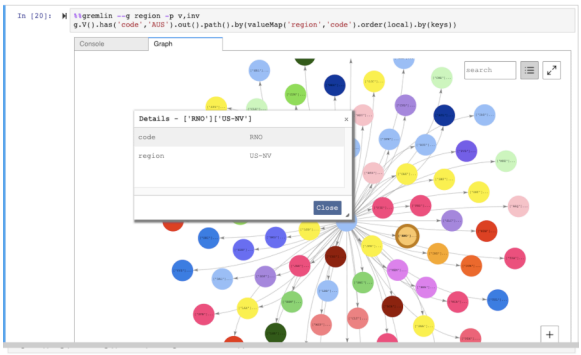
Los nombres de las sugerencias reflejan los pasos de Gremlin que se utilizan habitualmente al atravesar vértices, y se comportan en consecuencia. Se pueden usar varias sugerencias combinadas, separadas por comas o sin espacios entre ellas. Las sugerencias utilizadas deben coincidir con los pasos de Gremlin correspondientes a la consulta que se está visualizando. A continuación se muestra un ejemplo:

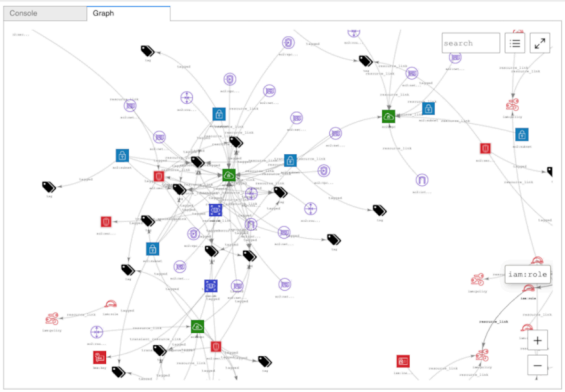
```
%%gremlin -p v,oute,inv
g.V().hasLabel('airport').outE().inV().path().by('code').by('dist').limit(5)
```

Las sugerencias de visualización disponibles son las siguientes:

```
v
inv
outv
e
ine
oute
```

A continuación, se muestran algunos ejemplos de visualizaciones de gráficos que utilizan grupos:





Visualización de resultados de las consultas de SPARQL

El entorno de trabajo de Neptune crea una visualización de los resultados de las consultas de cualquier consulta de SPARQL que adopte una de estas formas:

- `SELECT ?subject ?predicate ?object`
- `SELECT ?s ?p ?o`

Para ver la visualización, seleccione la pestaña Gráfico situada a la derecha de la pestaña Tabla, debajo de la consulta, después de ejecutarla.

De forma predeterminada, una visualización de SPARQL solo incluye patrones triples en los que o? es `uri` o bnode (nodo en blanco). Todos los demás tipos de enlaces `?o`, como las cadenas literales

o los números enteros, se tratan como propiedades del nodo `?s` y se pueden ver mediante el panel detalles de la pestaña Gráfico.

Sin embargo, en muchos casos, es posible que desee incluir dichos valores literales como vértices en la visualización. Para ello, use la sugerencia de consulta `--expand-all` que aparece después del comando mágico de celda `%%sparql`:

```
%%sparql --expand-all
```

Esto le indica al visualizador que incluya todos los resultados `?s ?p ?o` en el diagrama del gráfico, independientemente del tipo de enlace.

Puede ver que esta sugerencia se ha utilizado en todo el cuaderno de `Air-Routes-SPARQL.ipynb` y puede probar mediante la ejecución de las consultas con y sin la sugerencia para ver qué diferencia hay en la visualización.

Acceso a los cuadernos con tutoriales de visualización en el entorno de trabajo de Neptune

Los dos cuadernos con tutoriales de visualización que se incluyen con el entorno de trabajo de Neptune proporcionan una gran cantidad de ejemplos en Gremlin y en SPARQL sobre cómo consultar datos de gráficos de forma eficaz y visualizar los resultados.

Navegación hasta los cuadernos de visualización

1. En el panel de navegación de la izquierda, seleccione el botón Abrir cuaderno de la derecha.
2. Una vez que se abra el entorno de trabajo de Neptune, que ejecuta Jupyter, verá una carpeta de Neptune en el nivel superior. Selecciónela para abrir la carpeta.
3. En el siguiente nivel hay una carpeta denominada 02-Visualization. Abra esta carpeta. En su interior hay varios cuadernos en los que se explican diferentes formas de consultar los datos de los gráficos, en Gremlin y en SPARQL, y se explica cómo visualizar los resultados de las consultas:

   - Air-Routes-Gremlin
   - Air-Routes-SPARQL
   - Blog de visualización del entorno de trabajo
   - EPL-Gremlin

- EPL-SPARQL

Seleccione un cuaderno para probar con las consultas que incluye.

Configuración de Neptune

Le damos la bienvenida a Amazon Neptune. En esta sección, le ayudaremos a crear un nuevo clúster de base de datos de Neptune y a encontrar lo que busca en la documentación de Neptune.

> **Note**
>
> Para obtener información AWS sobre las arquitecturas de referencia de bases de datos gráficas y las arquitecturas de implementación de referencia, consulte Amazon Neptune Resources. Estos recursos pueden servirle para proporcionar información de sus elecciones sobre los modelos de datos de gráficos y lenguajes de consulta, y acelerar su proceso de desarrollo.

Temas

Elección del tipo de instancia de base de datos de Neptune correcto

Amazon Neptune dispone de varios tamaños y familias de instancias diferentes, que ofrecen distintas capacidades que se adaptan a diferentes cargas de trabajo de gráficos. El objetivo de esta sección es ayudarle a elegir el mejor tipo de instancia para sus necesidades.

Para conocer los precios de cada tipo de instancia de estas familias, consulte la página de precios de Neptune.

Información general sobre la asignación de recursos de instancias

Cada tipo y tamaño de instancia de Amazon EC2 que se utiliza en Neptune tiene una cantidad definida de computación (vCPU) y memoria del sistema. El almacenamiento principal de Neptune es externo a las instancias de base de datos de un clúster, lo que permite que la capacidad de procesamiento y almacenamiento se escale de forma independiente.

Esta sección se centra en cómo se pueden escalar los recursos de computación y en las diferencias entre cada una de las distintas familias de instancias.

En todas las familias de instancias, los recursos de vCPU se asignan para admitir dos (2) subprocesos de ejecución de consultas por vCPU. Esta capacidad viene determinada por el tamaño de la instancia. Al determinar el tamaño adecuado de una instancia de base de datos de Neptune específica, debe tener en cuenta la posible simultaneidad de la aplicación y la latencia media de las consultas. Puede calcular la cantidad de vCPU necesaria de la siguiente manera, donde la latencia se mide como la latencia media de consultas en segundos y la simultaneidad se mide como el número objetivo de consultas por segundo:

$$vCPUs = \frac{latency \times concurrency}{2}$$

> **Note**
>
> En determinadas circunstancias, las consultas de SPARQL, las consultas de openCypher y las consultas de lectura de Gremlin que utiliza el motor de consultas de DFE pueden emplear más de un subproceso de ejecución por consulta. Al dimensionar inicialmente el clúster de base de datos, comience con la suposición de que cada consulta consume un único subproceso de ejecución por ejecución y escale verticalmente si observa una contrapresión en la cola de consultas. Esto se puede observar mediante el uso de `/sparql/status` las API `/gremlin/status/oc/status`, o también se puede observar mediante la métrica. `MainRequestsPendingRequestsQueue` CloudWatch

La memoria del sistema de cada instancia se divide en dos asignaciones principales: la caché del grupo de búferes y la memoria de subprocesos de ejecución de consultas.

Aproximadamente dos tercios de la memoria disponible de una instancia se asigna a la caché del grupo de búferes. La caché del grupo de búferes se usa para almacenar en caché los componentes del gráfico utilizados más recientemente para acceder más rápidamente a las consultas que acceden repetidamente a esos componentes. Las instancias con una mayor cantidad de memoria del sistema tienen cachés de grupos de búferes más grandes que pueden almacenar una mayor parte del gráfico de forma local. Un usuario puede ajustar la cantidad adecuada de caché del búfer agrupado supervisando las métricas de aciertos y errores de caché disponibles en el búfer. CloudWatch

Es posible que desee aumentar el tamaño de la instancia si la tasa de aciertos de caché desciende por debajo del 99,9 % durante un período de tiempo constante. Esto sugiere que el conjunto de búferes no es lo suficientemente grande y que el motor tiene que buscar datos del volumen de almacenamiento subyacente con una frecuencia que no es eficiente.

El tercio restante de la memoria del sistema se distribuye de manera uniforme entre los subprocesos de ejecución de consultas, con algo de memoria restante para el sistema operativo y un pequeño grupo dinámico para que los subprocesos la utilicen según sea necesario. La memoria disponible para cada subproceso aumenta ligeramente de un tamaño de instancia a otro hasta llegar a un tipo de instancia `8xl`, a cuyo tamaño la memoria asignada por subproceso alcanza el máximo.

El momento de añadir más memoria de subprocesos es cuando se encuentra con una `OutOfMemoryException` (OOM). Las excepciones OOM se producen cuando un subproceso necesita más memoria que la máxima que se le ha asignado (esto no es lo mismo que toda la instancia se quede sin memoria).

Tipos de instancias **`t3`** y **`t4g`**

La familia de instancias `t3` y `t4g` es una opción económica para empezar a utilizar una base de datos de gráficos y también para el desarrollo y las pruebas iniciales. Estas instancias son aptas para la oferta de capa gratuita de Neptune, que permite a los nuevos clientes utilizar Neptune sin coste alguno durante las primeras 750 horas de instancia utilizadas en una AWS cuenta independiente o acumuladas en una AWS organización con facturación unificada (cuenta de pagador).

Las instancias `t3` y `t4g` solo se ofrecen en la configuración de tamaño medio (`t3.medium` y `t4g.medium`).

No se han diseñado para utilizarse en un entorno de producción.

Dado que estas instancias tienen recursos muy limitados, no se recomiendan para probar el tiempo de ejecución de las consultas o el rendimiento general de la base de datos. Para evaluar el rendimiento de las consultas, actualice a una de las otras familias de instancias.

Familia **r4** de tipos de instancias

EN DESUSO: la familia `r4` se ofreció en 2018 cuando se lanzó Neptune, pero ahora los tipos de instancias más recientes tienen una relación precio-rendimiento mucho mejor. A partir de la versión 1.1.0.0 del motor, Neptune ya no admite tipos de instancias `r4`.

Familia **r5** de tipos de instancias

La familia `r5` contiene tipos de instancias optimizadas para memoria que funcionan bien en la mayoría de los casos de uso de gráficos. La familia `r5` contiene tipos de instancias desde `r5.large` hasta `r5.24xlarge`. Escalan linealmente el rendimiento de computación a medida que aumenta el tamaño. Por ejemplo, un `r5.xlarge` (4 vCPU y 32 GiB de memoria) tiene el doble de vCPU y memoria que un `r5.large` (2 vCPU y 16 GiB de memoria), y un `r5.2xlarge` (8 vCPU y 64 GiB de memoria) tiene el doble de vCPU y memoria que un `r5.xlarge`. Puede esperar que el rendimiento de las consultas se escale directamente con la capacidad de computación con tipos de instancias de hasta `r5.12xlarge`.

La familia de instancias `r5` tiene una arquitectura de CPU Intel de 2 sockets. Los tipos `r5.12xlarge` y los más pequeños utilizan un solo socket y la memoria del sistema es propiedad de ese procesador de un solo socket. Los tipos `r5.16xlarge` y `r5.24xlarge` utilizan los dos sockets y la memoria disponible. Dado que se requiere cierta sobrecarga en la administración de la memoria entre dos procesadores físicos de una arquitectura de 2 sockets, las ganancias de rendimiento si se escala verticalmente de un tipo de instancia `r5.12xlarge` a un `r5.16xlarge` o `r5.24xlarge` no son tan lineales como las que se obtienen escalando verticalmente los tamaños más pequeños.

Familia **r5d** de tipos de instancias

Neptune tiene una característica de caché de búsqueda que se puede utilizar para mejorar el rendimiento de las consultas que necesitan recuperar y devolver un gran número de valores de propiedades y literales. Esta característica la utilizan principalmente los clientes con consultas que necesitan devolver muchos atributos. La caché de búsqueda mejora el rendimiento de estas consultas al obtener estos valores de atributos de forma local en lugar de buscarlos una y otra vez en el almacenamiento indexado de Neptune.

La caché de búsqueda se implementa mediante un volumen de EBS asociado a NVMe en un tipo de instancia `r5d`. Se habilita mediante un grupo de parámetros de un clúster. A medida que los datos se obtienen del almacenamiento indexado de Neptune, los valores de las propiedades y los literales de RDF se almacenan en caché en este volumen de NVMe.

Si no necesita la característica de caché de búsqueda, utilice un tipo de instancia `r5` estándar en lugar de `r5d` para evitar el mayor costo que supone `r5d`.

La familia `r5d` tiene tipos de instancias del mismo tamaño que la familia `r5`, desde `r5d.large` hasta `r5d.24xlarge`.

Familia **`r6g`** de tipos de instancias

AWS ha desarrollado su propio procesador basado en ARM, llamado Graviton, que ofrece una mejor relación precio/rendimiento que sus equivalentes de Intel y AMD. La familia `r6g` utiliza el procesador Graviton2. En nuestras pruebas, el procesador Graviton2 ofrece un rendimiento entre un 10 y un 20 % mejor para las consultas gráficas de estilo OLTP (restringidas). Sin embargo, las consultas más grandes, tipo OLAP, pueden tener un rendimiento ligeramente inferior con los procesadores Graviton2 que con los procesadores Intel, debido a que el rendimiento de paginación de memoria es ligeramente inferior.

También es importante tener en cuenta que la familia `r6g` tiene una arquitectura de un solo socket, lo que significa que el rendimiento se escala linealmente con la capacidad de computación, desde un `r6g.large` a un `r6g.16xlarge` (el tipo más grande de la familia).

Familia **`r6i`** de tipos de instancias

Las instancias Amazon R6i funcionan con procesadores escalables Intel Xeon de tercera generación (nombre en código Ice Lake) y son ideales para cargas de trabajo que hacen un uso intensivo de la memoria. Como regla general, ofrecen hasta un 15 % más de rendimiento de computación y hasta un 20 % más de ancho de banda de memoria por vCPU que los tipos de instancias R5 comparables.

Familia **`x2g`** de tipos de instancias

Algunos casos de uso de gráficos tienen mejor rendimiento cuando las instancias tienen cachés de grupos de búferes más grandes. La familia `x2g` se lanzó para facilitar mejor esos casos de uso. La `x2g` familia tiene una relación de memory-to-v CPU mayor que la familia or. `r5` `r6g` Las instancias `x2g` también utilizan el procesador Graviton2 y tienen muchas de las mismas características de rendimiento que los tipos de instancias `r6g`, además de una memoria caché de grupos de búferes más grande.

Si tiene tipos de instancias `r5` o `r6g` con un bajo uso de la CPU y una alta tasa de errores de caché del grupo de búferes, pruebe a usar la familia `x2g` en su lugar. De esta forma, conseguirá la memoria adicional que necesita sin tener que pagar más capacidad de CPU.

Tipo de instancia **`serverless`**

La característica Neptune sin servidor puede escalar el tamaño de la instancia de forma dinámica en función de las necesidades de recursos de la carga de trabajo. En lugar de calcular cuántas vCPU se necesitan para su aplicación, Neptune sin servidor le permite establecer unos límites inferior y superior en la capacidad de computación (medidos en unidades de capacidad de Neptune) para las instancias de su clúster de base de datos. Los costos de las cargas de trabajo con distintos usos se pueden optimizar si se utilizan instancias sin servidor en lugar de aprovisionadas.

Puede configurar instancias aprovisionadas y sin servidor en el mismo clúster de base de datos para lograr una configuración con una relación costo-rendimiento óptima.

Elegir el tipo de almacenamiento adecuado para su clúster de base de datos de Neptune

Neptune ofrece dos tipos de almacenamiento con un modelo de precios diferente:

- Almacenamiento estándar: el almacenamiento estándar proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a bajo.
- Almacenamiento optimizado para E/S: con el almacenamiento optimizado para E/S, disponible a partir de la versión 1.3.0.0 del motor, solo paga por el almacenamiento y las instancias que utilice. Los costos de almacenamiento son más elevados que los del almacenamiento estándar y no paga nada por la E/S que utilice. Si el uso de E/S es elevado, el almacenamiento de IOPS aprovisionado puede reducir los costos de forma significativa.

  El almacenamiento optimizado para E/S está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S a un costo predecible, con una latencia de E/S baja y un rendimiento de E/S constante. Solo puede cambiar entre los tipos de almacenamiento estándar y optimizado para E/S una vez cada 30 días.

  Para obtener información sobre el almacenamiento optimizado para E/S, consulte la página de precios de Neptune. En la siguiente sección se describe cómo configurar el almacenamiento optimizado para E/S para un clúster de base de datos de Neptune.

Elección de E/S: almacenamiento optimizado para un clúster de base de datos Neptune

De forma predeterminada, los clústeres de base de datos de Neptune utilizan almacenamiento estándar. Puede habilitar el almacenamiento optimizado para E/S en un clúster de base de datos en el momento de crearlo, de la siguiente manera:

A continuación se muestra un ejemplo de cómo puede habilitar el almacenamiento optimizado para E/S al crear un clúster mediante la AWS CLI:

```
aws neptune create-db-cluster \
  --database-name (name for the new database) \
  --db-cluster-identifier (an ID for the cluster) \
  --engine neptune \
  --engine-version 1.3.0.0 \
  --storage-type iopt1
```

A continuación, cualquier instancia que cree automáticamente tendrá habilitado el almacenamiento optimizado para E/S:

```
aws neptune create-db-instance \
  --db-cluster-identifier (the ID of the new cluster) \
  --db-instance-identifier (an ID for the new instance) \
  --engine neptune \
  --db-instance-class db.r5.large
```

Asimismo, puede modificar un clúster de base de datos existente para habilitar el almacenamiento optimizado para E/S en él, de la siguiente manera:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the ID of a cluster without I/O-Optimized storage) \
  --storage-type iopt1 \
  --apply-immediately
```

Puede restaurar una instantánea de copia de seguridad en un clúster de base de datos con el almacenamiento optimizado para E/S activado:

```
aws neptune restore-db-cluster-from-snapshot \
  --db-cluster-identifier (an ID for the restored cluster) \
```

```
  --snapshot-identifier (the ID of the snapshot to restore from) \
  --engine neptune \
  --engine-version 1.3.0.0 \
  --storage-type iopt1
```

Puede determinar si un clúster utiliza un almacenamiento optimizado para E/S mediante cualquier llamada `describe-`. Si el almacenamiento optimizado para E/S está activado, la llamada devuelve un campo de tipo de almacenamiento establecido en `iop1`.

Creación de un nuevo clúster de base de datos de Neptune

La forma más sencilla de crear un nuevo clúster de base de datos de Amazon Neptune es utilizar una AWS CloudFormation plantilla que cree todos los recursos necesarios sin tener que hacerlo todo manualmente. La AWS CloudFormation plantilla realiza gran parte de la configuración por usted, incluida la creación de una instancia de Amazon Elastic Compute Cloud (Amazon EC2):

Para lanzar un nuevo clúster de base de datos de Neptune mediante una plantilla AWS CloudFormation

1. Cree un nuevo usuario de IAM con los permisos que necesitará para trabajar con su clúster de base de datos de Neptune, tal y como se explica en Permisos de usuario de IAM.
2. Configure los requisitos previos adicionales necesarios para usar la AWS CloudFormation plantilla, tal y como se explica en. Requisitos previos para su uso en AWS CloudFormation la configuración de Neptune
3. Invoque la AWS CloudFormation pila, tal y como se describe en. Uso de una AWS CloudFormation pila para crear un clúster de base de datos de Neptune

También puede crear una base de datos global de Neptune que abarque varias Regiones de AWS, lo que permita lecturas globales de baja latencia y proporcione una recuperación rápida en el raro caso de que una interrupción afecte a un todo. Región de AWS

Para obtener información sobre la creación manual de un clúster de Amazon Neptune mediante el AWS Management Console, consulte. Lanzamiento de un clúster de base de datos de Neptune mediante la AWS Management Console

También puede usar una AWS CloudFormation plantilla para crear una función Lambda para usarla con Neptune (consulte). Uso de AWS CloudFormation para crear una función de Lambda para usarla en Neptune

Para obtener información sobre la administración de clústeres e instancias en Neptune, consulte Administración de la base de datos de Amazon Neptune.

Requisitos previos para su uso en AWS CloudFormation la configuración de Neptune

Antes de crear un clúster de Amazon Neptune mediante una AWS CloudFormation plantilla, debe disponer de lo siguiente:

- Un par de claves de Amazon EC2.
- Los permisos necesarios para su uso AWS CloudFormation.

Cree un par de claves de Amazon EC2 para lanzar un clúster de Neptune mediante AWS CloudFormation

Para lanzar un clúster de base de datos de Neptune mediante una AWS CloudFormation plantilla, debe tener un par de claves de Amazon EC2 (y su archivo PEM asociado) disponible en la región en la que cree la pila. AWS CloudFormation

Si necesita crear el par de claves, consulte Creación de un par de claves con Amazon EC2 en la Guía del usuario de Amazon EC2 o Creación de un par de claves con Amazon EC2 en la Guía del usuario de Amazon EC2 para obtener instrucciones.

Añada políticas de IAM para conceder los permisos necesarios para usar la plantilla AWS CloudFormation

En primer lugar, debe tener un usuario de IAM configurado con los permisos necesarios para trabajar con Neptune, tal y como se describe en Creación de un usuario de IAM con permisos para Neptune.

A continuación, debe añadir la política AWS gestionada`AWSCloudFormationReadOnlyAccess`, a ese usuario.

Por último, debe crear la siguiente política administrada por el cliente y añadirla a ese usuario:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "rds:CreateDBCluster",
          "rds:CreateDBInstance"
      ],
```

```
      "Resource": [
          "arn:aws:rds:*:*:*"
      ],
      "Condition": {
          "StringEquals": {
              "rds:DatabaseEngine": ["graphdb","neptune"]
          }
      }
    },
    {
      "Action": [
        "rds:AddRoleToDBCluster",
        "rds:AddSourceIdentifierToSubscription",
        "rds:AddTagsToResource",
        "rds:ApplyPendingMaintenanceAction",
        "rds:CopyDBClusterParameterGroup",
        "rds:CopyDBClusterSnapshot",
        "rds:CopyDBParameterGroup",
        "rds:CreateDBClusterParameterGroup",
        "rds:CreateDBClusterSnapshot",
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSubnetGroup",
        "rds:CreateEventSubscription",
        "rds:DeleteDBCluster",
        "rds:DeleteDBClusterParameterGroup",
        "rds:DeleteDBClusterSnapshot",
        "rds:DeleteDBInstance",
        "rds:DeleteDBParameterGroup",
        "rds:DeleteDBSubnetGroup",
        "rds:DeleteEventSubscription",
        "rds:DescribeAccountAttributes",
        "rds:DescribeCertificates",
        "rds:DescribeDBClusterParameterGroups",
        "rds:DescribeDBClusterParameters",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBClusters",
        "rds:DescribeDBEngineVersions",
        "rds:DescribeDBInstances",
        "rds:DescribeDBLogFiles",
        "rds:DescribeDBParameterGroups",
        "rds:DescribeDBParameters",
        "rds:DescribeDBSecurityGroups",
        "rds:DescribeDBSubnetGroups",
```

```
        "rds:DescribeEngineDefaultClusterParameters",
        "rds:DescribeEngineDefaultParameters",
        "rds:DescribeEventCategories",
        "rds:DescribeEventSubscriptions",
        "rds:DescribeEvents",
        "rds:DescribeOptionGroups",
        "rds:DescribeOrderableDBInstanceOptions",
        "rds:DescribePendingMaintenanceActions",
        "rds:DescribeValidDBInstanceModifications",
        "rds:DownloadDBLogFilePortion",
        "rds:FailoverDBCluster",
        "rds:ListTagsForResource",
        "rds:ModifyDBCluster",
        "rds:ModifyDBClusterParameterGroup",
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
          "*"
      ]
    },
    {
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
```

```
        "kms:ListAliases",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
          "*"
      ]
    },
    {
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
            "iam:passedToService": "rds.amazonaws.com"
        }
      }
    },
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    }
  ]
}
```

> **Note**
>
> Los siguientes permisos solo son necesarios para eliminar una pila: `iam:DeleteRole`, `iam:RemoveRoleFromInstanceProfile`, `iam:DeleteRolePolicy`, `iam:DeleteInstanceProfile` y `ec2:DeleteVpcEndpoints`.
> Tenga en cuenta también que `ec2:*Vpc` concede permisos `ec2:DeleteVpc`.

Uso de una AWS CloudFormation pila para crear un clúster de base de datos de Neptune

Puede utilizar una AWS CloudFormation plantilla para configurar un clúster de base de datos de Neptune.

1. Para lanzar la AWS CloudFormation pila en la AWS CloudFormation consola, seleccione uno de los botones Iniciar pila de la siguiente tabla.

Región	Visualización	Ver en Designer	iniciar
Este de EE. UU. (Norte de Virginia)	Ver	Ver en Designer	
Este de EE. UU. (Ohio)	Ver	Ver en Designer	
Oeste de EE. UU. (Norte de California)	Ver	Ver en Designer	
Oeste de EE. UU. (Oregón)	Ver	Ver en Designer	
Canadá (centro)	Ver	Ver en Designer	
América del Sur (São Paulo)	Ver	Ver en Designer	
Europa (Estocolmo)	Ver	Ver en Designer	
Europa (Irlanda)	Ver	Ver en Designer	
Europa (Londres)	Ver	Ver en Designer	
Europa (París)	Ver	Ver en Designer	

Región	Visualización	Ver en Designer	iniciar
Europa (Fráncfort)	Ver	Ver en Designer	
Medio Oriente (Baréin)	Ver	Ver en Designer	
Medio Oriente (EAU)	Ver	Ver en Designer	
Israel (Tel Aviv)	Ver	Ver en Designer	
África (Ciudad del Cabo)	Ver	Ver en Designer	
Asia-Pacífico (Hong Kong)	Ver	Ver en Designer	
Asia-Pacífico (Tokio)	Ver	Ver en Designer	
Asia-Pacífico (Seúl)	Ver	Ver en Designer	
Asia-Pacífico (Singapur)	Ver	Ver en Designer	
Asia-Pacífico (Sídney)	Ver	Ver en Designer	
Asia-Pacífico (Bombay)	Ver	Ver en Designer	
China (Pekín)	Ver	Ver en Designer	
China (Ningxia)	Ver	Ver en Designer	
AWS GovCloud (EE. UU.-Oeste)	Ver	Ver en Designer	

Región	Visualización	Ver en Designer	iniciar
AWS GovCloud (EE. UU.-Este)	Ver	Ver en Designer	

2. En la página Select Template, elija Next.
3. En la página Especificar detalles, elija un key pair para el nombre EC2SSH KeyPair.

   Este par de claves es necesario para acceder a la instancia EC2. Asegúrese de que tiene el archivo PEM para el par de claves que elija.
4. Elija Siguiente.
5. En la página Opciones, seleccione Siguiente.
6. En la página Revisar, seleccione la primera casilla para confirmar que AWS CloudFormation debe crear los recursos de IAM. Seleccione la segunda casilla para confirmar `CAPABILITY_AUTO_EXPAND` para la nueva pila.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` confirma explícitamente que las macros se expandirán al crear la pila, sin revisión previa. Los usuarios suelen crear un conjunto de cambios a partir de una plantilla procesada para que los cambios realizados por las macros puedan revisarse antes de crear la pila. Para obtener más información, consulte la API AWS CloudFormation CreateStack.

   A continuación, seleccione Crear.

> **Note**
>
> También puede utilizar la AWS CloudFormation plantilla para actualizar la versión del motor del clúster de base de datos.

Configure la Amazon VPC en la que se encuentra el clúster de base de datos de Amazon Neptune

Un clúster de base de datos de Amazon Neptune solo se puede crear en una Amazon Virtual Private Cloud (Amazon VPC). Se puede acceder a sus puntos de conexión dentro de esa VPC.

Existen varias formas de configurar la VPC, según la forma en que desee acceder al clúster de base de datos.

Estos son algunos aspectos que se deben tener en cuenta al configurar la VPC en la que se encuentra el clúster de base de datos de Neptune:

- La VPC debe tener dos subredes como mínimo. Estas subredes deben estar en dos diferentes zonas de disponibilidad (AZ). Al distribuir las instancias de su clúster en al menos dos AZ, Neptune ayuda a garantizar que siempre haya instancias disponibles en su clúster de base de datos, incluso en el improbable caso de que falle una zona de disponibilidad. El volumen de su clúster de base de datos de Neptune siempre abarca tres zonas de disponibilidad para proporcionar un almacenamiento duradero con una probabilidad extremadamente baja de pérdida de datos.
- Los bloques de CIDR de cada subred deben ser lo suficientemente grandes como para proporcionar las direcciones IP que Neptune pueda necesitar durante las actividades de mantenimiento, la conmutación por error y el escalado.
- La VPC debe tener un grupo de subredes de base de datos que contenga las subredes que haya creado. Neptune elige una de las subredes del grupo de subredes y una dirección IP dentro de esa subred para asociarla con cada instancia de base de datos en el clúster de base de datos. La instancia de base de datos se coloca entonces en la misma zona de disponibilidad que la subred.
- La VPC debe tener el DNS habilitado (tanto los nombres de host de DNS como la resolución de DNS).
- La VPC debe tener un grupo de seguridad de VPC que permita el acceso al clúster de base de datos.
- La tenencia en una VPC de Neptune debe estar configurada como Predeterminada.

Adición de subredes a la VPC en la que se encuentra el clúster de base de datos de Neptune

Una subred es un rango de direcciones IP en su VPC. Puede iniciar recursos, como un clúster de base de datos de Neptune o una instancia EC2 en una subred específica. Al crear una subred,

debe especificar el bloque de CIDR IPv4 de la subred, que es un subconjunto del bloque de CIDR de la VPC. Cada subred debe residir enteramente en una zona de disponibilidad (AZ) y no puede abarcar otras zonas. Al iniciar instancias en distintas zonas de disponibilidad, puede proteger sus aplicaciones de un fallo en una de las zonas. Para obtener más información, consulte la documentación de subredes de VPC.

Un clúster de base de datos de Neptune requiere al menos dos subredes de VPC.

Para añadir subredes a una VPC

1. Inicie sesión en la consola de Amazon VPC AWS Management Console y ábrala en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación, elija Subnets (Subredes).
3. En Panel de VPC, elija Subredes y, luego, Crear subred.
4. En la página Crear subred, elija la VPC en la que desee crear la subred.
5. En Configuración de la subred, seleccione las siguientes opciones:
    a. Introduzca un nombre para la nueva subred en Nombre de la subred.
    b. Elija una zona de disponibilidad (AZ) para la subred o deje la opción en Sin preferencia.
    c. Introduzca el bloque de direcciones IP de la subred en el Bloque CIDR de IPv4.
    d. Añada etiquetas a la subred si es necesario.
    e. Elija
6. Si desea crear otra subred al mismo tiempo, elija Agregar nueva subred.
7. Elija Crear subred para crear las nuevas subredes.

Cree un grupo de subredes en la VPC

Cree un grupo de subredes.

Para crear un grupo de subredes de Neptune

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Elija Subnet Groups (Grupos de subredes) y, a continuación, elija Create DB Subnet Group (Crear grupo de subredes de base de datos).

3. Introduzca un nombre y una descripción para el nuevo grupo de subredes (la descripción es obligatoria).
4. En VPC, elija la VPC en la que desee que esté este grupo de subredes.
5. En Zona de disponibilidad, elija la AZ en la que desee que esté este grupo de subredes.
6. En Subred, añada una o más subredes de esta AZ a este grupo de subredes.
7. Elija Crear para crear el nuevo grupo de subredes.

Cree un grupo de seguridad con la consola de VPC

Los grupos de seguridad proporcionan acceso al clúster de base de datos de Neptune en la VPC. Actúan como firewall para el clúster de base de datos asociado y controlan el tráfico entrante y saliente en el nivel de la instancia. De manera predeterminada, una instancia de base de datos se crea con un firewall y un grupo de seguridad predeterminado que impide cualquier acceso a ella. Para habilitar el acceso, debe tener un grupo de seguridad de VPC con reglas adicionales.

El siguiente procedimiento muestra cómo añadir una regla TCP personalizada que especifique el rango de puertos y las direcciones IP que debe utilizar la instancia de Amazon EC2 para acceder a su clúster de base de datos de Neptune. Puede utilizar el grupo de seguridad de VPC asignado a la instancia EC2 en lugar de la dirección IP.

Para crear un grupo de seguridad de VPC para Neptune en la consola

1. Inicie sesión en la consola de Amazon VPC AWS Management Console y ábrala en https://console.aws.amazon.com/vpc/.
2. En la esquina superior derecha de la consola, elija la AWS región en la que desee crear un grupo de seguridad de VPC para Neptune. La lista de recursos de Amazon VPC para esa región debería mostrar que tiene al menos una VPC y varias subredes. Si no es así, no tiene una VPC predeterminada en esa región.
3. En el panel de navegación, en Seguridad, elija Grupos de seguridad.
4. Elija Crear grupo de seguridad. En la ventana Crear grupo de seguridad, introduzca el Nombre del grupo de seguridad, una Descripción y el identificador de la VPC en la que residirá el clúster de base de datos de Neptune.
5. Añada una regla de entrada para el grupo de seguridad de una instancia de Amazon EC2 que desee conectar a su clúster de base de datos de Neptune:

    a. En la sección Reglas de entrada, elija Agregar regla.

b. En la lista Tipos, deje seleccionada la opción TCP personalizado.

c. En el cuadro de texto Intervalo de puertos, escriba 8182, que es el valor de puerto predeterminado para Neptune.

d. En Origen, introduzca el rango de direcciones IP (valor de CIDR) desde el que accederá a Neptune o seleccione un nombre de grupo de seguridad existente.

e. Si necesita añadir más direcciones IP o diferentes intervalos de puertos, elija de nuevo Agregar regla.

6. En el área de reglas de salida, también puede añadir una o más reglas de salida si lo necesita.

7. Cuando termine, elija Create security group (Crear grupo de seguridad).

Puede utilizar este nuevo grupo de seguridad de VPC al crear un nuevo clúster de base de datos de Neptune.

Si se usa una VPC predeterminada, ya se ha creado un grupo de subredes predeterminado que abarca todas las subredes de la VPC. Al elegir Crear base de datos en la consola de Neptune, se utiliza la VPC predeterminada, a menos que especifique otra.

Asegúrese de que la VPC admite DNS

El sistema de nombres de dominio (DNS) es un estándar mediante el cual los nombres utilizados en Internet se resuelven a sus direcciones IP correspondientes. Un nombre de host de DNS designa a un equipo de forma exclusiva y se compone de un nombre de host y un nombre de dominio. Los servidores DNS resuelven los nombres de host DNS a sus direcciones IP correspondientes.

Compruebe que los nombres de host de DNS y la resolución de DNS están habilitados en la VPC. Los atributos de red de VPC `enableDnsHostnames` y `enableDnsSupport` deben establecerse en `true`. Para ver y modificar estos atributos, vaya a la consola de la VPC en https://console.aws.amazon.com/vpc/.

Para obtener más información, consulte Utilización de DNS con su VPC.

> **Note**
>
> Si utiliza Route 53, confirme que la configuración no sustituye los atributos de red de DNS en la VPC.

Conexión a su gráfico de Amazon Neptune

Una vez que haya creado un clúster de base de datos de Neptune, el siguiente paso es configurar las formas en que desea conectarse a él.

Configuración de `curl` o awscurl para comunicarse con el punto de conexión de Neptune

Es muy práctico disponer de una herramienta de línea de comandos para enviar consultas al clúster de base de datos de Neptune, como se ilustra en muchos de los ejemplos de esta documentación. La herramienta de línea de comandos curl es una excelente opción para comunicarse con los puntos de conexión de Neptune cuando la autenticación de IAM no está habilitada. Las versiones a partir de la 7.75.0 admiten la opción `--aws-sigv4` para firmar las solicitudes cuando la autenticación de IAM está habilitada.

Para los puntos conexión en los que la autenticación de IAM está habilitada, también puede utilizar awscurl, que utiliza prácticamente la misma sintaxis que `curl`, pero admite la firma de solicitudes que sea necesaria para la autenticación de IAM. Debido a la seguridad adicional que proporciona la autenticación de IAM, suele ser una buena idea habilitarla.

Para obtener más información sobre cómo utilizar `curl` (o `awscurl`), consulte la página de curl man y el libro Everything curl.

Para conectarse mediante HTTPS (algo que Neptune requiere), `curl` necesita acceder a los certificados correspondientes. Siempre que `curl` pueda localizar los certificados adecuados, gestionará las conexiones HTTPS de la misma forma que las conexiones HTTP, sin parámetros adicionales. Lo mismo sucede para `awscurl`. Los ejemplos de esta documentación se basan en ese escenario.

Para saber cómo obtener dichos certificados y cómo formatearlos correctamente en un almacén de certificados de autoridad de certificación (CA) que `curl` pueda utilizar, consulte SSL Certificate Verification en la documentación de `curl`.

A continuación, puede especificar la ubicación de este almacén de certificados de CA mediante la variable de entorno `CURL_CA_BUNDLE`. En Windows, `curl` lo busca de forma automática en un archivo llamado `curl-ca-bundle.crt`. Primero busca en el mismo directorio que `curl.exe` y después en el resto de sitios de la ruta. Para obtener más información, consulte Certificados SSL.

Diferentes formas de conectarse a un clúster de base de datos de Neptune

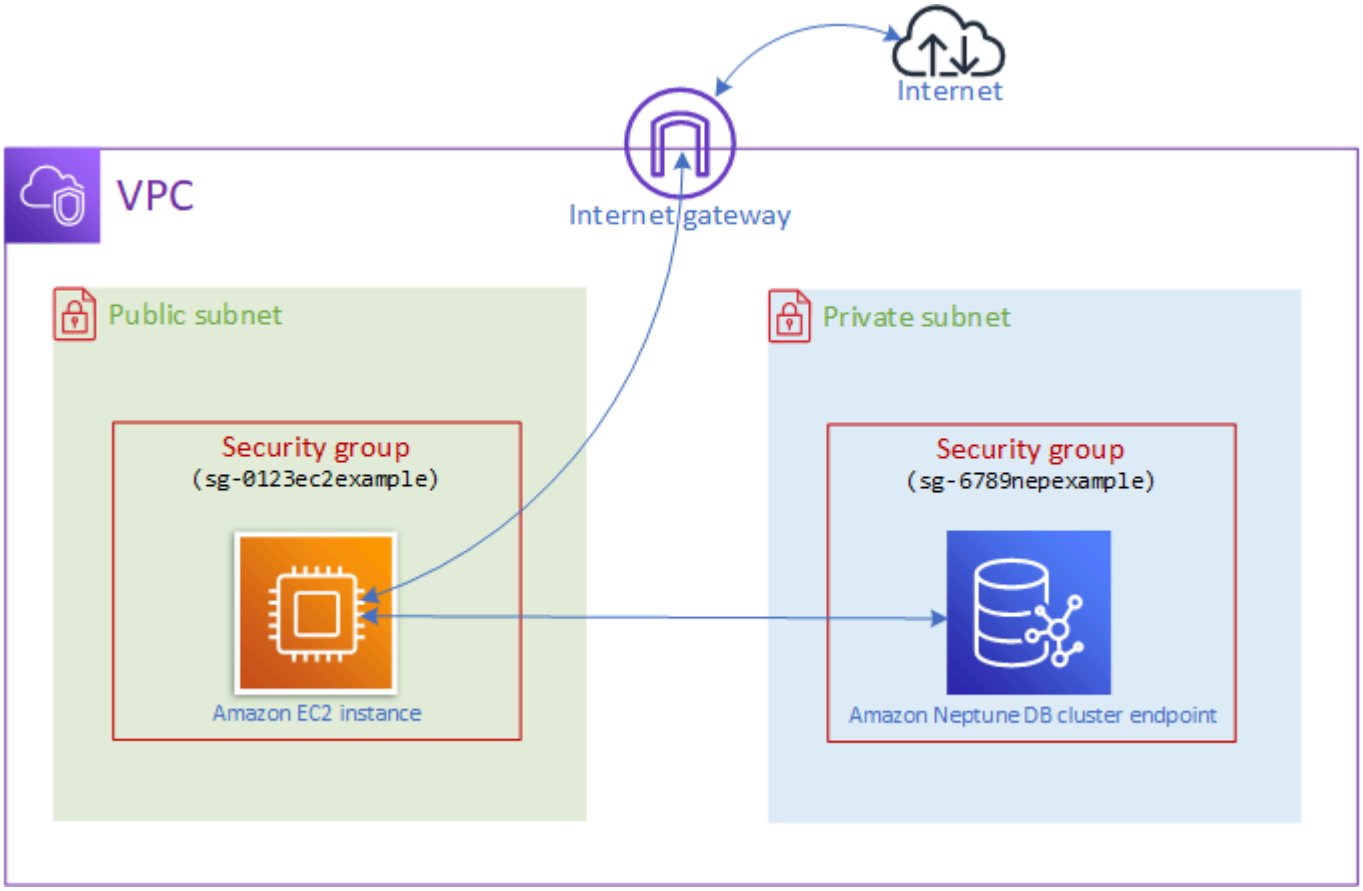
Un clúster de base de datos de Amazon Neptune solo se puede crear en una Amazon Virtual Private Cloud (Amazon VPC). A menos que habilite y configure los puntos de conexión públicos de Neptune para el clúster de base de datos, solo se puede acceder a sus puntos de enlace dentro de esa VPC.

Existen varias formas de configurar el acceso al clúster de base de datos de Neptune en su VPC:

- Conexión desde una instancia de Amazon EC2 en la misma VPC
- Conexión desde una instancia de Amazon EC2 en otra VPC
- Conexión desde una red privada

Conexión a un clúster de base de datos de Neptune desde una instancia de Amazon EC2 en la misma VPC

Una de las formas más comunes de conectarse a una base de datos de Neptune es desde una instancia de Amazon EC2 que está en la misma VPC que el clúster de base de datos de Neptune. Por ejemplo, la instancia EC2 podría ejecutar un servidor web que se conecta a Internet. En este caso, solo la instancia EC2 tiene acceso al clúster de base de datos de Neptune e Internet solo tiene acceso a la instancia EC2:



Para habilitar esta configuración, debe tener configurados los grupos de seguridad de VPC y los grupos de subredes correctos. El servidor web está alojado en una subred pública, para que pueda obtener acceso a la internet pública, y la instancia del clúster de Neptune está alojada en una subred privada para mantenerla segura. Consulte Configure la Amazon VPC en la que se encuentra el clúster de base de datos de Amazon Neptune.

Para que la instancia de Amazon EC2 se conecte a su punto de conexión de Neptune en, por ejemplo, el puerto 8182, deberá configurar un grupo de seguridad. Si su instancia de Amazon EC2 está utilizando un grupo de seguridad denominado, por ejemplo, `ec2-sg1`, debe crear otro grupo de seguridad de Amazon EC2 (supongamos, `db-sg1`) que tenga reglas de entrada para el puerto 8182 y tenga `ec2-sg1` como origen. A continuación, añada `db-sg1` a su clúster de Neptune para permitir la conexión.

Tras crear la instancia de Amazon EC2, puede iniciar sesión en ella mediante SSH y conectarse a su clúster de base de datos de Neptune. Para obtener información sobre cómo conectarse a una instancia EC2 mediante SSH, consulte Conectarse a su instancia de Linux en la Guía del usuario de Amazon EC2.

Si utiliza una línea de comandos de Linux o macOS para conectarse a la instancia EC2, puede pegar el comando SSH del elemento SSHAccess de la sección Salidas de la pila. AWS CloudFormation Es necesario que el archivo PEM esté en el directorio actual y que los permisos de dicho archivo estén establecidos en 400 (`chmod 400 keypair.pem`).

Para crear una VPC con subredes públicas y privadas

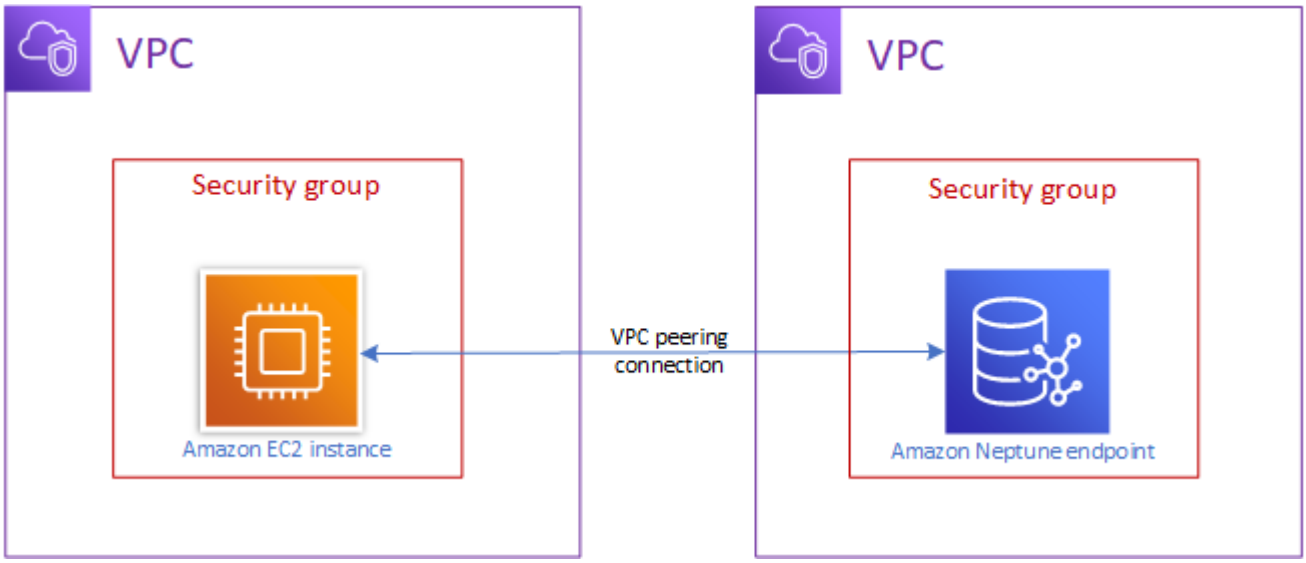
1. Inicie sesión en la consola de Amazon VPC AWS Management Console y ábrala en https://console.aws.amazon.com/vpc/.
2. En la esquina superior derecha de AWS Management Console, elige la región en la que quieres crear tu VPC.
3. En el Panel de VPC, elija Lanzar el asistente de VPC.
4. Complete el área de Configuración de VPC de la página Crear VPC:
    a. En Resources to create (Recursos que crear), elija VPC, subnets, etc. (VPC, subredes, etc.).
    b. Deje la etiqueta de nombre predeterminada como está, introduzca el nombre que desee o desactive la casilla de verificación Generar automáticamente para deshabilitar la generación de etiquetas de nombres.
    c. Deje el valor del bloque de CIDR de IPv4 en `10.0.0.0/16`.
    d. Deje el valor del bloque de CIDR IPv6 en Sin bloque CIDR de IPv6.
    e. Deje la Tenencia en Predeterminada.
    f. Deje el número de Zonas de disponibilidad (AZ) en 2.
    g. Deje Puertas de enlace NAT ($) en Ninguna, a menos que necesite una o más puertas de enlace NAT.
    h. Establezca Puertas de enlace de la VPC en Ninguno, a menos que vaya a utilizar Amazon S3.
    i. Debe marcar las opciones Habilitar nombres de host DNS y Habilitar la resolución de DNS.
5. Seleccione Crear VPC.

Acceso al clúster de base de datos desde una instancia de Amazon EC2 en otra VPC

Un clúster de base de datos de Amazon Neptune solo se puede crear en una Amazon Virtual Private Cloud (Amazon VPC) y solo se puede obtener acceso a sus puntos de conexión desde dicha VPC,

normalmente desde una instancia de Amazon Elastic Compute Cloud (Amazon EC2) que se ejecuta en dicha VPC.

Cuando su clúster de base de datos se encuentra en una VPC diferente de la instancia EC2 que está utilizando para acceder a él, puede utilizar el emparejamiento de VPC para realizar la conexión:



Una conexión de emparejamiento de VPC es una conexión de redes entre dos VPC que dirige el tráfico entre ellas de forma privada, para que las instancias de cualquier VPC puedan comunicarse como si estuviesen en la misma red. Puede crear una conexión de emparejamiento de VPC entre las VPC de su cuenta, entre una VPC de su cuenta AWS y una VPC de otra cuenta AWS , o con una VPC de una región diferente. AWS

AWS utiliza la infraestructura existente de una VPC para crear una conexión de emparejamiento de VPC. No es una puerta de enlace ni una conexión VPN AWS Site-to-Site, y no depende de un hardware físico independiente. No existen puntos de error de comunicaciones ni cuellos de botella de ancho de banda.

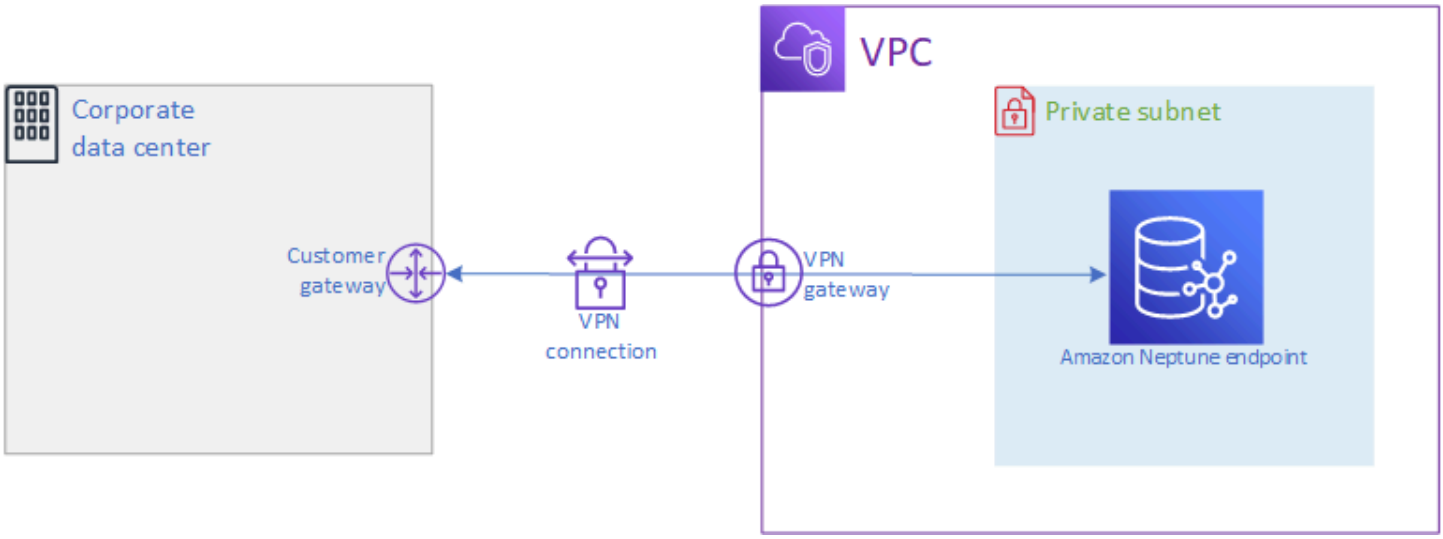
Para obtener más información acerca de los emparejamientos de VPC, consulte la Guía de interconexión de VPC de Amazon.

Acceso a su clúster de base de datos desde una red privada

Puede acceder a un clúster de base de datos de Neptune desde una red privada de dos maneras diferentes:

- Con una conexión de AWS Site-to-Site VPN.
- Con una conexión de AWS Direct Connect.

Los enlaces anteriores contienen información sobre estos métodos de conexión y sobre cómo configurarlos. La configuración de una conexión de AWS sitio a sitio podría tener el siguiente aspecto:



Seguridad de los datos en Amazon Neptune

Dispone de varias maneras de proteger sus clústeres de Amazon Neptune.

Uso de políticas de IAM para restringir el acceso a un clúster de base de datos de Neptune

Para controlar quién puede realizar las acciones de administración de Neptune en los clústeres y las instancias de base de datos de Neptune, utilice AWS Identity and Access Management (IAM).

Si utiliza una cuenta de IAM para acceder a la consola Neptuno, primero debe iniciar sesión en ella con su cuenta de IAM antes de abrir AWS Management Console la consola Neptuno en https://console.aws.amazon.com/neptune/home.

Cuando se conecta AWS mediante credenciales de IAM, su cuenta de IAM debe tener políticas de IAM que concedan los permisos necesarios para realizar las operaciones de administración de Neptune. Para obtener más información, consulte Uso de diferentes tipos de políticas de IAM para controlar el acceso a Neptune.

Uso de grupos de seguridad de VPC para restringir el acceso a un clúster de base de datos de Neptune

Los clústeres de base de datos de Neptune se deben crear en una Amazon Virtual Private Cloud (Amazon VPC). Para controlar qué dispositivos e instancias EC2 pueden abrir conexiones al punto de conexión y al puerto de la instancia de base de datos de los clústeres de base de datos de Neptune en una VPC, debe usar un grupo de seguridad de VPC. Para obtener más información acerca de VPC, consulte Cree un grupo de seguridad con la consola de VPC.

Uso de la autenticación de IAM para restringir el acceso a un clúster de base de datos de Neptune

Si habilita la autenticación AWS Identity and Access Management (IAM) en un clúster de base de datos de Neptune, cualquier persona que acceda al clúster de base de datos debe autenticarse primero. Para obtener información acerca de la configuración de la autenticación de IAM, consulte Descripción general de AWS Identity and Access Management (IAM) en Amazon Neptune.

Para obtener información sobre el uso de credenciales temporales para autenticarse, incluidos ejemplos para AWS CLI AWS Lambda, y Amazon EC2, consulte. the section called “Credenciales temporales”

Los siguientes enlaces proporcionan información adicional sobre la conexión a Neptune mediante la autenticación de IAM con los lenguajes de consulta individuales:

Uso de Gremlin con autenticación de IAM

- the section called “Consola de Gremlin”
- the section called “Java de Gremlin”
- the section called “Ejemplo de Python”

> **Note**
>
> Este ejemplo se aplica tanto a Gremlin como a SPARQL.

Uso de openCypher con autenticación de IAM

- the section called “Consola de Gremlin”

- the section called “Java de Gremlin”
- the section called “Ejemplo de Python”

> **Note**
>
> Este ejemplo se aplica tanto a Gremlin como a SPARQL.

Uso de SPARQL con autenticación de IAM

- the section called “SPARQL Java (RDF4J y Jena)”
- the section called “Ejemplo de Python”

> **Note**
>
> Este ejemplo se aplica tanto a Gremlin como a SPARQL.

Introducción al acceso al gráfico de Neptune

Una vez que haya creado un clúster de base de datos de Neptune y configurado la conexión a él, el siguiente paso es comunicarse con él para cargar datos, realizar consultas, etc. Para ello, la mayoría de las personas utilizan la `curl` o herramientas de la línea de comandos de `awscurl`.

Configuración de **`curl`** para comunicarse con el punto de conexión de Neptune

Tal y como se muestra en muchos de los ejemplos de esta documentación, la herramienta de línea de comandos curl es una opción muy útil para comunicarse con el punto de conexión de Neptune. Para obtener más información sobre la herramienta, consulte la página de gestión de curl y el libro Everything curl.

Para conectarse mediante HTTPS (tal y como se recomienda y como Neptune requiere en la mayoría de las regiones), `curl` debe tener acceso a los certificados adecuados. Para saber cómo obtener dichos certificados y cómo formatearlos correctamente en un almacén de certificados de autoridad de certificación (CA) que `curl` pueda utilizar, consulte SSL Certificate Verification en la documentación de `curl`.

A continuación, puede especificar la ubicación de este almacén de certificados de CA mediante la variable de entorno `CURL_CA_BUNDLE`. En Windows, `curl` lo busca de forma automática en un archivo llamado `curl-ca-bundle.crt`. Primero busca en el mismo directorio que `curl.exe` y después en el resto de sitios de la ruta. Para obtener más información, consulte Certificados SSL.

Siempre que `curl` pueda localizar los certificados adecuados, gestionará las conexiones HTTPS de la misma forma que las conexiones HTTP, sin parámetros adicionales. Los ejemplos de esta documentación se basan en ese escenario.

Uso de un lenguaje de consulta para acceder a datos de gráficos en su clúster de base de datos de Neptune

Una vez conectado, puede usar los lenguajes de consulta Gremlin y openCypher para crear y consultar un gráfico de propiedades, o el lenguaje de consulta SPARQL para crear y consultar un gráfico que contenga datos RDF.

Lenguajes de consulta de gráficos compatibles con Neptune

- Gremlin es un lenguaje de recorrido de gráficos para gráficos de propiedades. En Gremlin, una consulta es un recorrido compuesto por pasos discretos, cada uno de los cuales sigue un borde hasta un nodo. Consulte la documentación de Gremlin en Apache TinkerPop 3 para obtener más información.

  La implementación de Gremlin en Neptune tiene algunas diferencias con respecto a otras implementaciones, especialmente cuando se utiliza Gremlin-Groovy (consultas de Gremlin enviadas como texto serializado). Para obtener más información, consulte Conformidad con los estándares de Gremlin en Amazon Neptune.
- openCypher es un lenguaje de consulta declarativo para gráficos de propiedades que desarrolló originalmente Neo4j, luego de código abierto en 2015, y que contribuyó al proyecto openCypher en virtud de una licencia de código abierto Apache 2. Su sintaxis está documentada en Cypher Query Language Reference, versión 9.
- SPARQL es un lenguaje de consulta declarativo para datos RDF, que se basa en la coincidencia de patrones gráficos que se ajusta al estándar World Wide Web Consortium (W3C) y se describe en SPARQL 1.1 Overview y en la especificación de SPARQL 1.1 Query Language.

> **Note**
>
> Puede acceder a los datos de gráficos de propiedades de Neptune mediante Gremlin y openCypher, pero no con SPARQL. Del mismo modo, solo puede acceder a los datos RDF mediante SPARQL, no con Gremlin ni openCypher.

Uso de Gremlin para acceder al gráfico de Amazon Neptune

Puede usar la consola Gremlin para experimentar con TinkerPop gráficos y consultas en un entorno REPL (bucle). read-eval-print

En el siguiente tutorial, se describe el uso de la consola de Gremlin para añadir vértices, bordes, propiedades y mucho más a un gráfico de Neptune, y se destacan algunas diferencias en la implementación de Gremlin específica de Neptune.

> **Note**
>
> Este ejemplo asume que ha completado lo siguiente:
>
> - Se ha conectado a una instancia de Amazon EC2 mediante SSH.
> - Ha creado un clúster de Neptune tal y como se detalla en Creación de un clúster de base de datos.
> - Ha instalado la consola de Gremlin tal y como se describe enInstalación de la consola de Gremlin.

Uso de la consola de Gremlin

1. Cambie los directorios en la carpeta donde se descomprimen los archivos de la consola de Gremlin.

   ```
   cd apache-tinkerpop-gremlin-console-3.6.5
   ```

2. Escriba el comando siguiente para ejecutar la consola de Gremlin.

   ```
   bin/gremlin.sh
   ```

   Debería ver los siguientes datos de salida:

```
        \,,,/
        (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

Ahora se encuentra en `gremlin>`. Introduzca los pasos restantes en este punto.

3. En el símbolo del sistema `gremlin>`, escriba lo siguiente para conectarse a la instancia de base de datos de Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

4. En la entrada `gremlin>`, escriba lo siguiente para cambiar al modo remoto. Esto envía todas las consultas de Gremlin a la conexión remota.

```
:remote console
```

5. Añada un vértice con una etiqueta y una propiedad.

```
g.addV('person').property('name', 'justin')
```

Al vértice se le asigna un ID `string` que contiene un GUID. Todos los identificadores de vértices son cadenas en Neptune.

6. Añada un vértice con ID personalizado.

```
g.addV('person').property(id, '1').property('name', 'martin')
```

La propiedad `id` no se indica entre comillas. Se trata de una palabra clave para el ID del vértice. El ID del vértice indicado es una cadena con el número 1.

Los nombres de propiedades normales deben incluirse entre comillas.

7. Cambie la propiedad o añada una si no existe.

```
g.V('1').property(single, 'name', 'marko')
```

En este ejemplo, se cambia la propiedad `name` para el vértice del paso anterior. Esto elimina todos los valores existentes de la propiedad `name`.

Si no especificó `single`, en su lugar añade el valor a la propiedad `name` si aún no lo ha hecho.

8. Añada la propiedad, pero anéxela si ya tiene un valor.

```
g.V('1').property('age', 29)
```

Neptune utiliza la cardinalidad en conjuntos como acción predeterminada.

Este comando añade la propiedad `age` con el valor `29`, pero no reemplaza los valores existentes.

Si la propiedad `age` ya tenía un valor, este comando anexa `29` a la propiedad. Por ejemplo, si la propiedad `age` era `27`, el nuevo valor será `[ 27, 29 ]`.

9. Añada varios vértices.

```
g.addV('person').property(id, '2').property('name', 'vadas').property('age',
 27).iterate()
g.addV('software').property(id, '3').property('name', 'lop').property('lang',
 'java').iterate()
g.addV('person').property(id, '4').property('name', 'josh').property('age',
 32).iterate()
g.addV('software').property(id, '5').property('name', 'ripple').property('lang',
 'java').iterate()
g.addV('person').property(id, '6').property('name', 'peter').property('age', 35)
```

Puede enviar varias instrucciones a Neptune al mismo tiempo.

Las instrucciones se pueden separar mediante una nueva línea (`'\n'`), espacios (`' '`), punto y coma (`'; '`) o nada (por ejemplo: `g.addV('person').iterate()g.V()` es válido).

> **Note**
>
> La consola de Gremlin envía un comando independiente en cada nueva línea (`'\n'`), por lo que cada uno es una transacción independiente en ese caso. Este ejemplo tiene todos los comandos en líneas separadas para facilitar su lectura. Elimine los caracteres

de nueva línea (`'\n'`) para enviarlo como un único comando a través de la consola de Gremlin.

Todas las instrucciones a excepción de la última deben finalizar en un paso de terminación, como `.next()` o `.iterate()`, o no se ejecutarán. La consola de Gremlin no requiere estos pasos de terminación. Use `.iterate` siempre que no necesite que los resultados se serialicen.

Todas las instrucciones que se envían juntas se incluyen en una única transacción y se realizan correctamente o producen un error de forma conjunta.

10. Añada bordes.

```
g.V('1').addE('knows').to(__.V('2')).property('weight', 0.5).iterate()
g.addE('knows').from(__.V('1')).to(__.V('4')).property('weight', 1.0)
```

A continuación, se muestran dos formas distintas de añadir un borde.

11. Añada el resto del gráfico moderno.

```
g.V('1').addE('created').to(__.V('3')).property('weight', 0.4).iterate()
g.V('4').addE('created').to(__.V('5')).property('weight', 1.0).iterate()
g.V('4').addE('knows').to(__.V('3')).property('weight', 0.4).iterate()
g.V('6').addE('created').to(__.V('3')).property('weight', 0.2)
```

12. Elimine un vértice.

```
g.V().has('name', 'justin').drop()
```

Elimina el vértice con el valor de la propiedad `name` igual a `justin`.

> **Important**
>
> Deténgase aquí y tendrá el gráfico completo de Apache TinkerPop Modern. Los ejemplos de la sección transversal de la TinkerPop documentación utilizan el gráfico moderno.

13. Ejecute un recorrido.

```
g.V().hasLabel('person')
```

Devuelve todos los vértices `person`.

14. Ejecute un recorrido con valores (valueMap()).

```
g.V().has('name', 'marko').out('knows').valueMap()
```

Devuelve pares clave-valor para todos los vértices que `marko` "conoce".

15. Especifique varias etiquetas.

```
g.addV("Label1::Label2::Label3")
```

Neptune admite varias etiquetas para un vértice. Al crear una etiqueta, puede especificar varias si las separa mediante `::`.

Este ejemplo añade un vértice con tres etiquetas distintas.

El paso `hasLabel` busca coincidencias de este vértice con cualquiera de esas tres etiquetas: `hasLabel("Label1")` `hasLabel("Label2")` y `hasLabel("Label3")`.

El delimitador `::` está reservado solo para este uso.

No se pueden especificar varias etiquetas en el paso `hasLabel`. Por ejemplo, `hasLabel("Label1::Label2")` no tiene ninguna coincidencia.

16. Especifique la hora o la fecha.

```
g.V().property(single, 'lastUpdate', datetime('2018-01-01T00:00:00'))
```

Neptune no admite la clase Date de Java. Use la función `datetime()` en su lugar. La función `datetime()` toma un valor de cadena de `datetime` compatible con ISO8061.

Admite los siguientes formatos: `YYYY-MM-DD`, `YYYY-MM-DDTHH:mm`, `YYYY-MM-DDTHH:mm:SS` y `YYYY-MM-DDTHH:mm:SSZ`.

17. Elimine vértices, propiedades o bordes.

```
g.V().hasLabel('person').properties('age').drop().iterate()
g.V('1').drop().iterate()
g.V().outE().hasLabel('created').drop()
```

Estos son algunos ejemplos.

> **Note**
>
> El paso `.next()` no funciona con `.drop()`. En su lugar, use `.iterate()`.

18. Cuando haya terminado, escriba lo siguiente para salir de la consola de Gremlin.

```
:exit
```

> **Note**
>
> Utilice punto y coma (`;`) o un carácter de nueva línea (`\n`) para separar las instrucciones. Cada recorrido anterior al final debe terminar en `iterate()` para ejecutarse. Solo se devuelven los datos del recorrido final.

Uso de openCypher para acceder al gráfico de Amazon Neptune

Para empezar a utilizar OpenCypheropenCypher, consulte o utilice los cuadernos OpenCypher del repositorio de cuadernos gráficos de Neptune. GitHub

Uso de RDF y SPARQL para acceder al gráfico en Amazon Neptune

SPARQL es un lenguaje de consulta para el marco de descripción de recursos (RDF), que es un formato de datos de gráficos diseñado para la web. Amazon Neptune es compatible con SPARQL 1.1. Esto significa que puede conectarse a una instancia de base de datos de Neptune y consultar el gráfico utilizando el lenguaje de consulta descrito en la especificación de SPARQL 1.1 Query Language.

Una consulta en SPARQL se compone de una cláusula `SELECT` para especificar las variables que se devolverán y una cláusula `WHERE` para especificar los datos del gráfico que deben corresponderse. Si no está familiarizado con las consultas SPARQL, consulte la sección sobre escritura de consultas sencillas en la documentación del lenguaje de consulta SPARQL 1.1.

El punto de conexión HTTP para las consultas de SPARQL a una instancia de base de datos de Neptune es `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

Para conectarse a SPARQL

1. Puede obtener el punto final de SPARQL para su clúster de Neptune desde SparqlEndpointel elemento de la sección Salidas de AWS CloudFormation la pila.
2. Escriba lo siguiente para enviar un comando **`UPDATE`** de SPARQL a través de HTTP `POST` y el comando curl.

   ```
   curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
   test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
   ```

   El ejemplo anterior inserta el siguiente triple en el gráfico predeterminado de SPARQL: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`
3. Escriba lo siguiente para enviar un comando **`QUERY`** de SPARQL a través de HTTP `POST` y el comando curl.

   ```
   curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
    https://your-neptune-endpoint:port/sparql
   ```

   El ejemplo anterior devuelve hasta 10 de los triples (sujeto-predicado-objeto) del gráfico utilizando la consulta `?s ?p ?o` con un límite de 10. Para otras consultas, sustitúyalo por otra consulta SPARQL.

   > **Note**
   >
   > El tipo MIME predeterminado de una respuesta es `application/sparql-results+json` para las consultas `SELECT` y `ASK`.
   > El tipo MIME predeterminado de una respuesta es `application/n-quads` para las consultas `CONSTRUCT` y `DESCRIBE`.
   > Para ver la lista de tipos MIME disponibles, consulte API HTTP de SPARQL.

Carga de datos en Neptune

Amazon Neptune ofrece un proceso para cargar datos de archivos externos directamente en una instancia de base de datos de Neptune. Puede utilizar este proceso en lugar de ejecutar un gran número de instrucciones `INSERT`, pasos `addV` y `addE` u otras llamadas a la API.

Los siguientes son enlaces a información de carga adicional.

- Métodos para cargar datos: Carga de datos
- Formatos de datos compatibles con el programa de carga masiva: the section called “Formatos de los datos”
- Ejemplo de carga: the section called “Ejemplo de carga”

Monitorización de Amazon Neptune

Amazon Neptune admite los siguientes métodos de monitorización.

- Amazon CloudWatch: Amazon Neptune envía automáticamente las métricas a las alarmas CloudWatch y también las admite CloudWatch . Para obtener más información, consulte the section called “Usando CloudWatch”.
- AWS CloudTrail— Amazon Neptune admite el registro de API mediante. CloudTrail Para obtener más información, consulte the section called “Registro de llamadas a la API de Neptune con AWS CloudTrail”.
- Etiquetado: utilice etiquetas para añadir metadatos a sus recursos de Neptune y realizar un seguimiento del uso en función de las etiquetas. Para obtener más información, consulte the section called “Etiquetado de recursos de Neptune”.
- Archivos de registro de auditoría: consulte, descargue o vea los archivos de registro de base de datos con la consola de Neptune. Para obtener más información, consulte the section called “Registros de auditoría con Neptune”.
- Estado de la instancia: compruebe el estado del motor de base de datos de gráficos de una instancia de Neptune, averigüe qué versión del motor está instalada y obtenga otra información sobre el estado del motor mediante la API de estado de la instancia.

Solución de problemas y prácticas recomendadas en Neptune

Los siguientes enlaces pueden ser útiles para resolver problemas con Amazon Neptune.

- Prácticas recomendadas: para obtener soluciones a problemas comunes y sugerencias de rendimiento, consulte Prácticas recomendadas.
- Errores de servicio: para obtener una lista de errores tanto para las API de administración como para las conexiones de bases de datos de gráficos, consulte Errores de Neptune.

- Límites de servicio: para obtener información acerca de los límites de servicio de Neptune, consulte Límites de Neptune.
- Versiones de motores: para obtener información sobre las versiones de los motores de gráficos, incluidas las notas de la versión, consulte Versiones del motor.
- Foros de soporte: para unirse a un debate sobre Neptune, consulte el foro de Amazon Neptune.
- Precios: para obtener información sobre los costos de usar Amazon Neptune, consulte los precios de Amazon Neptune.
- AWS Support: para obtener ayuda y orientación de los expertos, consulte AWS Support.

Creación de una base de datos global de Neptune

Una base de datos Amazon Neptune Global Database abarca varias Regiones de AWS, lo que permite lecturas globales de baja latencia y proporciona una recuperación rápida en el caso excepcional de que se produzca una interrupción que afecte a toda una Región de AWS.

Una base de datos global de Neptune tiene un clúster de base de datos principal en una región y hasta cinco clústeres de base de datos secundarios en diferentes regiones.

Las escrituras solo se pueden realizar en la región principal. Las regiones secundarias solo admiten lecturas. Cada región secundaria puede tener hasta 16 instancias de lector.

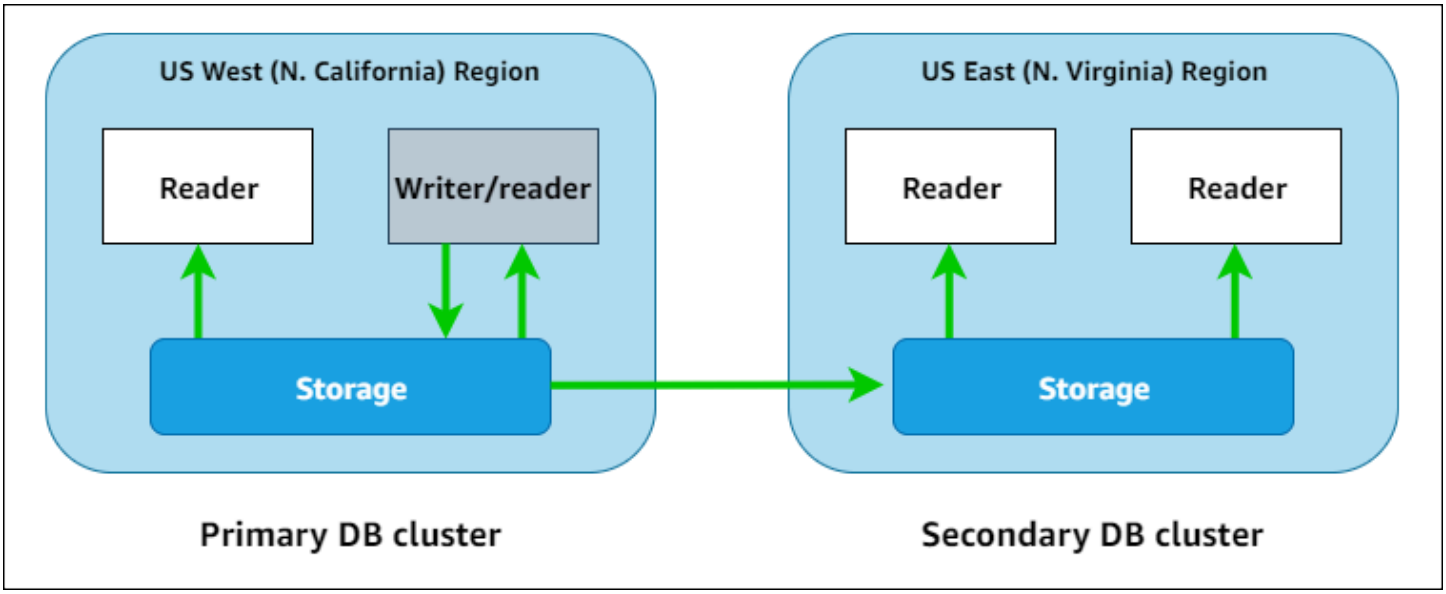
Temas

Información general sobre las bases de datos globales de Amazon Neptune

Mediante una base de datos global de Neptune, puede ejecutar sus aplicaciones distribuidas globalmente con una única base de datos que abarque varias Regiones de AWS.

Una base de datos global de Neptune tiene un clúster de base de datos en una Región de AWS principal en la que se pueden escribir datos y hasta cinco clústeres de base de datos de solo lectura en Regiones de AWS secundarias. Cuando realiza una operación de escritura en el clúster de base de datos principal, Neptune replica los datos escritos en todos los clústeres de base de datos secundarios con una infraestructura dedicada y con una latencia que suele ser inferior a un segundo.

En el siguiente diagrama se muestra un ejemplo de base de datos global que abarca dos Regiones de AWS:



Puede escalar cada clúster secundario de forma independiente para gestionar cargas de trabajo de solo lectura. Para ello, añada una o varias instancias de réplica de lectura.

Para realizar operaciones de escritura, debe conectar el punto de conexión del clúster de base de datos del clúster de base de datos principal. Solo el clúster principal realiza operaciones de escritura. A continuación, tal y como se muestra en el diagrama anterior, la replicación la realiza el volumen de almacenamiento del clúster, no el motor de la base de datos.

Las bases de datos globales de Neptune están diseñadas para aplicaciones con una huella mundial. Los clústeres de base de datos secundarios de solo lectura admiten operaciones de lectura más cercanas a los usuarios de la aplicación.

Una base de datos global de Neptune admite dos enfoques diferentes para la conmutación por error:

- Para recuperarse de una interrupción en la región principal, utilice el proceso de desconexión y promoción manual no planificado, en el que se desconecta uno de los clústeres secundarios, se convierte en un clúster independiente y, a continuación, se convierte en el nuevo clúster principal.
- Para los procedimientos operativos planificados, como el mantenimiento, utilice la conmutación por error planificada administrada, donde reubica el clúster principal en una de sus regiones secundarias sin que se pierdan los datos.

Ventajas de usar bases de datos globales de Amazon Neptune

Mediante el uso de una base de datos global, puede obtener las siguientes ventajas:

- Lecturas globales con latencia local: si tiene oficinas en todo el mundo, una base de datos global le permite a las oficinas de regiones secundarias acceder a los datos de su propia región con latencia local.
- Clústeres de bases de datos de Neptune secundarios y escalables: añada instancias de base de datos de réplica de lectura para escalar clústeres secundarios. Dado que los clústeres secundarios son de solo lectura, cada uno puede admitir hasta 16 réplicas de lectura en lugar del límite habitual de 15.
- Replicación rápida en clústeres de base de datos secundarios: la replicación de clústeres de base de datos principales a secundarios es rápida, con una latencia que suele ser inferior a un segundo y con poco impacto en el rendimiento del clúster de base de datos principal. Dado que la replicación se realiza en el nivel de almacenamiento, los recursos de la instancia de base de datos están totalmente disponibles para las cargas de trabajo de lectura y escritura de las aplicaciones.
- Recuperación de interrupciones en toda la región: los clústeres de bases de datos secundarios le permiten trasladar el clúster principal a una nueva región de forma más rápida con menor RTO y menos pérdida de datos (menor RPO) que las soluciones de replicación tradicionales.

Limitaciones de las bases de datos globales de Amazon Neptune

Las limitaciones siguientes se aplican actualmente a las bases de datos globales de :

- Las bases de datos globales de Neptune están disponibles en las siguientes Regiones de AWS:
  - Este de EE. UU. (Norte de Virginia): `us-east-1`
  - Este de EE. UU. (Ohio): `us-east-2`
  - Oeste de EE. UU. (Norte de California): `us-west-1`
  - Oeste de EE. UU. (Oregón): `us-west-2`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londres): `eu-west-2`
  - Asia-Pacífico (Tokio): `ap-northeast-1`
- Las bases de datos globales de Neptune actualmente no admiten el escalado automático para clústeres de bases de datos secundarios.
- No puede aplicar un grupo de parámetros personalizado al clúster de base de datos global mientras realiza una actualización importante de la versión de esa base de datos global. En su lugar, cree grupos de parámetros personalizados en cada región del clúster global y luego aplíquelos manualmente a los clústeres regionales después de la actualización.

- No puede detener ni iniciar de forma individual los clústeres de base de datos en una base de datos global.
- Las instancias de réplica de lectura de un clúster de base de datos secundario pueden reiniciarse en determinadas circunstancias. Si la instancia de escritor del clúster principal se reinicia o se conmuta por error, también se reiniciarán todas las instancias de las regiones secundarias. El clúster secundario no estará disponible hasta que todas las instancias vuelvan a estar sincronizadas con la instancia de escritor del clúster de base de datos principal.

Configuración de una base de datos global de Amazon Neptune

Puede crear una base de datos global de Neptune de una de las siguientes maneras:

- Cree una base de datos global con todos los clústeres e instancias de bases de datos nuevos.
- Cree una base de datos global con un clúster de base de datos de Neptune existente como clúster principal.

Temas

Requisitos de configuración para una base de datos global de Amazon Neptune

Una base de datos global de Neptune abarca al menos dos Regiones de AWS. La Región de AWS principal cuenta con un clúster de base de datos de Neptune que tiene una instancia de escritor. De una a cinco Regiones de AWS secundarias; cada una cuenta con un clúster de base de datos de Neptune de solo lectura compuesto por instancias de réplica de lectura. Se necesita cómo mínimo una Región de AWS secundaria.

Los clústeres de base de datos de Neptune que componen una base de datos global tienen los siguientes requisitos específicos:

- Requisitos de clase de instancia de base de datos: una base de datos global requiere clases de instancia de base de datos `r5` o `r6g` optimizadas para cargas de trabajo con uso intensivo de memoria, como, por ejemplo, un tipo de instancia `db.r5.large`.
- Requisitos de Región de AWS: una base de datos global necesita un clúster de base de datos de Neptune principal en una Región de AWS y al menos un clúster de base de datos de Neptune secundario en otra región. Puede crear hasta cinco clústeres de base de datos de Neptune secundarios (de solo lectura) y cada uno debe estar en una región diferente. Es decir, no hay dos clústeres de base de datos de Neptune en una base de datos global de Neptune en la misma Región de AWS.
- Requisitos de la versión del motor: la versión del motor de Neptune utilizada por todos los clústeres de base de datos de la base de datos global debe ser la misma y debe ser `1.2.0.0` o posterior. Si no especifica la versión del motor al crear una nueva base de datos global, clúster o instancia, se utilizará la versión del motor más reciente.

> **Important**
>
> Si bien los grupos de parámetros del clúster de base de datos se pueden configurar de forma independiente para cada clúster de base de datos de una base de datos global, se recomienda mantener la misma configuración en todos los clústeres para evitar cambios de comportamiento inesperados en caso de que un clúster secundario se convierta en principal. Por ejemplo, utilice la misma configuración de índices de objetos, transmisiones, etc. en todos los clústeres de base de datos.

Uso de la AWS CLI para crear una base de datos global de Amazon Neptune

> **Note**
>
> Los ejemplos de esta sección siguen la convención de UNIX de utilizar una barra diagonal inversa (\) como carácter de extensión de línea. En Windows, sustituya la barra diagonal inversa por un signo de intercalación (^).

Para crear una base de datos global con la AWS CLI

1. Comience por crear una base de datos global vacía con el comando `create-global-cluster` de la AWS CLI (que incluye la API de CreateGlobalCluster). Especifique el nombre de la Región de AWS que desee que sea principal, establezca Neptune como motor de base de datos y, si lo desea, especifique la versión del motor que desee utilizar (debe ser la versión 1.2.0.0 o posterior):

```
aws neptune create-global-cluster
  --region (primary region, such as us-east-1) \
  --global-cluster-identifier (ID for the global database) \
  --engine neptune \
  --engine-version (engine version; this is optional)
```

2. La base de datos global puede tardar unos minutos en estar disponible, así que antes de continuar con el siguiente paso, utilice el comando `describe-global-clusters` de la CLI (que incluye la API de DescribeGlobalClusters) para comprobar que la base de datos global esté disponible:

```
aws neptune describe-global-clusters \
  --region (primary region) \
  --global-cluster-identifier (global database ID)
```

3. Cuando la base de datos global de Neptune esté disponible, puede crear un nuevo clúster de base de datos de Neptune para que sea el clúster principal:

```
aws neptune create-db-cluster \
  --region (primary region) \
  --db-cluster-identifier (ID for the primary DB cluster) \
  --engine neptune \
  --engine-version (engine version; must be >= 1.2.0.0) \
  --global-cluster-identifier (global database ID)
```

4. Utilice el comando `describe-db-clusters` de la AWS CLI para confirmar que el nuevo clúster de base de datos esté listo para añadir su instancia de base de datos principal:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

Cuando la respuesta indique `"Status": "available"`, continúe con el siguiente paso.

5. Cree la instancia de base de datos principal para el clúster principal con el comando `create-db-instance` de la AWS CLI. Debe usar uno de los tipos de instancia `r5` o `r6g` optimizados para la memoria, como, por ejemplo, `db.r5.large`.

```
aws neptune create-db-instance \
  --region (primary region) \
  --db-cluster-identifier (primary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Si tiene previsto añadir datos al nuevo clúster de base de datos principal mediante el programa de carga masiva Neptune, hágalo antes de añadir regiones secundarias. Este proceso es más rápido y rentable que realizar una carga masiva una vez que la base de datos global esté completamente configurada.

Ahora añada una o varias regiones secundarias a la nueva base de datos global, tal y como se describe en Añadir una región secundaria con la AWS CLI.

Convertir un clúster de base de datos existente en una base de datos global

Para convertir un clúster de base de datos existente en una base de datos global, utilice el comando `create-global-cluster` de la AWS CLI para crear una nueva base de datos global en la misma Región de AWS en la que se encuentra el clúster de base de datos existente y establezca su parámetro `--source-db-cluster-identifier` en Nombre de recurso de Amazon (ARN) del clúster existente ubicado allí:

```
aws neptune create-global-cluster \
  --region (region where the existing cluster is located) \
  --global-cluster-identifier (provide an ID for the new global database) \
  --source-db-cluster-identifier (the ARN of the existing DB cluster) \
  --engine neptune \
```

```
  --engine-version (engine version; this is optional)
```

Ahora añada una o varias regiones secundarias a la nueva base de datos global, tal y como se describe en Añadir una región secundaria con la AWS CLI.

Uso de un clúster de base de datos restaurado a partir de una instantánea como clúster principal

Puede convertir un clúster de base de datos restaurado a partir de una instantánea en una base de datos global de Neptune. Una vez finalizada la restauración, convierta el clúster de base de datos que ha creado en el clúster principal de una nueva base de datos global, tal y como se ha descrito anteriormente.

Añadir regiones de bases de datos globales secundarias a una región principal en Amazon Neptune

Una base de datos global de Neptune necesita al menos un clúster de base de datos de Neptune secundario en una Región de AWS diferente a la del clúster de base de datos principal. Puede adjuntar hasta cinco clústeres de base de datos secundarios al clúster de base de datos principal.

Cada clúster de base de datos secundario que añada reduce en uno el número máximo de instancias de réplica de lectura que puede tener en el clúster principal. Por ejemplo, si hay cuatro clústeres secundarios, el número máximo de instancias de réplica de lectura que puede tener en el clúster principal es `15 - 4 = 11`. Esto significa que si tiene 14 instancias de lector en el clúster de base de datos principal y un clúster secundario, no podrá añadir otro clúster secundario.

Uso de AWS CLI para añadir una región secundaria a una base de datos global de Neptune

Para añadir una Región de AWS secundaria a la base de datos global de Neptune con la AWS CLI

1. Utilice el comando `create-db-cluster` de la AWS CLI para crear un nuevo clúster de base de datos en una región diferente a la del clúster principal y establezca su parámetro `--global-cluster-identifier` para especificar el identificador de la base de datos global:

   ```
   aws neptune create-db-cluster \
     --region (the secondary region) \
     --db-cluster-identifier (ID for the new secondary DB cluster) \
   ```

```
  --global-cluster-identifier (global database ID)
  --engine neptune \
  --engine-version (optional: engine version)
```

2. Utilice el comando describe-db-clusters de la AWS CLI para confirmar que el nuevo clúster de base de datos esté listo para añadir su instancia de base de datos principal:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Cuando la respuesta indique `"Status": "available"`, continúe con el siguiente paso.

3. Cree la instancia de base de datos principal para el clúster principal mediante el comando create-db-instance de la AWS CLI con un tipo de instancia de la clase de instancia `r5` o `r6g`:

```
aws neptune create-db-instance \
  --region (secondary region) \
  --db-cluster-identifier (secondary cluster ID) \
  --db-instance-class (instance class) \
  --db-instance-identifier (ID for the DB instance) \
  --engine neptune \
  --engine-version (optional: engine version)
```

> **Note**
>
> Si no tiene previsto atender a un gran número de solicitudes de lectura en la región secundaria, y lo que más le preocupa es mantener copias de seguridad de sus datos de forma fiable en esa región, puede crear un clúster de base de datos secundario con instancias sin bases de datos. Esto le permite ahorrar dinero, ya que entonces solo paga por el almacenamiento del clúster secundario, que Neptune mantendrá sincronizado con el almacenamiento del clúster de base de datos principal.

Conexión a una base de datos global de Neptune

La forma en que se conecta a una base de datos global de Neptune depende de si necesita escribir en ella o leer de ella:

- Para solicitudes o consultas de solo lectura, conéctese al punto de conexión del lector del clúster de Neptune en su Región de AWS.
- Para ejecutar consultas de mutación, conéctese al punto de conexión del clúster de base de datos principal, que puede estar en una Región de AWS diferente al de la aplicación.

Administración de una base de datos Amazon Neptune Global Database

A excepción del proceso de conmutación por error planificada administrada, realiza la mayor parte de las operaciones de administración en los clústeres individuales que componen una base de datos global de Neptune. El proceso de conmutación por error planificada administrada solo está disponible para bases de datos globales de Neptune, no para clústeres de base de datos individuales de Neptune. Para obtener más información, consulte Ejecución de la conmutación por error planificada administrada para bases de datos globales de Neptune.

Para recuperar una base de datos global de Neptune de una interrupción no planificada en la región principal, consulte Desconexión y promoción de una base de datos global de Neptune en caso de que se produzca una interrupción imprevista.

Si bien puede configurar los grupos de parámetros del clúster de base de datos de forma independiente para cada clúster de Neptune de una base de datos global, se recomienda mantener la misma configuración en todos los clústeres para evitar cambios de comportamiento inesperados en caso de que un clúster secundario se convierta en principal. Por ejemplo, utilice la misma configuración de índices de objetos, transmisiones, etc. en todos los clústeres de base de datos.

Eliminación de un clúster de una base de datos global de Neptune

Hay varios motivos por los que desee eliminar un clúster de base de datos de una base de datos global. Por ejemplo:

- Si el clúster principal se degrada o se aísla, puede eliminarlo de la base de datos global, y se convierte en un clúster aprovisionado independiente que se puede utilizar para crear una nueva base de datos global (consulte Desconexión y promoción de una base de datos global de Neptune en caso de que se produzca una interrupción imprevista).
- Si desea eliminar una base de datos global, primero tiene que eliminar (desconectar) todos los clústeres asociados y dejar el principal para lo último (consulte Eliminación de una base de datos global de Neptune).

Puede usar el comando remove-from-global-cluster de la CLI (que incluye la API de RemoveFromGlobalCluster) para desconectar un clúster de base de datos de Neptune de una base de datos global:

```
aws neptune remove-from-global-cluster \
  --region (region of the cluster to remove) \
  --global-cluster-identifier (global database ID) \
  --db-cluster-identifier (ARN of the cluster to remove)
```

El clúster de base de datos desconectado se convierte en un clúster de base de datos independiente.

Eliminación de una base de datos global de Neptune

No puede eliminar una base de datos global y los clústeres asociados en un único paso. Por tanto, debe eliminar sus componentes uno por uno:

1. Desconecte todos los clústeres de base de datos secundarios de la base de datos global, tal y como se describe en Eliminación de un clúster. Si lo desea, ya puede eliminarlos de forma individual.
2. Elimine el clúster de base de datos principal de la base de datos global.
3. Elimine todas las instancias de base de datos de réplica de lectura del clúster principal.
4. Elimine la instancia de base de datos principal (escritor) del clúster principal. Si lo hace en la consola, también se eliminará el clúster de base de datos.
5. Elimine la base de datos global. Para ello, utilice la AWS CLI; utilice el comando delete-global-cluster de la CLI (que incluye la API de DeleteGlobalCluster) de la siguiente manera:

   ```
   aws neptune delete-global-cluster \
     --region (region of the DB cluster to delete) \
     --global-cluster-identifier (global database ID)
   ```

Modificación de una base de datos global de Neptune

Los grupos de parámetros del clúster de base de datos se pueden configurar de forma independiente para cada clúster de base de datos de Neptune de una base de datos global, pero se recomienda mantener la misma configuración en todos los clústeres para evitar cambios de comportamiento inesperados en caso de que un clúster secundario se convierta en principal.

Puede modificar la configuración de la propia base de datos global mediante el comando modify-global-cluster de la CLI (que incluye la API de ModifyGlobalCluster). Por ejemplo, puede cambiar el identificador de la base de datos global y, al mismo tiempo, desactivar la protección contra la eliminación de la siguiente manera:

```
aws neptune modify-global-cluster \
  --region (region of the DB cluster to modify) \
  --global-cluster-identifier (current global database ID) \
  --new-global-cluster-identifier (new global database ID to assign) \
  --deletion-protection false
```

Uso de la conmutación por error en una base de datos global de Neptune

Una base de datos global de Neptune proporciona capacidades de conmutación por error más completas que un clúster de base de datos de Neptune independiente. Al utilizar una base de datos global, puede planificar y recuperarse de desastres con bastante rapidez. Por lo general, la recuperación de desastres se valora mediante la evaluación del objetivo de tiempo de recuperación (RTO) y el objetivo de punto de recuperación (RPO):

- Objetivo de tiempo de recuperación (RTO): la rapidez con la que un sistema vuelve a un estado operativo después de un desastre. En otras palabras, el RTO mide el tiempo de inactividad. Para una base de datos global de Neptune, el RTO puede estar en el orden de minutos.
- Objetivo de punto de recuperación (RPO): cantidad de tiempo durante la cual se pierden los datos. Para una base de datos global de Neptune, el RPO se mide normalmente en segundos (consulte Ejecución de la conmutación por error planificada administrada para bases de datos globales de Neptune).

Para una base de datos global de Neptune, hay dos enfoques diferentes para la conmutación por error:

- Desconexión y promoción (recuperación manual no planificada): para recuperarse de una interrupción no planificada o para llevar a cabo pruebas de recuperación de desastres (pruebas de DR), realice una desconexión y promoción entre regiones en uno de los clústeres de base de datos secundarios de la base de datos global. El RTO para este proceso manual depende de la rapidez con la que pueda realizar las tareas enumeradas en Desconexión y promoción. Por lo

general, el RPO es un número en segundos, pero esto depende del retraso de replicación del almacenamiento en la red en el momento en el que se produce el error.

- Conmutación por error planificada administrada: este enfoque está diseñado para el mantenimiento operativo y otros procedimientos operativos planificados, como la reubicación del clúster de base de datos principal de la base de datos global en una de las regiones secundarias. Dado que este proceso sincroniza los clústeres secundarios de base de datos con el principal antes de realizar cualquier otro cambio, el RPO es efectivamente 0 (es decir, sin pérdida de datos). Consulte Ejecución de la conmutación por error planificada administrada para bases de datos globales de Neptune.

Desconexión y promoción de una base de datos global de Neptune en caso de que se produzca una interrupción imprevista

En la improbable situación en la que la base de datos global de Neptune experimente una interrupción inesperada en su Región de AWS principal, el cluster de base de datos principal de Neptune y su nodo de escritor dejarán de estar disponibles, y se detendrá la replicación entre el clúster principal y los secundarios. Para reducir al mínimo el tiempo de inactividad (RTO) y la pérdida de datos (RPO), realice rápidamente una desconexión y promoción entre regiones para reconstruir la base de datos global.

> **Tip**
>
> Es buena idea entender este proceso antes de usarlo y tener un plan para proceder rápidamente ante el primer indicio de que se produzca un problema en toda la región.
>
> - Utilice periódicamente Amazon CloudWatch para realizar un seguimiento de los tiempos de retraso de los clústeres secundarios para que pueda identificar la región secundaria con el menor tiempo de retraso en caso de necesitar una conmutación por error.
> - Asegúrese de probar el plan para verificar que sus procedimientos sean completos y precisos.
> - Utilice un entorno simulado para asegurarse de que el personal esté capacitado y preparado para realizar una conmutación por error de recuperación de desastres rápidamente en caso de que sea necesaria.

Para conmutar por error a un clúster secundario después de una interrupción no planificada en la región principal

1. Deje de realizar consultas de mutación y otras operaciones de escritura en el clúster de base de datos principal.
2. Identifique un clúster de base de datos en una Región de AWS secundaria para usarlo como el nuevo clúster principal de la base de datos global. Si la base de datos global tiene dos o varias Regiones de AWS secundarias, elija el clúster secundario que tenga el menor tiempo de retraso.
3. Desconecte el clúster de base de datos secundario que haya elegido de la base de datos global de Neptune.

   La eliminación de un clúster de base de datos secundario de una base de datos global detiene inmediatamente la replicación de datos del principal al secundario y la promueve a un clúster de base de datos independiente con capacidades completas de lectura/escritura. El resto de clústeres secundarios de la base de datos global seguirán estando disponibles y podrán aceptar llamadas de lectura de la aplicación.

   Antes de volver a crear la base de datos global de Neptune, también tendrá que desconectar los demás clústeres secundarios para evitar incoherencias de datos entre los clústeres (consulte Eliminación de un clúster).
4. Vuelva a configurar la aplicación para enviar todas las operaciones de escritura al clúster de base de datos de Neptune independiente que ha elegido para convertirse en el nuevo clúster principal con su nuevo punto de conexión. Si ha aceptado los nombres predeterminados al crear la base de datos global de Neptune, puede cambiar el punto de conexión. Para ello, elimine `-ro` de la cadena del punto de conexión del clúster en la aplicación.

   Por ejemplo, el punto de conexión del clúster secundario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` se convierte en `my-global.cluster-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` cuando ese clúster se desconecta de la base de datos global.

   Este clúster de base de datos de Neptune se convierte en el clúster principal de una nueva base de datos global de Neptune cuando, en el siguiente paso, comienza a añadirle regiones.
5. Agregue una Región de AWS al clúster de base de datos. Al hacerlo, comienza el proceso de reproducción de clúster principal a secundario. Consulte Añadir regiones de bases de datos globales secundarias a una región principal en Amazon Neptune .

6. Agregue más Regiones de AWS según sea necesario para recrear la topología necesaria para admitir su aplicación.

Asegúrese de que las escrituras de la aplicación se envíen al clúster de base de datos de Neptune correcto antes, durante y después de realizar estos cambios. De este modo, se evitan incoherencias con respecto a los datos entre los clústeres de la base de datos global de Neptune (problemas del tipo cerebro dividido).

Ejecución de la conmutación por error planificada administrada para bases de datos globales de Neptune

La conmutación por error planificada administrada le permite reubicar el clúster principal de la base de datos global de Neptune en otra Región de AWS cuando lo desee. Algunas organizaciones querrán rotar periódicamente las ubicaciones de sus clústeres principales.

> **Note**
>
> La conmutación por error planificada administrada está diseñada para utilizarse en una base de datos global de Neptune en buen estado. Para recuperarse de una interrupción no planificada o para realizar pruebas de recuperación de desastres (DR), siga el proceso de desconectar y promover.

Durante una conmutación por error planificada administrada, el clúster principal se conmuta por error a la región secundaria que elija mientras se mantiene la topología de replicación existente de la base de datos global. Antes de que comience el proceso de conmutación por error planificada administrada, la base de datos global sincroniza todos los clústeres secundarios con su clúster principal. Después de asegurarse de que todos los clústeres están sincronizados, comienza la conmutación por error administrada. El clúster de base de datos de la región principal se convierte en solo de lectura, y el clúster secundario elegido promueve una de sus instancias de solo lectura al estado de escritor completo, lo que permite que el clúster asuma el rol de clúster principal. Dado que todos los clústeres secundarios se sincronizaron con el principal al principio del proceso, el nuevo clúster principal continúa las operaciones para la base de datos global sin perder ningún dato. La base de datos no estará disponible durante un breve periodo, mientras los clústeres principales y secundarios seleccionados asumen nuevos roles.

Para optimizar la disponibilidad de las aplicaciones, realice la conmutación por error durante los horarios menos concurridos, en un momento en que las escrituras en el clúster principal de base de datos sean mínimas. Asimismo, siga estos pasos antes de iniciar la conmutación por error:

- Desconecte las aplicaciones siempre que sea posible para reducir las escrituras en el clúster principal.
- Compruebe los tiempos de retraso de todos los clústeres de base de datos secundarios de Neptune en la base de datos global y elija el secundario con el menor tiempo de retraso total para que se convierta en el principal. Utilice Amazon CloudWatch para ver la métrica de `NeptuneGlobalDBProgressLag` de todos los clústeres secundarios. Esta métrica le indica qué tan lejos (en milisegundos) está un clúster secundario con respecto al clúster principal de base de datos. Su valor es directamente proporcional al tiempo que tardará Neptune en completar la conmutación por error. Dicho de otro modo, cuanto mayor sea el valor de retraso, más extensa será la interrupción por conmutación por error, así que elija el clúster secundario con el menor retraso. Para obtener más información, consulte Métricas de Neptune CloudWatch .

Durante una conmutación por error planificada administrada, el clúster de base de datos secundario elegido se promueve a su nuevo rol de clúster principal, pero no hereda la configuración completa del clúster principal de base de datos. Una falta de coincidencia en la configuración puede provocar problemas de rendimiento, incompatibilidades de carga de trabajo y otros comportamientos anómalos. Para evitar estos problemas, solucione los siguientes tipos de diferencias de configuración entre los clústeres de bases de datos globales antes de la conmutación por error:

- Configurar los parámetros del nuevo clúster principal para que coincidan con el principal actual.
- Configurar alarmas, opciones y herramientas de monitorización: configure el clúster de base de datos que será el nuevo clúster principal con la misma capacidad de registro, alarmas, etc. que tiene el principal actual.
- Configurar integraciones con otros servicios de AWS: si la base de datos global de Neptune se integra con servicios de AWS, como AWS Identity and Access Management (IAM), Amazon S3 o AWS Lambda, asegúrese de que estén configurados según sea necesario para integrarlos con el nuevo clúster de base de datos principal.

Cuando finalice el proceso de conmutación por error y el clúster de base de datos promovido esté preparado para gestionar las operaciones de escritura en la base de datos global, asegúrese de cambiar las aplicaciones para que utilicen el nuevo punto de conexión del nuevo clúster principal.

Uso de la AWS CLI para iniciar una conmutación por error planificada administrada

Utilice el comando failover-global-cluster de la CLI (que incluye la API de FailoverGlobalCluster) para realizar la conmutación por error de la base de datos global de Neptune:

```
aws neptune failover-global-cluster \
  --region (the region where the primary cluster is located) \
  --global-cluster-identifier (global database ID) \
  --target-db-cluster-identifier (the ARN of the secondary DB cluster to promote)
```

> **Note**
>
> La API de `failover-global-cluster` no está disponible en la versión preliminar. Formará parte de la versión de GA.

Monitorización de una base de datos global de Neptune mediante métricas de CloudWatch

Neptune admite las siguientes métricas de CloudWatch que puede utilizar para monitorizar una base de datos global de Neptune:

- **`GlobalDbDataTransferBytes`**: el número de bytes de datos de registro redo transferidos de la Región de AWS principal a una Región de AWS secundaria en una base de datos global de Neptune.
- **`GlobalDbReplicatedWriteIO`**: el número de operaciones de E/S de escritura replicadas desde la Región de AWS principal de la base de datos global hasta el volumen de clúster de una Región de AWS secundaria.

  Los cálculos de facturación de cada clúster de base de datos de una base de datos global de Neptune utilizan la métrica `VolumeWriteIOPS` para contabilizar las escrituras realizadas dentro del clúster. En el caso del clúster de base de datos principal, los cálculos de facturación utilizan `NeptuneGlobalDbReplicatedWriteIO` para contabilizar la replicación entre regiones en clústeres de bases de datos secundarios.
- **`GlobalDbProgressLag`**: el número de milisegundos que un clúster secundario está por detrás del clúster principal tanto para las transacciones de usuario como para las del sistema.

Métrica	Dimensión	Publicado en	Unidades
`GlobalDbDataTransferBytes`	SourceRegion, DBClusterIdentifier	Secondary	Bytes
`GlobalDbReplicatedWriteIO`	SourceRegion, DBClusterIdentifier	Secondary	Recuento
`GlobalDbProgressLag`	DBClusterIdentifier, SecondaryRegion: en Primary; DBCluster Identifier, SourceReg ion: en Secondary	Primary, Secondary	Milisegundos

Información general de las características de Amazon Neptune

En esta sección, se proporciona información general acerca de las características específicas de Neptune, entre las que se incluyen:

- Conformidad de Neptune con los estándares de los lenguajes de consulta.
- Modelo de datos de gráficos de Neptune.
- Una explicación de la semántica de las transacciones de Neptune.
- Introducción a los clústeres e instancias de Neptune.
- El almacenamiento, la fiabilidad y la disponibilidad de Neptune.
- Una explicación de los puntos de conexión de Neptune.
- Cómo los identificadores de consulta personalizados de Neptune le permiten comprobar el estado de la consulta.
- Uso del modo laboratorio de Neptune para activar las características experimentales.
- Descripción del motor DFE de Neptune.
- La conectividad JDBC de Neptune.
- Una lista de las versiones del motor de Neptune y cómo actualizarlo.

> **Note**
>
> En esta sección, no se describe el uso de los lenguajes de consulta que se pueden usar para acceder a los datos de un gráfico de Neptune.
> Para obtener información sobre cómo conectarse con un clúster de base de datos de Neptune en ejecución con Gremlin, consulte Acceso al gráfico de Neptune con Gremlin.
> Para obtener información sobre cómo conectarse con un clúster de base de datos de Neptune en ejecución con openCypher, consulte Acceso al gráfico de Neptune con openCypher.
> Para obtener información sobre cómo conectarse con un clúster de base de datos de Neptune en ejecución con SPARQL, consulte Acceso al gráfico de Neptune con SPARQL.

Temas

Notas sobre la conformidad con los estándares de Amazon Neptune

Amazon Neptune cumple los estándares aplicables en la implementación de los lenguajes de consulta de gráficos Gremlin y SPARQL en la mayoría de los casos.

En estas secciones, se describen los estándares, así como las áreas en las que Neptune los amplía o se diferencia de ellos.

Temas

Conformidad con los estándares de Gremlin en Amazon Neptune

Las siguientes secciones proporcionan una descripción general de la implementación de Gremlin en Neptune y en qué se diferencia de la implementación de Apache. TinkerPop

Neptune implementa algunos pasos de Gremlin de forma nativa en su motor y usa la implementación de Apache TinkerPop Gremlin para procesar otros (consulte). Compatibilidad nativa con pasos de Gremlin en Amazon Neptune

> **Note**
>
> Para ver algunos ejemplos concretos de estas diferencias de implementación que se muestran en la consola de Gremlin y Amazon Neptune, consulte la sección the section called “Utilice Gremlin” del inicio rápido.

Temas

- Ejecución de scripts
- Sesiones
- Transacciones
- Identificador de vértice y borde
- Identificadores proporcionados por el usuario
- Identificadores de propiedades de vértice
- Cardinalidad de las propiedades de vértice
- Actualización de una propiedad de vértice
- Etiquetas
- Caracteres de escape
- Limitaciones de Groovy
- Serialización
- Pasos de Lambda
- Métodos de Gremlin no admitidos
- Pasos de Gremlin no admitidos
- Características de gráficos de Gremlin en Neptune

Estándares aplicables de Gremlin

- El lenguaje Gremlin se define en la TinkerPop documentación de Apache y en la TinkerPop implementación de Gremlin en Apache, más que en una especificación formal.
- En el caso de los formatos numéricos, Gremlin sigue el estándar IEEE 754 (IEEE 754-2019 - Estándar IEEE para aritmética de punto flotante. Para obtener más información, consulte también la página de la Wikipedia sobre IEEE 754).

Variables y parámetros en los scripts

En lo que respecta a las variables previamente enlazadas, el objeto de recorrido `g` está enlazado previamente en Neptune y no se admite el objeto `graph`.

Aunque Neptune no admite variables de Gremlin ni la parametrización en los scripts, a menudo encontrará en Internet ejemplos de scripts del servidor de Gremlin que contienen declaraciones de variables, como:

```
String query = "x = 1; g.V(x)";
List<Result> results = client.submit(query).all().get();
```

También hay muchos ejemplos que utilizan la parametrización (o enlaces) al enviar consultas, como:

```
Map<String,Object> params = new HashMap<>();
params.put("x",1);
String query = "g.V(x)";
List<Result> results = client.submit(query).all().get();
```

Los ejemplos de parámetros suelen estar asociados a advertencias sobre penalizaciones en el rendimiento si no se parametriza cuando es posible. Puede encontrar muchos ejemplos de este tipo, y todos parecen bastante convincentes en cuanto TinkerPop a la necesidad de parametrizar.

Sin embargo, tanto la función de declaración de variables como la función de parametrización (junto con las advertencias) solo se aplican al servidor Gremlin cuando se utiliza TinkerPop el. `GremlinGroovyScriptEngine` No se aplican cuando el servidor de Gremlin utiliza la gramática ANTLR `gremlin-language` de Gremlin para analizar las consultas. La gramática de ANTLR no admite ni las declaraciones de variables ni la parametrización, por lo que cuando utilice ANTLR, no tiene que preocuparse por no poder parametrizar. Como la gramática del ANTLR es un componente más reciente TinkerPop, el contenido más antiguo que puedas encontrar en Internet no suele reflejar esta distinción.

Neptune utiliza la gramática de ANTLR en su motor de procesamiento de consultas en lugar del `GremlinGroovyScriptEngine`, por lo que no admite variables, parametrización ni la propiedad `bindings`. Como resultado, los problemas relacionados con la falta de parametrización no se aplican en Neptune. Con Neptune, es perfectamente seguro enviar la consulta tal cual, cuando lo normal sería parametrizarla. Como resultado, el ejemplo anterior se puede simplificar sin ninguna penalización en el rendimiento de la siguiente manera:

```
String query = "g.V(1)";
List<Result> results = client.submit(query).all().get();
```

TinkerPop enumeraciones

Neptune no admite nombres completos de clase para los valores de enumeración. Por ejemplo, debe utilizar `single` y no `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` en la solicitud de Groovy.

El tipo de enumeración viene determinado por el tipo de parámetro.

En la siguiente tabla se muestran los valores de enumeración permitidos y el nombre TinkerPop completo correspondiente.

Valores permitidos	Clase
`id`, `key`, `label`, `value`	org.apache.tinkerpop.gremlin.structure.T
`T.id`, `T.key`, `T.label`, `T.value`	org.apache.tinkerpop.gremlin.structure.T
`set`, `single`	org.apache.tinkerpop.gremlin.structure. VertexProperty. Cardinalidad
`asc`, `desc`, `shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`Order.asc` , `Order.desc` , `Order.shuffle`	org.apache.tinkerpop.gremlin.process.traversal.Order
`global`, `local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`Scope.global` , `Scope.local`	org.apache.tinkerpop.gremlin.process.traversal.Scope
`all`, `first`, `last`, `mixed`	org.apache.tinkerpop.gremlin.process.traversal.Pop
`normSack`	org.apache.tinkerpop.gremlin.process.traversal. SackFunctions.Barrera
`addAll`, `and`, `assign`, `div`, `max`, `min`, `minus`, `mult`, `or`, `sum`, `sumLong`	org.apache.tinkerpop.gremlin.process.traversal.Operator
`keys`, `values`	org.apache.tinkerpop.gremlin.structure.Column
`BOTH`, `IN`, `OUT`	org.apache.tinkerpop.gremlin.structure.Direction

any, none	org.apache.tinkerpop.gremlin.process.traversal.step. TraversalOptionParent.Pick

Código Java

Neptune no admite las llamadas a métodos que se definen mediante llamadas arbitrarias a Java o a bibliotecas de Java, aparte de las API de Gremlin admitidas. Por ejemplo `java.lang.*`, `Date()` y `g.V().tryNext().orElseGet()` no se permiten.

Propiedades de los elementos

Neptune no admite la `materializeProperties` bandera que se introdujo en la TinkerPop versión 3.7.0 para devolver las propiedades de los elementos. Como resultado, Neptune solo devolverá vértices o aristas como referencias solo con su y. `id` `label`

Ejecución de scripts

Todas las consultas debe comenzar por g, el objeto de recorrido.

En los envíos de consulta de cadena, se pueden emitir varios recorridos separados por punto y coma (`;`) o por un carácter de nueva línea (\n). Para ejecutarse, todas las instrucciones, aparte de la última, deben finalizar con un paso `.iterate()`. Solo se devuelven los datos del recorrido final. Tenga en cuenta que esto no se aplica a los envíos de consultas a GLV ByteCode .

Sesiones

Las sesiones en Neptune se limitan a una duración de 10 minutos. Consulte Sesiones basadas en scripts de Gremlin la referencia de la TinkerPop sesión para obtener más información.

Transacciones

Neptune abre una nueva transacción al principio de cada recorrido de Gremlin y la cierra una vez que este se completa correctamente. La transacción se revierte si se produce un error.

Varias instrucciones separadas por punto y coma (`;`) o por un carácter de nueva línea (\n) se incluyen en una sola transacción. Todas las instrucciones, aparte de la última, deben finalizar con un paso `next()` para ejecutarse. Solo se devuelven los datos del recorrido final.

No se admite la lógica de transacción manual que utiliza `tx.commit()` y `tx.rollback()`.

> **Important**
>
> Esto solo se aplica a aquellos casos en los que envíe la consulta de Gremlin como una cadena de texto (consulte Transacciones de Gremlin).

Identificador de vértice y borde

Los identificadores de vértice y borde de Gremlin en Neptune deben ser de tipo `String`. Estas cadenas de identificación admiten caracteres Unicode y su tamaño no puede superar los 55 MB.

Se admiten los identificadores proporcionados por el usuario, pero son opcionales en el uso normal. Si no proporciona un identificador cuando añade un vértice o un borde, Neptune genera un UUID y lo convierte en una cadena, de una forma parecida a esta: `"48af8178-50ce-971a-fc41-8c9a954cea62"`. Estos UUID no cumplen el estándar RFC, por lo que si necesita UUID estándar, debe generarlos externamente y proporcionarlos cuando añada vértices o bordes.

> **Note**
>
> Sin embargo, el comando `Load` de Neptune requiere que proporcione identificadores mediante el campo ~id en formato CSV de Neptune.

Identificadores proporcionados por el usuario

Neptune Gremlin admite los identificadores proporcionados por el usuario con las siguientes condiciones.

- Los ID proporcionados son opcionales.
- Solo se admiten vértices y bordes.
- Solo se admite el tipo `String`.

Para crear un vértice con un ID personalizado, utilice el paso `property` con la palabra clave `id`: `g.addV().property(id, 'customid')`.

> **Note**
>
> No añada comillas alrededor de la palabra clave `id`. Hace referencia a `T.id`.

Todos los identificadores de vértice deben ser únicos y todos los identificadores de borde deben ser únicos. Sin embargo, Neptune permite que un vértice y un borde tengan el mismo identificador.

Si intenta crear un vértice con `g.addV()` y ya existe un vértice con ese ID, la operación dará error. La excepción a esta norma es que, si especifica una nueva etiqueta para el vértice, la operación se completará correctamente, pero añadirá la nueva etiqueta y las propiedades adicionales especificadas al vértice existente. No se sobrescribirá ninguna. No se creará un nuevo vértice. El ID del vértice no cambia y continúa siendo único.

Por ejemplo, los siguientes comandos de la consola de Gremlin se ejecutan correctamente:

```
gremlin> g.addV('label1').property(id, 'customid')
gremlin> g.addV('label2').property(id, 'customid')
gremlin> g.V('customid').label()
==>label1::label2
```

Identificadores de propiedades de vértice

Los ID de propiedades de vértice se generan automáticamente y pueden aparecer como números positivos o negativos al realizar consultas.

Cardinalidad de las propiedades de vértice

Neptune admite la cardinalidad en conjuntos y la cardinalidad simple. Si no se especifica, se selecciona la cardinalidad en conjuntos. Esto significa que si establece un valor de propiedad, añade un nuevo valor a la propiedad, pero solo si este no aparece en el conjunto de valores. Este es el valor de enumeración de Gremlin de Set (Establecer).

`List` no se admite. Para obtener más información sobre la cardinalidad de las propiedades, consulte el tema Vértices en el Gremlin. JavaDoc

Actualización de una propiedad de vértice

Para actualizar un valor de propiedad sin añadir un valor al conjunto de valores, especifique la cardinalidad `single` en el paso `property`.

```
g.V('exampleid01').property(single, 'age', 25)
```

Esto elimina todos los valores existentes para la propiedad.

Etiquetas

Neptune admite varias etiquetas para un vértice. Al crear una etiqueta, puede especificar varias si las separa mediante `::`. Por ejemplo, `g.addV("Label1::Label2::Label3")` añade un vértice con tres etiquetas distintas. El paso `hasLabel` busca coincidencias de este vértice con cualquiera de esas tres etiquetas: `hasLabel("Label1")` `hasLabel("Label2")` y `hasLabel("Label3")`.

> **Important**
>
> El delimitador `::` está reservado solo para este uso. No se pueden especificar varias etiquetas en el paso `hasLabel`. Por ejemplo, `hasLabel("Label1::Label2")` no tiene ninguna coincidencia.

Caracteres de escape

Neptune resuelve todos los caracteres de escape tal y como se describe en la sección Escaping Special Characters de la documentación del lenguaje Apache Groovy.

Limitaciones de Groovy

Neptune no admite los comandos de Groovy que no empiezan por g. Esto incluye expresiones matemáticas (por ejemplo, 1+1), llamadas al sistema (por ejemplo, `System.nanoTime()`) y definiciones de variable (por ejemplo, 1+1).

> **Important**
>
> Neptune no admite nombres completos de clase. Por ejemplo, debe utilizar `single` y no `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` en la solicitud de Groovy.

Serialización

Neptune admite las siguientes serializaciones en función del tipo MIME solicitado.

Tipo MIME	Serialización	Configuración

application/vnd.gremlin-v1.0+gryo	GryoMessageSerializerV1	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]
application/vnd.gremlin-v1.0+gryo-stringd	GryoMessageSerializerV1	serializeResultToString: true}
application/vnd.gremlin-v3.0+gryo	GryoMessageSerializerV3	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]
application/vnd.gremlin-v3.0+gryo-stringd	GryoMessageSerializerV3	serializeResultToString: true
application/vnd.gremlin-v1.0+json	GraphSONMessageSerializerGremlinV1	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV1]
application/vnd.gremlin-v2.0+json	GraphSONMessageSerializerV2 (solo funciona con) WebSockets	ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV2]
application/vnd.gremlin-v3.0+json	GraphSONMessageSerializerV3	

`application/json`	`GraphSONMessageSerializerV3`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerIoRegistryV3]`
`application/vnd.graphbinary-v1.0`	`GraphBinaryMessageSerializerV1`	

Si bien Neptune admite estos diferentes tipos de serializadores, la guía para su uso es bastante sencilla. Si se conecta a Neptune a través de HTTP, priorice el uso de, `application/vnd.gremlin-v3.0+json;types=false` ya que los tipos embebidos en la versión alternativa de GraphSon 3 dificultan el trabajo con ellos. Si utilizas los TinkerPop controladores de Apache, es probable que no tengas que tomar ninguna decisión, ya que utilizarías el predeterminado. `application/vnd.graphbinary-v1.0` El resto de los formatos permanecen presentes por motivos heredados.

> **Note**
>
> La tabla de serializadores que se muestra aquí se refiere a la nomenclatura a partir de TinkerPop la versión 3.7.0. Si desea obtener más información sobre este cambio, consulte la TinkerPop documentación de actualización. La compatibilidad con la serialización de Gryo quedó obsoleta en la versión 3.4.3 y se eliminó oficialmente en la versión 3.6.0. Si utilizas Gryo de forma explícita o utilizas una versión de controlador que lo utiliza de forma predeterminada, deberías cambiar a tu controlador o actualizarlo. GraphBinary

Pasos de Lambda

Neptune no admite los pasos de Lambda.

Métodos de Gremlin no admitidos

Neptune no admite los siguientes métodos de Gremlin:

- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.progra`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.sideEf`

- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.from(o`
- `org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal.to(org`

Por ejemplo, el recorrido siguiente no está permitido: `g.V().addE('something').from(__.V().next()).to(__.V().next())`.

> **Important**
>
> Esto solo se aplica a aquellos casos en los que envíe la consulta de Gremlin como una cadena de texto.

Pasos de Gremlin no admitidos

Neptune no admite los siguientes pasos de Gremlin:

- El paso io() de Gremlin solo se admite parcialmente en Neptune. Se puede usar en un contexto de lectura, como en `g.io(url).read()`, pero no para escribir.

Características de gráficos de Gremlin en Neptune

La implementación de Gremlin en Neptune no expone el objeto `graph`. Las siguientes tablas muestran las características de Gremlin e indican si Neptune las admite o no.

Compatibilidad de Neptune con características de **`graph`**

Las características de gráficos, si se admiten, son las mismas que devolvería el comando `graph.features()`.

Característica de gráfico	¿Habilitada?
`Transactions`	true
`ThreadedTransactions`	false
`Computer`	false
`Persistence`	true

`ConcurrentAccess`	true

Compatibilidad de Neptune con características de variables

Característica de variable	¿Habilitada?
`Variables`	false
`SerializableValues`	false
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	false
`ByteValues`	false
`DoubleValues`	false
`FloatValues`	false
`IntegerValues`	false
`LongValues`	false
`MapValues`	false
`MixedListValues`	false
`StringValues`	false
`ByteArrayValues`	false
`FloatArrayValues`	false

`LongArrayValues`	false

Compatibilidad de Neptune con características de vértices

Característica de vértice	¿Habilitada?
`MetaProperties`	false
`DuplicateMultiProperties`	false
`AddVertices`	true
`RemoveVertices`	true
`MultiProperties`	true
`UserSuppliedIds`	true
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	false
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Compatibilidad de Neptune con características de propiedades de vértices

Característica de propiedad de vértice	¿Habilitada?
`UserSuppliedIds`	false
`AddProperty`	true

RemoveProperty	true
NumericIds	true
StringIds	true
UuidIds	false
CustomIds	false
AnyIds	false
Properties	true
SerializableValues	false
UniformListValues	false
BooleanArrayValues	false
DoubleArrayValues	false
IntegerArrayValues	false
StringArrayValues	false
BooleanValues	true
ByteValues	true
DoubleValues	true
FloatValues	true
IntegerValues	true
LongValues	true
MapValues	false
MixedListValues	false
StringValues	true

`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Compatibilidad de Neptune con características de bordes

Característica de borde	¿Habilitada?
`AddEdges`	true
`RemoveEdges`	true
`UserSuppliedIds`	true
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	false
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Compatibilidad de Neptune con características de propiedades de bordes

Característica de propiedad de borde	¿Habilitada?
`Properties`	true
`SerializableValues`	false
`UniformListValues`	false

`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true
`LongValues`	true
`MapValues`	false
`MixedListValues`	false
`StringValues`	true
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Conformidad con los estándares de SPARQL en Amazon Neptune

Tras enumerar los estándares de SPARQL aplicables, en las siguientes secciones se proporcionan detalles específicos sobre cómo la implementación de SPARQL de Neptune amplía o se aparta de dichos estándares.

Temas

- Estándares aplicables de SPARQL

- Prefijos de espacio de nombres predeterminados en Neptune SPARQL
- Gráfico predeterminado SPARQL y gráficos con nombre
- Funciones del constructor XPath de SPARQL compatibles con Neptune
- IRI base predeterminado para consultas y actualizaciones
- Valores xsd:dateTime en Neptune
- Gestión de Neptune de los valores especiales de coma flotante
- Limitación en Neptune de los valores de longitud arbitraria
- Neptune amplía la comparación de valores iguales en SPARQL
- Gestión de literales fuera de rango en SPARQL de Neptune

Amazon Neptune cumple los siguientes estándares en la implementación del lenguaje de consulta de gráficos SPARQL.

Estándares aplicables de SPARQL

- SPARQL se define en la recomendación Lenguaje de consulta SPARQL 1.1 del W3C del 21 de marzo de 2013.
- El lenguaje de consulta y el protocolo de SPARQL Update están definidos en la especificación SPARQL 1.1 Update del W3C.
- En el caso de los formatos numéricos, SPARQL sigue la especificación de W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, que es coherente con la especificación IEEE 754 (Estándar IEEE 754-2019 - IEEE para aritmética de coma flotante; consulte también la página de la Wikipedia sobre IEEE 754). Sin embargo, las características que se incorporaron después de la versión `IEEE 754-1985` no están incluidas en la especificación.

Prefijos de espacio de nombres predeterminados en Neptune SPARQL

Neptune define los siguientes prefijos de forma predeterminada para su uso en consultas de SPARQL. Para obtener más información, consulte Nombres con prefijos en la especificación de SPARQL.

- `rdf – http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- `rdfs – http://www.w3.org/2000/01/rdf-schema#`
- `owl – http://www.w3.org/2002/07/owl#`

- `xsd – http://www.w3.org/2001/XMLSchema#`

Gráfico predeterminado SPARQL y gráficos con nombre

Amazon Neptune asocia cada triple con un gráfico con nombre. El gráfico predeterminado se define como la unión de todos los gráficos con nombre.

Gráfico predeterminado para consultas

Si envía una consulta SPARQL sin especificar explícitamente un gráfico mediante la palabra clave `GRAPH` o construcciones como `FROM NAMED`, Neptune siempre tiene en cuenta todos los triples en su instancia de base de datos. Por ejemplo, la siguiente consulta devuelve todos los triples desde un punto de conexión de SPARQL de Neptune:

```
SELECT * WHERE { ?s ?p ?o }
```

Los triples que aparecen en más de un gráfico se devuelven solo una vez.

Para obtener información sobre la especificación de gráfico predeterminado, consulte la sección Conjunto de datos de RDF de la especificación del lenguaje de consulta SPARQL 1.1.

Especificación del gráfico con nombre para carga, inserciones o actualizaciones

Si no especifica un gráfico con nombre al cargar, insertar o actualizar triples, Neptune utiliza el gráfico con nombre alternativo definido por el URI `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Cuando emite una solicitud `Load` de Neptune utilizando un formato basado en triples, se puede especificar el gráfico con nombre que se va a utilizar para todos los triples mediante el parámetro `parserConfiguration: namedGraphUri`. Para obtener información acerca de la sintaxis del comando `Load`, consulte the section called “Comando Loader”.

> **Important**
>
> Si no utiliza este parámetro y no especifica un gráfico con nombre, se utiliza la URI alternativa: `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

También se utiliza este gráfico con nombre alternativo si se cargan triples a través de `SPARQL UPDATE` sin proporcionar explícitamente un destino de gráfico con nombre.

Puede utilizar el formato N-Quads basado en cuádruples para especificar un gráfico con nombre para cada triple en la base de datos.

> **Note**
>
> El uso de N-Quads le permite dejar el gráfico con nombre en blanco. En este caso, se usa `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.
> Puede anular el gráfico con nombre predeterminado para N-Quads con la opción de configuración del analizador `namedGraphUri`.

Funciones del constructor XPath de SPARQL compatibles con Neptune

El estándar SPARQL permite que los motores SPARQL admitan un conjunto ampliable de funciones de constructores de XPath. Neptune admite actualmente las siguientes funciones de constructores, donde el prefijo `xsd` se define como `http://www.w3.org/2001/XMLSchema#`:

- `xsd:boolean`
- `xsd:integer`
- `xsd:double`
- `xsd:float`
- `xsd:decimal`
- `xsd:long`
- `xsd:unsignedLong`

IRI base predeterminado para consultas y actualizaciones

Dado que un clúster de Neptune tiene varios puntos de conexión diferentes, el uso de la URL de solicitud de una consulta o actualización como IRI base podría generar resultados inesperados al resolver los IRI relativos.

A partir de la versión 1.2.1.0 del motor, Neptune utiliza `http://aws.amazon.com/neptune/default/` como IRI base si no hay una IRI base explícita que forme parte de la solicitud.

En la siguiente solicitud, el IRI base forma parte de la solicitud:

```
BASE <http://example.org/default/>
```

```
INSERT DATA { <node1> <id> "n1" }

BASE <http://example.org/default/>
SELECT * { <node1> ?p ?o }
```

Y el resultado sería:

```
?p                                                  ?o
http://example.org/default/id                       n1
```

Sin embargo, en esta solicitud no se incluye ningún IRI base:

```
INSERT DATA { <node1> <id> "n1" }

SELECT * { <node1> ?p ?o }
```

En ese caso, el resultado sería:

```
?p                                                  ?o
http://aws.amazon.com/neptune/default/id            n1
```

Valores xsd:dateTime en Neptune

Por razones de rendimiento, Neptune siempre almacena valores de fecha/hora como hora universal coordinada (UTC). Esto hace que las comparaciones directas sean muy eficientes.

Esto también significa que, si introduce un valor `dateTime` que especifica una zona horaria determinada, Neptune traduce el valor a UTC y descarta esa información de zona horaria. A continuación, cuando recupere el valor `dateTime` más tarde, se expresará en UTC, no en la hora de la zona horaria original, y no podrá indicar cuál era la zona horaria original.

Gestión de Neptune de los valores especiales de coma flotante

Neptune gestiona los valores especiales de coma flotante en SPARQL de la siguiente manera.

Gestión de NaN de SPARQL en Neptune

En Neptune, SPARQL puede aceptar un valor de `NaN` en una consulta. No se hace ninguna distinción entre los valores de señalización y los valores `NaN` silenciosos. Neptune trata todos los valores `NaN` como silenciosos.

Semánticamente, no es posible ninguna comparación de un valor `NaN`, porque nada es mayor, menor o igual que `NaN`. Esto significa que un valor de `NaN` en un lado de una comparación en teoría nunca coincide con nada del otro lado.

Sin embargo, la especificación XSD trata dos valores `xsd:double` o `xsd:float` de `NaN` como iguales. Neptune hace esto para el filtro `IN`, para el operador igual en las expresiones de filtro y para la semántica de coincidencia (que tiene a `NaN` en la posición del objeto de un patrón triple).

Gestión de valores infinitos de SPARQL en Neptune

En Neptune, SPARQL puede aceptar un valor de `INF` o `-INF` en una consulta. `INF` es mayor que cualquier otro valor numérico y `-INF` es menor que cualquier otro valor numérico.

Dos valores INF con signos coincidentes se comparan como iguales entre sí independientemente del tipo (por ejemplo, un flotante `-INF` se compara como igual a un `-INF` doble).

Por supuesto, no es posible ninguna comparación con `NaN` porque nada es mayor, menor o igual que `NaN`.

Gestión de cero negativo de SPARQL en Neptune

Neptune normaliza un valor cero negativo a un cero sin signo. Puede utilizar los valores cero negativos se pueden utilizar en una consulta, pero no se registran como tales en la base de datos y se comparan como iguales a ceros sin signo.

Limitación en Neptune de los valores de longitud arbitraria

Neptune limita el tamaño de almacenamiento de valores de enteros, coma flotante y decimales XSD en SPARQL a 64 bits. Si se utilizan valores mayores, se producirá un error `InvalidNumericDataException`.

Neptune amplía la comparación de valores iguales en SPARQL

El estándar SPARQL define una lógica ternaria para las expresiones de valor, donde una expresión de valor puede evaluar a `true`, `false` o `error`. La semántica predeterminada para la igualdad de términos tal y como se define en la especificación SPARQL 1.1, que se aplica a las comparaciones `=` y `!=` en condiciones `FILTER`, produce un `error` al comparar los tipos de datos que no son explícitamente comparables en la tabla de operadores en la especificación.

Este comportamiento puede conllevar resultados poco intuitivos, como en el siguiente ejemplo.

Datos:

```
<http://example.com/Server/1> <http://example.com/ip> "127.0.0.1"^^<http://example.com/
datatype/IPAddress>
```

Consulta 1:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o = "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Consulta 2:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o != "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Con la semántica de SPARQL predeterminada que Neptune usaba antes de la versión 1.0.2.1, ambas consultas devolverían el resultado vacío. La razón es que, cuando `?o = "127.0.0.2"^^<http://example.com/IPAddress>` se evalúa para `?o := "127.0.0.1"^^<http://example.com/IPAddress>`, se produce un `error` distinto de `false` porque no hay reglas de comparación explícitas especificadas para el tipo de datos personalizados `<http://example.com/IPAddress>`. Como resultado, la versión negada en la segunda consulta también produce un `error`. En ambas consultas, el `error` provoca que se excluya la solución candidata.

A partir de la versión 1.0.2.1, Neptune ha ampliado el operador de desigualdad de SPARQL de acuerdo con la especificación. Consulte la sección SPARQL 1.1 sobre la extensibilidad de operadores, que permite a los motores definir reglas adicionales sobre cómo efectuar la comparación entre tipos de datos integrados definidos por el usuario y no comparables.

Mediante esta opción, Neptune ahora trata una comparación de dos tipos de datos cualesquiera que no estén explícitamente definidos en la tabla de mapeo de operadores como si se evaluaran en `true` si los valores literales y los tipos de datos son sintácticamente iguales y en false en caso contrario. En cualquier caso no se produce un `error`.

Con esta nueva semántica, la segunda consulta devolvería `"127.0.0.1"^^<http://example.com/IPAddress>` en lugar de un resultado vacío.

Gestión de literales fuera de rango en SPARQL de Neptune

La semántica XSD define cada tipo numérico con su espacio de valor, excepto para `integer` y `decimal`. Estas definiciones limitan cada tipo a un rango de valores. Por ejemplo, el rango de un rango `xsd:byte` va de -128 a +127, ambos incluidos. Cualquier valor fuera de este rango no se considera válido.

Si intenta asignar un valor literal fuera del espacio de valores de un tipo (por ejemplo, si intenta establecer un `xsd:byte` valor literal de 999), Neptune acepta el out-of-range valor tal cual, sin redondearlo ni truncarlo. Pero no lo conserva como un valor numérico ya que el tipo determinado no puede representarlo.

Es decir, Neptune acepta `"999"^^xsd:byte` aunque sea un valor fuera del rango de valores `xsd:byte` definido. Sin embargo, después de que el valor se conserve en la base de datos, solo se puede utilizar en la semántica de coincidencia exacta, en una posición de objeto de un patrón triple. No se puede ejecutar ningún filtro de rango en él porque los out-of-range literales no se tratan como valores numéricos.

La especificación de SPARQL 1.1 define operadores de rango con el formato `numeric`-operador-`numeric`, `string`-operador-`string`, `literal`-operador-`literal`, etc. Neptune no puede ejecutar un operador de comparación de rangos con algo como `invalid-literal`-operador-`numeric-value`.

Conformidad con las especificaciones de OpenCypher en Amazon Neptune

La versión de openCypher de Amazon Neptune suele admitir las cláusulas, los operadores, las expresiones, las funciones y la sintaxis que se definen en la especificación de openCypher actual, que es Cypher Query Language Reference Version 9. A continuación, se detallan las limitaciones y diferencias en la compatibilidad de Neptune con openCypher.

> **Note**
>
> La implementación actual de Neo4j de Cypher contiene una funcionalidad que no está incluida en la especificación de openCypher mencionada anteriormente. Si va a migrar el código de Cypher actual a Neptune, consulte Compatibilidad de Neptune con Neo4j y Reescritura de consultas de Cypher para ejecutarlas en openCypher en Neptune para obtener más información.

Compatibilidad con las cláusulas de openCypher en Neptune

Neptune admite las siguientes cláusulas, salvo que se indique lo contrario:

- `MATCH`: se admite, pero *`shortestPath()`* y *`allShortestPaths()`* no se admiten en este momento.
- `OPTIONAL MATCH`
- *`MANDATORY MATCH`*: no se admite en este momento en Neptune. Sin embargo, Neptune admite valores de ID personalizados en las consultas `MATCH`.
- `RETURN`: se admite, excepto cuando se utiliza con valores no estáticos para `SKIP` o `LIMIT`. Por ejemplo, lo siguiente no funciona actualmente:

```
MATCH (n)
RETURN n LIMIT toInteger(rand())    // Does NOT work!
```

- `WITH`: se admite, excepto cuando se utiliza con valores no estáticos para `SKIP` o `LIMIT`. Por ejemplo, lo siguiente no funciona actualmente:

```
MATCH (n)
WITH n SKIP toInteger(rand())
WITH count() AS count
RETURN count > 0 AS nonEmpty    // Does NOT work!
```

- `UNWIND`
- `WHERE`
- `ORDER BY`
- `SKIP`
- `LIMIT`
- `CREATE`: Neptune permite crear valores de identificadores personalizados en las consultas `CREATE`.
- `DELETE`
- `SET`
- `REMOVE`
- `MERGE`: Neptune admite valores de identificadores personalizados en las consultas `MERGE`.
- *`CALL[YIELD...]`*: no se admite en este momento en Neptune.

- `UNION, UNION ALL`: se admiten consultas de solo lectura, pero actualmente no se admiten consultas de mutación.

Compatibilidad con las operadores de openCypher en Neptune

Neptune admite los siguientes operadores, salvo que se indique lo contrario:

Operadores generales

- `DISTINCT`
- El operador **`.`** para acceder a las propiedades de un mapa literal anidado.

Operadores matemáticos

- El operador de suma **`+`**.
- El operador de resta **`-`**.
- El operador de multiplicación **`*`**.
- El operador de división **`/`**.
- El operador de división de módulo **`%`**.
- *`NO se admite`* el operador de exponenciación **`^`**.

Operadores de comparación

- El operador de suma **`=`**.
- El operador de desigualdad **`<>`**.
- Se admite el operador menor que **`<`**, excepto cuando alguno de los argumentos es una ruta, una lista o un mapa.
- Se admite el operador mayor que **`>`**, excepto cuando alguno de los argumentos es una ruta, una lista o un mapa.
- Se admite el operador **`<=`** less-than-or-equal -to excepto cuando alguno de los argumentos es una ruta, una lista o un mapa.
- Se admite el operador **`>=`** greater-than-or-equal -to excepto cuando alguno de los argumentos es una ruta, una lista o un mapa.
- `IS NULL`

- `IS NOT NULL`
- `STARTS WITH` se admite si los datos que se buscan son una cadena.
- `ENDS WITH` se admite si los datos que se buscan son una cadena.
- `CONTAINS` se admite si los datos que se buscan son una cadena.

Operadores booleanos

- `AND`
- `OR`
- `XOR`
- `NOT`

Operadores de cadena

- El operador de concatenación **+**.

Operadores de lista

- El operador de concatenación **+**.
- `IN` (comprueba la presencia de un elemento en una lista)

Compatibilidad con expresiones de openCypher en Neptune

Neptune admite las siguientes expresiones, salvo que se indique lo contrario:

- `CASE`
- La expresión **`[]`** no se admite actualmente en Neptune para acceder a claves de propiedad calculadas dinámicamente dentro de un nodo, una relación o un mapa. Por ejemplo, lo siguiente no funciona:

```
MATCH (n)
WITH [5, n, {key: 'value'}] AS list
RETURN list[1].name
```

Compatibilidad con las funciones de openCypher en Neptune

Neptune admite las siguientes funciones, salvo que se indique lo contrario:

Funciones de predicados

- `exists()`

Funciones escalares

- `coalesce()`
- `endNode()`
- `epochmillis()`
- `head()`
- `id()`
- `last()`
- `length()`
- `randomUUID()`
- `properties()`
- `removeKeyFromMap`
- `size()`: este método sobrecargado solo funciona actualmente para expresiones de patrones, listas y cadenas
- `startNode()`
- `timestamp()`
- `toBoolean()`
- `toFloat()`
- `toInteger()`
- `type()`

Funciones de agregación

- `avg()`
- `collect()`
- `count()`

- `max()`
- `min()`
- `percentileDisc()`
- `stDev()`
- `percentileCont()`
- `stDevP()`
- `sum()`

Lista de funciones

- `join()` (concatena las cadenas de una lista en una sola cadena)
- `keys()`
- `labels()`
- `nodes()`
- `range()`
- `relationships()`
- `reverse()`
- `tail()`

Funciones matemáticas: numéricas

- `abs()`
- `ceil()`
- `floor()`
- `rand()`
- `round()`
- `sign()`

Funciones matemáticas: logarítmicas

- `e()`
- `exp()`

- `log()`
- `log10()`
- `sqrt()`

Funciones matemáticas: trigonométricas

- `acos()`
- `asin()`
- `atan()`
- `atan2()`
- `cos()`
- `cot()`
- `degrees()`
- `pi()`
- `radians()`
- `sin()`
- `tan()`

Funciones de cadena

- `join()` (concatena las cadenas de una lista en una sola cadena)
- `left()`
- `lTrim()`
- `replace()`
- `reverse()`
- `right()`
- `rTrim()`
- `split()`
- `substring()`
- `toLower()`
- `toString()`

- `toUpper()`
- `trim()`

Funciones definidas por el usuario

`Las funciones definidas por el usuario` no se admiten actualmente en Neptune.

Detalles de la implementación de openCypher específica de Neptune

En las siguientes secciones, se describen las formas en las que la implementación de openCypher de Neptune puede diferir o ir más allá de las especificaciones de openCypher.

Evaluaciones de la ruta de longitud variable (VLP) en Neptune

Las evaluaciones de rutas de longitud variable (VLP) detectan rutas entre nodos en el gráfico. La longitud de la ruta no puede estar restringida en una consulta. Para evitar ciclos, en la especificación de openCypher se indica que cada borde debe recorrerse como máximo una vez por solución.

Para las VLP, la implementación de Neptune difiere de la especificación de openCypher en el sentido de que solo admite valores constantes para los filtros de igualdad de propiedades. Tomemos como ejemplo la siguiente consulta:

```
MATCH (x)-[:route*1..2 {dist:33, code:x.name}]->(y) return x,y
```

Dado que el valor del filtro de igualdad de la propiedad `x.name` no es una constante, esta consulta da como resultado una `UnsupportedOperationException` con el mensaje: `Property predicate over variable-length relationships with non-constant expression is not supported in this release.`

Soporte temporal en la implementación OpenCypher de Neptune (base de datos Neptune 1.3.1.0 y versiones anteriores)

Actualmente, Neptune proporciona una compatibilidad limitada con la función temporal en openCypher. Admite el tipo de datos `DateTime` para los tipos temporales.

La función `datetime()` se puede utilizar para obtener la fecha y hora UTC actuales de la siguiente manera:

```
RETURN  datetime() as res
```

Los valores de fecha y hora se pueden convertir a partir de los datos almacenados en Neptune de la siguiente manera:

```
MATCH (n) RETURN datetime(n.createdDate)
```

Los valores de fecha y hora se pueden analizar a partir de cadenas en un formato de "fechaThora" en el que la fecha y la hora se expresan en uno de los formatos admitidos que se indican a continuación:

Formatos de fecha admitidos

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Formatos de tiempo admitidos

- `HH:mm:ssZ`
- `HHmmssZ`
- `HH:mm:ssZ`
- `HH:mmZ`
- `HHmmZ`
- `HHZ`
- `HHmmss`
- `HH:mm:ss`
- `HH:mm`
- `HHmm`
- `HH`

Por ejemplo:

```
RETURN datetime('2022-01-01T00:01')      // or another example:
RETURN datetime('2022T0001')
```

Tenga en cuenta que todos los valores de fecha y hora en openCypher de Neptune se almacenan y recuperan como valores UTC.

openCypher de Neptune usa un reloj `statement`, lo que significa que se usa el mismo instante en el tiempo durante toda la consulta. Una consulta diferente dentro de la misma transacción podría utilizar un instante diferente en el tiempo.

Neptune no admite el uso de una función en una llamada a `datetime()`. Por ejemplo, lo siguiente no funciona:

```
CREATE (:n {date:datetime(tostring(2021))})  // ---> NOT ALLOWED!
```

Neptune admite la función `epochmillis()` que convierte una `datetime` en `epochmillis`. Por ejemplo:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

Neptune no admite actualmente otras funciones y operaciones en objetos `DateTime`, como la suma y la resta.

Soporte temporal en la implementación de Neptune OpenCypher (Neptune Analytics y Neptune Database 1.3.2.0 y versiones posteriores)

La siguiente funcionalidad de fecha y hora OpenCypher se aplica a Neptune Analytics. Como alternativa, puede usar el parámetro labmode `DatetimeMillisecond=enabled` para habilitar la siguiente funcionalidad de fecha y hora en la versión 1.3.2.0 y posteriores del motor Neptune Engine. Para obtener más información sobre el uso de esta funcionalidad en labmode, consulte. Soporte extendido de fecha y hora

- Support para milisegundos. El literal de fecha y hora siempre se devolverá en milisegundos, incluso si los milisegundos son 0. (El comportamiento anterior consistía en truncar los milisegundos).

  ```
  CREATE (:event {time: datetime('2024-04-01T23:59:59Z')})
  ```

```
# Returning the date returns with 000 suffixed representing milliseconds
MATCH(n:event)
RETURN n.time as datetime

{
  "results" : [ {
    "n" : {
      "~id" : "0fe88f7f-a9d9-470a-bbf2-fd6dd5bf1a7d",
      "~entityType" : "node",
      "~labels" : [ "event" ],
      "~properties" : {
        "time" : "2024-04-01T23:59:59.000Z"
      }
    }
  } ]
}
```

- Support para llamar a la función datetime () sobre propiedades almacenadas o resultados intermedios. Por ejemplo, las siguientes consultas no eran posibles antes de esta función.

  Datetime () sobre las propiedades:

  ```
  // Create node with property 'time' stored as string
  CREATE (:event {time: '2024-04-01T23:59:59Z'})

  // Match and return this property as datetime
  MATCH(n:event)
  RETURN datetime(n.time) as datetime
  ```

  Datetime () sobre los resultados intermedios:

  ```
  // Parse datetime from parameter
  UNWIND $list as myDate
  RETURN datetime(myDate) as d
  ```

- Ahora también es posible guardar las propiedades de fecha y hora que se crean en los casos mencionados anteriormente.

  Guardar la fecha y hora de la propiedad de cadena de una propiedad en otra:

  ```
  // Create node with property 'time' stored as string
  CREATE (:event {time: '2024-04-01T23:59:59Z', name: 'crash'})
  ```

```
// Match and update the same property to datetime type
MATCH(n:event {name: 'crash'})
SET n.time = datetime(n.time)

// Match and update another node's property
MATCH(e:event {name: 'crash'})
MATCH(n:server {name: e.servername})
SET n.time = datetime(e.time)
```

Cree nodos por lotes a partir de un parámetro con una propiedad de fecha y hora:

```
// Batch create from parameter
UNWIND $list as events
CREATE (n:crash) {time: datetime(events.time)}
// Parameter value
{
  "x":[
    {"time":"2024-01-01T23:59:29", "name":"crash1"},
    {"time":"2023-01-01T00:00:00Z", "name":"crash2"}
  ]
}
```

- Support para un subconjunto mayor de formatos de fecha y hora ISO8601. Consulte a continuación.

Formatos admitidos

El formato de un valor de fecha y hora es [Fecha] T [Hora] [Zona horaria], donde T es el separador. Si no se proporciona una zona horaria explícita, se asume que UTC (Z) es la predeterminada.

Zona horaria

Los formatos de zona horaria admitidos son:

- +/-HH:mm
- +/-HHmm
- +/-HH

La presencia de una zona horaria en una cadena de fecha y hora es opcional. En caso de que el desplazamiento de zona horaria sea 0, se puede usar Z en lugar del sufijo de zona horaria anterior para indicar la hora UTC. El rango admitido de una zona horaria es de - 14:00 a + 14:00.

Date

Si no hay ninguna zona horaria o la zona horaria es UTC (Z), los formatos de fecha admitidos son los siguientes:

> **Note**
>
> La DDD hace referencia a una fecha ordinal, que representa un día del año comprendido entre 001 y 365 (366 en años bisiestos). Por ejemplo, 2024-002 representa el 2 de enero de 2024.

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyyMM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Si se elige una zona horaria distinta de la Z, los formatos de fecha admitidos se limitan a los siguientes:

- `yyyy-MM-dd`
- `yyyy-DDD`
- `yyyyDDD`

El intervalo de fechas admitido es del 01/01/2001 al 31/12/9999.

Tiempo

Si no hay ninguna zona horaria o la zona horaria es UTC (Z), los formatos de hora admitidos son:

- `HH:mm:ss.SSS`
- `HH:mm:ss`
- `HHmmss.SSS`
- `HHmmss`
- `HH:mm`
- `HHmm`
- `HH`

Si se elige una zona horaria distinta de la Z, los formatos de hora admitidos se limitan a los siguientes:

- `HH:mm:ss`
- `HH:mm:ss.SSS`

Diferencias en la semántica del lenguaje openCypher de Neptune

Neptune representa los identificadores de nodos y relaciones como cadenas en lugar de enteros. El identificador es igual al identificador proporcionado a través del programa de carga de datos. Si hay un espacio de nombres para la columna, se utiliza el espacio de nombres más el identificador. En consecuencia, la función `id` devuelve una cadena en lugar de un número entero.

El tipo de datos `INTEGER` está limitado a 64 bits. Al convertir valores de cadena o coma flotante más grandes en un número entero mediante la función `TOINTEGER`, los valores negativos se truncan a `LLONG_MIN` y los valores positivos se truncan a `LLONG_MAX`.

Por ejemplo:

```
RETURN TOINTEGER(2^100)
>  9223372036854775807

RETURN TOINTEGER(-1 * 2^100)
>  -9223372036854775808
```

La función **`join()`** específica de Neptune

Neptune implementa una función `join()` que no está presente en la especificación de openCypher. Crea un literal de cadena a partir de una lista de literales de cadena y un delimitador de cadenas. Adopta dos argumentos:

- El primer argumento es una lista de literales de cadena.
- El segundo argumento es la cadena delimitadora, que puede tener cero, uno o más caracteres.

Ejemplo:

```
join(["abc", "def", "ghi"], ", ")    // Returns "abc, def, ghi"
```

La función **`removeKeyFromMap()`** específica de Neptune

Neptune implementa una función `removeKeyFromMap()` que no está presente en la especificación de openCypher. Elimina una clave específica de un mapa y devuelve el nuevo mapa resultante.

La función toma dos argumentos:

- El primer argumento es el mapa del que se debe eliminar la clave.
- El segundo argumento es la clave que hay que eliminar del mapa.

La función `removeKeyFromMap()` resulta especialmente útil en situaciones en las que se desean establecer valores para un nodo o una relación presentando una lista de mapas. Por ejemplo:

```
UNWIND [{`~id`: 'id1', name: 'john'}, {`~id`: 'id2', name: 'jim'}] as val
CREATE (n {`~id`: val.`~id`})
SET n = removeKeyFromMap(val, '~id')
```

Valores de identificadores personalizados para las propiedades de los nodos y las relaciones

A partir de la versión 1.2.0.2 del motor, Neptune ha ampliado la especificación de openCypher para que pueda especificar los valores `id` para los nodos y las relaciones en las cláusulas `CREATE`, `MERGE` y `MATCH`. Esto le permite asignar cadenas fáciles de usar en lugar de UUID generados por el sistema para identificar nodos y relaciones.

> **Warning**
>
> Esta extensión de la especificación de openCypher no es compatible con versiones anteriores, ya que ahora `~id` se considera un nombre de propiedad reservada. Si ya utiliza `~id` como propiedad en sus datos y consultas, tendrá que migrar la propiedad existente a una nueva clave de propiedad y eliminar la antigua. Consulte Qué hacer si actualmente utiliza `~id` como propiedad.

A continuación, tenemos un ejemplo que muestra cómo crear nodos y relaciones con identificadores personalizados:

```
CREATE (n {`~id`: 'fromNode', name: 'john'})
  -[:knows {`~id`: 'john-knows->jim', since: 2020}]
  ->(m {`~id`: 'toNode', name: 'jim'})
```

Si intenta crear un identificador personalizado que ya esté en uso, Neptune mostrará un error `DuplicateDataException`.

A continuación, se muestra un ejemplo del uso de un identificador personalizado en una cláusula `MATCH`:

```
MATCH (n {`~id`: 'id1'})
RETURN n
```

A continuación, se muestra un ejemplo del uso de identificadores personalizados en una cláusula `MERGE`:

```
MATCH (n {name: 'john'}), (m {name: 'jim'})
MERGE (n)-[r {`~id`: 'john->jim'}]->(m)
RETURN r
```

Qué hacer si actualmente utiliza **`~id`** como propiedad

Con la versión 1.2.0.2 del motor, la clave `~id` de las cláusulas de openCypher ahora se trata como `id` y no como una propiedad. Esto significa que si tiene una propiedad denominada `~id`, es imposible acceder a ella.

Si utiliza una propiedad `~id`, lo que debe hacer antes de actualizarla a la versión del motor `1.2.0.2` o posterior es migrar primero la propiedad `~id` existente a una nueva clave de propiedad y, a continuación, eliminar la propiedad `~id`. Por ejemplo, la siguiente consulta:

- Crea una nueva propiedad denominada 'newID' para todos los nodos,
- copia el valor de la propiedad '~id' en la propiedad 'newId'
- y elimina la propiedad '~id' de los datos.

```
MATCH (n)
WHERE exists(n.`~id`)
SET n.newId = n.`~id`
REMOVE n.`~id`
```

Se debe hacer lo mismo con cualquier relación en los datos que tenga una propiedad `~id`.

También tendrá que cambiar las consultas que esté utilizando que hagan referencia a una propiedad `~id`. Por ejemplo, esta consulta:

```
MATCH (n)
WHERE n.`~id` = 'some-value'
RETURN n
```

...se convertiría en esto:

```
MATCH (n)
WHERE n.newId = 'some-value'
RETURN n
```

Otras diferencias entre openCypher de Neptune y Cypher

- Neptune solo admite conexiones TCP para el protocolo de Bolt. WebSocketsno se admiten las conexiones para Bolt.
- openCypher de Neptune elimina los espacios en blanco tal como los define Unicode en las funciones `trim()`, `ltrim()` y `rtrim()`.
- En openCypher de Neptune, `tostring(`double`)` no cambia automáticamente a la notación E para valores grandes del doble.

- Aunque CREATE de openCypher no crea propiedades con varios valores, sí que pueden existir en los datos creados con Gremlin. Si openCypher de Neptune encuentra una propiedad con varios valores, uno de los valores se elige arbitrariamente, lo que da lugar a un resultado no determinista.

Modelo de datos de gráficos de Neptune.

La unidad básica de los datos de gráficos de Amazon Neptune es un elemento de cuatro posiciones (cuádruple), que es similar a un cuádruple del marco de descripción de recursos (RDF). A continuación, se indican las cuatro posiciones de un cuádruple de Neptune:

- `subject  (S)`
- `predicate  (P)`
- `object  (O)`
- `graph  (G)`

Cada cuádruple es una instrucción que realiza una afirmación sobre uno o varios recursos. Una instrucción puede afirmar la existencia de una relación entre dos recursos o puede asociar una propiedad (par clave/valor) a un recurso. Por lo general, puede considerar el valor de predicado de cuádruple como el verbo de la instrucción. Describe el tipo de relación o propiedad que se va a definir. El objeto es el objetivo de la relación o el valor de la propiedad. A continuación se muestran algunos ejemplos:

- Una relación entre dos vértices se puede representar almacenando el identificador del vértice de origen en la posición `S`, el identificador del vértice de destino en la posición `O` y la etiqueta de borde en la posición `P`.
- Una propiedad se puede representar almacenando el identificador del elemento en la posición `S`, la clave de propiedad en la posición `P` y el valor de la propiedad en la posición `O`.

La posición `G` del gráfico se utiliza de forma diferente en las distintas pilas. En el caso de los datos RDF de Neptune, la posición `G` contiene un identificador de gráfico con nombre. En los gráficos de propiedad en Gremlin, se utiliza para almacenar el valor del ID de borde en el caso de un borde. En todos los demás casos, se usa de forma predeterminada un valor fijo.

Un conjunto de instrucciones de cuádruples con identificadores de recursos compartidos crea un gráfico.

Diccionario de valores orientados al usuario

Neptune no almacena la mayoría de los valores orientados al usuario directamente en los distintos índices que mantiene. En su lugar, los almacena por separado en un diccionario y los reemplaza en los índices por identificadores de 8 bytes.

- Todos los valores orientados al usuario que se incluirían en los índices `S`, `P` o `G` se almacenan en el diccionario de esta forma.
- En el índice 0, los valores numéricos se almacenan directamente en el índice (en línea). Esto incluye los valores `date` y `datetime` (representados como milisegundos a partir de la época).
- Todos los demás valores orientados al usuario que se incluirían en el índice 0 se almacenan en el diccionario y se representan en el índice mediante identificadores.

El diccionario contiene un mapeo directo de los valores orientados al usuario a identificadores de 8 bytes en un índice `value_to_id`.

Almacena el mapeo inverso de los identificadores de 8 bytes a los valores de uno de los dos índices, según el tamaño de los valores:

- Un índice `id_to_value` mapea los identificadores a valores orientados al usuario que son inferiores a 767 bytes después de la codificación interna.
- Un índice `id_to_blob` mapea los identificadores a valores más grandes orientados al usuario.

Cómo se indexan las instrucciones en Neptune

Cuando se realiza una consulta en un gráfico de cuádruples, para cada posición de cuádruple puede especificar una restricción de valor o no. La consulta devuelve todos los cuádruples que coinciden con las restricciones de valor que ha especificado.

Neptune utiliza índices para resolver consultas. En el documento de 2005, Optimized Index Structures for Querying RDF from the Web, Andreas Harth y Stefan Decker observaron que hay 16 (2^4) patrones de acceso posibles para las cuatro posiciones del cuadrante. Puede consultar los 16 patrones de forma eficiente sin tener que escanear y filtrar utilizando seis índices de instrucciones cuádruples. Cada índice de instrucción cuádruple utiliza una clave compuesta por los cuatro valores de posición concatenados en un orden diferente.

```
     Access Pattern                                          Index key order
-------------------------------------------------    --------------
 1.  ????  (No constraints; returns every quad)            SPOG
 2.  SPOG  (Every position is constrained)                 SPOG
 3.  SPO?  (S, P, and O are constrained; G is not)         SPOG
 4.  SP??  (S and P are constrained; O and G are not)      SPOG
 5.  S???  (S is constrained; P, O, and G are not)         SPOG
```

```
 6.  S??G  (S and G are constrained; P and O are not)       SPOG

 7.  ?POG  (P, O, and G are constrained; S is not)          POGS
 8.  ?PO?  (P and O are constrained; S and G are not)       POGS
 9.  ?P??  (P is constrained; S, O, and G are not)          POGS

10.  ?P?G  (P and G are constrained; S and O are not)       GPSO
11.  SP?G  (S, P, and G are constrained; O is not)          GPSO
12.  ???G  (G is constrained; S, P, and O are not)          GPSO

13.  S?OG  (S, O, and G are constrained; P is not)          OGSP
14.  ??OG  (O and G are constrained; S and P are not)       OGSP
15.  ??O?  (O is constrained; S, P, and G are not)          OGSP

16.  S?O?  (S and O are constrained; P and G are not)       OSGP
```

De forma predeterminada, Neptune crea y mantiene solo tres de esos seis índices:

- `SPOG` – utiliza una clave compuesta de `Subject + Predicate + Object + Graph`.
- `POGS` – utiliza una clave compuesta de `Predicate + Object + Graph + Subject`.
- `GPSO` – utiliza una clave compuesta de `Graph + Predicate + Subject + Object`.

Estos tres índices administran muchos de los patrones de acceso más comunes. El mantenimiento de solo tres índices de instrucciones completos en lugar de seis reduce considerablemente los recursos que necesita para permitir un acceso rápido sin necesidad de escanear ni filtrar. Por ejemplo, el índice `SPOG` permite una búsqueda eficiente siempre que un prefijo de las posiciones, como el vértice o el identificador de vértice y propiedad, esté vinculado. El índice `POGS` permite un acceso eficiente cuando solo se vincula la etiqueta de borde o de propiedad almacenada en la posición `P`.

La API de bajo nivel para encontrar instrucciones toma un patrón de instrucción en el que algunas posiciones son conocidas y el resto se dejan para su detección por parte de la búsqueda por índice. Al componer las posiciones conocidas en un prefijo de clave de acuerdo con el orden de la clave de índice para uno de los índices de la instrucción, Neptune realiza un análisis de rango para recuperar todas las instrucciones que coincidan con las posiciones conocidas.

Sin embargo, uno de los índices de la instrucción que Neptune no crea de forma predeterminada es un índice `OSGP` de recorrido inverso, que puede reunir predicados en objetos y sujetos. En su lugar, Neptune realiza de forma predeterminada un seguimiento de predicados diferentes en un índice

independiente que utiliza para realizar un análisis de unión de `{all P x POGS}`. Cuando trabaja con Gremlin, un predicado se corresponde con una propiedad o una etiqueta de borde.

La estrategia de acceso predeterminada de Neptune puede ser ineficiente si el número de predicados diferentes de un gráfico se vuelve grande. En Gremlin, por ejemplo, un paso `in()` en el que no se proporcionan etiquetas de borde, o cualquier paso que utilice `in()` internamente como `both()` o `drop()`, puede resultar bastante ineficiente.

Habilitación de la creación de índices OSGP mediante el modo lab

Si el modelo de datos crea un gran número de predicados distintos, es posible que experimente una reducción del rendimiento y un aumento de los costos operativos que se pueden mejorar considerablemente utilizando el modo de laboratorio para habilitar el índice OSGP además de los tres índices que Neptune mantiene de manera predeterminada.

> **Note**
>
> Esta característica está disponible a partir de la versión 1.0.1.0.200463.0 del motor de Neptune.

Habilitar el índice OSGP puede tener algunos inconvenientes:

- La tasa de inserción puede disminuir hasta un 23 %.
- El almacenamiento aumenta hasta un 20 %.
- Las consultas de lectura que modifican todos los índices por igual (lo cual es bastante raro) pueden tener mayores latencias.

Sin embargo, en general merece la pena habilitar el índice OSGP para clústeres de base de datos con un gran número de predicados distintos. Las búsquedas basadas en objetos se vuelven muy eficientes (por ejemplo, encontrar todos los bordes entrantes en un vértice o todos los sujetos conectados a un objeto determinado) y, como resultado, la eliminación de vértices también se vuelve mucho más eficiente.

> **Important**
>
> Solo puede habilitar el índice OSGP en un clúster de base de datos vacío, antes de cargar cualquier dato en él.

Instrucciones de Gremlin en el modelo de datos de Neptune

Los datos de gráficos de propiedades de Gremlin se expresan en el modelo SPOG mediante tres clases de instrucciones:

- Instrucciones de etiqueta de vértice
- Instrucciones de borde
- Instrucciones de propiedades

Para saber cómo se utilizan en las consultas de Gremlin, consulte Descripción de cómo funcionan las consultas de Gremlin en Neptune.

La caché de búsqueda de Neptune puede acelerar las consultas de lectura

Amazon Neptune implementa una caché de búsqueda que utiliza el SSD basado en NVMe de la instancia `R5d` para mejorar el rendimiento de lectura de las consultas con búsquedas frecuentes y repetitivas de valores de propiedades o literales RDF. La caché de búsqueda almacena temporalmente estos valores en el volumen del SSD NVMe, donde se puede acceder a ellos rápidamente.

Esta característica solo está disponible a partir de Versión 1.0.4.2.R2 del motor de Amazon Neptune (01/06/2021).

Las consultas de lectura que devuelven las propiedades de un gran número de vértices y bordes, o de muchos triples RDF, pueden tener una latencia alta si es necesario recuperar valores o literales de las propiedades de los volúmenes de almacenamiento del clúster y no de la memoria. Entre algunos ejemplos, se incluyen las consultas de lectura de larga duración que devuelven una gran cantidad de nombres completos de un gráfico de identidad o de direcciones IP de un gráfico de detección de fraudes. A medida que aumenta el número de valores de propiedades o literales RDF que devuelve la consulta, la memoria disponible disminuye y la ejecución de la consulta puede degradarse considerablemente.

Casos de uso de la caché de búsqueda de Neptune

La caché de búsqueda solo ayuda cuando las consultas de lectura devuelven las propiedades de un gran número de vértices y bordes o de triples RDF.

Para optimizar el rendimiento de las consultas, Amazon Neptune usa el tipo de instancia `R5d` para crear una caché grande para dichos valores de propiedad o literales. Por lo tanto, recuperarlos de la memoria caché es mucho más rápido que recuperarlos de los volúmenes de almacenamiento del clúster.

Como norma general, solo vale la pena habilitar la caché de búsqueda si se cumplen las tres condiciones siguientes:

- Ha observado un aumento de la latencia en las consultas de lectura.
- También observa una caída en la `BufferCacheHitRatio` CloudWatch métrica al ejecutar consultas de lectura (consulteMonitorización de Neptune con Amazon CloudWatch).

- Sus consultas de lectura dedican mucho tiempo a materializar los valores devueltos antes de representarlos (consulte el ejemplo del perfil de Gremlin que aparece a continuación para determinar cuántos valores de propiedades se están materializando para una consulta).

> **Note**
>
> Esta característica solo es útil en el escenario específico descrito anteriormente. Por ejemplo, la caché de búsqueda no ayuda en absoluto en las consultas de agregación. A menos que esté ejecutando consultas que podrían beneficiarse de la caché de búsqueda, no hay ninguna razón para usar un tipo de instancia R5d en lugar de un tipo de instancia R5 equivalente y menos costosa.

Si utiliza Gremlin, puede evaluar los costos de materialización de una consulta con API `profile` de Gremlin. En la sección “Operaciones indexadas”, se muestra el número de términos que se materializaron durante la ejecución:

```
Index Operations
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 5273
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 32693
```

El número de términos no numéricos que se materializan es directamente proporcional al número de búsquedas de términos que Neptune debe realizar.

Uso de la caché de búsqueda

La caché de búsqueda solo está disponible en un tipo de instancia R5d, donde se habilita automáticamente de forma predeterminada. Las instancias R5d de Neptune tienen las mismas especificaciones que las instancias R5, además de hasta 1,8 TB de almacenamiento SSD local basado en NVMe. Las cachés de búsqueda son específicas de cada instancia y las cargas de trabajo

que se benefician se pueden dirigir específicamente a las instancias R5d de un clúster de Neptune, mientras que otras cargas de trabajo se pueden dirigir a R5 u otros tipos de instancias.

Para usar la caché de búsqueda en una instancia de Neptune, solo tiene que actualizar esa instancia al tipo de instancia R5d. Al hacerlo, Neptune establece automáticamente el parámetro del clúster de base de datos neptune_lookup_cache en `'enabled'` y crea la caché de búsqueda en esa instancia concreta. A continuación, puede usar la API Estado de la instancia para confirmar que la caché está habilitada.

Del mismo modo, para deshabilitar la caché de búsqueda en una instancia determinada, escale verticalmente la instancia de un tipo de instancia R5d a un tipo de instancia R5 equivalente.

Cuando se lanza una instancia R5d, la caché de búsqueda se habilita en modo de arranque en frío, lo que significa que está vacía. Neptune comprueba primero en la caché de búsqueda los valores de las propiedades o los literales RDF mientras procesa las consultas y los añade si aún no están presentes. Esto calienta gradualmente la memoria caché.

Al dirigir las consultas de lectura que requieren búsquedas de valores de propiedad o literales RDF a una instancia de lector R5d, el rendimiento de lectura se reduce ligeramente mientras la caché se calienta. Sin embargo, cuando la caché se calienta, el rendimiento de lectura se acelera considerablemente y es posible que también se produzca una disminución en los costos de E/S debido a que las búsquedas se realizan en la memoria caché y no en el almacenamiento en clúster. La utilización de la memoria también mejora.

Si la instancia de escritor es R5d, calienta su caché de búsqueda automáticamente en cada operación de escritura. Este enfoque aumenta ligeramente la latencia de las consultas de escritura, pero calienta la caché de búsqueda de manera más eficiente. A continuación, si dirige las consultas de lectura que requieren búsquedas de valores de propiedades o literales RDF a la instancia del escritor, el rendimiento de lectura mejorará inmediatamente, ya que los valores ya se han almacenado en caché allí.

Además, si ejecuta el programa de carga masiva en una instancia de escritor R5d, es posible que note que su rendimiento se reduce ligeramente debido a la caché.

Como la caché de búsqueda es específica para cada nodo, la sustitución del host restablece la caché para que se inicie en frío.

Puede deshabilitar temporalmente la caché de búsqueda en todas las instancias del clúster de base de datos configurando el parámetro del clúster de base de datos neptune_lookup_cache

en `'disabled'`. Sin embargo, en general, tiene más sentido deshabilitar la caché en instancias específicas reduciéndolas verticalmente del tipo de instancia `R5d` a `R5`.

Semántica de transacciones en Neptune

Amazon Neptune está diseñado para admitir cargas de trabajo de procesamiento transaccional en línea (OLTP) enormemente simultáneas a través de gráficos de datos. La especificación del lenguaje de consultas SPARQL para RDF del W3C y la documentación del lenguaje transversal de gráficos de Apache TinkerPop Gremlin no definen la semántica de las transacciones para el procesamiento simultáneo de consultas. Puesto que la compatibilidad con ACID y las garantías de transacciones bien definidas pueden ser muy importantes, hemos aplicado una semántica estricta para ayudarle a evitar anomalías.

En esta sección, se define dicha semántica y se muestra cómo se aplican a varios casos de uso comunes en Neptune.

Temas

Definición de niveles de aislamiento

La "I" de `ACID` significa aislamiento. El grado de aislamiento de una transacción determina el grado en el que otras transacciones simultáneas pueden afectar a los datos en los que opera.

El estándar SQL:1992 creó un vocabulario para describir los niveles de aislamiento. Define tres tipos de interacciones (que denomina fenómenos) que pueden producirse entre dos transacciones simultáneas, `Tx1` y `Tx2`:

- `Dirty read`: esto ocurre cuando `Tx1` modifica un elemento y, a continuación, `Tx2` lee ese elemento antes de que `Tx1` haya realizado el cambio. A continuación, si `Tx1` nunca consigue confirmar el cambio o lo restaura, `Tx2` leerá un valor que nunca leyó en la base de datos.
- `Non-repeatable read`: esto ocurre cuando `Tx1` lee un elemento y, a continuación, `Tx2` lo modifica o elimina, confirma el cambio y, a continuación, `Tx1` intenta volver a leer el elemento. `Tx1` ahora lee un valor diferente al anterior o descubre que el elemento ya no existe.
- `Phantom read`: esto ocurre cuando `Tx1` lee un conjunto de elementos que cumplen un criterio de búsqueda y, a continuación, `Tx2` añade un nuevo elemento que cumple el criterio de búsqueda y

luego `Tx1` repite la búsqueda. `Tx1` ahora obtiene un conjunto de elementos diferente al que tenía antes.

Cada uno de estos tres tipos de interacción puede provocar incoherencias en los datos resultantes en una base de datos.

El estándar SQL:1992 definió cuatro niveles de aislamiento que tienen diferentes garantías en cuanto a los tres tipos de interacción y las incoherencias que pueden producir. En los cuatro niveles, se puede garantizar que una transacción se ejecute por completo o que no se ejecute en absoluto:

- `READ UNCOMMITTED`: permite los tres tipos de interacciones (es decir, lecturas incorrectas, lecturas no repetibles y lecturas fantasma).
- `READ COMMITTED`: las lecturas incorrectas no son posibles, pero sí las no repetibles y las lecturas fantasma.
- `REPEATABLE READ`: las lecturas incorrectas y las lecturas no repetibles no son correctas, pero las lecturas fantasmas sí lo son.
- `SERIALIZABLE`: no se puede producir ninguno de los tres tipos de fenómenos de interacción.

El control de simultaneidad multiversión (MVCC) permite otro tipo de aislamiento, es decir, el aislamiento SNAPSHOT . Esto garantiza que una transacción opera en una instantánea de datos en el momento en el que aparece cuando comienza la transacción y que ninguna otra transacción puede cambiar esa instantánea.

Niveles de aislamiento de transacciones en Neptune

Amazon Neptune implementa diferentes niveles de aislamiento de transacciones para consultas de solo lectura y para consultas de mutación. Las consultas SPARQL y Gremlin se clasifican como de solo lectura o mutación en función de los siguientes criterios:

- En SPARQL, existe una distinción clara entre las consultas de lectura (`SELECT`, `ASK`, `CONSTRUCT` y `DESCRIBE` según se define en la especificación del lenguaje de consulta SPARQL 1.1) y las consultas de mutación (`INSERT` y `DELETE` tal como se define en la especificación SPARQL 1.1 Update).

  Tenga en cuenta que Neptune trata varias consultas de mutación que se envían juntas (por ejemplo, en un mensaje `POST`, separadas por punto y coma) como una sola transacción. Se

garantiza que se efectúan de forma correcta o incorrecta como una unidad atómica y, en caso de fallo, se revierten los cambios parciales.

- Sin embargo, en Gremlin, Neptune considera una consulta como de solo lectura o de mutación en función de si contiene algún paso de ruta de consulta como `addE()`, `addV()`, `property()` o `drop()` que manipula los datos. Si la consulta contiene este paso de ruta, se clasifica y ejecuta como una consulta de mutación.

También es posible utilizar sesiones activas en Gremlin. Para obtener más información, consulte Sesiones basadas en scripts de Gremlin. En estas sesiones, todas las consultas, incluidas las de solo lectura, se ejecutan con el mismo aislamiento que las consultas de mutación en el punto de conexión del escritor.

Al utilizar sesiones de lectura-escritura tipo bolt en openCypher, todas las consultas, incluidas las de solo lectura, se ejecutan con el mismo aislamiento que las consultas de mutación, en el punto de conexión del escritor.

Temas

- Aislamiento de consultas de solo lectura en Neptune
- Aislamiento de consultas de mutación en Neptune
- Resolución de conflictos mediante tiempos de espera de bloqueo
- Bloqueos de rango y conflictos falsos

Aislamiento de consultas de solo lectura en Neptune

Neptune evalúa las consultas de solo lectura con semántica de aislamiento de instantáneas. Esto significa que una consulta de solo lectura opera de forma lógica en una instantánea coherente de la base de datos realizada cuando comienza la evaluación de la consulta. Neptune puede garantizar entonces que no se produzcan los siguientes fenómenos:

- `Dirty reads`: las consultas de solo lectura en Neptune nunca tendrán datos no confirmados de una transacción simultánea.
- `Non-repeatable reads`: una transacción de solo lectura que lea los mismos datos más de una vez siempre obtendrá los mismos valores.
- `Phantom reads`: una transacción de solo lectura nunca leerá los datos que se hayan añadido después del inicio de la transacción.

Puesto que el aislamiento de instantáneas se consigue mediante el control de simultaneidad multiversión (MVCC), las consultas de solo lectura no tienen necesidad de bloquear datos y, por lo tanto, no bloquean las consultas de mutación.

Las réplicas de lectura solo aceptan consultas de solo lectura, por lo que todas las consultas de réplicas de lectura se ejecutan bajo semántica de aislamiento `SNAPSHOT`.

La única consideración adicional a la hora de consultar una réplica de lectura es que puede haber un pequeño retraso de replicación entre el escritor y las réplicas de lectura. Esto significa que una actualización realizada en el escritor puede tardar poco tiempo en propagarse a la réplica de lectura desde la que está leyendo. El tiempo de replicación real depende de la carga de escritura en la instancia principal. La arquitectura Neptune admite la replicación de baja latencia y el retraso de la replicación se instrumenta en una métrica de Amazon. CloudWatch

Sin embargo, debido al nivel de aislamiento de `SNAPSHOT`, las consultas de lectura siempre ven un estado coherente de la base de datos, incluso si no es el más reciente.

En los casos en los que necesite una garantía sólida de que una consulta observe el resultado de una actualización anterior, envíe la consulta al propio punto de enlace de escritor en lugar de a una réplica de lectura.

Aislamiento de consultas de mutación en Neptune

Las lecturas realizadas como parte de las consultas de mutación se ejecutan con aislamiento de transacciones `READ COMMITTED`, lo que descarta la posibilidad de lecturas sucias. Al ir más allá de las garantías habituales proporcionadas para el aislamiento de transacciones `READ COMMITTED`, Neptune proporciona la garantía sólida de que no pueden producirse lecturas `NON-REPEATABLE` ni `PHANTOM`.

Estas sólidas garantías se logran bloqueando registros y rangos de registros al leer datos. Esto impide que las transacciones simultáneas realicen inserciones o eliminaciones en rangos de índice después de leerlos, lo que garantiza lecturas repetibles.

> **Note**
>
> Sin embargo, podría comenzar una transacción de mutación simultánea `Tx2` después del inicio de la transacción `Tx1` de mutación, y podría confirmar un cambio antes de que `Tx1` hubiera bloqueado los datos para leerlos. En ese caso, `Tx1` vería el cambio de `Tx2` como si

`Tx2` se hubiera completado antes de que se iniciara `Tx1`. Puesto que esto solo se aplica a los cambios confirmados, no se podría producir nunca `dirty read`.

Para comprender el mecanismo de bloqueo que utiliza Neptune para las consultas de mutación, primero es útil comprender los detalles de Modelo de datos de gráficos y Estrategia de indexación de Neptune. Neptune administra los datos mediante tres índices: `SPOG`, `POGS` y `GPSO`.

Para conseguir lecturas repetibles para el nivel de transacción `READ COMMITTED`, Neptune toma bloqueos de rango en el índice que se está utilizando. Por ejemplo, si una consulta de mutación lee todas las propiedades y los bordes salientes de un vértice denominado `person1`, el nodo bloquearía todo el rango definido por el prefijo `S=person1` en el índice `SPOG` antes de leer los datos.

Se aplica el mismo mecanismo cuando se utilizan otros índices. Por ejemplo, cuando una transacción de mutación busca todos los pares de vértices de origen-destino para una determinada etiqueta de borde mediante el índice `POGS`, el rango de la etiqueta de borde en la posición `P` se bloquearía. Cualquier transacción simultánea, independientemente de si se trata de una consulta de solo lectura o de mutación, podría realizar lecturas dentro del rango bloqueado. Sin embargo, cualquier mutación que implique la inserción o eliminación de registros nuevos en el rango de prefijos bloqueados requeriría un bloqueo exclusivo y se evitaría.

En otras palabras, cuando una transacción de mutación ha leído un rango del índice, existe una gran garantía de que este rango no se modificará mediante ninguna transacción simultánea hasta el final de la transacción de lectura. Esto garantiza que no se producirá `non-repeatable reads`.

Resolución de conflictos mediante tiempos de espera de bloqueo

Si una segunda transacción intenta modificar un registro en un rango que ha bloqueado una primera transacción, Neptune detecta el conflicto inmediatamente y bloquea la segunda transacción.

Si no se detecta un interbloqueo de dependencias, Neptune aplica automáticamente un mecanismo de tiempo de espera de bloqueo, en el que la transacción bloqueada espera hasta 60 segundos a que la transacción que contiene el bloqueo finalice y libere el bloqueo.

- Si el tiempo de espera de bloqueo vence antes de que se libere el bloqueo, la transacción bloqueada se revisa.
- Si el bloqueo se libera dentro del tiempo de espera del bloqueo, la segunda transacción se desbloquea y puede finalizar correctamente sin necesidad de reintentarlo.

Sin embargo, si Neptune detecta un interbloqueo de dependencia entre las dos transacciones, no es posible que se produzca una conciliación automática del conflicto. En este caso, Neptune cancela y revierte inmediatamente una de las dos transacciones sin iniciar un tiempo de espera de bloqueo. Neptune hace todo lo posible para revertir la transacción que tiene el menor número de registros insertados o eliminados.

Bloqueos de rango y conflictos falsos

Neptune toma bloqueos de alcance utilizando bloqueos de espacio. Un bloqueo de espacio consiste en bloquear un espacio entre los registros del índice o bloquear el espacio antes del primer registro de índice o después del último.

Neptune usa lo que se denomina una tabla de diccionario para asociar valores de identificadores numéricos con literales de cadena específicos. Aquí tenemos un ejemplo del estado de una tabla de diccionario de Neptune:

Cadena	ID
type	1
default_graph	2
person_3	3
person_1	5
knows	6
person_2	7
age	8
edge_1	9
lives_in	10
Nueva York	11
Persona	12
Place	13

Cadena	ID
edge_2	14

Las cadenas anteriores pertenecen a un modelo de gráficos de propiedades, pero los conceptos se aplican por igual a todos los modelos de gráficos RDF.

El estado correspondiente del índice SPOG (sujeto-predicado-objeto-gráfico) se muestra abajo, a la izquierda. A la derecha, se muestran las cadenas correspondientes para ayudar a entender el significado de los datos del índice.

S (ID)	P (ID)	O (ID)	G (ID)		S (cadena)	P (cadena)	O (cadena)	G (cadena)
3	1	12	2		person_3	type	Persona	default_graph
5	1	12	2		person_1	type	Persona	default_graph
5	6	3	9		person_1	knows	person_3	edge_1
5	8	40	2		person_1	age	40	default_graph
5	10	11	14		person_1	lives_in	Nueva York	edge_2
7	1	12	2		person_2	type	Persona	default_graph
11	1	13	2		Nueva York	type	Place	default_graph

Ahora, si una consulta de mutación leyera todas las propiedades y los bordes salientes de un vértice denominado `person_1`, el nodo bloquearía todo el rango definido por el prefijo `S=person_1` en el índice SPOG antes de leer los datos. El bloqueo de rango colocaría bloqueos de espacio en todos los registros coincidentes y en el primer registro que no coincida. Los registros coincidentes se

bloquearían y los no coincidentes no se bloquearían. Neptune colocaría los espacios de la siguiente manera:

- `5 1 12 2` (espacio 1)
- `5 6 3 9` (espacio 2)
- `5 8 40 2` (espacio 3)
- `5 10 11 14` (espacio 4)
- `7 1 12 2` (espacio 5)

Esto bloquea los siguientes registros:

- `5 1 12 2`
- `5 6 3 9`
- `5 8 40 2`
- `5 10 11 14`

En este estado, las siguientes operaciones se bloquean legítimamente:

- Inserción de una nueva propiedad o borde para `S=person_1`. Una nueva propiedad que no sea `type` o un borde nuevo tendrían que ir en el espacio 2, el espacio 3, el espacio 4 o el espacio 5, todos los cuales están bloqueados.
- Eliminación de cualquiera de los registros existentes.

Al mismo tiempo, algunas operaciones simultáneas se bloquearían falsamente (lo que generaría conflictos falsos):

- Todas las inserciones de propiedades o bordes de `S=person_3` se bloquean porque deberían estar en el espacio 1.
- Cualquier inserción de un nuevo vértice a la que se le asigne un identificador entre 3 y 5 se bloqueará porque tendría que ir en el espacio 1.
- Cualquier inserción de un nuevo vértice a la que se le asigne un identificador entre 5 y 7 se bloqueará porque tendría que ir en el espacio 5.

Los bloqueos de espacios no son lo suficientemente precisos como para bloquear el espacio de un predicado específico (por ejemplo, para bloquear el espacio 5 para el predicado `S=5`).

Los bloqueos de rango solo se colocan en el índice donde se produce la lectura. En el caso anterior, los registros se bloquean únicamente en el índice SPOG, no en POGS ni GPSO. Las lecturas de una consulta se pueden realizar en todos los índices en función de los patrones de acceso, que se pueden enumerar mediante las API de `explain` (para Sparql y Gremlin).

> **Note**
>
> También se pueden utilizar bloqueos de espacios para garantizar la seguridad de las actualizaciones simultáneas de los índices subyacentes, lo que también puede provocar conflictos falsos. Estos bloqueos de espacios se colocan independientemente del nivel de aislamiento o de las operaciones de lectura realizadas por la transacción.

Los conflictos falsos pueden producirse no solo cuando las transacciones simultáneas colisionan debido a bloqueos de espacios, sino también, en algunos casos, cuando se vuelve a intentar realizar una transacción tras algún tipo de error. Si la reversión provocada por el error sigue en curso y los bloqueos previamente realizados para la transacción aún no se han liberado por completo, al reintentarlo se detectará un conflicto falso y se producirá un error.

Con una carga elevada, normalmente se produce un error en el 3-4 % de las consultas de escritura debido a conflictos falsos. En el caso de un cliente externo, estos conflictos falsos son difíciles de predecir y deben solucionarse mediante reintentos.

Ejemplos de semántica de transacciones de Neptune

Los siguientes ejemplos muestran diferentes casos de uso de semántica de transacciones en Amazon Neptune.

Temas

Ejemplo 1: inserción de una propiedad solo si no existe

Supongamos que desea asegurarse de que una propiedad se establezca solo una vez. Por ejemplo, suponga que varias consultas intentan asignar a una persona una puntuación de crédito de forma simultánea. Solo desea que se inserte una instancia de la propiedad y que las demás consultas devuelvan un error, ya que la propiedad ya se ha establecido.

```
# GREMLIN:
g.V('person1').hasLabel('Person').coalesce(has('creditScore'), property('creditScore',
 'AAA+'))

# SPARQL:
INSERT { :person1 :creditScore "AAA+" .}
WHERE  { :person1 rdf:type :Person .
         FILTER NOT EXISTS { :person1 :creditScore ?o .} }
```

El el paso `property()` de Gremlin, se inserta una propiedad con la clave y el valor determinados. En el paso `coalesce()`, se ejecuta el primer argumento del primer paso y, si se produce un error, se ejecuta el segundo paso:

Antes de insertar el valor de la propiedad `creditScore` para un vértice `person1` determinado, una transacción debe intentar leer el posible valor `creditScore` inexistente de `person1`. Este intento de lectura bloquea el rango SP de `S=person1` y `P=creditScore` en el índice`SPOG`, donde el valor `creditScore` existe o se va a sobrescribir.

Si se utiliza este bloqueo de rango, se impide que cualquier transacción simultánea inserte un valor `creditScore` de forma simultánea. Cuando hay varias transacciones en paralelo, como máximo una de ellas puede actualizar el valor a la vez. Esto descarta la anomalía de más de una propiedad `creditScore` que se crea.

Ejemplo 2: afirmación de que el valor de una propiedad es único de forma global

Supongamos que desea insertar una persona con un número de la Seguridad Social como clave principal. Desea que su consulta de mutación garantice que, a nivel global, nadie más de la base de datos tenga el mismo número de Seguridad Social:

```
# GREMLIN:
g.V().has('ssn', 123456789).fold()
  .coalesce(__.unfold(),
            __.addV('Person').property('name', 'John Doe').property('ssn', 123456789'))
```

```
# SPARQL:
INSERT { :person1 rdf:type :Person .
         :person1 :name "John Doe" .
         :person1 :ssn 123456789 .}
WHERE  { FILTER NOT EXISTS { ?person :ssn 123456789 } }
```

Este ejemplo es similar al anterior. La principal diferencia es que el bloqueo de rango se realiza en el índice `POGS` en lugar de en el índice `SPOG`.

La transacción que ejecuta la consulta debe leer el patrón, `?person :ssn 123456789`, en el que están vinculadas las posiciones `P` y `0`. El bloqueo de rango se realiza en el índice `POGS` para `P=ssn` y `O=123456789`.

- Si el patrón existe, no se realiza ninguna acción.
- Si no existe, el bloqueo impide que cualquier transacción simultánea inserte también ese número de Seguridad Social.

Ejemplo 3: cambio de una propiedad si otra propiedad tiene un valor específico

Supongamos que varios eventos de un juego mueven a una persona del nivel uno al nivel dos y les asignan una nueva propiedad `level2Score` establecida en cero. Debe asegurarse de que varias instancias simultáneas de dicha transacción no puedan crear varias instancias de la propiedad de puntuación de nivel dos. Las consultas en Gremlin y SPARQL podrían tener un aspecto similar al siguiente.

```
# GREMLIN:
g.V('person1').hasLabel('Person').has('level', 1)
 .property('level2Score', 0)
 .property(Cardinality.single, 'level', 2)

# SPARQL:
DELETE { :person1 :level 1 .}
INSERT { :person1 :level2Score 0 .
         :person1 :level 2 .}
WHERE  { :person1 rdf:type :Person .
         :person1 :level 1 .}
```

En Gremlin, cuando se especifica `Cardinality.single`, el paso `property()` añade una nueva propiedad o sustituye un valor de propiedad existente por el nuevo valor que se especifica.

Cualquier actualización de un valor de propiedad, como aumentar `level` de 1 a 2, se implementa como una eliminación del registro actual y la inserción de un nuevo registro con el nuevo valor de propiedad. En este caso, se elimina el registro con el número de nivel 1 y se vuelve a insertar un registro con el número de nivel 2.

Para que la transacción pueda añadir `level2Score` y actualizar `level` de 1 a 2, primero debe validar que el valor `level` sea actualmente igual a 1. Al hacerlo, toma un bloqueo de rango en el prefijo SPO de `S=person1`, `P=level` y `O=1` en el índice SPOG. Este bloqueo impide que las transacciones simultáneas eliminen el triple de la versión 1 y, como resultado, no se pueden producir actualizaciones simultáneas en conflicto.

Ejemplo 4: sustitución de una propiedad existente

Algunos eventos pueden actualizar la puntuación de crédito de una persona a un nuevo valor (aquí BBB). Sin embargo, desea asegurarse de que los eventos simultáneos de ese tipo no puedan crear varias propiedades de puntuación de crédito para una persona.

```
# GREMLIN:
g.V('person1').hasLabel('Person')
 .sideEffect(properties('creditScore').drop())
 .property('creditScore', 'BBB')

# SPARQL:
DELETE { :person1 :creditScore ?o .}
INSERT { :person1 :creditScore "BBB" .}
WHERE  { :person1 rdf:type :Person .
         :person1 :creditScore ?o .}
```

Este caso es similar al ejemplo 3, salvo que en lugar de bloquear el prefijo SPO, Neptune bloquea el prefijo SP solo con `S=person1` y `P=creditScore`. Esto impide que las transacciones simultáneas inserten o eliminen triples con la propiedad `creditScore` para el asunto `person1`.

Ejemplo 5: evitar propiedades o bordes pendientes

La actualización de una entidad no debe dejar un elemento pendiente, es decir, una propiedad o borde asociado a una entidad que no esté escrita. Esto solo es un problema en SPARQL, ya que Gremlin tiene restricciones integradas para evitar elementos pendientes.

```
# SPARQL:
```

```
tx1: INSERT { :person1 :age 23 } WHERE { :person1 rdf:type :Person }
tx2: DELETE { :person1 ?p ?o }
```

La consulta `INSERT` debe leer y bloquear el prefijo `SPO` con `S=person1`, `P=rdf:type` y `O=Person` en el índice `SPOG`. El bloqueo impide que la consulta `DELETE` se realice correctamente en paralelo.

En la carrera entre la consulta `DELETE` que intenta eliminar el registro `:person1 rdf:type :Person` y la consulta `INSERT` que lee el registro y crea un bloqueo de rango en su `SPO` en el índice `SPOG`, son posibles los siguientes resultados:

- Si la consulta `INSERT` se confirma antes de que la consulta `DELETE` lea y elimine todos los registros de `:person1`, `:person1` se elimina en su totalidad de la base de datos, incluido el registro recién insertado.
- Si la consulta `DELETE` se confirma antes de que la consulta `INSERT` intente leer el registro `:person1 rdf:type :Person`, la lectura observa el cambio confirmado. Es decir, no encuentra ningún registro `:person1 rdf:type :Person` y, por lo tanto, se convierte en una instrucción no-op (sin operación).
- Si la consulta `INSERT` se lee antes que la consulta `DELETE`, el triple `:person1 rdf:type :Person` se bloquea y la consulta `DELETE` se bloquea hasta que se confirma la consulta INSERT, como en el primer caso anterior.
- Si `DELETE` se lee antes que la consulta `INSERT` y la consulta `INSERT` intenta leer y bloquear el prefijo `SPO` del registro, se detecta un conflicto. Esto se debe a que el triple se ha marcado para su eliminación y, por lo tanto, se produce un fallo en `INSERT`.

En todas estas diferentes secuencias posibles de eventos, no se crea ningún borde pendiente.

Control de excepciones y reintentos

Cuando las transacciones se cancelan debido a conflictos que no se pueden resolver o a tiempos de espera de bloqueo, Amazon Neptune responde con un `ConcurrentModificationException`. Para obtener más información, consulte Códigos de error del motor. Como práctica recomendada, los clientes siempre deben detectar y gestionar estas excepciones.

En muchos casos, cuando el número de instancias `ConcurrentModificationException` es bajo, un mecanismo de reintento exponencial basado en retardo funciona bien como una forma de gestionarlas. En este enfoque de reintento, el número máximo de reintentos y el tiempo de espera suele depender del tamaño y la duración máximos de las transacciones.

Sin embargo, si la aplicación tiene cargas de trabajo de actualización muy simultáneas y observa un gran número de eventos `ConcurrentModificationException`, es posible que pueda modificar la aplicación para reducir el número de modificaciones simultáneas en conflicto.

Por ejemplo, elijamos una aplicación que realiza actualizaciones frecuentes en un conjunto de vértices y utiliza varios subprocesos simultáneos para estas actualizaciones con el fin de optimizar el rendimiento de escritura. Si cada subproceso ejecuta continuamente consultas que actualizan una o varias propiedades de nodo, las actualizaciones simultáneas del mismo nodo pueden producir `ConcurrentModificationExceptions`. Esto, a su vez, puede reducir el rendimiento de escritura.

Puede disminuir en gran medida la probabilidad de este tipo de colisiones si puede serializar actualizaciones que es probable que entren en conflicto entre sí. Por ejemplo, si puede asegurarse de que todas las consultas de actualización de un nodo determinado se realicen en el mismo subproceso (quizás mediante una asignación basada en hash), puede estar seguro de que se ejecutarán una tras otra en lugar de simultáneamente. Aunque sigue siendo posible que un bloqueo de rango tomado en un nodo vecino pueda provocar una exepción `ConcurrentModificationException`, se eliminarán las actualizaciones simultáneas del mismo nodo.

Clústeres e instancias de base de datos de Amazon Neptune

Un clúster de base de datos de Amazon Neptune administra el acceso a sus datos mediante consultas. Un clúster se compone de:

- Una instancia de base de datos principal .
- Hasta 15 instancias de base de datos de réplica de lectura.

Todas las instancias de un clúster comparten el mismo volumen de almacenamiento administrado subyacente, que está diseñado para ofrecer fiabilidad y alta disponibilidad.

Para conectarse a las instancias de base de datos de su clúster de base de datos, debe utilizar los puntos de conexión de Neptune.

La instancia de base de datos principal de un clúster de base de datos de Neptune

La instancia de base de datos principal coordina todas las operaciones de escritura en el volumen de almacenamiento subyacente del clúster de base de datos. También admite operaciones de lectura.

Solo puede haber una instancia de base de datos principal en un clúster de base de datos de Neptune. Si la instancia principal deja de estar disponible, Neptune conmuta automáticamente por error a una de las instancias de réplica de lectura con una prioridad que puede especificar.

Instancias de base de datos de réplica de lectura en un clúster de base de datos de Neptune

Después de crear la instancia principal de un clúster de base de datos, puede crear un máximo de 15 réplicas de lectura en el clúster de base de datos para permitir las consultas de solo lectura.

Las instancias de base de datos de réplica de lectura de Neptune funcionan bien para escalar la capacidad de lectura porque están totalmente dedicadas a operaciones de lectura en el volumen de su clúster. Todas las operaciones de escritura se administran en la instancia principal. Cada instancia de base de datos de réplica de lectura tiene su propio punto de conexión.

Como el volumen de almacenamiento del clúster se comparte entre todas las instancias de un clúster, todas las instancias de réplica de lectura devuelven los mismos datos para los resultados de

las consultas con un retardo de replicación muy reducido. Este retardo suele ser muy inferior a 100 milisegundos después de que la instancia principal escriba una actualización, aunque puede ser algo mayor cuando el volumen de operaciones de escritura es muy grande.

Tener una o más instancias de réplica de lectura disponibles en distintas zonas de disponibilidad puede aumentar la disponibilidad, ya que las réplicas de lectura sirven como destinos de conmutación por error para la instancia principal. Es decir, si la instancia principal falla, Neptune promueve una instancia de réplica de lectura para convertirla en la instancia primaria. Cuando esto ocurre, se produce una breve interrupción mientras se reinicia la instancia promovida, durante la cual las solicitudes de lectura y escritura que se realizan a la instancia principal fallan con una excepción.

Por el contrario, si su clúster de base de datos no incluye ninguna instancia de réplica de lectura, su clúster de base de datos seguirá sin estar disponible cuando la instancia principal falle hasta que se vuelva a crear. Volver a crear la instancia principal lleva mucho más tiempo que promover una réplica de lectura.

Para garantizar una alta disponibilidad, le recomendamos que cree una o más instancias de réplica de lectura que tengan la misma clase de instancia de base de datos que la instancia principal y que estén ubicadas en zonas de disponibilidad diferentes a las de la instancia principal. Consulte Tolerancia a errores para un clúster de base de datos de Neptune.

Con la consola, puede crear un despliegue Multi-AZ especificando Multi-AZ al crear una instancia de clúster de base de datos. Si un clúster de base de datos se encuentra en una única zona de disponibilidad, puede convertirlo en un clúster de base de datos Multi-AZ añadiendo una réplica de Neptune en una zona de disponibilidad diferente.

> **Note**
>
> No puede crear una instancia de réplica de lectura cifrada para un clúster de base de datos de Neptune sin cifrar ni una instancia de réplica de lectura sin cifrar para un clúster de base de datos de Neptune cifrado.

Para obtener más información sobre cómo crear una instancia de base de datos de réplica de lectura de Neptune, consulte Creación de una instancia de lector de Neptune con la consola.

Dimensionamiento de las instancias de base de datos en un clúster de base de datos de Neptune

Cambie el tamaño de las instancias del clúster de base de datos de Neptune en función de los requisitos de CPU y memoria. La cantidad de vCPU en una instancia determina la cantidad de subprocesos de consulta que gestionan las consultas entrantes. La cantidad de memoria de una instancia determina el tamaño de la memoria caché del búfer, que se utiliza para almacenar copias de las páginas de datos extraídas del volumen de almacenamiento subyacente.

Cada instancia de base de datos de Neptune tiene un número de subprocesos de consulta que equivale a 2 veces el número de vCPU en esa instancia. Un `r5.4xlarge`, por ejemplo, con 16 vCPU, tiene 32 subprocesos de consulta y, por lo tanto, puede procesar 32 consultas simultáneamente.

Las consultas adicionales que llegan mientras todos los subprocesos de consulta están ocupados se colocan en una cola del lado del servidor y se procesan en orden FIFO a medida que los subprocesos de consulta vuelven a estar disponibles. Esta cola del servidor puede contener aproximadamente 8000 solicitudes pendientes. Una vez que esté llena, Neptune responde a las solicitudes adicionales con un `ThrottlingException`. Puede monitorizar el número de solicitudes pendientes con la `MainRequestQueuePendingRequests` CloudWatch métrica o utilizando el punto final de estado de la consulta de Gremlin con el parámetro. `includeWaiting`

El tiempo de ejecución de la consulta desde la perspectiva del cliente incluye el tiempo que se pasa en la cola, además del tiempo que se tarda en ejecutar realmente la consulta.

Lo ideal es que una carga de escritura simultánea sostenida que utilice todos los subprocesos de consulta de la instancia de base de datos principal muestre un 90 % o más de utilización de la CPU, lo que indica que todos los subprocesos de consulta del servidor se dedican activamente a realizar un trabajo útil. Sin embargo, el uso real de la CPU suele ser algo inferior, incluso con una carga de escritura simultánea sostenida. Esto suele deberse a que los subprocesos de consulta esperan a que se completen las operaciones de E/S del volumen de almacenamiento subyacente. Neptune utiliza escrituras de cuórum para realizar seis copias de los datos en tres zonas de disponibilidad, y cuatro de esos seis nodos de almacenamiento deben reconocer una escritura para que se considere duradera. Mientras un subproceso de consulta espera este cuórum desde el volumen de almacenamiento, se detiene, lo que reduce el uso de la CPU.

Si tiene una carga de escritura en serie en la que realiza una escritura tras otra y espera a que se complete la primera antes de comenzar la siguiente, es de esperar que la utilización de la CPU

sea aún menor. La cantidad exacta dependerá de la cantidad de vCPU y subprocesos de consulta (cuantos más subprocesos de consulta, menos CPU total por consulta), y se reducirá en cierta medida debido a la espera de E/S.

Para obtener más información acerca de la mejor manera de dimensionar las instancias de base de datos, consulte Elección del tipo de instancia de base de datos de Neptune correcto. Para conocer los precios de cada tipo de instancia, consulte la página de precios de Neptune.

Monitorización del rendimiento de las instancias de base de datos en Neptune

Puede utilizar CloudWatch las métricas de Neptune para supervisar el rendimiento de las instancias de base de datos y realizar un seguimiento de la latencia de las consultas observada por el cliente. Consulte Utilización CloudWatch para supervisar el rendimiento de las instancias de base de datos en Neptune.

Almacenamiento, fiabilidad y disponibilidad de Amazon Neptune

Amazon Neptune utiliza una arquitectura de almacenamiento distribuido y compartido que se escala automáticamente a medida que aumentan las necesidades de almacenamiento de la base de datos.

Los datos de Neptune se almacenan en un volumen de clúster, que es un volumen único y virtual que utiliza unidades de estado sólido de memoria rápida no volátil (NVMe). El volumen del clúster se compone de un conjunto de bloques lógicos conocidos como segmentos. A cada uno de estos segmentos se le asignan 10 gigabytes (GB) de almacenamiento. Los datos de cada segmento se replican en seis copias, que luego se distribuyen entre tres zonas de disponibilidad (AZ) en la región de AWS en la que reside el clúster de base de datos.

Cuando se crea un clúster de base de datos de Neptune, se le asigna un único segmento de 10 GB. A medida que el volumen de datos aumenta y supera el almacenamiento asignado actualmente, Neptune amplía automáticamente el volumen del clúster añadiendo nuevos segmentos. El volumen de un clúster de Neptune puede crecer hasta un tamaño máximo de 128 tebibytes (TiB) en todas las regiones compatibles, excepto en China GovCloud y, donde está limitado a 64 TiB. Sin embargo, para las versiones de motores anteriores a la Versión: 1.0.2.2 (09/03/2020), el tamaño de los volúmenes de los clústeres está limitado a 64 TiB en todas las regiones.

El volumen del clúster de base de datos contiene todos sus datos de usuario, índices y diccionarios (que se describen en la sección Modelo de datos de gráficos de Neptune.), así como metadatos internos como registros de transacciones internas. Todos estos datos de gráficos, incluidos índices y registros internos, no pueden superar el tamaño máximo del volumen del clúster.

Opción de almacenamiento optimizado para E/S

Neptune ofrece dos modelos de precios para el almacenamiento:

- Almacenamiento estándar: el almacenamiento estándar proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a bajo.
- Almacenamiento optimizado para E/S: con el almacenamiento optimizado para E/S, solo paga por el almacenamiento que utiliza, a un costo mayor que el del almacenamiento estándar, y no paga nada por la E/S que utiliza.

  El almacenamiento optimizado para E/S está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S a un costo predecible, con una latencia de E/S baja y un rendimiento de E/S constante.

Para obtener más información, consulte Almacenamiento optimizado para E/S.

Asignación de almacenamiento de Neptune

Aunque un volumen de clúster de Neptune puede crecer hasta 128 TiB (o 64 TiB en algunas regiones), solo se le cobrará por el espacio que se asigne realmente. El espacio total asignado lo determina el límite máximo de almacenamiento, que es la cantidad máxima asignada al volumen del clúster en cualquier momento de su existencia.

Esto significa que, incluso si los datos de usuario se eliminan del volumen de un clúster, por ejemplo, mediante una consulta de colocación como `g.V().drop()`, el espacio total asignado sigue siendo el mismo. Neptune optimiza automáticamente el espacio asignado no utilizado para reutilizarlo en el futuro.

Además de los datos del usuario, dos tipos adicionales de contenido consumen espacio de almacenamiento interno: los datos del diccionario y los registros de transacciones internos. Aunque los datos del diccionario se almacenan junto con los datos de los gráficos, persisten indefinidamente, incluso cuando los datos de gráficos que admiten se han eliminado, lo que significa que las entradas se pueden reutilizar si se vuelven a introducir los datos. Los datos del registro interno se almacenan en un espacio de almacenamiento interno independiente que tiene su propio límite máximo. Cuando un registro interno caduca, el almacenamiento que ocupaba se puede reutilizar para otros registros, pero no para datos de gráficos. La cantidad de espacio interno que se ha asignado a los registros se incluye en el espacio total que indica la métrica. `VolumeBytesUsed` CloudWatch

Compruebe en Prácticas recomendadas de almacenamiento las formas de reducir al mínimo el almacenamiento asignado y de reutilizar el espacio.

Facturación del almacenamiento de Neptune

Los costos de almacenamiento se facturan en función del límite máximo de almacenamiento, tal y como se ha descrito anteriormente. Aunque los datos se replican en seis copias, solo se le facturará una copia de los datos.

Puede determinar cuál es el límite máximo de almacenamiento actual de su clúster de base de datos supervisando la `VolumeBytesUsed` CloudWatch métrica (consulteMonitorización de Neptune con Amazon CloudWatch).

Otros factores que pueden afectar a los costos de almacenamiento de Neptune incluyen las instantáneas y copias de seguridad de las bases de datos, que se facturan por separado como

almacenamiento de copia de seguridad y se basan en los costos de almacenamiento de Neptune (consulte Métricas de CloudWatch que son útiles para administrar el almacenamiento de copias de seguridad de Neptune).

Sin embargo, si crea un clon de la base de datos, ese clon apunta al mismo volumen de clúster que utiliza el propio clúster de base de datos, por lo que no se cobrará ningún cargo adicional por el almacenamiento de los datos originales. Los cambios posteriores en el clon utilizan el copy-on-write protocolo y, de hecho, conllevan costes de almacenamiento adicionales.

Para obtener más información acerca de los precios de Neptune, consulte Precios de Amazon Neptune.

Prácticas recomendadas de almacenamiento de Neptune

Dado que ciertos tipos de datos consumen almacenamiento permanente en Neptune, utilice estas prácticas recomendadas para evitar grandes picos en el crecimiento del almacenamiento:

- Al diseñar su modelo de datos de gráficos, evite en la medida de lo posible utilizar claves de propiedad y valores orientados al usuario que sean de naturaleza temporal.
- Si tiene pensado realizar cambios en su modelo de datos, no cargue datos en un clúster de base de datos existente con el nuevo modelo hasta que haya borrado los datos de ese clúster de base de datos mediante la API de restablecimiento rápido. Lo mejor suele ser cargar los datos que utilizan un modelo nuevo en un clúster de base de datos nuevo.
- Las transacciones que funcionan con grandes cantidades de datos generan registros internos proporcionalmente grandes, lo que puede aumentar permanentemente el límite máximo del espacio de registros interno. Por ejemplo, una sola transacción que elimine todos los datos de su clúster de base de datos podría generar un registro interno enorme que requeriría asignar una gran cantidad de almacenamiento interno y, por lo tanto, reduciría permanentemente el espacio disponible para los datos de gráficos.

  Para evitarlo, divida las transacciones grandes en otras más pequeñas y deje pasar tiempo entre ellas para que los registros internos asociados tengan la posibilidad de caducar y liberar su almacenamiento interno para que se reutilice en los registros posteriores.
- Para monitorear el crecimiento del volumen de su cúmulo de Neptune, puede configurar una CloudWatch alarma en la `VolumeBytesUsed` CloudWatch métrica. Esto puede resultar especialmente útil si los datos alcanzan el tamaño máximo del volumen del clúster. Para obtener más información, consulta Cómo usar CloudWatch las alarmas de Amazon.

La única forma de reducir el espacio de almacenamiento que utiliza su clúster de base de datos cuando tiene una gran cantidad de espacio asignado sin utilizar consiste en exportar todos los datos de gráficos y, a continuación, volver a cargarlos en un nuevo clúster de base de datos. Consulte el servicio y la utilidad de exportación de datos de Neptune para conocer una forma sencilla de exportar datos desde un clúster de base de datos, y el programa de carga masiva de Neptune para conocer una forma sencilla de importar datos a Neptune.

> **Note**
>
> La creación y restauración de una instantánea no reduce la cantidad de almacenamiento asignada al clúster de base de datos, ya que la instantánea conserva la imagen original del almacenamiento subyacente del clúster. Si no se está utilizando una cantidad sustancial del almacenamiento asignado, la única forma de reducirlo es exportar los datos de sus gráficos y volver a cargarlos en un nuevo clúster de base de datos.

Fiabilidad y alta disponibilidad del almacenamiento de Neptune

Amazon Neptune se ha diseñado para ofrecer fiabilidad, durabilidad y tolerancia a errores.

El hecho de que se mantengan seis copias de los datos de Neptune en tres zonas de disponibilidad (AZ) garantiza que el almacenamiento de los datos sea muy duradero y que la probabilidad de pérdida de datos sea muy baja. Los datos se replican automáticamente en todas las zonas de disponibilidad, independientemente de si hay instancias de base de datos en ellas, y la cantidad de réplicas es independiente de la cantidad de instancias de base de datos de su clúster.

Esto significa que puede añadir una réplica de lectura rápidamente, ya que Neptune no hace una nueva copia de los datos de los gráficos. En su lugar, la réplica de lectura se conecta al volumen del clúster que ya contiene sus datos. Del mismo modo, al eliminar una réplica de lectura, no se elimina ninguno de los datos subyacentes.

Puede eliminar el volumen del clúster y sus datos solo después de eliminar todas sus instancias de base de datos.

Neptune también detecta automáticamente los errores de los segmentos que integran el volumen de clúster. Cuando una copia de los datos de un segmento está dañada, Neptune repara inmediatamente ese segmento y utiliza otras copias de los datos dentro del mismo segmento para garantizar que los datos reparados estén actualizados. Como resultado, Neptune evita la pérdida de

datos y reduce la necesidad de realizar una point-in-time restauración para recuperarse de una falla en el disco.

Conexión a los puntos de conexión de Amazon Neptune

Amazon Neptune utiliza un clúster de instancias de base de datos en lugar de una sola instancia. Cada conexión de Neptune la gestiona una instancia de base de datos específica. Al conectarse a un clúster de base de datos de Neptune, el nombre de host y el puerto especificados apuntan a un controlador intermedio denominado punto de conexión. Un punto de conexión es una URL que contiene una dirección de host y un puerto. Los puntos de conexión de Neptune utilizan conexiones de capa de conexión segura (SSL)/seguridad de la capa de transporte (TLS/SSL).

Neptune utiliza el mecanismo de punto de conexión para abstraer estas conexiones, de modo que no tenga que codificar los nombres de host o escribir su propia lógica para redirigir las conexiones cuando algunas instancias de base de datos no están disponibles.

Al usar puntos de conexión, puede mapear cada conexión a la instancia o grupo de instancias adecuados en función de su caso de uso. Los puntos de conexión personalizados le permiten conectarse a subconjuntos de instancias de base de datos. Los siguientes puntos de conexión están disponibles en un clúster de base de datos de Neptune.

Puntos de conexión del clúster de Neptune

Un punto de conexión de clúster es un punto de conexión de una base de datos de Neptune que se conecta a la instancia principal de base de datos actual de ese clúster de base de datos. Cada clúster de base de datos de Neptune tiene un punto de conexión de clúster y una instancia de base de datos principal.

El punto de enlace del clúster proporciona soporte de conmutación por error para conexiones de lectura/escritura al clúster de bases de datos. Utilice el punto de enlace del clúster para todas las operaciones de escritura en el clúster de la base de datos, incluidos inserciones, actualizaciones, eliminaciones y cambios de lenguaje de definición de datos (DDL). También puede usar el punto de conexión del clúster para operaciones de lectura, como por ejemplo consultas.

Si se produce un error en la instancia de base de datos principal actual de un clúster de base de datos, Neptune conmuta por error automáticamente a una nueva instancia de base de datos principal. Durante una conmutación por error, el clúster de bases de datos continúa atendiendo solicitudes de conexión al punto de enlace del clúster de la nueva instancia de base de datos principal, con una interrupción del servicio mínima.

En el siguiente ejemplo, se ilustra un punto de conexión del clúster de un clúster de base de datos de Neptune.

```
mydbcluster.cluster-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Puntos de conexión del lector de Neptune

Un punto de conexión del lector es un punto de conexión de un clúster de base de datos de Neptune que se conecta a una de las réplicas de Neptune disponibles de ese clúster de base de datos. Cada clúster de base de datos de Neptune tiene un punto de conexión del lector. Si hay más de una réplica, el punto de conexión del lector dirige cada solicitud de conexión a una de las réplicas de Neptune.

El punto de enlace del lector proporciona direccionamiento de turnos rotativos para conexiones de solo lectura al clúster de base de datos. Utilice el punto de conexión del lector para operaciones de lectura, como por ejemplo consultas .

No se puede utilizar el punto de enlace del lector para operaciones de escritura a menos que tenga un clúster de una sola instancia (un clúster sin réplicas de lectura). En ese caso y solo en ese caso, se puede utilizar el lector para operaciones de escritura, así como para operaciones de lectura.

El enrutamiento de turnos rotativos para puntos de enlace de lectura funciona mediante el cambio del host al que apunta la entrada DNS. Cada vez que resuelva las DNS; obtendrá una IP diferente y se abrirán conexiones con dichas IP. Después de establecer una conexión, todas las solicitudes para dicha conexión se envían al mismo alojamiento. El cliente debe crear una nueva conexión y resolver el registro de DNS de nuevo para obtener una conexión a una réplica de lectura potencialmente diferente.

> **Note**
>
> WebSockets las conexiones suelen mantenerse activas durante períodos prolongados. Para obtener diferentes réplicas de lectura, debe:
>
> - Asegúrese de que su cliente resuelva la entrada de DNS cada vez que se conecte.
> - Cierre la conexión y vuelva a conectar.

Los distintos clientes de software podrían resolver las DNS de formas distintas. Por ejemplo, si su cliente resuelve las DNS y, a continuación, utiliza la IP para cada conexión, dirigirá todas las solicitudes a un único host.

El almacenamiento en caché de DNS para los clientes o proxies resuelve el nombre de DNS para el mismo punto de enlace del caché. Este problema se da tanto en escenarios de direccionamiento de turnos rotativos como de conmutación por error.

> **Note**
>
> Desactive la configuración de caché para las DNS de modo que se fuerce una resolución de DNS todas las veces.

El clúster de base de datos distribuye las solicitudes de conexión al punto de conexión del lector entre las réplicas de Neptune disponibles. Si el clúster de base de datos contiene solo una instancia de base de datos principal, el punto de enlace del lector atiende solicitudes de conexión de la instancia de base de datos principal. Si se crea una réplica de Neptune para ese clúster de base de datos, el punto de conexión del lector continúa atendiendo solicitudes de conexión al punto de conexión del lector de la nueva réplica de Neptune, con una interrupción del servicio mínima.

En el siguiente ejemplo, se ilustra un punto de conexión del lector de un clúster de base de datos de Neptune.

```
mydbcluster.cluster-ro-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Puntos de conexión de instancias de Neptune

Un punto de conexión de una instancia es un punto de conexión de base de datos de un clúster de base de datos de Neptune que se conecta a esa instancia de base de datos específica. Cada instancia de base de datos de un clúster de base de datos, independientemente del tipo de instancia, tiene su propio punto de enlace de la instancia único. Por tanto, hay un punto de enlace de instancia de la instancia de base de datos principal actual del clúster de la base de datos. También hay un punto de conexión de la instancia para cada una de las réplicas de Neptune en el clúster de base de datos.

El punto de enlace de la instancia proporciona un control directo sobre las conexiones al clúster de bases de datos, en los casos en los que el uso del punto de enlace del clúster o del lector puede no ser adecuado. Por ejemplo, su aplicación cliente podría necesitar un balanceo de carga detallado en función del tipo de carga de trabajo. En este caso, puede configurar varios clientes para que se conecten a distintas réplicas de Neptune de un clúster de base de datos con el fin de distribuir las cargas de trabajo de lectura.

En el siguiente ejemplo, se ilustra un punto de conexión de una instancia de base de datos de un clúster de base de datos de Neptune.

```
mydbinstance.123456789012.us-east-1.neptune.amazonaws.com:8182
```

Puntos de conexión personalizados de Neptune

Un punto de conexión personalizado de un clúster de Neptune representa un conjunto de instancias de base de datos que ha elegido. Cuando se conecta al punto de conexión, Neptune elige una de las instancias del grupo para gestionar la conexión. Defina las instancias a las que hace referencia este punto de enlace y decida el objetivo de este.

Un clúster de base de datos de Neptune no tiene puntos de conexión personalizados hasta que cree uno, y puede crear hasta cinco puntos de conexión personalizados para cada clúster de Neptune aprovisionado.

El punto de enlace personalizado proporciona conexiones de base de datos con equilibrio de carga en función de otros criterios aparte de la capacidad de solo lectura o de lectura/escritura de las instancias de base de datos. Dado que la conexión puede dirigirse a cualquier instancia de base de datos asociada al punto de conexión, asegúrese de que todas las instancias de ese grupo comparten las mismas características de rendimiento y capacidad de memoria. Al usar puntos de enlace personalizados, normalmente no usa el punto de enlace del lector de ese clúster.

Esta característica está destinada a usuarios avanzados con tipos de cargas de trabajo especializados donde no resulta práctico que todas las réplicas de Neptune del clúster sean idénticas. Con los puntos de conexión personalizados, puede ajustar la capacidad de las instancias de base de datos que se utilizan para cada conexión.

Por ejemplo, si define varios puntos de conexión personalizados que se conectan a grupos de instancias con diferentes clases de instancias, puede dirigir a los usuarios con diferentes necesidades de rendimiento a los puntos de conexión que mejor se adapten a sus casos de uso.

En el siguiente ejemplo, se ilustra un punto de conexión personalizado de una instancia de base de datos de un clúster de base de datos de Neptune.

```
myendpoint.cluster-custom-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Para obtener más información, consulte Trabajo con puntos de conexión personalizados.

Consideraciones sobre puntos de conexión de Neptune

Considere los siguientes problemas a la hora de trabajar con los puntos de conexión de Neptune:

- Antes de usar un punto de enlace de la instancia para conectarse a una instancia de base de datos específica de un clúster de base de datos, valore la posibilidad de usar el punto de enlace del clúster o el punto de enlace del lector del clúster de base de datos en su lugar.

  El punto de enlace del clúster y el del lector proporcionan soporte para situaciones que requieren alta disponibilidad. Si se produce un error en la instancia de base de datos principal de un clúster de base de datos, Neptune conmuta por error automáticamente a una nueva instancia de base de datos principal. Lo hace promoviendo una réplica de Neptune existente a una nueva instancia de base de datos principal o creando una instancia de base de datos principal. Si se produce una conmutación por error, puede usar el punto de conexión del clúster para volver a conectarse a la instancia de base de datos recién promovida o creada, o bien usar el punto de conexión del lector para volver a conectarse a una de las otras réplicas de Neptune del clúster de base de datos.

  Si no adopta este enfoque, aún puede asegurarse de que se conecta a la instancia de base de datos adecuada del clúster de base de datos para la operación esperada. Para hacerlo, puede encontrar manualmente o mediante programación el conjunto resultante de instancias de base de datos disponibles del clúster de base de datos y confirmar sus tipos de instancia tras la conmutación por error, antes de usar el punto de enlace de la instancia de una instancia de base de datos específica.

  Para obtener más información acerca de las conmutaciones por error, consulte Tolerancia a errores para un clúster de base de datos de Neptune.

- El punto de conexión del lector solo dirige las conexiones a las réplicas de Neptune disponibles de un clúster de base de datos de Neptune. No dirige consultas específicas.

  > **Important**
  >
  > Neptune no equilibra las cargas.

Si desea equilibrar la carga de las consultas para distribuir la carga de trabajo de lectura de un clúster, tendrá que administrarlo en su aplicación. Debe utilizar los puntos de conexión de la instancia para conectarse directamente a las réplicas de Neptune para equilibrar así la carga.

- El enrutamiento de turnos rotativos para puntos de enlace de lectura funciona mediante el cambio del host al que apunta la entrada DNS. El cliente debe crear una nueva conexión y resolver el registro de DNS de nuevo para obtener una conexión a una réplica de lectura potencialmente nueva.

- Durante una conmutación por error, el punto de conexión del lector podría dirigir las conexiones a la nueva instancia de base de datos principal de un clúster de base de datos durante un breve periodo de tiempo cuando una réplica de Neptune se convierte en la nueva instancia de base de datos principal.

Uso de puntos de conexión personalizados en Neptune

Al añadir una instancia de base de datos a un punto de enlace personalizado o quitarla de un punto de enlace personalizado, cualquier conexión existente a esa instancia de base de datos permanece activa.

Puede definir una lista de instancias de base de datos para incluirlas en un punto de conexión personalizado (la lista estática) o una para excluir del punto de conexión personalizado (la lista de exclusiones). Puede utilizar el mecanismo de inclusión/exclusión para subdividir las instancias de base de datos en grupos y asegurarse de que los puntos de conexión personalizados cubren todas las instancias de base de datos del clúster. Cada punto de enlace personalizado puede contener solo uno de estos tipos de lista.

En el AWS Management Console, la elección se representa mediante la casilla de verificación Adjuntar futuras instancias agregadas a este clúster. Al no seleccionar la casilla de verificación, el punto de enlace personalizado usa una lista estática que contiene solo las instancias de base de datos especificadas en el cuadro de diálogo. Al seleccionar la casilla de verificación, el punto de conexión personalizado usa una lista de exclusión. En este caso, el punto de conexión personalizado representa todas las instancias de base de datos del clúster (incluidas las añadidas en el futuro), excepto las que se han dejado desactivadas en el cuadro de diálogo.

Neptune no cambia las instancias de base de datos especificadas en las listas estáticas o de exclusión cuando las instancias de base de datos cambian los roles entre la instancia principal y la réplica de Neptune debido a una conmutación por error o a una promoción.

Puede asociar una instancia de base de datos a más de un punto de enlace personalizado. Por ejemplo, supongamos que añade una nueva instancia de base de datos a un clúster. En estos casos, la instancia de base de datos se añade a todos los puntos de conexión personalizados para los que cumple los requisitos. La lista estática o de exclusiones definida determina qué instancia de base de datos se le puede añadir.

Si un punto de conexión incluye una lista estática de instancias de base de datos, las réplicas de Neptune recién añadidas no se añaden a ese punto de conexión. Por el contrario, si el punto de conexión tiene una lista de exclusión, las réplicas de Neptune recién añadidas se añaden a esta última, siempre y cuando no se mencionen en la lista de exclusión.

Si una réplica de Neptune deja de estar disponible, permanece asociada a cualquier punto de conexión personalizado. Esto ocurre si está en mal estado, se detiene, se reinicia o no está disponible por otro motivo. Sin embargo, mientras no esté disponible, no podrá conectarse a ella a través de ningún punto de conexión.

Dado que los clústeres de Neptune recién creados no tienen ningún punto de conexión personalizado, debe crear y administrar estos objetos usted mismo. Esto también se aplica a los clústeres de Neptune restaurados a partir de instantáneas, ya que los puntos de conexión personalizados no se incluyen en la instantánea. Tiene que volver a crearlos tras la restauración y elegir nuevos nombres de punto de conexión si el clúster restaurado está en la misma región que el original.

Creación de un punto de conexión personalizado

Administre puntos de conexión personalizados mediante la consola de Neptune. Para ello, vaya a la página de detalles del clúster de Neptune y utilice los controles de la sección Custom Endpoints.

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Navegue hasta la página de detalles del clúster.
3. Elija la acción `Create custom endpoint` en la sección Puntos de conexión.
4. Elija un nombre para el punto de conexión personalizado, que sea único para su identificador de usuario y región. El nombre debe tener 63 caracteres o menos y debe utilizar el siguiente formato:

endpointName.cluster-custom-*customerDnsIdentifier*.*dnsSuffix*

Dado que los nombres de punto de enlace personalizado no incluyen el nombre de su clúster, no tiene que cambiar esos nombres si cambia el nombre de un clúster. No puede volver a usar el mismo nombre de punto de conexión personalizado para más de un clúster en la misma región. Especifique un nombre para cada punto de enlace personalizado que sea único en los clústeres que pertenecen a su ID de usuario dentro de una región particular.

5. Para elegir una lista de instancias de base de datos que se conserve igual incluso a medida que se amplía el clúster, deje sin seleccionar la casilla de verificación Attach future instances added to this cluster (Asociar futuras instancias añadidas a este clúster). Al seleccionar esa casilla de verificación, el punto de conexión personalizado añade de forma dinámica cualquier nueva instancia a medida que se añaden al clúster.

Visualización de puntos de enlace personalizados

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Navegue hasta la página de detalles de su clúster de base de datos.
3. La sección Puntos de conexión solo contiene información sobre los puntos de conexión personalizados (los detalles sobre los puntos de conexión integrados se muestran en la sección principal de Detalles). Para ver los detalles de un punto de conexión personalizado específico, seleccione su nombre para abrir la página de detalles para ese punto de conexión.

Edición de un punto de enlace personalizado

Puede editar las propiedades de un punto de conexión personalizado para cambiar las instancias de base de datos asociadas a él. También puede cambiar entre una lista estática y una lista de exclusión.

No puede conectarse a un punto de enlace personalizado o usarlo mientras los cambios de una acción de edición están en curso. Podrían pasar algunos minutos después de realizar un cambio hasta que el estado del punto de conexión vuelva a ser Disponible y usted pueda volver a conectarse.

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.

2. Navegue hasta la página de detalles del clúster.
3. En la sección Puntos de conexión, elija el nombre del punto de conexión personalizado que desea editar.
4. En la página de detalles de ese punto de conexión, elija la acción Editar.

Eliminación de un punto de enlace personalizado

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Navegue hasta la página de detalles del clúster.
3. En la sección Puntos de conexión, elija el punto de conexión que desea eliminar.
4. En la página de detalles de ese punto de conexión, elija la acción Eliminar.

Inserte un identificador personalizado en una consulta de Neptune Gremlin o SPARQL

De forma predeterminada, Neptune asigna un valor `queryId` único a cada consulta. Puede utilizar este ID para obtener información sobre una consulta en ejecución (consulte API del estado de la consulta de Gremlin o API de estado de la consulta SPARQL) o cancelarla (consulte Cancelación de consultas de Gremlin o Cancelación de consultas SPARQL).

Neptune también le permite especificar su propio valor `queryId` para una consulta de Gremlin o SPARQL, ya sea en el encabezado HTTP o para una consulta SPARQL mediante la sugerencia de la consulta `queryId`. La asignación de su propio `queryID` facilita la realización de un seguimiento de una consulta para obtener el estado o cancelarlo.

> **Note**
>
> Esta característica solo está disponible a partir de Versión 1.0.1.0.200463.0 (15/10/2019).

Inserción de un valor **`queryId`** personalizado mediante el encabezado HTTP

Tanto para Gremlin como para SPARQL, el encabezado HTTP se puede utilizar para insertar su propio valor `queryId` en una consulta.

Ejemplo de Gremlin

```
curl -XPOST https://your-neptune-endpoint:port \
    -d "{\"gremlin\": \
        \"g.V().limit(1).count()\" , \
        \"queryId\":\"4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47\"  }"
```

Ejemplo de SPARQL

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=SELECT * WHERE { ?s ?p ?o } " \
       --data-urlencode \
       "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Inserción de un valor **queryId** personalizado mediante una sugerencia de consulta SPARQL

A continuación se muestra un ejemplo de cómo utilizaría la sugerencia de la consulta `queryId` SPARQL para insertar un valor `queryId` personalizado en una consulta SPARQL:

```
curl https://your-neptune-endpoint:port/sparql \
    -d "PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#> \
       SELECT * WHERE { hint:Query hint:queryId \"4d5c4fae-
aa30-41cf-9e1f-91e6b7dd6f47\" \
       {?s ?p ?o}}"
```

Uso del valor **queryId** para comprobar el estado de la consulta

Ejemplo de Gremlin

```
curl https://your-neptune-endpoint:port/gremlin/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Ejemplo de SPARQL

```
curl https://your-neptune-endpoint:port/sparql/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Modo de laboratorio de Neptune

Puede usar el modo de laboratorio de Amazon Neptune para habilitar las nuevas características que se incluyen en la versión actual del motor de Neptune, pero que aún no están listas para su uso en producción y no están habilitadas de forma predeterminada. Esto le permite probar estas características en sus entornos de desarrollo y pruebas.

> **Note**
>
> Esta característica solo está disponible a partir de Versión 1.0.1.0.200463.0 (15/10/2019).

Uso del modo de laboratorio de Neptune

Utilice el parámetro del clúster de base de datos `neptune_lab_mode` para habilitar o deshabilitar características. Para ello, incluya *`(feature name)`*`=enabled` o *`(feature name)`*`=disabled` en el valor del parámetro `neptune_lab_mode` en el grupo de parámetros del clúster de base de datos.

Por ejemplo, en esta versión del motor podría establecer el parámetro `neptune_lab_mode` en `Streams=disabled, ReadWriteConflictDetection=enabled`.

Para obtener información sobre cómo editar el grupo de parámetros de clúster de base de datos para su base de datos, consulte Edición de un grupo de parámetros. Tenga en cuenta que no puede editar el grupo de parámetros de clúster de base de datos predeterminado; si utiliza el grupo predeterminado, debe crear un nuevo grupo de parámetros de clúster de base de datos para poder establecer el parámetro `neptune_lab_mode`.

> **Note**
>
> Al realizar un cambio en un parámetro estático de un clúster de base de datos como, por ejemplo, `neptune_lab_mode`, debe volver a iniciar la instancia principal (escritor) del clúster para que el cambio se aplique. Antes de Versión: 1.2.0.0 (21/07/2022), todas las réplicas de lectura de un clúster de base de datos se reiniciaban automáticamente cuando se reiniciaba la instancia principal.
>
> A partir de Versión: 1.2.0.0 (21/07/2022), el reinicio de la instancia principal no provoca el reinicio de ninguna de las réplicas. Esto significa que debe reiniciar cada instancia por separado para detectar un cambio en los parámetros del clúster de base de datos (consulte Grupos de parámetros).

Important

En la actualidad, si proporciona parámetros incorrectos del modo de laboratorio o si su solicitud falla por otro motivo, es posible que no se le notifique el error. Siempre debe comprobar que una solicitud de cambio del modo de laboratorio se ha realizado correctamente. Para ello, llame a la API de estado, tal y como se muestra a continuación:

```
curl -G https://your-neptune-endpoint:port/status
```

Los resultados de estado incluyen información del modo de laboratorio que indica si se han realizado o no los cambios solicitados:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "LookupCache":{"status":"Available"},
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Actualmente se accede a las siguientes características utilizando el modo de laboratorio:

El índice OSGP

Neptune ahora puede mantener un cuarto índice, el índice OSGP, que es útil para los conjuntos de datos que tienen un gran número de predicados (consulte Habilitación de un índice OSGP).

> **Note**
>
> Esta característica está disponible a partir de la versión 1.0.2.1 del motor de Neptune.

Puede habilitar un índice OSGP en un clúster de base de datos de Neptune nuevo y vacío configurando `ObjectIndex=enabled` en el parámetro de clúster de base de datos `neptune_lab_mode`. Un índice OSGP solo se puede habilitar en un clúster de base de datos nuevo y vacío.

De forma predeterminada, el índice OSGP está deshabilitado.

> **Note**
>
> Tras configurar el parámetro del clúster de base de datos `neptune_lab_mode` para habilitar el índice OSGP, debe reiniciar la instancia del escritor del clúster para que el cambio se aplique.

> **Warning**
>
> Si deshabilita un índice OSGP activado configurando `ObjectIndex=disabled` y luego lo vuelve a habilitar después de añadir más datos, el índice no se generará correctamente. No se admite la reconstrucción del índice bajo demanda, por lo que solo debe habilitar el índice OSGP cuando la base de datos esté vacía.

Semántica de transacciones formalizada

Neptune ha actualizado la semántica formal para transacciones simultáneas (consulte Semántica de transacciones en Neptune).

Utilice `ReadWriteConflictDetection` como el nombre en el parámetro `neptune_lab_mode` que habilita o deshabilita la semántica de transacciones formalizada.

De forma predeterminada, la semántica de transacción formalizada ya está habilitada. Si desea volver al comportamiento anterior, incluya `ReadWriteConflictDetection=disabled` en el valor establecido para el parámetro `neptune_lab_mode` del clúster de base de datos.

Soporte extendido de fecha y hora

Neptune ha ampliado el soporte para la funcionalidad de fecha y hora. Para habilitar la fecha y hora con formatos extendidos, inclúyala `DatetimeMillisecond=enabled` en el valor establecido para el parámetro del clúster de base de datos. `neptune_lab_mode`

El motor de consultas alternativo (DFE) de Amazon Neptune (DFE)

Amazon Neptune tiene un motor de consultas alternativo denominado DFE que utiliza los recursos de la instancia de base de datos, como los núcleos de la CPU, la memoria y la E/S, de forma más eficiente que el motor de Neptune original.

> **Note**
>
> Con conjuntos de datos de gran tamaño, es posible que el motor DFE no funcione bien en las instancias t3.

El motor DFE ejecuta consultas SPARQL, Gremlin y openCypher, y es compatible con una amplia variedad de tipos de planes, incluidos planes left-deep, bushy e híbridos. Los operadores de planes pueden invocar tanto operaciones de computación, que se ejecutan en un conjunto reservado de núcleos de computación, como operaciones de E/S, cada una de las cuales se ejecuta en su propio subproceso en un grupo de subprocesos de E/S.

El DFE utiliza estadísticas pregeneradas sobre los datos de los gráficos de Neptune para tomar decisiones informadas sobre cómo estructurar las consultas. Consulte Estadísticas del DFE para obtener información sobre cómo se generan estas estadísticas.

La elección del tipo de plan y el número de subprocesos de computación utilizados se realiza automáticamente en función de las estadísticas generadas previamente y de los recursos disponibles en el nodo principal de Neptune. El orden de los resultados no está predeterminado para los planes que tienen un paralelismo de computación interno.

Control sobre dónde se usa el motor DFE de Neptune

De forma predeterminada, el parámetro de instancia neptune_dfe_query_engine de una instancia está establecido en `viaQueryHint`, lo que hace que el motor DFE solo se utilice para las consultas de openCypher y para las consultas de Gremlin y SPARQL que incluyen explícitamente la sugerencia de consulta `useDFE` establecida en `true`.

Para habilitar por completo el motor DFE para que se utilice siempre que sea posible, defina el parámetro de instancia `neptune_dfe_query_engine` en `enabled`.

También puede deshabilitar el DFE incluyendo la sugerencia de consulta `useDFE` para una consulta de Gremlin o una consulta de SPARQL en particular. Esta sugerencia de consulta le permite impedir que el DFE ejecute esa consulta concreta.

Puede determinar si el DFE está habilitado o no en una instancia mediante una llamada Estado de la instancia, de la siguiente manera:

```
curl -G https://your-neptune-endpoint:port/status
```

A continuación, la respuesta de estado especifica si el DFE está habilitado o no:

```
{
  "status":"healthy",
  "startTime":"Wed Dec 29 02:29:24 UTC 2021",
  "dbEngineVersion":"development",
  "role":"writer",
  "dfeQueryEngine":"viaQueryHint",
  "gremlin":{"version":"tinkerpop-3.5.2"},
  "sparql":{"version":"sparql-1.1"},
  "opencypher":{"version":"Neptune-9.0.20190305-1.0"},
  "labMode":{
    "ObjectIndex":"disabled",
    "ReadWriteConflictDetection":"enabled"
  },
  "features":{
    "ResultCache":{"status":"disabled"},
    "IAMAuthentication":"disabled",
    "Streams":"disabled",
    "AuditLog":"disabled"
  },
  "settings":{"clusterQueryTimeoutInMs":"120000"}
}
```

Los resultados de `explain` y `profile` de Gremlin indican si el DFE está ejecutando una consulta. Consulte Información contenida en un informe `explain` de Gremlin para `explain` y Informes `profile` de DFE para `profile`.

Del mismo modo, `explain` de SPARQL le indica si el DFE está ejecutando una consulta de SPARQL. Para obtener más información, consulte Ejemplo de salida `explain` de SPARQL cuando el DFE está habilitado y operador `DFENode`.

Componentes de consulta compatibles con el DFE de Neptune

Actualmente, el DFE de Neptune admite un subconjunto de componentes de consultas de SPARQL y Gremlin.

En el caso de SPARQL, se trata del subconjunto de patrones gráficos básicos conjuntivos.

En el caso de Gremlin, generalmente es el subconjunto de consultas que contiene una cadena de recorridos que no contienen algunos de los pasos más complejos.

Puede averiguar si el DFE ejecuta una de sus consultas total o parcialmente de la siguiente manera:

- En Gremlin, los resultados de `explain` y `profile` indican qué partes de una consulta está ejecutando el DFE, si las hay. Consulte Información contenida en un informe `explain` de Gremlin para `explain` y Informes `profile` de DFE para `profile`. Consulte también Ajuste de las consultas de Gremlin mediante `explain` y `profile`.

  Los detalles sobre la compatibilidad del motor de Neptune para pasos individuales de Gremlin están documentados en Compatibilidad con pasos de Gremlin.
- Del mismo modo, `explain` de SPARQL le indica si el DFE está ejecutando una consulta de SPARQL. Para obtener más información, consulte Ejemplo de salida `explain` de SPARQL cuando el DFE está habilitado y operador `DFENode`.

Administración de las estadísticas para que las utilice el DFE de Neptune

> **Note**
>
> La compatibilidad con openCypher depende del motor de consultas DFE de Neptune.
> El motor DFE estuvo disponible por primera vez en el modo de laboratorio en la versión 1.0.3.0 del motor de Neptune y, a partir de la versión 1.0.5.0 del motor de Neptune, pasó a estar habilitado de forma predeterminada, pero solo para su uso con sugerencias de consulta y para la compatibilidad con openCypher.
> A partir de la versión 1.1.1.0 del motor de Neptune, el motor DFE ya no está en modo de laboratorio y ahora se controla mediante el parámetro de instancia neptune_dfe_query_engine del grupo de parámetros de base de datos de una instancia.

El motor DFE utiliza la información sobre los datos de gráficos de Neptune para realizar compensaciones efectivas a la hora de planificar la ejecución de las consultas. Esta información adopta la forma de estadísticas que incluyen los denominados conjuntos de características y estadísticas de predicados que pueden guiar la planificación de las consultas.

A partir de la versión 1.2.1.0 del motor, puede recuperar información resumida sobre su gráfico a partir de estas estadísticas mediante la API de GetGraphresumen o el punto final. `summary`

Actualmente, estas estadísticas del DFE se vuelven a generar cada vez que se modifica más del 10 % de los datos del gráfico o cuando las estadísticas más recientes tienen más de 10 días de antigüedad. Sin embargo, estos desencadenadores pueden cambiar en el futuro.

> **Note**
>
> La generación de estadísticas está deshabilitada en las instancias `T3` y `T4g` porque puede superar la capacidad de memoria de esos tipos de instancias.

Puede administrar la generación de estadísticas de DFE a través de uno de los siguientes puntos de conexión:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics` (para SPARQL).

- `https://your-neptune-host:port/propertygraph/statistics` (para Gremlin y openCypher) y su versión alternativa: `https://your-neptune-host:port/pg/statistics`.

> **Note**
>
> A partir de la versión 1.1.1.0 del motor, el punto de conexión de las estadísticas de Gremlin (`https://your-neptune-host:port/gremlin/statistics`) está en desuso a favor del punto de conexión `propertygraph` o `pg`. Sigue siendo compatible con versiones anteriores, pero es posible que se elimine en futuras versiones.
> A partir de la versión 1.2.1.0 del motor, el punto de conexión de las estadísticas de SPARQL (`https://your-neptune-host:port/sparql/statistics`) está en desuso a favor del punto de conexión `rdf`. Sigue siendo compatible con versiones anteriores, pero es posible que se elimine en futuras versiones.

En los ejemplos siguientes, `$STATISTICS_ENDPOINT` se refiere a cualquiera de estas direcciones URL de punto de conexión.

> **Note**
>
> Si un punto de conexión de las estadísticas del DFE está en una instancia del lector, las únicas solicitudes que puede procesar son las solicitudes de estado. Otras solicitudes fallarán con una `ReadOnlyViolationException`.

Límites de tamaño para la generación de estadísticas de DFE

Actualmente, la generación de estadísticas del DFE se detiene si se alcanza alguno de los siguientes límites de tamaño:

- El número de conjuntos de características generados no puede superar los 50 000.
- El número de estadísticas de predicados generadas no puede superar el millón.

Estos límites pueden cambiar.

Estado actual de las estadísticas del DFE

Puede comprobar el estado actual de las estadísticas del DFE mediante la siguiente solicitud `curl`:

```
curl -G "$STATISTICS_ENDPOINT"
```

La respuesta a una solicitud de estado contiene los siguientes campos:

- `status`: el código de retorno HTTP de la solicitud. Si la solicitud se ha realizado correctamente, el código es `200`. Consulte Errores comunes para obtener una lista de los errores comunes.
- `payload`:
  - `autoCompute`: (booleano) indica si la generación automática de estadísticas está habilitada o no.
  - `active`: (booleano) indica si la generación de estadísticas del DFE está habilitada o no.
  - `statisticsId` : indica el identificador de la ejecución de generación de estadísticas actual. Un valor de `-1` indica que no se ha generado ninguna estadística.
  - `date`: la hora UTC a la que se generaron las estadísticas del DFE por última vez, en formato ISO 8601.

    > **Note**
    >
    > Antes de la versión 1.2.1.0 del motor, se representaba con una precisión minuciosa, pero a partir de la versión 1.2.1.0 del motor, se representa con una precisión de milisegundos (por ejemplo, `2023-01-24T00:47:43.319Z`).

  - `note`: una nota sobre los problemas en el caso de que las estadísticas no sean válidas.
  - `signatureInfo`: contiene información sobre los conjuntos de características generados en las estadísticas (antes de la versión 1.2.1.0 del motor, este campo se llamaba `summary`). Por lo general, no se pueden procesar directamente:
    - `signatureCount`: el número total de firmas en todos los conjuntos de características.
    - `instanceCount`: el número total de instancias del conjunto de características.
    - `predicateCount`: el número total de predicados únicos.

La respuesta a una solicitud de estado cuando no se han generado estadísticas es la siguiente:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : -1
   }
}
```

Si las estadísticas del DFE están disponibles, la respuesta tiene un aspecto similar al siguiente:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : true,
    "statisticsId" : 1588893232718,
    "date" : "2020-05-07T23:13Z",
    "summary" : {
      "signatureCount" : 5,
      "instanceCount" : 1000,
      "predicateCount" : 20
    }
  }
}
```

Si las estadísticas del DFE no se pudieron generar, por ejemplo, porque se superó el límite de tamaño de las estadísticas, la respuesta tendrá el siguiente aspecto:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : 1588713528304,
    "date" : "2020-05-05T21:18Z",
    "note" : "Limit reached: Statistics are not available"
  }
}
```

Deshabilitación de la generación automática de estadísticas del DFE

De forma predeterminada, la generación automática de estadísticas del DFE se habilita cuando se habilita el DFE.

Puede deshabilitar la generación automática de la siguiente manera:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "disableAutoCompute" }'
```

Si la solicitud se realiza correctamente, el código de respuesta HTTP es `200` y la respuesta es:

```
{
  "status" : "200 OK"
}
```

Para confirmar que la generación automática está deshabilitada, emita una solicitud de estado y compruebe que el campo `autoCompute` de la respuesta esté configurado en `false`.

Al deshabilitar la generación automática de estadísticas, no finaliza un cálculo de estadísticas en curso.

Si realiza una solicitud para deshabilitar la generación automática en una instancia de lectura en lugar de en la instancia de escritura de su clúster de base de datos, la solicitud fallará con un código de retorno HTTP 400 y tendrá un resultado similar al siguiente:

```
{
  "detailedMessage" : "Writes are not permitted on a read replica instance",
  "code" : "ReadOnlyViolationException",
  "requestId":"8eb8d3e5-0996-4a1b-616a-74e0ec32d5f7"
}
```

Consulte Errores comunes para obtener una lista de otros errores comunes.

Volver a habilitar la generación automática de estadísticas del DFE

De forma predeterminada, la generación automática de estadísticas del DFE ya está habilitada cuando se habilita el DFE. Si deshabilita la generación automática, puede volver a habilitarla más adelante de la siguiente manera:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "enableAutoCompute" }'
```

Si la solicitud se realiza correctamente, el código de respuesta HTTP es `200` y la respuesta es:

```
{
  "status" : "200 OK"
}
```

Para confirmar que la generación automática está habilitada, emita una solicitud de estado y compruebe que el campo `autoCompute` de la respuesta esté configurado en `true`.

Activación manual de la generación de estadísticas del DFE

Puede iniciar la generación de estadísticas del DFE manualmente de la siguiente manera:

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "refresh" }'
```

Si la solicitud se realiza correctamente, el resultado es el siguiente, con un código de retorno HTTP 200:

```
{
  "status" : "200 OK",
  "payload" : {
    "statisticsId" : 1588893232718
  }
}
```

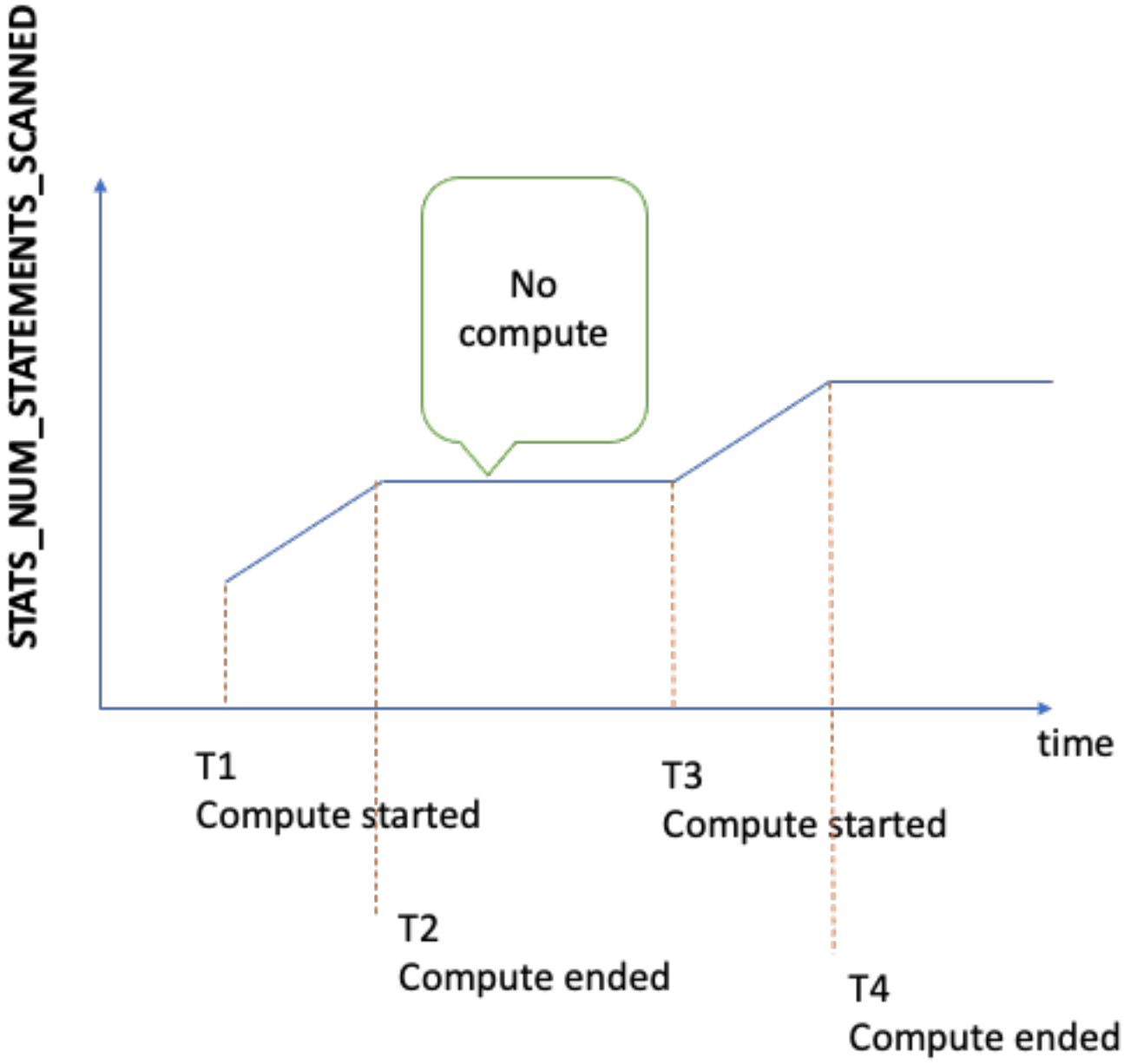
El `statisticsId` en la salida es el identificador de la ejecución de generación de estadísticas que se está produciendo actualmente. Si una ejecución ya estaba en marcha en el momento de la solicitud, la solicitud devuelve el identificador de esa ejecución en lugar de iniciar una nueva. Solo puede realizar una ejecución de generación de estadísticas a la vez.

Si se produce una conmutación por error mientras se generan las estadísticas del DFE, el nuevo nodo de escritor recogerá el último punto de comprobación procesado y reanudará la ejecución de las estadísticas desde allí.

Uso de la **`StatsNumStatementsScanned`** CloudWatch métrica para supervisar el cálculo de las estadísticas

La `StatsNumStatementsScanned` CloudWatch métrica devuelve el número total de sentencias escaneadas para el cálculo de estadísticas desde que se inició el servidor. Se actualiza en cada segmento de cálculo de estadísticas.

Cada vez que se activa el cálculo de estadísticas, este número aumenta y, cuando no se realiza ningún cálculo, permanece constante. Por lo tanto, al observar una gráfica de valores `StatsNumStatementsScanned` a lo largo del tiempo, se obtiene una idea bastante clara de cuándo se estaba realizando el cálculo de las estadísticas y con qué rapidez:



Cuando se realiza el cálculo, la pendiente del gráfico muestra con qué rapidez (cuanto más pronunciada sea la pendiente, más rápido se calculan las estadísticas).

Si el gráfico es simplemente una línea plana en 0, la característica de estadísticas se ha habilitado, pero no se ha calculado ninguna estadística. Si la característica de estadísticas se ha

deshabilitado o si está utilizando una versión del motor que no admite el cálculo de estadísticas, `StatsNumStatementsScanned` no existe.

Como se mencionó anteriormente, puede deshabilitar el cálculo de estadísticas mediante la API de estadísticas, pero si no lo hace, puede provocar que las estadísticas no estén actualizadas, lo que, a su vez, puede provocar una generación deficiente del plan de consultas para el motor DFE.

Consulte Monitorización de Neptune con Amazon CloudWatch para obtener información sobre cómo utilizar CloudWatch.

Uso de la autenticación AWS Identity and Access Management (IAM) con puntos finales de estadísticas del DFE

Puede acceder a los puntos de conexión de estadísticas del DFE de forma segura con la autenticación de IAM mediante awscurl o cualquier otra herramienta que funcione con HTTPS e IAM. Consulte Uso de `awscurl` con credenciales temporales para conectarse de forma segura a un clúster de base de datos con la autenticación de IAM habilitada para saber cómo configurar las credenciales adecuadas. Una vez que lo haya hecho, podrá realizar una solicitud de estado como la siguiente:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

O, por ejemplo, puede crear un archivo JSON denominado `request.json` que contenga:

```
{ "mode" : "refresh" }
```

Luego, puede iniciar la generación de estadísticas manualmente de la siguiente manera:

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db \
    -X POST -d @request.json
```

Eliminación de estadísticas del DFE

Puede eliminar todas las estadísticas de la base de datos realizando una solicitud HTTP DELETE al punto de conexión de las estadísticas:

```
curl -X "DELETE" "$STATISTICS_ENDPOINT"
```

Los códigos de retorno HTTP válidos son:

- `200`: la eliminación se ha realizado correctamente.

  En este caso, una respuesta típica sería la siguiente:

  ```
  {
    "status" : "200 OK",
    "payload" : {
        "active" : false,
        "statisticsId" : -1
    }
  }
  ```

- `204`: no había ninguna estadística que eliminar.

  En este caso, la respuesta está en blanco (no hay respuesta).

Si envía una solicitud de eliminación a un punto de conexión de las estadísticas de un nodo de lector, se emite una `ReadOnlyViolationException`.

Códigos de error comunes para la solicitud de estadísticas del DFE

A continuación, se ofrece una lista de los errores más comunes que se pueden producir al realizar una solicitud a un punto de conexión de estadísticas:

- `AccessDeniedException` – Código de retorno: `400`. Mensaje: `Missing Authentication Token`.
- `BadRequestException` (para Gremlin y openCypher) – Código de retorno: `400`. Mensaje: `Bad route: /pg/statistics`.
- `BadRequestException` (para datos RDF) – Código de retorno: `400`. Mensaje: `Bad route: /rdf/statistics`.
- `InvalidParameterException` – Código de retorno: `400`. Mensaje: `Statistics command parameter 'mode' has unsupported value '`*`the invalid value`*`'`.
- `MissingParameterException` – Código de retorno: `400`. Mensaje: `Content-type header not specified.`.

- `ReadOnlyViolationException` – Código de retorno: `400`. Mensaje: `Writes are not permitted on a read replica instance`.

Por ejemplo, si realiza una solicitud cuando el DFE y las estadísticas no están habilitados, obtendrá una respuesta como la siguiente:

```
{
  "code" : "BadRequestException",
  "requestId" : "b2b8f8ee-18f1-e164-49ea-836381a3e174",
  "detailedMessage" : "Bad route: /sparql/statistics"
}
```

Obtención de un informe resumido rápido sobre su gráfico

La API de resumen de gráficos de Neptune recupera la siguiente información sobre su gráfico:

- En el caso de los gráficos de propiedades (PG), la API de resumen de gráficos devuelve una lista de solo lectura de etiquetas de nodos y bordes y claves de propiedades, junto con el recuento de nodos, bordes y propiedades.
- En el caso de los gráficos del marco de descripción de recursos (RDF), la API de resumen de gráficos devuelve una lista de clases y claves de predicados de solo lectura, junto con el recuento de cuádruples, sujetos y predicados.

> **Note**
>
> La API de resumen de gráficos se introdujo en la versión 1.2.1.0 del motor de Neptune.

Con la API de resumen de gráficos, puede obtener rápidamente una comprensión de alto nivel del tamaño y el contenido de los datos de sus gráficos. También puede utilizar la API de forma interactiva en un cuaderno de Neptune mediante el comando mágico `%summary` de Neptune Workbench. En una aplicación de gráficos, la API se puede utilizar para mejorar los resultados de la búsqueda al proporcionar etiquetas de nodos o bordes descubiertos como parte de la búsqueda.

Los datos resumidos de los gráficos se extraen de las estadísticas del DFE que ha calculado el motor DFE de Neptune durante el tiempo de ejecución y están disponibles siempre que lo estén las estadísticas del DFE. Las estadísticas se habilitan de forma predeterminada cuando crea un nuevo clúster de base de datos de Neptune.

> **Note**
>
> La generación de estadísticas está deshabilitada en los tipos de instancias `t3` y `t4` (es decir, en los tipos de instancias `db.t3.medium` y `db.t4g.medium`) para ahorrar memoria. Como resultado, los datos resumidos de los gráficos tampoco están disponibles en esos tipos de instancias.

Puede comprobar el estado de las estadísticas del DFE mediante la API de estado de estadísticas. Mientras no se haya deshabilitado la generación automática de estadísticas, las estadísticas se actualizarán automáticamente de forma periódica.

Si quiere asegurarse de que las estadísticas estén lo más actualizadas posible cuando solicite un resumen de gráficos, puede activar manualmente una actualización de las estadísticas justo antes de recuperar el resumen. Si el gráfico está cambiando mientras se calculan las estadísticas, estas se retrasarán necesariamente un poco, pero no mucho.

Uso de la API de resumen de gráficos para recuperar información de resumen de gráficos

En el caso de un gráfico de propiedades que consulte con Gremlin u openCypher, puede recuperar un resumen de gráficos desde el punto de conexión del resumen del gráfico de propiedades. Hay un URI largo y uno corto para este punto de conexión:

- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics/summary`
- `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics/summary`

En el caso de un gráfico RDF que consulte mediante SPARQL, puede recuperar un resumen de gráficos desde el punto de conexión del resumen de RDF:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics/summary`

Estos puntos de conexión son de solo lectura y solo admiten una operación `GET` HTTP. Si $GRAPH_SUMMARY_ENDPOINT se establece en la dirección del punto de conexión que desee consultar, puede recuperar los datos del resumen mediante `curl` y `GET` HTTP de la siguiente manera:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT"
```

Si no hay estadísticas disponibles al intentar recuperar un resumen de gráficos, la respuesta tendrá el siguiente aspecto:

```
{
  "detailedMessage": "Statistics are not available. Summary can only be generated after
 statistics are available.",
  "requestId": "48c1f788-f80b-b69c-d728-3f6df579a5f6",
```

```
  "code": "StatisticsNotAvailableException"
}
```

El parámetro de consulta de URL **`mode`** para la API de resumen de gráficos

La API de resumen de gráficos acepta un parámetro de consulta de URL denominado `mode`, que puede tomar uno de dos valores: `basic` (el predeterminado) y `detailed`. En el caso de un gráfico RDF, la respuesta del resumen de gráficos del modo `detailed` contiene un campo `subjectStructures` adicional. En el caso de un gráfico de propiedades, la respuesta de resumen de gráficos detallada contiene dos campos adicionales: `nodeStructures` y `edgeStructures`.

Para solicitar una respuesta resumida en el gráfico `detailed`, incluya el parámetro `mode` de la siguiente manera:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT?mode=detailed"
```

Si el parámetro `mode` no está presente, se utiliza el modo `basic` de forma predeterminada, por lo que, aunque es posible especificar `?mode=basic` de forma explícita, no es necesario.

Respuesta del resumen de gráficos para un gráfico de propiedades (PG)

En el caso de un gráfico de propiedades vacío, la respuesta del resumen de gráficos detallado tiene el siguiente aspecto:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numNodes" : 0,
      "numEdges" : 0,
      "numNodeLabels" : 0,
      "numEdgeLabels" : 0,
      "nodeLabels" : [ ],
      "edgeLabels" : [ ],
      "numNodeProperties" : 0,
      "numEdgeProperties" : 0,
      "nodeProperties" : [ ],
      "edgeProperties" : [ ],
      "totalNodePropertyValues" : 0,
```

```
      "totalEdgePropertyValues" : 0,
      "nodeStructures" : [ ],
      "edgeStructures" : [ ]
    }
  }
}
```

Una respuesta de resumen de gráficos de propiedades (PG) tiene los siguientes campos:

- **`status`**: el código de retorno HTTP de la solicitud. Si la solicitud se ha realizado correctamente, el código es 200.

  Consulte Errores comunes del resumen de gráficos para obtener una lista de los errores comunes.
- **`payload`**
  - **`version`**: la versión de esta respuesta del resumen de gráficos.
  - **`lastStatisticsComputationTime`** : la marca temporal, en formato ISO 8601, de la hora en que Neptune calculó las estadísticas por última vez.
  - **`graphSummary`**
    - **`numNodes`**: número de nodos del gráfico.
    - **`numEdges`**: número de bordes del gráfico.
    - **`numNodeLabels`**: número de etiquetas de nodos distintas del gráfico.
    - **`numEdgeLabels`**: número de etiquetas de bordes distintas del gráfico.
    - **`nodeLabels`**: lista de etiquetas de nodos distintas del gráfico.
    - **`edgeLabels`**: lista de etiquetas de bordes distintas del gráfico.
    - **`numNodeProperties`**: número de propiedades de nodos distintas del gráfico.
    - **`numEdgeProperties`**: número de propiedades de bordes distintas del gráfico.
    - **`nodeProperties`**: lista de las propiedades de nodos distintas del gráfico, junto con el recuento de nodos en los que se utiliza cada propiedad.
    - **`edgeProperties`**: lista de las propiedades de bordes distintas del gráfico, junto con el recuento de bordes en los que se utiliza cada propiedad.
    - **`totalNodePropertyValues`**: número total de usos de todas las propiedades de los nodos.
    - **`totalEdgePropertyValues`**: número total de usos de todas las propiedades de los bordes.
    - **`nodeStructures`**: este campo solo está presente cuando *`mode=detailed`* se especifica en la solicitud. Contiene una lista de estructuras de nodos, cada una de las cuales contiene los siguientes campos:

- **`count`**: número de nodos que tienen esta estructura específica.
- **`nodeProperties`**: lista de propiedades de los nodos presentes en esta estructura específica.
- **`distinctOutgoingEdgeLabels`**: lista de las etiquetas de borde salientes distintas presentes en esta estructura específica.

- **`edgeStructures`**: este campo solo está presente cuando *`mode=detailed`* se especifica en la solicitud. Contiene una lista de estructuras de borde, cada una de las cuales contiene los siguientes campos:
  - **`count`**: número de bordes que tienen esta estructura específica.
  - **`edgeProperties`**: lista de propiedades de los bordes presentes en esta estructura específica.

Respuesta de resumen de gráficos para un gráfico RDF

En el caso de un gráfico RDF vacío, la respuesta del resumen de gráficos detallado tiene el siguiente aspecto:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-01-10T07:58:47.972Z",
    "graphSummary" : {
      "numDistinctSubjects" : 0,
      "numDistinctPredicates" : 0,
      "numQuads" : 0,
      "numClasses" : 0,
      "classes" : [ ],
      "predicates" : [ ],
      "subjectStructures" : [ ]
    }
  }
}
```

Una respuesta de resumen de gráficos RDF tiene los siguientes campos:

- **`status`**: el código de retorno HTTP de la solicitud. Si la solicitud se ha realizado correctamente, el código es 200.

Consulte Errores comunes del resumen de gráficos para obtener una lista de los errores comunes.

- **payload**
  - **version**: la versión de esta respuesta del resumen de gráficos.
  - **lastStatisticsComputationTime** : la marca temporal, en formato ISO 8601, de la hora en que Neptune calculó las estadísticas por última vez.
  - **graphSummary**
    - **numDistinctSubjects**: el número de sujetos distintos del gráfico.
    - **numDistinctPredicates**: el número de predicados distintos del gráfico.
    - **numQuads**: el número de cuádruples del gráfico.
    - **numClasses**: el número de clases del gráfico.
    - **classes**: lista de clases del gráfico.
    - **predicates**: lista de predicados del gráfico, junto con los recuentos de predicados.
    - **subjectStructures**: este campo solo está presente cuando *mode=detailed* se especifica en la solicitud. Contiene una lista de estructuras de sujetos, cada una de las cuales contiene los siguientes campos:
      - **count**: número de apariciones de esta estructura específica.
      - **predicates**: lista de predicados presentes en esta estructura específica.

Ejemplo de respuesta del resumen de gráficos de propiedades (PG)

Esta es la respuesta de resumen detallada para un gráfico de propiedades que contiene el conjunto de datos de rutas aéreas de un gráfico de propiedades de ejemplo:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:35:03.804Z",
    "graphSummary" : {
      "numNodes" : 3748,
      "numEdges" : 51300,
      "numNodeLabels" : 4,
      "numEdgeLabels" : 2,
      "nodeLabels" : [
        "continent",
```

```
        "country",
        "version",
        "airport"
      ],
      "edgeLabels" : [
        "contains",
        "route"
      ],
      "numNodeProperties" : 14,
      "numEdgeProperties" : 1,
      "nodeProperties" : [
        {
          "desc" : 3748
        },
        {
          "code" : 3748
        },
        {
          "type" : 3748
        },
        {
          "country" : 3503
        },
        {
          "longest" : 3503
        },
        {
          "city" : 3503
        },
        {
          "lon" : 3503
        },
        {
          "elev" : 3503
        },
        {
          "icao" : 3503
        },
        {
          "region" : 3503
        },
        {
          "runways" : 3503
        },
```

```
        {
          "lat" : 3503
        },
        {
          "date" : 1
        },
        {
          "author" : 1
        }
      ],
      "edgeProperties" : [
        {
          "dist" : 50532
        }
      ],
      "totalNodePropertyValues" : 42773,
      "totalEdgePropertyValues" : 50532,
      "nodeStructures" : [
        {
          "count" : 3471,
          "nodeProperties" : [
            "city",
            "code",
            "country",
            "desc",
            "elev",
            "icao",
            "lat",
            "lon",
            "longest",
            "region",
            "runways",
            "type"
          ],
          "distinctOutgoingEdgeLabels" : [
            "route"
          ]
        },
        {
          "count" : 161,
          "nodeProperties" : [
            "code",
            "desc",
            "type"
```

```
        ],
        "distinctOutgoingEdgeLabels" : [
          "contains"
        ]
      },
      {
        "count" : 83,
        "nodeProperties" : [
          "code",
          "desc",
          "type"
        ],
        "distinctOutgoingEdgeLabels" : [ ]
      },
      {
        "count" : 32,
        "nodeProperties" : [
          "city",
          "code",
          "country",
          "desc",
          "elev",
          "icao",
          "lat",
          "lon",
          "longest",
          "region",
          "runways",
          "type"
        ],
        "distinctOutgoingEdgeLabels" : [ ]
      },
      {
        "count" : 1,
        "nodeProperties" : [
          "author",
          "code",
          "date",
          "desc",
          "type"
        ],
        "distinctOutgoingEdgeLabels" : [ ]
      }
    ],
```

```
      "edgeStructures" : [
        {
          "count" : 50532,
          "edgeProperties" : [
            "dist"
          ]
        }
      ]
    }
  }
}
```

Ejemplo de respuesta de resumen de un gráfico RDF

Este es el resumen detallado de la respuesta de un gráfico RDF que contiene el conjunto de datos de rutas aéreas RDF de ejemplo:

```
{
  "status" : "200 OK",
  "payload" : {
    "version" : "v1",
    "lastStatisticsComputationTime" : "2023-03-01T14:54:13.903Z",
    "graphSummary" : {
      "numDistinctSubjects" : 54403,
      "numDistinctPredicates" : 19,
      "numQuads" : 158571,
      "numClasses" : 4,
      "classes" : [
        "http://kelvinlawrence.net/air-routes/class/Version",
        "http://kelvinlawrence.net/air-routes/class/Airport",
        "http://kelvinlawrence.net/air-routes/class/Continent",
        "http://kelvinlawrence.net/air-routes/class/Country"
      ],
      "predicates" : [
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/route" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/dist" : 50656
        },
        {
          "http://kelvinlawrence.net/air-routes/objectProperty/contains" : 7004
        },
```

```
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/code" : 3747
        },
        {
          "http://www.w3.org/2000/01/rdf-schema#label" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/desc" : 3747
        },
        {
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" : 3747
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/icao" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lat" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/region" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/runways" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/longest" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/elev" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/lon" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/country" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/city" : 3502
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/author" : 1
```

```
        },
        {
          "http://kelvinlawrence.net/air-routes/datatypeProperty/date" : 1
        }
      ],
      "subjectStructures" : [
        {
          "count" : 50656,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/dist"
          ]
        },
        {
          "count" : 3471,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://kelvinlawrence.net/air-routes/objectProperty/route",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 238,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://kelvinlawrence.net/air-routes/objectProperty/contains",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
```

```
          "count" : 31,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/city",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/country",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/elev",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/icao",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lat",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/lon",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/longest",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/region",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/runways",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 6,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        },
        {
          "count" : 1,
          "predicates" : [
            "http://kelvinlawrence.net/air-routes/datatypeProperty/author",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/code",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/date",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/desc",
            "http://kelvinlawrence.net/air-routes/datatypeProperty/type",
            "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
            "http://www.w3.org/2000/01/rdf-schema#label"
          ]
        }
      ]
    }
  }
```

```
}
```

Uso de la autenticación AWS Identity and Access Management (IAM) con puntos finales resumidos en gráficos

Puede acceder a los puntos de conexión de resúmenes de gráficos de forma segura con la autenticación de IAM mediante awscurl o cualquier otra herramienta que funcione con HTTPS e IAM. Consulte Uso de `awscurl` con credenciales temporales para conectarse de forma segura a un clúster de base de datos con la autenticación de IAM habilitada para saber cómo configurar las credenciales adecuadas. Una vez que lo haya hecho, podrá realizar solicitudes como la siguiente:

```
awscurl "$GRAPH_SUMMARY_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

> **Important**
>
> La identidad o el rol de IAM que crea las credenciales temporales debe tener una política de IAM adjunta que permita la GetGraph acción resumida de IAM.

Consulte Errores de autenticación de IAM para ver una lista de los errores de IAM más comunes que puede encontrar.

Códigos de error comunes que puede devolver una solicitud de resumen de gráficos

Código de error de servicio de Neptune	Esta HTT	Mensaje	Escenario de error	Mitigación
`AccessDeniedException`	403	Falta token de autenticación	Se envió una solicitud sin firmar o firmada incorrectamente a la base de datos de Neptune con la IAM habilitada.	Firme la solicitud con SigV4 antes de enviarla (consulte IAM y resúmenes de gráficos).

Código de error de servicio de Neptune	Esta HTT	Mensaje	Escenario de error	Mitigación
	403	*El usuario: (ARN del usuario) no está autorizado a realizar: neptune-db: GetGraphSummary on resource: (ARN del recurso).*	La política de IAM no permite la acción GetGraphResumen cuando la solicitud de resumen del gráfico se envió a la base de datos de Neptune con la IAM habilitada.	Asegúrese de que la política de IAM asociada al usuario o rol que realiza la solicitud permita la acción `GetGraphSummary` .
`BadRequestException`	400	Las estadísticas están deshabilitadas, por lo que el resumen de gráficos también está deshabilitado.	Intento de obtener un resumen de los tipos de instancias ampliables (`t3` o `t4g`) en las que las estadísticas están deshabilitadas.	Use un tipo de instancia en el que la generación de estadísticas esté habilitada (todas las instancias compatibles excepto `t3` y `t4g`).
	400	Ruta incorrecta: */rdf/statistics/summarypathapi*	Solicitud enviada a una ruta no válida.	Utilice la ruta correcta para el punto de conexión del resumen de gráficos.
`InvalidParameterException`	400	La solicitud contiene parámetros desconocidos: '*(parámetro o parámetros desconocidos)* '.	Cuando se especifica un parámetro no válido en la solicitud.	Utilice únicamente parámetros válidos (por ejemplo, `mode`) en la solicitud.

Código de error de servicio de Neptune	Esta HTT	Mensaje	Escenario de error	Mitigación
InvalidParameterException	400	El parámetro de consulta de URI 'mode' tiene un valor no admitido '*(valor no válido)*'.	Cuando el parámetro de URL 'mode' de la solicitud va seguido de un valor no válido.	Utilice valores válidos (como `basic` o `detailed`) al especificar el parámetro de URL 'mode'.
MethodNotAllowedException	405	Método no permitido	Llamada al punto de conexión del resumen con cualquier método HTTP que no sea `GET` (por ejemplo, `POST` o `DELETE`).	Utilice el método HTTP `GET` al llamar al punto de conexión del resumen.
StatisticsNotAvailableException	400	Las estadísticas aún no se han calculado, el resumen gráfico estará disponible una vez que se realice este cálculo.	No hay estadísticas disponibles cuando la solicitud se envía al punto de conexión del resumen.	Espere hasta que se generen las estadísticas. Puede comprobar el estado de la generación de las estadísticas mediante la API de estado de estadísticas.
	400	Se ha alcanzado el límite de estadísticas, por lo que el resumen gráfico no está disponible.	La generación de estadísticas se detuvo porque se alcanzaron los límites de tamaño de las estadísticas.	El resumen de gráficos no está disponible en este gráfico.

Por ejemplo, si realiza una solicitud para representar gráficamente el punto de conexión del resumen en una base de datos de Neptune que tiene habilitada la autenticación de IAM y los permisos

necesarios no están presentes en la política de IAM del solicitante, obtendrá una respuesta como la siguiente:

```
{
  "detailedMessage": "User: arn:aws:iam::(account ID):(user or user name) is not
 authorized to perform: neptune-db:GetGraphSummary on resource: arn:aws:neptune-
db:(region):(account ID):(cluster resource ID)/*",
  "requestId": "7ac2b98e-b626-d239-1d05-74b4c88fce82",
  "code": "AccessDeniedException"
}
```

Conectividad JDBC de Amazon Neptune

Amazon Neptune ha publicado un controlador JDBC de código abierto que admite consultas de openCypher, Gremlin, SQL-Gremlin y SPARQL. La conectividad JDBC facilita la conexión a Neptune con herramientas de inteligencia empresarial (BI), como, por ejemplo, Tableau. El uso del controlador JDBC con Neptune no supone ningún costo adicional; solo tendrá que pagar los recursos de Neptune que se consuman.

El controlador es compatible con JDBC 4.2 y requiere al menos Java 8. Para obtener más información sobre el controlador JDBC, consulte la documentación de la API de Java JDBC.

El GitHub proyecto, en el que puede archivar problemas y abrir solicitudes de funciones, contiene documentación detallada para el controlador:

Controlador JDBC para Amazon Neptune

- Uso de SQL con el controlador JDBC
- Uso de Gremlin con el controlador JDBC
- Uso de openCypher con el controlador JDBC
- Uso de SPARQL con el controlador JDBC

Introducción al controlador JDBC de Neptune

Para usar el controlador JDBC de Neptune para conectarse a una instancia de Neptune, el controlador JDBC debe implementarse en una instancia de Amazon EC2 en la misma VPC que el clúster de base de datos de Neptune, o la instancia debe estar disponible a través de un túnel SSH o un equilibrador de carga. Se puede configurar un túnel SSH en el controlador interna o externamente.

Puede descargar el controlador desde aquí. El controlador viene empaquetado como un único archivo JAR con un nombre similar a `neptune-jdbc-1.0.0-all.jar`. Para usarlo, coloque el archivo JAR en la `classpath` de la aplicación. O bien, si la aplicación usa Maven o Gradle, puede usar los comandos de Maven o Gradle adecuados para instalar el controlador desde el JAR.

El controlador necesita una URL de conexión JDBC para conectarse con Neptune, en un formato como este:

```
jdbc:neptune:(connection
 type)://(host);property=value;property=value;...;property=value
```

Las secciones de cada lenguaje de consulta del GitHub proyecto describen las propiedades que puede configurar en la URL de conexión JDBC para ese lenguaje de consulta.

Si el archivo JAR está en la `classpath` de su aplicación, no es necesaria ninguna otra configuración. Puede conectar el controlador mediante la interfaz `DriverManager` de JDBC y una cadena de conexión de Neptune. Por ejemplo, si se puede acceder al clúster de base de datos de Neptune a través del punto de conexión `neptune-example.com` del puerto 8182, podrá conectarse con openCypher de la siguiente manera:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

void example() {
    String url = "jdbc:neptune:opencypher://bolt://neptune-example:8182";

    Connection connection = DriverManager.getConnection(url);
    Statement statement = connection.createStatement();

    connection.close();
}
```

Las secciones de documentación del GitHub proyecto para cada lenguaje de consulta describen cómo construir la cadena de conexión cuando se utiliza ese lenguaje de consulta.

Uso de Tableau con el controlador JDBC de Neptune

Para usar Tableau con el controlador JDBC de Neptune, comience por descargar e instalar la versión más reciente de Tableau Desktop. Descargue el archivo JAR del controlador JDBC de Neptune y también el archivo del conector Tableau de Neptune (un archivo `.taco`).

Para conectarse a Tableau para Neptune en un Mac

1. Coloque el archivo JAR del controlador JDBC de Neptune en la carpeta `/Users/(your user name)/Library/Tableau/Drivers`.
2. Coloque el archivo `.taco` del conector de Tableau de Neptune en la carpeta `/Users/(your user name)/Documents/My Tableau Repository/Connectors`.

3. Si tiene habilitada la autenticación de IAM, configure el entorno para ello. Tenga en cuenta que las variables de entorno configuradas en `.zprofile/`, `.zshenv/`, `.bash_profile`, etc., no funcionarán. Las variables de entorno deben configurarse de manera que una aplicación de interfaz gráfica de usuario pueda cargarlas.

   Una forma de configurar las credenciales consiste en colocar la clave de acceso y la clave secreta en el archivo `/Users/(your user name)/.aws/credentials`.

   Una forma sencilla de configurar la región de servicio consiste en abrir un terminal e introducir el siguiente comando, utilizando la región de la aplicación (por ejemplo, `us-east-1`):

   ```
   launchctl setenv SERVICE_REGION region name
   ```

   Hay otras formas de configurar las variables de entorno que persisten después de un reinicio, pero sea cual sea la técnica que utilice, debe establecer variables a las que pueda acceder una aplicación con interfaz gráfica de usuario.

4. Para que las variables de entorno se carguen en una GUI en el Mac, introduzca este comando en un terminal:

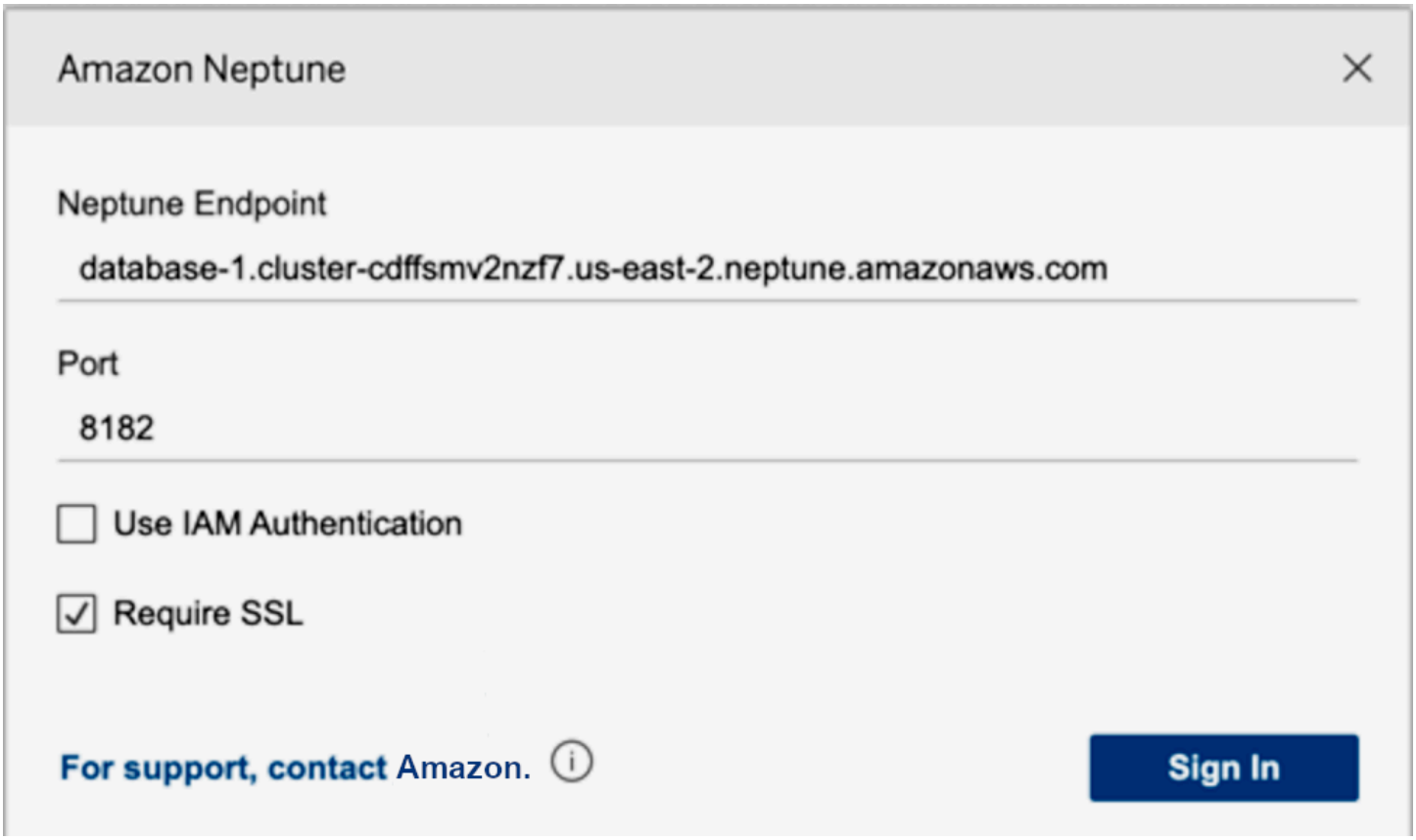
   ```
   /Applications/Tableau/Desktop/2021.1.app/Contents/MacOS/Tableau
   ```

Para conectarse a Tableau para Neptune en una máquina Windows

1. Coloque el archivo JAR del controlador JDBC de Neptune en la carpeta `C:\Program Files \Tableau\Drivers`.
2. Coloque el archivo `.taco` del conector de Tableau de Neptune en la carpeta `C:\Users\(your user name)\Documents\My Tableau Repository\Connectors`.
3. Si tiene habilitada la autenticación de IAM, configure el entorno para ello.

   Puede ser tan sencillo como configurar las variables de entorno `ACCESS_KEY`, `SECRET_KEY` y `SERVICE_REGION` del usuario.

Con Tableau abierto, seleccione Más en el lado izquierdo de la ventana. Si el archivo del conector de Tableau está ubicado correctamente, puede seleccionar Amazon Neptune por AWS en la lista que aparece:



No debería ser necesario editar el puerto ni añadir ninguna opción de conexión. Introduzca su punto de conexión de Neptune y establezca la configuración de IAM y SSL (debe habilitar SSL si utiliza IAM).

Al seleccionar Iniciar sesión, la conexión podría tardar más de 30 segundos si el gráfico es grande. Tableau recopila tablas de vértices y bordes y une los vértices en los bordes, además de crear visualizaciones.

Solución de problemas de conexión con un controlador JDBC

Si el controlador no se conecta al servidor, utilice la función `isValid` del objeto `Connection` de JDBC para comprobar si la conexión es válida. Si la función devuelve `false`, lo que significa que la conexión no es válida, compruebe que el punto de conexión al que se está conectando es correcto y que se encuentra en la VPC de su clúster de base de datos de Neptune o que tiene un túnel SSH válido hacia el clúster.

Si recibe una respuesta `No suitable driver found for` *`(connection string)`* de la llamada `DriverManager.getConnection`, es probable que haya un problema al principio de la cadena de conexión. Asegúrese de que la cadena de conexión comience así:

```
jdbc:neptune:opencypher://...
```

Para recopilar más información sobre la conexión, puede añadir un `LogLevel` a su cadena de conexión de la siguiente manera:

```
jdbc:neptune:opencypher://(JDBC URL):(port);logLevel=trace
```

Como alternativa, puede añadir `properties.put("logLevel", "trace")` en las propiedades de entrada para registrar la información de rastreo.

Actualizaciones del motor de Amazon Neptune

Amazon Neptune publica actualizaciones del motor con regularidad. Puede determinar qué versión del motor tiene instalada actualmente utilizando la API instance-status.

Las versiones de motor se enumeran en Versiones del motor para Amazon Neptune y los parches, en Últimas novedades.

En esta sección, encontrará más información sobre cómo se publican las actualizaciones y cómo actualizar el motor de Neptune en su base de datos en Mantenimiento de clústeres. Por ejemplo, la numeración de las versiones del motor se explica en Números de la versión del motor.

Seguridad en Amazon Neptune

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores independientes prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener más información acerca de los programas de conformidad que se aplican a Amazon Neptune, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. Usted también es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Neptune. En los siguientes temas, se le mostrará cómo configurar Neptune para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a usar otros AWS servicios que le ayudan a monitorear y proteger sus recursos de Neptune.

Temas

- Resiliencia en Amazon Neptune

Protección de datos en Amazon Neptune

El modelo de se aplica a protección de datos en Amazon Neptune. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-2 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabajas con Neptune u otro dispositivo Servicios de AWS mediante la consola, la API o AWS los AWS CLI SDK. Cualquier dato que

ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

> **Important**
>
> TLS 1.3 solo es compatible con la versión 1.3.2.0 y posteriores del motor de Neptune.

Utiliza las llamadas a la API AWS publicadas para administrar Neptune a través de la red. Los clientes deben ser compatibles con la seguridad de la capa de transporte (TLS) 1.2 o una versión posterior que utilice conjuntos de cifrado sólidos, tal y como se describe en Cifrado en tránsito. La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

En las secciones siguientes, se explica más en profundidad cómo se protegen los datos de Neptune.

Temas

Cada clúster de base de datos de Amazon Neptune reside en una Amazon VPC

Un clúster de base de datos de Amazon Neptune solo se puede crear en una Amazon Virtual Private Cloud (Amazon VPC) y solo se puede obtener acceso a sus puntos de conexión desde dicha VPC, normalmente desde una instancia de Amazon Elastic Compute Cloud (Amazon EC2) que se ejecuta en dicha VPC.

Puede proteger sus datos de Neptune limitando el acceso a la VPC en la que se encuentra el clúster de base de datos de Neptune, tal y como se describe en Conexión a su gráfico de Amazon Neptune.

Cifrado en tránsito: conexión con Neptune mediante SSL/HTTPS

A partir de la versión 1.0.4.0 del motor, Amazon Neptune solo permite conexiones de capa de conexión segura (SSL) a través de HTTPS a cualquier instancia o punto de conexión del clúster.

Neptune requiere al menos la versión 1.2 de TLS y utiliza los siguientes conjuntos de cifrado seguros:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

A partir de la versión 1.3.2.0 del motor Neptune, Neptune admite la versión 1.3 de TLS mediante los siguientes conjuntos de cifrado:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384

Incluso cuando las conexiones HTTP estaban permitidas en las versiones anteriores del motor, cualquier clúster de base de datos que utilice un nuevo grupo de parámetros de clúster de base de datos debe utilizar SSL de forma predeterminada. Para proteger sus datos, los puntos de conexión de Neptune en la versión de motor *`1.0.4.0`* y posteriores solo admiten solicitudes HTTPS. Para obtener más información, consulte Uso del punto de conexión HTTP REST para conectarse a una instancia de base de datos de Neptune.

Neptune proporciona automáticamente certificados SSL para las instancias de base de datos de Neptune. No tiene que solicitar ningún certificado. Los certificados se facilitan cuando crea una nueva instancia.

Neptune asigna un único certificado SSL comodín a las instancias de su cuenta para cada región. AWS El certificado proporciona entradas para los puntos de enlace del clúster, los puntos de enlace de solo lectura del clúster y los puntos de enlace de instancia.

Detalles del certificado

Las siguientes entradas se incluyen en el certificado suministrado:

- Punto de conexión de clúster: `*.cluster-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`
- Punto de conexión de solo lectura: `*.cluster-ro-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`

- Puntos de conexión de instancia: `*.a1b2c3d4wxyz.region.neptune.amazonaws.com`

Solo se admiten las entradas enumeradas aquí.

Conexiones de proxy

Los certificados solo admiten los nombres de host que se enumeran en la sección anterior.

Si utiliza un balanceador de carga o un servidor proxy (como HAProxy), debe utilizar la terminación SSL y tener su propio certificado SSL en el servidor proxy.

El acceso directo SSL no funciona porque los certificados SSL proporcionados no coinciden con el nombre de host del servidor proxy.

Certificados de CA raíz.

Los certificados de las instancias de Neptune normalmente se validan utilizando el almacén de confianza local del sistema operativo o el SDK (como el SDK de Java).

Si debe proporcionar un certificado raíz manualmente, puede descargar el certificado de CA raíz de Amazon en formato PEM desde el repositorio de políticas de confianza de Amazon Services.

Más información

Para obtener más información acerca de la conexión a puntos de conexión de Neptune con SSL, consulte the section called “Instalación de la consola de Gremlin” y the section called “HTTP REST”.

Cifrado de los recursos de Neptune en reposo

Las instancias cifradas de Neptune ofrecen una capa adicional de protección de datos al ayudarle a proteger los datos del acceso no autorizado al almacenamiento subyacente. Puede utilizar el cifrado de Neptune para aumentar la protección de datos de las aplicaciones implementadas en la nube. También puede usarlo para cumplir con los requisitos de conformidad en materia de cifrado. data-at-rest

Para administrar las claves utilizadas para cifrar y descifrar los recursos de Neptune, utilice ().AWS Key Management ServiceAWS KMS AWS KMS combina hardware y software seguros y de alta disponibilidad para proporcionar un sistema de administración de claves adaptado a la nube. Con AWS KMSél, puede crear claves de cifrado y definir las políticas que controlan cómo se pueden utilizar estas claves. AWS KMS es compatible AWS CloudTrail, por lo que puede auditar el uso de

las claves para comprobar que las claves se utilizan de forma adecuada. Puede utilizar AWS KMS las llaves en combinación con Neptune y AWS servicios compatibles, como Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS) y Amazon Redshift. Para obtener una lista de los servicios compatibles AWS KMS, consulte Cómo se usan AWS los servicios AWS KMS en la AWS Key Management Service Guía para desarrolladores.

Todos los registros, copias de seguridad e instantáneas de una instancia cifrada de Neptune están cifrados.

Habilitación del cifrado para una instancia de base de datos de Neptune

Para habilitar el cifrado en una instancia de base de datos de Neptune nueva, elija Sí en la sección Habilitar el cifrado de la consola de Neptune. Para obtener información acerca de la creación de una instancia de base de datos de Neptune, consulte Creación de un nuevo clúster de base de datos de Neptune.

Al crear una instancia de base de datos de Neptune cifrada, también puede proporcionar el identificador de AWS KMS clave para su clave de cifrado. Si no especifica un identificador de AWS KMS clave, Neptune utiliza la clave de cifrado de Amazon RDS predeterminada (`aws/rds`) para la nueva instancia de base de datos Neptune. AWS KMS crea la clave de cifrado predeterminada de Neptune para su AWS cuenta. Su AWS cuenta tiene una clave de cifrado predeterminada diferente para cada AWS región.

Después de crear una instancia de base de datos de Neptune cifrada, no puede cambiar la clave de cifrado de dicha instancia. Por tanto, asegúrese de determinar los requisitos de clave de cifrado antes de crear la instancia de base de datos de Neptune cifrada.

Puede utilizar el Nombre de recurso de Amazon (ARN) de una clave de otra cuenta para cifrar una instancia de base de datos de Neptune. Si crea una instancia de base de datos Neptune con la misma AWS cuenta propietaria de la clave de AWS KMS cifrado que se utiliza para cifrar esa nueva instancia de base de datos Neptune, el ID de AWS KMS clave que pase puede ser el alias de la clave en lugar del AWS KMS ARN de la clave.

> **Important**
>
> Si Neptune pierde el acceso a la clave de cifrado de una instancia de base de datos de Neptune (por ejemplo, cuando se revoca el acceso de Neptune a una clave), la instancia de base de datos cifrada se coloca en un estado de terminal y solo se puede restaurar desde una copia de seguridad. Recomendamos que siempre habilite las copias de seguridad para

las instancias de bases de datos cifradas de Neptune con el fin de protegerse contra la pérdida de los datos cifrados de dichas bases de datos.

Se necesitan permisos de claves para habilitar el cifrado

El rol o usuario de IAM que crea una instancia de base de datos de Neptune cifrada debe tener al menos los siguientes permisos para la clave KMS:

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:GenerateDataKey"`
- `"kms:ReEncryptTo"`
- `"kms:GenerateDataKeyWithoutPlaintext"`
- `"kms:CreateGrant"`
- `"kms:ReEncryptFrom"`
- `"kms:DescribeKey"`

A continuación, se muestra un ejemplo de una política de claves que incluye los permisos necesarios:

```
{
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable Permissions for root principal",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key for Neptune",
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ReEncryptTo",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:CreateGrant",
        "kms:ReEncryptFrom",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Deny use of the key for non Neptune",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/NeptuneFullAccess"
      },
      "Action": [
        "kms:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "kms:ViaService": "rds.us-east-1.amazonaws.com"
        }
      }
    }
  ]
}
```

- La primera instrucción de esta política es opcional. Da acceso a la entidad principal raíz del usuario.
- La segunda declaración proporciona acceso a todas las AWS KMS API necesarias para esta función, que dependen del director de servicio de RDS.

- La tercera afirmación refuerza aún más la seguridad al exigir que este rol no pueda utilizar esta clave para ningún otro servicio. AWS

También puede reducir aún más los permisos de `createGrant` añadiendo:

```
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
```

Limitaciones del cifrado de Neptune

Existen las siguientes limitaciones para cifrar clústeres de Neptune:

- No puede convertir un clúster de bases de datos sin cifrar en uno cifrado.

  Sin embargo, sí es posible restaurar una instantánea de clúster de base de datos de sin cifrar en un clúster de base de datos de cifrado. Para hacer esto, especifique una clave de cifrado KMS cuando restaure a partir de una instantánea de clúster de base de datos sin cifrar.
- No puede convertir una instancia de bases de datos sin cifrar en una cifrada. Solo puede habilitar el cifrado para una instancia de base de datos al crearla.
- Además, las instancias de bases de datos que están cifradas no se pueden modificar para deshabilitar el cifrado.
- No se puede tener una réplica de lectura cifrada de una instancia de base de datos sin cifrar ni una réplica de lectura sin cifrar de una instancia de base de datos cifrada.
- Las réplicas de lectura cifradas deben cifrarse con la misma clave que la instancia de base de datos de origen.

Descripción general de AWS Identity and Access Management (IAM) en Amazon Neptune

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los recursos. AWS Los administradores de IAM controlan quién puede estar autenticado (ha iniciado sesión) y autorizado (tiene permisos) para

utilizar recursos de Neptune. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Puede usar AWS Identity and Access Management (IAM) para autenticarse en su instancia de base de datos o clúster de base de datos de Neptune. Cuando la autenticación de bases de datos de IAM está habilitada, cada solicitud debe firmarse con AWS la versión 4 de Signature.

AWS La versión 4 de Signature agrega información de autenticación a AWS las solicitudes. Por motivos de seguridad, todas las solicitudes a los clústeres de bases de datos de Neptune con autenticación de IAM habilitada se deben firmar con una clave de acceso. La clave consta de un ID de clave de acceso y una clave de acceso secreta. La autenticación se administra de forma externa mediante políticas de IAM.

Neptune se autentica en la conexión y, en el caso de WebSockets las conexiones, verifica los permisos periódicamente para garantizar que el usuario siga teniendo acceso.

> **Note**
>
> - La revocación, eliminación o rotación de las credenciales asociadas al usuario de IAM no es recomendable porque no finaliza ninguna conexión que ya esté abierta.
> - Hay límites en el número de WebSocket conexiones simultáneas por instancia de base de datos y en cuanto al tiempo que una conexión puede permanecer abierta. Para obtener más información, consulte WebSockets Límites.

El uso de IAM depende de la función

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que se realice en Neptune.

Usuario de servicio: si utiliza el servicio de Neptune para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita para utilizar el plano de datos de Neptune. A medida que necesite más acceso para realizar su trabajo, comprender cómo se administra el acceso a los datos puede ayudarle a solicitar los permisos adecuados a su administrador.

Administrador de servicio: si está a cargo de los recursos de Neptune en su empresa, probablemente tenga acceso a las acciones de administración de Neptune, que corresponden a la API de administración de Neptune. También puede ser su trabajo determinar qué acciones y recursos de

acceso a datos de Neptune necesitan los usuarios del servicio para realizar su trabajo. En ese caso, un administrador de IAM puede aplicar políticas de IAM para cambiar los permisos de los usuarios de su servicio.

Administrador de IAM: si es administrador de IAM, tendrá que escribir políticas de IAM para la administración y el acceso de datos a Neptune. Para ver ejemplos de políticas basadas en la identidad de Neptune que puede utilizar, consulte Uso de diferentes tipos de políticas de IAM para controlar el acceso a Neptune.

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar las solicitudes de la AWS API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWSen la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console cambiando de función. Puede

asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para conocer la diferencia entre las funciones y las políticas basadas en recursos para el acceso entre cuentas, consulte el tema sobre el acceso a los recursos entre cuentas en IAM en la Guía del usuario de IAM.
- Acceso entre servicios: algunos utilizan funciones en otros. Servicios de AWS Servicios de AWS Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para

obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

- Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
- Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Habilitación de la autenticación de base de datos de IAM en Neptune

De forma predeterminada, la autenticación de bases de datos de IAM está deshabilitada al crear un clúster de bases de datos de Amazon Neptune. Puede habilitarla (o deshabilitarla de nuevo) mediante la AWS Management Console.

Para crear un nuevo clúster de base de datos de Neptune con autenticación de IAM mediante la consola, debe seguir las instrucciones para crear un clúster de base de datos de Neptune en Lanzamiento de un clúster de base de datos de Neptune mediante la AWS Management Console.

En la segunda página del proceso de creación, en Enable IAM DB Authentication (Habilitar autenticación de bases de datos de IAM), seleccione Yes (Sí).

Para habilitar o deshabilitar la autenticación de IAM para una instancia o clúster de base de datos existente

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. Elija el clúster de base de datos Neptune que desea modificar y elija Acciones del clúster. A continuación, seleccione Modify cluster (Modificar clúster).
4. En la sección Database options (Opciones de base de datos), en IAM DB Authentication (Autenticación de base de datos de IAM), elija Enable IAM DB authorization (Habilitar autorización de base de datos de IAM) o No (para deshabilitarla). Después elija Continue (Continuar).
5. Para aplicar los cambios inmediatamente, elija Apply immediately.
6. Elija Modify Cluster (Modificar clúster).

Conexión y firma con AWS Signature, versión 4

Los recursos de Amazon Neptune que tienen habilitada la autenticación de base de datos de IAM requieren que todas las solicitudes HTTP se firmen con AWS Signature Version 4. Para obtener información general sobre la firma de las solicitudes con la versión 4 de AWS Signature, consulte Firmar las solicitudes de la AWS API.

AWS La versión 4 de Signature es el proceso para añadir información de autenticación a AWS las solicitudes. Por motivos de seguridad, la mayoría de las solicitudes AWS deben firmarse con una clave de acceso, que consiste en un identificador de clave de acceso y una clave de acceso secreta.

> **Note**
>
> Si utiliza las credenciales temporales, caducan después de un intervalo especificado, incluido el token de sesión.
> Tiene que actualizar el token de sesión cuando solicite nuevas credenciales. Para obtener más información, consulte Uso de credenciales de seguridad temporales para solicitar acceso a AWS los recursos.

> **Important**
>
> Para tener acceso a Neptune con la autenticación basada en IAM, debe crear solicitudes HTTP y firmarlas usted mismo.

Cómo funciona Signature Version 4

1. Se crea una solicitud canónica.
2. Utiliza la solicitud canónica y alguna otra información para crear una. string-to-sign
3. Usas tu clave de acceso AWS secreta para obtener una clave de firma y, a continuación, usas esa clave de firma y la string-to-sign para crear una firma.
4. Se añade la firma resultante a la solicitud HTTP en un encabezado o como parámetro de cadena de consulta.

Cuando Neptune recibe la solicitud, lleva a cabo los mismos pasos realizados para calcular la firma. A continuación, Neptune compara la firma calculada con la que se ha enviado con la solicitud. Si las firmas coinciden, la solicitud se procesa. Si las firmas no coinciden, la solicitud se deniega.

Para obtener información general sobre la firma de las solicitudes con AWS la versión 4 de la firma, consulte el proceso de firma de la versión 4 en Referencia general de AWS.

Las siguientes secciones contienen ejemplos que muestran cómo enviar solicitudes firmadas a los puntos de conexión de Gremlin y SPARQL de una instancia de base de datos de Neptune con la autenticación de IAM habilitada.

Temas

- Requisitos previos en Amazon Linux EC2
- Uso de una herramienta de línea de comandos para enviar consultas a su clúster de base de datos de Neptune
- Conexión con Neptune mediante la consola de Gremlin con firma de Signature Version 4
- Conexión con Neptune mediante Java y Gremlin con firma de Signature Version 4
- Conexión con Neptune mediante Java y SPARQL con firma de Signature Version 4 (RDF4J y Jena)
- Conexión con Neptune mediante SPARQL y Node.js con firma de Signature Version 4
- Ejemplo: conexión con Neptune mediante Python con firma de Signature Version 4

Requisitos previos en Amazon Linux EC2

A continuación se muestran instrucciones para instalar Apache Maven y Java 8 en una instancia de Amazon EC2. Estos son necesarios para los ejemplos de autenticación de Signature Version 4 de Amazon Neptune.

Para instalar Apache Maven y Java 8 en la instancia EC2

1. Conéctese con la instancia de Amazon EC2 por medio de un cliente SSH.
2. Instale Apache Maven en la instancia EC2. En primer lugar, escriba lo siguiente para añadir un repositorio con un paquete de Maven.

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Escriba lo siguiente para establecer el número de versión de los paquetes.

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   A continuación, puede usar el comando yum para instalar Maven.

   ```
   sudo yum install -y apache-maven
   ```
3. Las bibliotecas de Gremlin requieren Java 8. Escriba lo siguiente para instalar Java 8 en la instancia EC2.

   ```
   sudo yum install java-1.8.0-devel
   ```
4. Escriba lo siguiente para establecer Java 8 como tiempo de ejecución predeterminado en la instancia EC2.

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Cuando se le solicite, escriba el número para Java 8.
5. Escriba lo siguiente para establecer Java 8 como compilador predeterminado en la instancia EC2.

   ```
   sudo /usr/sbin/alternatives --config javac
   ```

Cuando se le solicite, escriba el número para Java 8.

Uso de una herramienta de línea de comandos para enviar consultas a su clúster de base de datos de Neptune

Es muy práctico disponer de una herramienta de línea de comandos para enviar consultas al clúster de base de datos de Neptune, como se ilustra en muchos de los ejemplos de esta documentación. La herramienta curl es una excelente opción para comunicarse con los puntos de conexión de Neptune si la autenticación de IAM no está habilitada.

Sin embargo, para mantener sus datos seguros, es mejor habilitar la autenticación de IAM.

Cuando la autenticación de IAM está habilitada, cada una de las solicitudes debe firmarse con Signature Version 4 (Sig4). La herramienta de línea de comandos awscurl de terceros utiliza la misma sintaxis que `curl` y puede firmar consultas mediante la firma Sig4. En la sección Uso de `awscurl` a continuación, se explica cómo utilizar `awscurl` de forma segura con credenciales temporales.

Configuración de una herramienta de línea de comandos para usar HTTPS

Neptune requiere que todas las conexiones usen HTTPS. Cualquier herramienta de línea de comandos, como `curl` o `awscurl`, necesita acceso a los certificados correspondientes para poder usar HTTPS. Mientras `curl` o `awscurl` puedan localizar los certificados apropiados, gestionan las conexiones HTTPS igual que las HTTP, sin necesidad de parámetros adicionales. Los ejemplos de esta documentación se basan en ese escenario.

Para saber cómo obtener dichos certificados y cómo formatearlos correctamente en un almacén de certificados de autoridad de certificación (CA) que `curl` pueda utilizar, consulte SSL Certificate Verification en la documentación de `curl`.

A continuación, puede especificar la ubicación de este almacén de certificados de CA mediante la variable de entorno `CURL_CA_BUNDLE`. En Windows, `curl` lo busca de forma automática en un archivo llamado `curl-ca-bundle.crt`. Primero busca en el mismo directorio que `curl.exe` y después en el resto de sitios de la ruta. Para obtener más información, consulte Certificados SSL.

Uso de **awscurl** con credenciales temporales para conectarse de forma segura a un clúster de base de datos con la autenticación de IAM habilitada

La herramienta awscurl usa la misma sintaxis que `curl`, pero también necesita información adicional:

- **`--access_key`**: una clave de acceso válida. Si no se proporciona con este parámetro, debe proporcionarse en la variable de entorno `AWS_ACCESS_KEY_ID` o en un archivo de configuración.
- **`--secret_key`**: clave de acceso secreta que corresponde a la clave de acceso. Si no se proporciona con este parámetro, debe proporcionarse en la variable de entorno `AWS_SECRET_ACCESS_KEY` o en un archivo de configuración.
- **`--security_token`**: un token de sesión válido. Si no se proporciona con este parámetro, debe proporcionarse en la variable de entorno `AWS_SECURITY_TOKEN` o en un archivo de configuración.

En el pasado, era una práctica habitual utilizar credenciales persistentes con `awscurl`, como las credenciales de usuario de IAM o incluso las credenciales raíz, pero no se recomienda. En su lugar, genere credenciales temporales mediante una de las API del AWS Security Token Service (STS) o uno de sus encapsuladores de AWS CLI.

Es mejor colocar los valores `AccessKeyId`, `SecretAccessKey` y `SessionToken` que devuelve la llamada de STS en las variables de entorno adecuadas de la sesión del intérprete de comandos, en lugar de en un archivo de configuración. A continuación, cuando el intérprete de comandos finaliza, las credenciales se descartan automáticamente, lo que no ocurre con un archivo de configuración. Del mismo modo, no solicite una duración mayor para las credenciales temporales de la que es probable que necesite.

El siguiente ejemplo muestra los pasos que puede seguir en un intérprete de comandos de Linux para obtener credenciales temporales que sean válidas durante media hora utilizando sts assume-role y, a continuación, colocarlas en variables de entorno donde `awscurl` pueda encontrarlas:

```
aws sts assume-role \
    --duration-seconds 1800 \
    --role-arn "arn:aws:iam::(account-id):role/(rolename)" \
    --role-session-name AWSCLI-Session > $output
AccessKeyId=$(cat $output | jq '.Credentials''.AccessKeyId')
SecretAccessKey=$(cat $output | jq '.Credentials''.SecretAccessKey')
SessionToken=$(cat $output | jq '.Credentials''.SessionToken')
```

```
export AWS_ACCESS_KEY_ID=$AccessKeyId
export AWS_SECRET_ACCESS_KEY=$SecretAccessKey
export AWS_SESSION_TOKEN=$SessionToken
```

A continuación, puede utilizar `awscurl` para realizar una solicitud firmada a su clúster de base de datos de una forma similar a la siguiente:

```
awscurl (your cluster endpoint):8182/status \
    --region us-east-1 \
    --service neptune-db
```

Conexión con Neptune mediante la consola de Gremlin con firma de Signature Version 4

La forma en que se conecte a Amazon Neptune mediante la consola Gremlin con la autenticación Signature Version 4 depende de si utiliza la TinkerPop versión superior `3.4.11` o anterior. En cualquiera de los casos, estos son los requisitos previos:

- Debe tener las credenciales de IAM necesarias para firmar las solicitudes. Consulte Uso de la cadena de proveedores de credenciales predeterminada en la Guía para desarrolladores. AWS SDK for Java
- Debe tener instalada una versión de consola de Gremlin que sea compatible con la versión del motor de Neptune que utiliza su clúster de base de datos.

Si usa credenciales temporales, estas caducan después de un intervalo específico, al igual que el token de sesión, por lo que debe actualizar su token de sesión cuando solicite nuevas credenciales. Consulte Uso de credenciales de seguridad temporales para solicitar acceso a AWS los recursos en la Guía del usuario de IAM.

Si necesita ayuda para conectarse mediante SSL/TLS, consulte Configuración de SSL/TLS.

Usar TinkerPop 3.4.11 o superior para conectarse a Neptune con la firma Sig4

Con la versión TinkerPop 3.4.11 o superior, utilizarás`handshakeInterceptor()`, que proporciona una forma de conectar un firmante Sigv4 a la conexión establecida por el comando. `:remote`
Al igual que con el enfoque utilizado para Java, es necesario configurar el objeto `Cluster` manualmente y, a continuación, pasarlo al comando `:remote`.

Tenga en cuenta que esto es muy diferente de la situación típica en la que el comando `:remote` utiliza un archivo de configuración para formar la conexión. El enfoque del archivo de configuración no funciona porque `handshakeInterceptor()` debe configurarse mediante programación y no puede cargar su configuración desde un archivo.

Conecta la consola Gremlin (TinkerPop 3.4.11 y superior) con la firma Sig4

1. Inicie la consola de Gremlin:

   ```
   $ bin/gremlin.sh
   ```

2. En el símbolo del sistema de `gremlin>`, instale la biblioteca `amazon-neptune-sigv4-signer` (esto solo debe hacerse una vez para la consola):

   ```
   :install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
   ```

   Si tienes problemas con este paso, puede ser útil consultar la documentación sobre la TinkerPop configuración de Grape.

   > **Note**
   >
   > Si utiliza un proxy HTTP, es posible que encuentre errores en este paso si el comando `:install` no se completa. Para solucionar este problema, ejecute los siguientes comandos para informar a la consola acerca del proxy:
   >
   > ```
   > System.setProperty("https.proxyHost", "(the proxy IP address)")
   > System.setProperty("https.proxyPort", "(the proxy port)")
   > ```

3. Importe la clase necesaria para gestionar el inicio de sesión en `handshakeInterceptor()`:

   ```
   :import com.amazonaws.auth.DefaultAWSCredentialsProviderChain
   :import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer
   ```

4. Si utiliza credenciales temporales, también tendrá que proporcionar su token de sesión de la siguiente manera:

   ```
   System.setProperty("aws.sessionToken","(your session token)")
   ```

5. Si no ha establecido las credenciales de su cuenta de otro modo, puede asignarlas de la siguiente manera:

```
System.setProperty("aws.accessKeyId","(your access key)")
System.setProperty("aws.secretKey","(your secret key)")
```

6. Construya manualmente el objeto `Cluster` para conectarlo a Neptune:

```
cluster = Cluster.build("(host name)")  \
                .enableSsl(true) \
                .handshakeInterceptor { r ->  \
                  def sigV4Signer = new NeptuneNettyHttpSigV4Signer("(Amazon
 region)", \
                                        new DefaultAWSCredentialsProviderChain()); \
                  sigV4Signer.signRequest(r); \
                  return r; } \
                .create()
```

   Para obtener ayuda para encontrar el nombre de host de una instancia de base de datos de Neptune, consulte Conexión a los puntos de conexión de Amazon Neptune.

7. Establezca la conexión `:remote` mediante el nombre de la variable del objeto `Cluster` en el paso anterior:

```
:remote connect tinkerpop.server cluster
```

8. Introduzca el siguiente comando para cambiar al modo remoto. Esto envía todas las consultas de Gremlin a la conexión remota:

```
:remote console
```

Uso de una versión TinkerPop anterior a la 3.4.11 para conectarse a Neptune con la firma Sig4

Con la TinkerPop versión 3.4.10 o inferior, utilice la `amazon-neptune-gremlin-java-sigv4` biblioteca proporcionada por Neptune para conectar la consola a Neptuno con la firma Sigv4, tal y como se describe a continuación:

Conecta la consola Gremlin (TinkerPop versiones anteriores a la 3.4.11) con la firma Sig4

1. Inicie la consola de Gremlin:

```
$ bin/gremlin.sh
```

2. En el símbolo del sistema de `gremlin>`, instale la biblioteca `amazon-neptune-sigv4-signer` (esto solo debe hacerse una vez para la consola):

```
:install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
```

> **Note**
>
> Si utiliza un proxy HTTP, es posible que encuentre errores en este paso si el comando `:install` no se completa. Para solucionar este problema, ejecute los siguientes comandos para informar a la consola acerca del proxy:
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```
>
> También puede ser útil consultar la documentación sobre la configuración de GrapeTinkerPop .

3. En el subdirectorio `conf` del directorio extraído, cree un archivo llamado `neptune-remote.yaml`.

   Si utilizó la AWS CloudFormation plantilla para crear su clúster de base de datos de Neptune, ya existirá un `neptune-remote.yaml` archivo. En ese caso, todo lo que tiene que hacer es editar el archivo existente para incluir la configuración del canalizador que se muestra a continuación.

   De lo contrario, copie el texto siguiente en el archivo y sustituya *(nombre de host)* por el nombre de host o la dirección IP de su instancia de base de datos de Neptune. Tenga en cuenta que los corchetes ([]) que rodean el nombre del host son obligatorios.

```
hosts: [(host name)]
port: 8182
connectionPool: {
  channelizer: org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer,
  enableSsl: true
```

```
}
serializer: { className:
 org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: true }}
```

4.

> **Important**
>
> Debe proporcionar credenciales de IAM para firmar las solicitudes. Introduzca los siguientes comandos para definir sus credenciales como variables de entorno, sustituyendo los elementos pertinentes por sus credenciales.
>
> ```
> export AWS_ACCESS_KEY_ID=access_key_id
> export AWS_SECRET_ACCESS_KEY=secret_access_key
> export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or
>  ca-central-1 or
>                       sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or
>  eu-west-3 or eu-central-1 or me-south-1 or
>                       me-central-1 or il-central-1 or af-south-1 or
>  ap-east-1 or ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-
> southeast-2 or ap-south-1 or
>                       cn-north-1 or cn-northwest-1 or
>                       us-gov-east-1 or us-gov-west-1
> ```
>
> El firmante de Neptune Versión 4 utiliza la cadena predeterminada de proveedores de credenciales. Para conocer métodos adicionales para proporcionar credenciales, consulte Using the Default Credential Provider Chain en la Guía para desarrolladores de AWS SDK for Java .
> La variable `SERVICE_REGION` es obligatoria, incluso cuando se utiliza un archivo de credenciales.

5. Establezca la conexión `:remote` mediante el archivo `.yaml`:

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

6. Introduzca el siguiente comando para cambiar al modo remoto, que envía todas las consultas de Gremlin a la conexión remota:

```
:remote console
```

Conexión con Neptune mediante Java y Gremlin con firma de Signature Version 4

Usar TinkerPop 3.4.11 o superior para conectarse a Neptune con la firma Sig4

A continuación, se muestra un ejemplo de cómo conectarse a Neptune mediante la API Java de Gremlin con firma Sig4 cuando se utiliza la versión TinkerPop 3.4.11 o superior (se supone tener conocimientos generales sobre el uso de Maven). Primero, defina las dependencias como parte del archivo `pom.xml`:

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-sigv4-signer</artifactId>
  <version>2.4.0</version>
</dependency>
```

Luego, utilice código como el siguiente:

```
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import com.amazonaws.neptune.auth.NeptuneSigV4SignerException;

 ...

System.setProperty("aws.accessKeyId","your-access-key");
System.setProperty("aws.secretKey","your-secret-key");

 ...

Cluster cluster = Cluster.build((your cluster))
                 .enableSsl(true)
                 .handshakeInterceptor( r ->
                  {
                    try {
                      NeptuneNettyHttpSigV4Signer sigV4Signer =
                        new NeptuneNettyHttpSigV4Signer("(your region)", new
 DefaultAWSCredentialsProviderChain());
                      sigV4Signer.signRequest(r);
                    } catch (NeptuneSigV4SignerException e) {
                      throw new RuntimeException("Exception occurred while signing the
 request", e);
```

```
                    }
                    return r;
                  }
                ).create();
try {
  Client client = cluster.connect();
  client.submit("g.V().has('code','IAD')").all().get();
} catch (Exception e) {
  throw new RuntimeException("Exception occurred while connecting to cluster", e);
}
```

> **Note**
>
> Si está actualizando desde `3.4.11`, elimine las referencias a la biblioteca de `amazon-neptune-gremlin-java-sigv4`. Ya no es necesario cuando se utiliza `handshakeInterceptor()` como se muestra en el ejemplo anterior. No intente utilizarla `handshakeInterceptor()` junto con el canalizador (`SigV4WebSocketChannelizer.class`), ya que se producirán errores.

Uso de una versión TinkerPop anterior a la 3.4.11 para conectarse a Neptune con la firma Sig4

TinkerPop las versiones anteriores `3.4.11` no eran compatibles con la `handshakeInterceptor()` configuración que se muestra en la sección anterior y, por lo tanto, deben basarse en el `amazon-neptune-gremlin-java-sigv4` paquete. Esta es una biblioteca de Neptune que contiene la `SigV4WebSocketChannelizer` clase, que reemplaza el TinkerPop Channelizer estándar por uno que puede inyectar automáticamente una firma SiGv4. Siempre que sea posible, actualice a la versión TinkerPop 3.4.11 o superior, ya que la biblioteca está obsoleta. `amazon-neptune-gremlin-java-sigv4`

Este es un ejemplo de cómo conectarse a Neptune mediante la API Java de Gremlin con firma Sig4 cuando se utilizan TinkerPop versiones anteriores a la 3.4.11 (se supone tener conocimientos generales sobre cómo usar Maven).

Primero, defina las dependencias como parte del archivo `pom.xml`:

```
<dependency>
  <groupId>com.amazonaws</groupId>
```

```
  <artifactId>amazon-neptune-gremlin-java-sigv4</artifactId>
  <version>2.4.0</version>
</dependency>
```

La dependencia anterior incluirá la versión del controlador de Gremlin `3.4.10`. Aunque es posible utilizar versiones más recientes del controlador de Gremlin (hasta la `3.4.13`), la actualización del controlador a partir de la versión 3.4.10 debería incluir un cambio para utilizar el modelo de `handshakeInterceptor()` descrito anteriormente.

A continuación, el objeto de clúster `gremlin-driver` debe configurarse de la siguiente manera en el código Java:

```
import org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer;

 ...

Cluster cluster = Cluster.build(your cluster)
                        .enableSsl(true)
                        .channelizer(SigV4WebSocketChannelizer.class)
                        .create();
Client client = cluster.connect();
client.submit("g.V().has('code','IAD')").all().get();
```

Conexión con Neptune mediante Java y SPARQL con firma de Signature Version 4 (RDF4J y Jena)

En esta sección, se muestra cómo conectarse a Neptune mediante RDF4J o Apache Jena con la autenticación Signature Version 4.

Requisitos previos

- Java 8 o superior.
- Apache Maven 3.3 o superior.

  Para obtener información sobre la instalación de estos requisitos previos en una instancia EC2 que ejecuta Amazon Linux, consulte Requisitos previos en Amazon Linux EC2.
- Credenciales de IAM para firmar las solicitudes. Para obtener más información, consulte Using the Default Credential Provider Chain en la Guía para desarrolladores de AWS SDK for Java .

> **Note**
>
> Si utiliza las credenciales temporales, caducan después de un intervalo especificado, incluido el token de sesión.
> Tiene que actualizar el token de sesión cuando solicite nuevas credenciales. Para obtener más información, consulte Uso de credenciales de seguridad temporales para solicitar acceso a los recursos en la Guía del usuario de IAM AWS.

- Establezca la variable `SERVICE_REGION` en una de las siguientes e indique la región de su instancia de base de datos de Neptune:
  - Este de EE. UU. (Norte de Virginia): `us-east-1`
  - Este de EE. UU. (Ohio): `us-east-2`
  - Oeste de EE. UU. (Norte de California): `us-west-1`
  - Oeste de EE. UU. (Oregón): `us-west-2`
  - Canadá (centro): `ca-central-1`
  - América del Sur (São Paulo): `sa-east-1`
  - Europa (Estocolmo): `eu-north-1`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londres): `eu-west-2`
  - Europa (París): `eu-west-3`
  - Europa (Fráncfort): `eu-central-1`
  - Medio Oriente (Baréin): `me-south-1`
  - Medio Oriente (EAU): `me-central-1`
  - Israel (Tel Aviv): `il-central-1`
  - África (Ciudad del Cabo): `af-south-1`
  - Asia Pacífico (Hong Kong): `ap-east-1`
  - Asia-Pacífico (Tokio): `ap-northeast-1`
  - Asia-Pacífico (Seúl): `ap-northeast-2`
  - Asia-Pacífico (Osaka): `ap-northeast-3`
  - Asia-Pacífico (Singapur): `ap-southeast-1`
  - Asia-Pacífico (Sídney): `ap-southeast-2`

- Asia-Pacífico (Bombay): `ap-south-1`
- China (Pekín): `cn-north-1`
- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (EEUU-Oeste): `us-gov-west-1`
- AWS GovCloud (EE. UU.-Este): `us-gov-east-1`

Para conectarse con Neptune mediante RDF4J o Apache Jena con firma de Signature Version 4

1. Clona el repositorio de muestras desde GitHub.

```
git clone https://github.com/aws/amazon-neptune-sparql-java-sigv4.git
```

2. Cambie al directorio clonado.

```
cd amazon-neptune-sparql-java-sigv4
```

3. Obtenga la versión más reciente del proyecto al revisar la ramificación con la etiqueta más reciente.

```
git checkout $(git describe --tags `git rev-list --tags --max-count=1`)
```

4. Introduzca uno de los siguientes comandos para compilar y ejecutar el código de ejemplo.

   Sustituya *`your-neptune-endpoint`* por el nombre de host o la dirección IP de su instancia de base de datos de Neptune. El puerto predeterminado es 8182.

   > **Note**
   >
   > Para obtener información acerca de cómo encontrar el nombre de host de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

   Eclipse RDF4J

   Para ejecutar el ejemplo de RDF4J, introduzca lo siguiente.

```
mvn compile exec:java \
```

```
    -Dexec.mainClass="com.amazonaws.neptune.client.rdf4j.NeptuneRdf4JSigV4Example"
 \
    -Dexec.args="https://your-neptune-endpoint:port"
```

Apache Jena

Para ejecutar el ejemplo de Apache Jena, introduzca lo siguiente.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.jena.NeptuneJenaSigV4Example" \
    -Dexec.args="https://your-neptune-endpoint:port"
```

5. Para ver el código fuente del ejemplo, consulte los ejemplos del directorio `src/main/java/com/amazonaws/neptune/client/`.

Para utilizar el controlador de firma SigV4 en su propia aplicación de Java, añada el paquete de Maven `amazon-neptune-sigv4-signer` a la sección `<dependencies>` del archivo `pom.xml`. Recomendamos que utilice los ejemplos como punto de partida.

Conexión con Neptune mediante SPARQL y Node.js con firma de Signature Version 4

Consultas mediante la firma Signature V4 y el AWS SDK para Javascript V3

A continuación, se muestra un ejemplo de cómo conectarse a Neptune SPARQL mediante Node.js con la autenticación Signature Version 4 y el AWS SDK para Javascript V3:

```
const { HttpRequest }  = require('@smithy/protocol-http');
const { fromNodeProviderChain } = require('@aws-sdk/credential-providers');
const { SignatureV4 } = require('@smithy/signature-v4');
const { Sha256 } = require('@aws-crypto/sha256-universal');
const https = require('https');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>
```

```
SELECT ?movies ?title WHERE {
  ?jel prop:name "James Earl Jones" .
  ?movies ?p2 ?jel .
  ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {
  var request = new HttpRequest({
    hostname: neptune_endpoint,
    port: 8182,
    path: 'sparql',
    body: encodeURI(query),
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
      'host': neptune_endpoint + ':8182',
    },
    method: 'POST',
  });

  const credentialProvider = fromNodeProviderChain();
  let credentials = credentialProvider();
  credentials.then(
    (cred)=>{
      var signer = new SignatureV4({credentials: cred, region: region, sha256: Sha256,
 service: 'neptune-db'});
      signer.sign(request).then(
        (req)=>{
          var responseBody = '';
          var sendreq = https.request(
            {
              host: req.hostname,
              port: req.port,
              path: req.path,
              method: req.method,
              headers: req.headers,
            },
          (res) => {
            res.on('data', (chunk) => { responseBody += chunk; });
            res.on('end', () => {
                console.log(JSON.parse(responseBody));
            });
```

```
          });
          sendreq.write(req.body);
          sendreq.end();
        }
      );
    },
    (err)=>{
      console.error(err);
    }
  );
}
```

Consultas mediante la firma Signature V4 y el SDK para Javascript V2 AWS

A continuación, se muestra un ejemplo de cómo conectarse a Neptune SPARQL mediante Node.js con la autenticación Signature Version 4 y el AWS SDK para Javascript V2:

```
var AWS = require('aws-sdk');

var region = 'us-west-2'; // e.g. us-west-1
var neptune_endpoint = 'your-Neptune-cluster-endpoint';  // like: 'cluster-
id.region.neptune.amazonaws.com'
var query = `query=PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX class: <http://aws.amazon.com/neptune/csv2rdf/class/>
PREFIX resource: <http://aws.amazon.com/neptune/csv2rdf/resource/>
PREFIX prop: <http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/>
PREFIX objprop: <http://aws.amazon.com/neptune/csv2rdf/objectProperty/>

SELECT ?movies ?title WHERE {
    ?jel prop:name "James Earl Jones" .
    ?movies ?p2 ?jel .
    ?movies prop:title ?title
} LIMIT 10`;

runQuery(query);

function runQuery(q) {

    var endpoint = new AWS.Endpoint(neptune_endpoint);
    endpoint.port = 8182;
    var request = new AWS.HttpRequest(endpoint, region);
    request.path += 'sparql';
    request.body = encodeURI(query);
```

```
    request.headers['Content-Type'] = 'application/x-www-form-urlencoded';
    request.headers['host'] = neptune_endpoint;
    request.method = 'POST';

    var credentials = new AWS.CredentialProviderChain();
    credentials.resolve((err, cred)=>{
        var signer = new AWS.Signers.V4(request, 'neptune-db');
        signer.addAuthorization(cred, new Date());
    });

    var client = new AWS.HttpClient();
    client.handleRequest(request, null, function(response) {
        console.log(response.statusCode + ' ' + response.statusMessage);
        var responseBody = '';
        response.on('data', function (chunk) {
            responseBody += chunk;
        });
        response.on('end', function (chunk) {
            console.log('Response body: ' + responseBody);
        });
    }, function(error) {
        console.log('Error: ' + error);
    });
}
```

Ejemplo: conexión con Neptune mediante Python con firma de Signature Version 4

En esta sección, se muestra un programa de ejemplo escrito en Python que ilustra cómo trabajar con Signature Version 4 para Amazon Neptune. Se basa en los ejemplos de la sección Proceso de firma Signature Version 4 en la Referencia general de Amazon Web Services.

Para utilizar este programa de ejemplo, se necesita lo siguiente:

- Python 3.x instalado en el equipo; puede obtenerlo en el sitio de Python. Estos programas se han probado con Python 3.6.
- La biblioteca requests de Python, que se utiliza en el script de ejemplo para realizar solicitudes web. Una forma cómoda de instalar paquetes de Python es utilizar `pip`, que obtiene los paquetes del sitio web Python Package Index. A continuación, puede instalar `requests` ejecutando `pip install requests` en la línea de comandos.

- Una clave de acceso (ID de clave de acceso y clave de acceso secreta) en las variables de entorno denominadas `AWS_ACCESS_KEY_ID` y `AWS_SECRET_ACCESS_KEY`. Como práctica recomendada, es conveniente que no incluya las credenciales en el código. Para más información, consulte Prácticas recomendadas para cuentas de AWS en la Guía de referencia de AWS Account Management .

  La región de su clúster de base de datos de Neptune en la variable de entorno denominada `SERVICE_REGION`.

  Si está utilizando credenciales temporales, debe especificar `AWS_SESSION_TOKEN` además de `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `SERVICE_REGION`.

  > **Note**
  >
  > Si utiliza las credenciales temporales, caducan después de un intervalo especificado, incluido el token de sesión.
  > Tiene que actualizar el token de sesión cuando solicite nuevas credenciales. Para obtener más información, consulte Uso de credenciales de seguridad temporales para solicitar acceso a los recursos de AWS.

En el siguiente ejemplo, se muestra cómo realizar solicitudes firmadas a Neptune mediante Python. Se realiza una solicitud GET o POST. La información de autenticación se pasa mediante el encabezado `Authorization` de la solicitud.

Este ejemplo también funciona como una AWS Lambda función. Para obtener más información, consulte the section called “Configuración de Lambda”.

Para realizar solicitudes firmadas a los puntos de conexión de Gremlin y SPARQL en Neptune

1. Cree un archivo con el nombre `neptunesigv4.py` y ábralo en un editor de texto.
2. Copie el siguiente código y péguelo en el archivo `neptunesigv4.py`.

```
# Amazon Neptune version 4 signing example (version v3)

# The following script requires python 3.6+
#    (sudo yum install python36 python36-virtualenv python36-pip)
# => the reason is that we're using urllib.parse() to manually encode URL
#    parameters: the problem here is that SIGV4 encoding requires whitespaces
```

```
#    to be encoded as %20 rather than not or using '+', as done by previous/
#    default versions of the library.


# See: https://docs.aws.amazon.com/general/latest/gr/sigv4_signing.html
import sys, datetime, hashlib, hmac
import requests  # pip3 install requests
import urllib
import os
import json
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace
from argparse import RawTextHelpFormatter
from argparse import ArgumentParser

# Configuration. https is required.
protocol = 'https'

# The following lines enable debugging at httplib level (requests->urllib3-
>http.client)
# You will see the REQUEST, including HEADERS and DATA, and RESPONSE with HEADERS
 but without DATA.
#
# The only thing missing will be the response.body which is not logged.
#
# import logging
# from http.client import HTTPConnection
# HTTPConnection.debuglevel = 1
# logging.basicConfig()
# logging.getLogger().setLevel(logging.DEBUG)
# requests_log = logging.getLogger("requests.packages.urllib3")
# requests_log.setLevel(logging.DEBUG)
# requests_log.propagate = True


# Read AWS access key from env. variables. Best practice is NOT
# to embed credentials in code.
access_key = os.getenv('AWS_ACCESS_KEY_ID', '')
secret_key = os.getenv('AWS_SECRET_ACCESS_KEY', '')
region = os.getenv('SERVICE_REGION', '')
```

```
# AWS_SESSION_TOKEN is optional environment variable. Specify a session token only
 if you are using temporary
# security credentials.
session_token = os.getenv('AWS_SESSION_TOKEN', '')

### Note same script can be used for AWS Lambda (runtime = python3.6).
## Steps to use this python script for AWS Lambda
# 1. AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY and AWS_SESSION_TOKEN and AWS_REGION
 variables are already part of Lambda's Execution environment
#    No need to set them up explicitly.
# 3. Create Lambda deployment package https://docs.aws.amazon.com/lambda/latest/dg/
lambda-python-how-to-create-deployment-package.html
# 4. Create a Lambda function in the same VPC and assign an IAM role with neptune
 access

def lambda_handler(event, context):
    # sample_test_input = {
    #     "host": "END_POINT:8182",
    #     "method": "GET",
    #     "query_type": "gremlin",
    #     "query": "g.V().count()"
    # }

    # Lambda uses AWS_REGION instead of SERVICE_REGION
    global region
    region = os.getenv('AWS_REGION', '')

    host = event['host']
    method = event['method']
    query_type = event['query_type']
    query =  event['query']

    return make_signed_request(host, method, query_type, query)

def validate_input(method, query_type):
    # Supporting GET and POST for now:
    if (method != 'GET' and method != 'POST'):
        print('First parameter must be "GET" or "POST", but is "' + method + '".')
        sys.exit()

    # SPARQL UPDATE requires POST
    if (method == 'GET' and query_type == 'sparqlupdate'):
        print('SPARQL UPDATE is not supported in GET mode. Please choose POST.')
        sys.exit()
```

```
def get_canonical_uri_and_payload(query_type, query, method):
    # Set the stack and payload depending on query_type.
    if (query_type == 'sparql'):
        canonical_uri = '/sparql/'
        payload = {'query': query}

    elif (query_type == 'sparqlupdate'):
        canonical_uri = '/sparql/'
        payload = {'update': query}

    elif (query_type == 'gremlin'):
        canonical_uri = '/gremlin/'
        payload = {'gremlin': query}
        if (method == 'POST'):
            payload = json.dumps(payload)

    elif (query_type == 'openCypher'):
        canonical_uri = '/openCypher/'
        payload = {'query': query}

    elif (query_type == "loader"):
        canonical_uri = "/loader/"
        payload = query

    elif (query_type == "status"):
        canonical_uri = "/status/"
        payload = {}

    elif (query_type == "gremlin/status"):
        canonical_uri = "/gremlin/status/"
        payload = {}

    elif (query_type == "openCypher/status"):
        canonical_uri = "/openCypher/status/"
        payload = {}

    elif (query_type == "sparql/status"):
        canonical_uri = "/sparql/status/"
        payload = {}

    else:
        print(
```

```
            'Third parameter should be from ["gremlin", "sparql", "sparqlupdate",
 "loader", "status] but is "' + query_type + '".')
        sys.exit()
    ## return output as tuple
    return canonical_uri, payload

def make_signed_request(host, method, query_type, query):
    service = 'neptune-db'
    endpoint = protocol + '://' + host

    print()
    print('+++++ USER INPUT +++++')
    print('host = ' + host)
    print('method = ' + method)
    print('query_type = ' + query_type)
    print('query = ' + query)

    # validate input
    validate_input(method, query_type)

    # get canonical_uri and payload
    canonical_uri, payload = get_canonical_uri_and_payload(query_type, query,
 method)

    # assign payload to data or params
    data = payload if method == 'POST' else None
    params = payload if method == 'GET' else None

    # create request URL
    request_url = endpoint + canonical_uri

    # create and sign request
    creds = SimpleNamespace(
        access_key=access_key, secret_key=secret_key, token=session_token,
 region=region,
    )

    request = AWSRequest(method=method, url=request_url, data=data, params=params)
    SigV4Auth(creds, service, region).add_auth(request)

    r = None

    # ************* SEND THE REQUEST *************
    if (method == 'GET'):
```

```
        print('++++ BEGIN GET REQUEST +++++')
        print('Request URL = ' + request_url)
        r = requests.get(request_url, headers=request.headers, verify=False,
 params=params)

    elif (method == 'POST'):

        print('\n+++++ BEGIN POST REQUEST +++++')
        print('Request URL = ' + request_url)
        if (query_type == "loader"):
            request.headers['Content-type'] = 'application/json'
        r = requests.post(request_url, headers=request.headers, verify=False,
 data=data)

    else:
        print('Request method is neither "GET" nor "POST", something is wrong
 here.')

    if r is not None:
        print()
        print('+++++ RESPONSE +++++')
        print('Response code: %d\n' % r.status_code)
        response = r.text
        r.close()
        print(response)

        return response

help_msg = '''
    export AWS_ACCESS_KEY_ID=[MY_ACCESS_KEY_ID]
    export AWS_SECRET_ACCESS_KEY=[MY_SECRET_ACCESS_KEY]
    export AWS_SESSION_TOKEN=[MY_AWS_SESSION_TOKEN]
    export SERVICE_REGION=[us-east-1|us-east-2|us-west-2|eu-west-1]

    python version >=3.6 is required.

    Examples: For help
    python3 program_name.py -h

    Examples: Queries
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/
status
```

```
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q
 sparqlupdate -d "INSERT DATA { <https://s> <https://p> <https://o> }"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
 "g.V().count()"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher
 -d "MATCH (n1) RETURN n1 LIMIT 1;"
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'
    python3 program_name.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'

    Environment variables must be defined as AWS_ACCESS_KEY_ID,
 AWS_SECRET_ACCESS_KEY and SERVICE_REGION.
    You should also set AWS_SESSION_TOKEN environment variable if you are using
 temporary credentials (ex. IAM Role or EC2 Instance profile).

    Current Limitations:
    - Query mode "sparqlupdate" requires POST (as per the SPARQL 1.1 protocol)
            '''

def exit_and_print_help():
    print(help_msg)
    exit()

def parse_input_and_query_neptune():


    parser = ArgumentParser(description=help_msg,
 formatter_class=RawTextHelpFormatter)
```

```
    group_host = parser.add_mutually_exclusive_group()
    group_host.add_argument("-ho", "--host", type=str)
    group_port = parser.add_mutually_exclusive_group()
    group_port.add_argument("-p", "--port", type=int, help="port ex. 8182,
default=8182", default=8182)
    group_action = parser.add_mutually_exclusive_group()
    group_action.add_argument("-a", "--action", type=str, help="http action,
default = GET", default="GET")
    group_endpoint = parser.add_mutually_exclusive_group()
    group_endpoint.add_argument("-q", "--query_type", type=str, help="query_type,
default = status ", default="status")
    group_data = parser.add_mutually_exclusive_group()
    group_data.add_argument("-d", "--data", type=str, help="data required for the
http action", default="")

    args = parser.parse_args()
    print(args)

    # Read command line parameters
    host = args.host
    port = args.port
    method = args.action
    query_type = args.query_type
    query = args.data

    if (access_key == ''):
        print('!!! ERROR: Your AWS_ACCESS_KEY_ID environment variable is
undefined.')
        exit_and_print_help()

    if (secret_key == ''):
        print('!!! ERROR: Your AWS_SECRET_ACCESS_KEY environment variable is
undefined.')
        exit_and_print_help()

    if (region == ''):
        print('!!! ERROR: Your SERVICE_REGION environment variable is undefined.')
        exit_and_print_help()

    if host is None:
        print('!!! ERROR: Neptune DNS is missing')
        exit_and_print_help()

    host = host + ":" + str(port)
```

```
    make_signed_request(host, method, query_type, query)


if __name__ == "__main__":
    parse_input_and_query_neptune()
```

3. En un terminal, vaya a la ubicación del archivo `neptunesigv4.py`.
4. Introduzca los siguientes comandos, reemplazando la clave de acceso, la clave secreta y la región por los valores correctos.

```
export AWS_ACCESS_KEY_ID=MY_ACCESS_KEY_ID
export AWS_SECRET_ACCESS_KEY=MY_SECRET_ACCESS_KEY
export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

Si está utilizando credenciales temporales, debe especificar `AWS_SESSION_TOKEN` además de `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `SERVICE_REGION`.

```
export AWS_SESSION_TOKEN=MY_AWS_SESSION_TOKEN
```

> **Note**
>
> Si utiliza las credenciales temporales, caducan después de un intervalo especificado, incluido el token de sesión.
> Tiene que actualizar el token de sesión cuando solicite nuevas credenciales. Para obtener más información, consulte Uso de credenciales de seguridad temporales para solicitar acceso a los recursos de AWS.

5. Introduzca uno de los siguientes comandos para enviar una solicitud firmada a la instancia de base de datos de Neptune. Estos ejemplos utilizan Python versión 3.6.

   Estado de punto de enlace

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q status
```

Gremlin

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin -d
 "g.V().count()"

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q gremlin -d
 "g.V().count()"
```

Estado de Gremlin

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q gremlin/
status
```

SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql -d
 "SELECT ?s WHERE { ?s ?p ?o }"
```

SPARQL UPDATE

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q sparqlupdate
 -d "INSERT DATA { <https://s> <https://p> <https://o> }"
```

Estado de SPARQL

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q sparql/status
```

openCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher -d
 "MATCH (n1) RETURN n1 LIMIT 1;"

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q openCypher -
d "MATCH (n1) RETURN n1 LIMIT 1;"
```

Estado de openCypher

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
```

Programa de carga

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{"loadId": "68b28dcc-8e15-02b1-133d-9bd0557607e6"}'

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q loader -d
 '{}'

python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a POST -q loader
 -d '{"source": "source", "format" : "csv", "failOnError": "fail_on_error",
 "iamRoleArn": "iam_role_arn", "region": "region"}'
```

6. La sintaxis para ejecutar el script de Python es la siguiente:

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p port -a GET|POST -q gremlin|
sparql|sparqlupdate|loader|status -d "string0data"
```

SPARQL UPDATE requiere POST.

Administración de acceso mediante políticas de IAM

Las políticas de IAM son objetos JSON que definen los permisos para usar acciones y recursos.

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidad

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Uso de políticas de control de servicios (SCP) con AWS las organizaciones

Las políticas de control de servicios (SCP) son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y administrar de forma centralizada varias AWS cuentas que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluido cada usuario raíz de la AWS cuenta. Para obtener más información sobre Organizations y los SCP, consulte Cómo funcionan los SCP en la Guía del AWS Organizations usuario.

Los clientes que implementen Amazon Neptune en una AWS cuenta de una AWS organización pueden aprovechar los SCP para controlar qué cuentas pueden usar Neptune. Para garantizar

el acceso a Neptune desde la cuenta de un miembro, asegúrese de permitir el acceso a las acciones de IAM del plano de control y del plano de datos mediante `neptune:*` y `neptune-db:*`, respectivamente.

Permisos necesarios para usar la consola de Amazon Neptune

Para que un usuario pueda trabajar con la consola de Amazon Neptune, debe tener un conjunto mínimo de permisos. Estos permisos dejan al usuario describir los recursos de Neptune de su cuenta de AWS y proporcionar otra información relacionada, incluida información de red y seguridad de Amazon EC2.

Si crea una política de IAM que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para los usuarios con esa política de IAM. Para asegurarse de que esos usuarios puedan seguir usando la consola de Neptune, asocie también la política administrada `NeptuneReadOnlyAccess` al usuario, según se explica en AWS políticas administradas (predefinidas) para Amazon Neptune.

No es necesario permitir permisos mínimos de consola a los usuarios que solo realizan llamadas a la API de Amazon Neptune AWS CLI o a la misma.

Asociación de una política de IAM a un usuario de IAM

Para aplicar una política administrada o personalizada, debe asociarla a un usuario de IAM. Para ver un tutorial acerca de este tema, consulte Crear y asociar su primera política administrada por el cliente en la Guía del usuario de IAM.

Mientras realiza el tutorial, puede usar uno de los ejemplos de política mostrados en esta sección como punto de partida y adaptarlo a sus necesidades. Al final del tutorial, tendrá un usuario de IAM con una política asociada que puede utilizar la acción `neptune-db:*`.

> **Important**
>
> - Los cambios realizados en una política de IAM pueden tardar hasta 10 minutos en aplicarse a los recursos de Neptune especificados.
> - Las políticas de IAM aplicadas a un clúster de base de datos de Neptune se aplican a todas las instancias incluidas en dicho clúster.

Uso de diferentes tipos de políticas de IAM para controlar el acceso a Neptune

Para proporcionar acceso a las acciones administrativas de Neptune o a los datos de un clúster de base de datos de Neptune, debe asociar políticas a un rol o usuario de IAM. Para obtener información sobre cómo asociar una política de IAM a un usuario, consulte Asociación de una política de IAM a un usuario de IAM. Para obtener información sobre cómo asociar una política a un rol, consulte Adición y eliminación de políticas de IAM en la Guía del usuario de IAM.

Para el acceso general a Neptune, puede utilizar una de las políticas administradas de Neptune. Para que el acceso sea más restringido, puede crear su propia política personalizada utilizando las acciones administrativas y recursos que admite Neptune.

En una política de IAM personalizada, puede utilizar dos tipos diferentes de declaraciones de políticas que controlan los distintos modos de acceso a un clúster de base de datos de Neptune:

- Declaraciones de políticas administrativas: las declaraciones de políticas administrativas proporcionan acceso a las API de administración de Neptune que se utilizan para crear, configurar y administrar un clúster de base de datos y sus instancias.

  Dado que Neptune comparte funcionalidades con Amazon RDS, las acciones administrativas, los recursos y las claves de condición de las políticas de Neptune utilizan un prefijo `rds:` por diseño.
- Declaraciones de políticas de acceso a los datos: las declaraciones de políticas de acceso a los datos utilizan acciones de acceso a los datos, recursos y claves de condición para controlar el acceso a los datos que contiene un clúster de base de datos.

  Las acciones de acceso a los datos, recursos y claves de condición de Neptune usan un prefijo. `neptune-db:`

Uso de claves de contexto de condición de IAM en Amazon Neptune

Puede especificar las condiciones en una declaración de la política de IAM que controle el acceso a Neptune. La declaración de la política solo se aplica si se cumplen las condiciones.

Por ejemplo, es posible que desee que una declaración de política entre en vigor solo después de una fecha específica o que permita el acceso solo cuando la solicitud contenga un valor específico.

Para expresar condiciones, se utilizan claves de condición predefinidas en el elemento `Condition` de una declaración de política, junto con operadores de política de condiciones de IAM, como igual o menor que.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una sola clave de condición, AWS evalúa la condición mediante una operación lógica`OR`. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para obtener más información, consulte Elementos de la política de IAM: Variables y etiquetas en la Guía del usuario de IAM.

El tipo de datos de una clave de condición determina qué operadores de condición puede utilizar para comparar los valores de la solicitud con los valores de la declaración de política. Si utiliza un operador de condición que no es compatible con ese tipo de datos, la coincidencia siempre falla y la declaración de política nunca se aplica.

Neptune admite diferentes conjuntos de claves de condición para las declaraciones de políticas administrativas que para las declaraciones de políticas de acceso a datos:

- Claves de condición para las declaraciones de políticas administrativas
- Claves de condición para las declaraciones de políticas de acceso a los datos

Compatibilidad con la política de IAM y las características de control de acceso en Amazon Neptune

La siguiente tabla muestra qué características de IAM admite Neptune para las declaraciones de políticas administrativas y las declaraciones de políticas de acceso a datos:

Características de IAM que puede utilizar con Neptune

Característica de IAM	Administración	Acceso a los datos
Políticas basadas en identidades	Sí	Sí

Característica de IAM	Administración	Acceso a los datos
Políticas basadas en recursos	No	No
Acciones de políticas	Sí	Sí
Recursos de políticas	Sí	Sí
Claves de condición global	Sí	(un subconjunto)
Claves de condición basadas en etiquetas	Sí	No
Listas de control de acceso (ACL)	No	No
Políticas de control de servicios (SCP)	Sí	Sí
Roles vinculados a servicios	Sí	No

Limitaciones de las políticas de IAM

Los cambios realizados en una política de IAM pueden tardar hasta 10 minutos en aplicarse a los recursos de Neptune especificados.

Las políticas de IAM aplicadas a un clúster de base de datos de Neptune se aplican a todas las instancias incluidas en dicho clúster.

Neptune no admite actualmente el control de acceso entre cuentas.

AWS políticas administradas (predefinidas) para Amazon Neptune

AWS aborda muchos casos de uso comunes al proporcionar políticas de IAM independientes que son creadas y administradas por. AWS Las políticas administradas conceden los permisos necesarios para casos de uso comunes, lo que le evita tener que investigar los permisos que se necesitan. Para más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

Las siguientes políticas AWS administradas, que puede adjuntar a los usuarios de su cuenta, son para usar las API de administración de Amazon Neptune:

- NeptuneReadOnlyAccess— Otorga acceso de solo lectura a todos los recursos de Neptune con fines administrativos y de acceso a los datos en la cuenta raíz. AWS
- NeptuneFullAcceso: otorga acceso total a todos los recursos de Neptune con fines administrativos y de acceso a los datos en la cuenta raíz. AWS Esto se recomienda si necesita acceso completo a Neptune desde el SDK AWS CLI o el SDK, pero no para AWS Management Console acceder.
- NeptuneConsoleFullAccess— Otorga acceso completo en la AWS cuenta raíz a todas las acciones y recursos administrativos de Neptune, pero no a ninguna acción o recurso de acceso a datos. También incluye permisos adicionales para simplificar el acceso a Neptune desde la consola, incluidos permisos limitados de IAM y Amazon EC2 (VPC).
- NeptuneGraphReadOnlyAccess — Proporciona acceso de solo lectura a todos los recursos de Amazon Neptune Analytics junto con permisos de solo lectura para los servicios dependientes
- AWSServiceRoleForNeptuneGraphPolicy— Permite que los gráficos de Neptune Analytics publiquen métricas y CloudWatch registros operativos y de uso.

Las políticas y roles de IAM de Neptune conceden cierto acceso a los recursos de Amazon RDS, ya que Neptune comparte tecnología operativa con Amazon RDS para determinadas características de administración. Esto incluye los permisos administrativos de la API, por lo que las acciones administrativas de Neptune tienen un prefijo `rds:`.

Actualizaciones de las políticas gestionadas por Neptune AWS

La siguiente tabla hace un seguimiento de las actualizaciones de las políticas administradas de Neptune a partir del momento en que Neptune comenzó a realizar el seguimiento de estos cambios:

Política	Descripción	Fecha
AWS políticas gestionadas para Amazon Neptune: actualización a las políticas existentes	Las políticas `NeptuneFullAccess` administradas `NeptuneReadOnlyAccess` y las administradas ahora incluyen `Sid` (ID de declaración) como identificador en la declaración de política.	22-01-2022

Política	Descripción	Fecha
NeptuneGraphReadOnlyAccess (publicado)	Se ha publicado para brindar acceso de solo lectura a gráficos y recursos de Neptune Analytics.	29-11-2020
AWSServiceRoleForNeptuneGraphPolicy(publicado)	Se lanzó para permitir el acceso a los gráficos de Neptune Analytics CloudWatch para publicar métricas y registros operativos y de uso. Consulte Uso de roles vinculados a servicios (SLR) en Neptune Analytics.	29-11-2020
NeptuneConsoleFullAccess(permisos añadidos)	Se han añadido permisos para brindar todo el acceso necesario para interactuar con los gráficos de Neptune Analytics.	29/11/2023
NeptuneFullAcceso (permisos añadidos)	Se han añadido permisos de acceso a los datos y permisos para las nuevas API de bases de datos globales.	28 de julio de 2022
NeptuneConsoleFullAccess(permisos añadidos)	Se han añadido permisos para las nuevas API de bases de datos globales.	2022-07-21
Se comenzó a realizar un seguimiento de los cambios en Neptune	Neptune comenzó a realizar un seguimiento de los cambios en sus políticas AWS gestionadas.	2022-07-21

Política AWS administrada por **NeptuneReadOnlyAccess**

La política NeptuneReadOnlyAccessgestionada que aparece a continuación otorga acceso de solo lectura a todas las acciones y recursos de Neptune con fines administrativos y de acceso a los datos.

> **Note**
>
> Esta política se actualizó el 21 de julio de 2022 para incluir permisos de acceso a los datos y permisos administrativos de solo lectura, e incluir permisos para acciones de bases de datos globales.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReadOnlyPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "AllowReadOnlyPermissionsForCloudwatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForEC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForKMS",
            "Effect": "Allow",
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnlyPermissionsForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
```

```
                "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
            ]
        },
        {
            "Sid": "AllowReadOnlyPermissionsForNeptuneDB",
            "Effect": "Allow",
            "Action": [
                "neptune-db:Read*",
                "neptune-db:Get*",
                "neptune-db:List*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Política AWS administrada por **`NeptuneFullAccess`**

La siguiente política de NeptuneFullacceso gestionado otorga acceso total a todas las acciones y recursos de Neptune con fines administrativos y de acceso a los datos. Se recomienda si necesita acceso total desde AWS CLI o desde un SDK, pero no desde. AWS Management Console

> **Note**
>
> Esta política se actualizó el 21 de julio de 2022 para incluir todos los permisos de acceso a los datos, así como los permisos administrativos completos, y para incluir permisos para acciones en bases de datos globales.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowNeptuneCreate",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBCluster",
                "rds:CreateDBInstance"
            ],
            "Resource": [
```

```
                "arn:aws:rds:*:*:*"
            ],
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": [
                        "graphdb",
                        "neptune"
                    ]
                }
            }
        },
        {
            "Sid": "AllowManagementPermissionsForRDS",
            "Effect": "Allow",
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBClusterEndpoint",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterEndpoint",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DeleteGlobalCluster",
                "rds:DescribeDBClusterEndpoints",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
```

```
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:FailoverGlobalCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterEndpoint",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime",
                "rds:StartDBCluster",
                "rds:StopDBCluster"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowOtherDepedentPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AllowPassRoleForNeptune",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:passedToService": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowCreateSLRForNeptune",
```

```
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowDataAccessForNeptune",
            "Effect": "Allow",
            "Action": [
                "neptune-db:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Política AWS administrada por **NeptuneConsoleFullAccess**

La siguiente política NeptuneConsoleFullAccessgestionada otorga acceso total a todas las acciones y recursos de Neptune con fines administrativos, pero no con fines de acceso a los datos. También incluye permisos adicionales para simplificar el acceso a Neptune desde la consola, incluidos permisos limitados de IAM y Amazon EC2 (VPC).

> **Note**
>
> Esta política se actualizó el 29 de noviembre de 2023 para incluir los permisos necesarios para interactuar con los gráficos de Neptune Analytics.
> Se actualizó el 21 de julio de 2022 para incluir permisos para acciones en bases de datos globales.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "AllowNeptuneCreate",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBCluster",
        "rds:CreateDBInstance"
      ],
      "Resource": [
        "arn:aws:rds:*:*:*"
      ],
      "Condition": {
        "StringEquals": {
          "rds:DatabaseEngine": [
            "graphdb",
            "neptune"
          ]
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForRDS",
      "Action": [
        "rds:AddRoleToDBCluster",
        "rds:AddSourceIdentifierToSubscription",
        "rds:AddTagsToResource",
        "rds:ApplyPendingMaintenanceAction",
        "rds:CopyDBClusterParameterGroup",
        "rds:CopyDBClusterSnapshot",
        "rds:CopyDBParameterGroup",
        "rds:CreateDBClusterParameterGroup",
        "rds:CreateDBClusterSnapshot",
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSubnetGroup",
        "rds:CreateEventSubscription",
        "rds:DeleteDBCluster",
        "rds:DeleteDBClusterParameterGroup",
        "rds:DeleteDBClusterSnapshot",
        "rds:DeleteDBInstance",
        "rds:DeleteDBParameterGroup",
        "rds:DeleteDBSubnetGroup",
        "rds:DeleteEventSubscription",
        "rds:DescribeAccountAttributes",
        "rds:DescribeCertificates",
        "rds:DescribeDBClusterParameterGroups",
```

```
        "rds:DescribeDBClusterParameters",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBClusters",
        "rds:DescribeDBEngineVersions",
        "rds:DescribeDBInstances",
        "rds:DescribeDBLogFiles",
        "rds:DescribeDBParameterGroups",
        "rds:DescribeDBParameters",
        "rds:DescribeDBSecurityGroups",
        "rds:DescribeDBSubnetGroups",
        "rds:DescribeEngineDefaultClusterParameters",
        "rds:DescribeEngineDefaultParameters",
        "rds:DescribeEventCategories",
        "rds:DescribeEventSubscriptions",
        "rds:DescribeEvents",
        "rds:DescribeOptionGroups",
        "rds:DescribeOrderableDBInstanceOptions",
        "rds:DescribePendingMaintenanceActions",
        "rds:DescribeValidDBInstanceModifications",
        "rds:DownloadDBLogFilePortion",
        "rds:FailoverDBCluster",
        "rds:ListTagsForResource",
        "rds:ModifyDBCluster",
        "rds:ModifyDBClusterParameterGroup",
        "rds:ModifyDBClusterSnapshotAttribute",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyDBSubnetGroup",
        "rds:ModifyEventSubscription",
        "rds:PromoteReadReplicaDBCluster",
        "rds:RebootDBInstance",
        "rds:RemoveRoleFromDBCluster",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds:RemoveTagsFromResource",
        "rds:ResetDBClusterParameterGroup",
        "rds:ResetDBParameterGroup",
        "rds:RestoreDBClusterFromSnapshot",
        "rds:RestoreDBClusterToPointInTime"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
```

```
    },
    {
      "Sid": "AllowOtherDepedentPermissions",
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "ec2:AllocateAddress",
        "ec2:AssignIpv6Addresses",
        "ec2:AssignPrivateIpAddresses",
        "ec2:AssociateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AssociateSubnetCidrBlock",
        "ec2:AssociateVpcCidrBlock",
        "ec2:AttachInternetGateway",
        "ec2:AttachNetworkInterface",
        "ec2:CreateCustomerGateway",
        "ec2:CreateDefaultSubnet",
        "ec2:CreateDefaultVpc",
        "ec2:CreateInternetGateway",
        "ec2:CreateNatGateway",
        "ec2:CreateNetworkInterface",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateVpc",
        "ec2:CreateVpcEndpoint",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeCustomerGateways",
        "ec2:DescribeInstances",
        "ec2:DescribeNatGateways",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribePrefixLists",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeSubnets",
```

```
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:ModifySubnetAttribute",
        "ec2:ModifyVpcAttribute",
        "ec2:ModifyVpcEndpoint",
        "iam:ListRoles",
        "kms:ListAliases",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "kms:ListRetirableGrants",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "sns:ListSubscriptions",
        "sns:ListTopics",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptune",
      "Action": "iam:PassRole",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowCreateSLRForNeptune",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
```

```
          "iam:AWSServiceName": "rds.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowManagementPermissionsForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": [
        "neptune-graph:CreateGraph",
        "neptune-graph:DeleteGraph",
        "neptune-graph:GetGraph",
        "neptune-graph:ListGraphs",
        "neptune-graph:UpdateGraph",
        "neptune-graph:ResetGraph",
        "neptune-graph:CreateGraphSnapshot",
        "neptune-graph:DeleteGraphSnapshot",
        "neptune-graph:GetGraphSnapshot",
        "neptune-graph:ListGraphSnapshots",
        "neptune-graph:RestoreGraphFromSnapshot",
        "neptune-graph:CreatePrivateGraphEndpoint",
        "neptune-graph:GetPrivateGraphEndpoint",
        "neptune-graph:ListPrivateGraphEndpoints",
        "neptune-graph:DeletePrivateGraphEndpoint",
        "neptune-graph:CreateGraphUsingImportTask",
        "neptune-graph:GetImportTask",
        "neptune-graph:ListImportTasks",
        "neptune-graph:CancelImportTask"
      ],
      "Resource": [
        "arn:aws:neptune-graph:*:*:*"
      ]
    },
    {
      "Sid": "AllowPassRoleForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:passedToService": "neptune-graph.amazonaws.com"
        }
      }
    },
    {
```

```
      "Sid": "AllowCreateSLRForNeptuneAnalytics",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/neptune-graph.amazonaws.com/
AWSServiceRoleForNeptuneGraph",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "neptune-graph.amazonaws.com"
        }
      }
    }
  ]
}
```

Política AWS administrada por **`NeptuneGraphReadOnlyAccess`**

La siguiente política de NeptuneGraphReadOnlyacceso gestionado proporciona acceso de solo lectura a todos los recursos de Amazon Neptune Analytics junto con permisos de solo lectura para los servicios dependientes.

Esta política incluye permisos para hacer lo siguiente:

- Para Amazon EC2: recupere información sobre las VPC, las subredes, los grupos de seguridad y las zonas de disponibilidad.
- Para AWS KMS: recuperar información sobre las claves y los alias de KMS.
- Para CloudWatch: recuperar información sobre CloudWatch las métricas.
- Para CloudWatch registros: recupera información sobre eventos y secuencias de CloudWatch registros.

> **Note**
>
> Esta política se publicó el 29 de noviembre de 2023.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowReadOnlyPermissionsForNeptuneGraph",
```

```
      "Effect": "Allow",
      "Action": [
        "neptune-graph:Get*",
        "neptune-graph:List*",
        "neptune-graph:Read*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForEC2",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeAvailabilityZones"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForKMS",
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:ListAliases"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForCloudwatch",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowReadOnlyPermissionsForLogs",
      "Effect": "Allow",
      "Action": [
```

```
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ]
    }
  ]
}
```

Política AWS administrada por **AWSServiceRoleForNeptuneGraphPolicy**

La siguiente política AWSServiceRoleForNeptuneGraphPolicygestionada permite acceder a los gráficos CloudWatch para publicar registros y métricas operativas y de uso. Consulte nan-service-linked-roles.

> **Note**
>
> Esta política se publicó el 29 de noviembre de 2023.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GraphMetrics",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/Neptune",
            "AWS/Usage"
          ]
        }
      }
    },
    {
      "Sid": "GraphLogGroup",
```

```
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    },
    {
      "Sid": "GraphLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/neptune/*:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    }
  ]
}
```

Claves de contexto de condición de IAM compatibles con Amazon Neptune

Puede especificar condiciones en las políticas de IAM que controlan el acceso a las acciones de administración y recursos de Neptune. La declaración de la política solo se aplica si se cumplen las condiciones.

Por ejemplo, es posible que desee que una declaración de política entre en vigor solo después de una fecha específica o que permita el acceso solo cuando la solicitud a la API contenga un valor específico.

Para expresar condiciones, se utilizan claves de condición predefinidas en el elemento `Condition` de una declaración de política, junto con operadores de política de condiciones de IAM, como igual o menor que.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una sola clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para obtener más información, consulte Elementos de la política de IAM: Variables y etiquetas en la Guía del usuario de IAM.

El tipo de datos de una clave de condición determina qué operadores de condición puede utilizar para comparar los valores de la solicitud con los valores de la declaración de política. Si utiliza un operador de condición que no es compatible con ese tipo de datos, la coincidencia siempre falla y la declaración de política nunca se aplica.

Claves de condición de IAM para las declaraciones de políticas administrativas de Neptune

- Claves de condición globales: puede utilizar la mayoría de las claves de condición AWS globales en las declaraciones de política administrativa de Neptune.
- Claves de condición específicas del servicio: son claves que se definen para servicios específicos. AWS Las que Neptune admite para las declaraciones de políticas administrativas se enumeran en Las claves de condición están disponibles en las declaraciones de políticas administrativas de IAM de Neptune.

Claves de condición de IAM para las declaraciones de políticas de acceso a datos de Neptune

- Claves de condición globales: el subconjunto de estas claves que admite Neptune en las declaraciones de políticas de acceso a los datos aparece en AWS claves de contexto de condiciones globales respaldadas por Neptune en las declaraciones de política de acceso a datos.
- Las claves de condición específicas del servicio que Neptune define para las declaraciones de políticas de acceso a los datos se enumeran en Claves de condición.

Declaraciones de políticas administrativas de IAM personalizadas de Amazon Neptune

Las declaraciones de políticas administrativas le permiten controlar lo que un usuario de IAM puede hacer para administrar una base de datos de Neptune.

Una declaración de política administrativa de Neptune otorga acceso a una o más acciones administrativas y recursos administrativos que admite Neptune. También puede utilizar Claves de condición para hacer que los permisos administrativos sean más específicos.

> **Note**
>
> Dado que Neptune comparte funcionalidades con Amazon RDS, las acciones administrativas, los recursos y las claves de condición específicas del servicio en las declaraciones de políticas administrativa sutilizan el `rds:` prefijo por diseño.

Temas

- Acciones disponibles en las declaraciones de políticas administrativas de IAM de Neptune
- Tipos de recursos disponibles en las declaraciones de políticas administrativas de IAM de Neptune
- Las claves de condición están disponibles en las declaraciones de políticas administrativas de IAM de Neptune
- Ejemplos de declaraciones de políticas administrativas de IAM para Neptune

Acciones disponibles en las declaraciones de políticas administrativas de IAM de Neptune

Puede utilizar las acciones administrativas que se enumeran a continuación en el elemento `Action` de una declaración de política de IAM para controlar el acceso a las API de administración de Neptune. Cuando utiliza una acción en una política, normalmente permite o deniega el acceso a la operación de la API o comandos de la CLI con el mismo nombre. No obstante, en algunos casos, una sola acción controla el acceso a más de una operación. Asimismo, algunas operaciones requieren varias acciones diferentes.

El campo `Resource type` de la siguiente lista indica si cada acción admite permisos de nivel de recurso. Si no hay ningún valor en este campo, debe especificar todos los recursos ("*") en el elemento `Resource` de la declaración de la política. Si la columna incluye un tipo de recurso,

puede especificar un ARN de recurso en una instrucción con dicha acción. Los tipos de recursos administrativos de Neptune se muestran en esta página.

Los recursos necesarios se indican en la lista siguiente con un asterisco (*). Si especifica un ARN de permiso de recursos en una instrucción mediante esta acción, deberá ser de este tipo. Algunas acciones admiten varios tipos de recursos. Si un tipo de recurso es opcional (es decir, no está marcado con un asterisco), no es necesario que lo incluya.

Para obtener más información sobre los campos que se muestran aquí, consulte la tabla de acciones de la Guía del usuario de IAM.

rds: ToDbCluster AddRole

`AddRoleToDBCluster` asocia un rol de IAM a un clúster de base de datos de Neptune.

Nivel de acceso: `Write`.

Acciones dependientes: `iam:PassRole`.

Tipo de recurso: clúster (obligatorio).

rds: Suscripción AddSource IdentifierTo

`AddSourceIdentifierToSubscription` añade un identificador de origen a una suscripción de notificación de eventos existente de Neptune.

Nivel de acceso: `Write`.

Tipo de recurso: es (obligatorio)

rds: AddTags ToResource

`AddTagsToResource` asocia un rol de IAM a un clúster de base de datos de Neptune.

Nivel de acceso: `Write`.

Tipos de recursos:

- db
- es
- pg
- cluster-snapshot
- subgrp

Claves de condición:

- *aws:RequestTag//tag-key*
- leyes: TagKeys

Ards: ApplyPending MaintenanceAction

ApplyPendingMaintenanceAction aplica una acción de mantenimiento pendiente a un recurso.

Nivel de acceso: `Write`.

Tipo de recurso: db (obligatorio).

ClusterParameterRDS: Grupo CopyDB

CopyDBClusterParameterGroup copia el grupo de parámetros de clúster de base de datos especificado.

Nivel de acceso: `Write`.

Tipo de recurso: cluster-pg (obligatorio).

RDS: CopyDB ClusterSnapshot

CopyDBClusterSnapshot copia una instantánea de un clúster de base de datos.

Nivel de acceso: `Write`.

Tipo de recurso: cluster-snapshot (obligatorio).

RDS: CopyDB ParameterGroup

CopyDBParameterGroup copia el grupo de parámetros de base de datos especificado.

Nivel de acceso: `Write`.

Tipo de recurso: pg (obligatorio).

rds:CreateDBCluster

CreateDBCluster crea un nuevo clúster de base de datos de Neptune.

Nivel de acceso: `Tagging`.

Acciones dependientes: `iam:PassRole`.

Tipos de recursos:

- cluster (obligatorio).
- cluster-pg (obligatorio).
- subgrp (obligatorio).

Claves de condición:

- *aws:/tag-key RequestTag*
- leyes: TagKeys
- neptune-rds_ DatabaseEngine

RDS: creó un grupo B ClusterParameter

`CreateDBClusterParameterGroup` crea un nuevo grupo de parámetros del clúster de base de datos.

Nivel de acceso: `Tagging`.

Tipo de recurso: cluster-pg (obligatorio).

Claves de condición:

- *aws:/tag-key RequestTag*
- leyes: TagKeys

RDS: creado B ClusterSnapshot

`CreateDBClusterSnapshot` crea una instantánea de un clúster de base de datos.

Nivel de acceso: `Tagging`.

Tipos de recursos:

- cluster (obligatorio).
- cluster-snapshot (obligatorio).

Claves de condición:

- *aws:/tag-key RequestTag*
- leyes: TagKeys

rds:CreateDBInstance

CreateDBInstance crea una nueva instancia de base de datos.

Nivel de acceso: Tagging.

Acciones dependientes: iam:PassRole.

Tipos de recursos:

- db (obligatorio).
- pg (obligatorio).
- subgrp (obligatorio).

Claves de condición:

- aws:RequestTag/*tag-key*
- leyes: TagKeys

RDS: creado B ParameterGroup

CreateDBParameterGroup crea un nuevo grupo de parámetros de base de datos.

Nivel de acceso: Tagging.

Tipo de recurso: pg (obligatorio).

Claves de condición:

- *aws:/tag-key RequestTag*
- leyes: TagKeys

RDS: creado B SubnetGroup

CreateDBSubnetGroup crea un nuevo grupo de subred de base de datos.

Nivel de acceso: `Tagging`.

Tipo de recurso: subgrp (obligatorio).

Claves de condición:

- *aws:/tag-key RequestTag*
- leyes: TagKeys

rds: Suscripción CreateEvent

`CreateEventSubscription` crea una suscripción de notificación de eventos de Neptune.

Nivel de acceso: `Tagging`.

Tipo de recurso: es (obligatorio)

Claves de condición:

- *aws:RequestTag/tag-key*
- leyes: TagKeys

rds:DeleteDBCluster

`DeleteDBCluster` elimina un clúster de base de datos de Neptune existente.

Nivel de acceso: `Write`.

Tipos de recursos:

- cluster (obligatorio).
- cluster-snapshot (obligatorio).

RDSClusterParameter: grupo B eliminado

`DeleteDBClusterParameterGroup` elimina un determinado grupo de parámetros del clúster de base de datos.

Nivel de acceso: `Write`.

Tipo de recurso: cluster-pg (obligatorio).

RDS: borrado B ClusterSnapshot

DeleteDBClusterSnapshot elimina una instantánea de clúster de base de datos.

Nivel de acceso: Write.

Tipo de recurso: cluster-snapshot (obligatorio).

rds:DeleteDBInstance

DeleteDBInstance elimina una instancia de base de datos especificada.

Nivel de acceso: Write.

Tipo de recurso: db (obligatorio).

RDS: borrado B ParameterGroup

DeleteDBParameterGroupelimina una base de datos especificada. ParameterGroup

Nivel de acceso: Write.

Tipo de recurso: pg (obligatorio).

RDS: borrado B SubnetGroup

DeleteDBSubnetGroup elimina un grupo de subred de base de datos.

Nivel de acceso: Write.

Tipo de recurso: subgrp (obligatorio).

rds: Suscripción DeleteEvent

DeleteEventSubscription elimina una suscripción a notificaciones de eventos.

Nivel de acceso: Write.

Tipo de recurso: es (obligatorio)

ClusterParameterRDS: grupos B descritos

DescribeDBClusterParameterGroupsdevuelve una lista de descripciones de bases de datos. ClusterParameterGroup

Nivel de acceso: List.

Tipo de recurso: cluster-pg (obligatorio).

RDS: descrito como B ClusterParameters

`DescribeDBClusterParameters` devuelve la lista detallada de parámetros para un grupo de parámetros de clúster de base de datos en particular.

Nivel de acceso: `List`.

Tipo de recurso: cluster-pg (obligatorio).

RDS: atributos descritos ClusterSnapshot de B

`DescribeDBClusterSnapshotAttributes` devuelve una lista de nombres y valores de atributos de instantáneas de clúster de base de datos y valores para una instantánea manual del clúster de base de datos.

Nivel de acceso: `List`.

Tipo de recurso: cluster-snapshot (obligatorio).

RDS: Descrito B ClusterSnapshots

`DescribeDBClusterSnapshots` devuelve información acerca de instantáneas del clúster de base de datos.

Nivel de acceso: `Read`.

rds:DescribeDBClusters

`DescribeDBClusters` devuelve información acerca de un clúster de base de datos aprovisionado de Neptune.

Nivel de acceso: `List`.

Tipo de recurso: clúster (obligatorio).

RDS: Descrito B EngineVersions

`DescribeDBEngineVersions` devuelve una lista con los motores de base de datos disponibles.

Nivel de acceso: `List`.

Tipo de recurso: pg (obligatorio).

rds:DescribeDBInstances

`DescribeDBInstances` devuelve información acerca de las instancias de base de datos.

Nivel de acceso: `List`.

Tipo de recurso: es (obligatorio)

RDS: Descrito B ParameterGroups

`DescribeDBParameterGroups`devuelve una lista de descripciones de bases de datos. ParameterGroup

Nivel de acceso: `List`.

Tipo de recurso: pg (obligatorio).

rds:DescribeDBParameters

`DescribeDBParameters` devuelve la lista detallada de parámetros para un grupo de parámetros de base de datos determinado.

Nivel de acceso: `List`.

Tipo de recurso: pg (obligatorio).

RDS: descrito como B SubnetGroups

`DescribeDBSubnetGroups`devuelve una lista de descripciones de bases de datos. SubnetGroup

Nivel de acceso: `List`.

Tipo de recurso: subgrp (obligatorio).

rds: Categorías DescribeEvent

`DescribeEventCategories` devuelve una lista de categorías de todos los tipos de origen de eventos o, si se especifica, para un tipo de origen especificado.

Nivel de acceso: `List`.

rds: Suscripciones DescribeEvent

`DescribeEventSubscriptions` muestra todas las descripciones de la suscripción para una cuenta de cliente.

Nivel de acceso: `List`.

Tipo de recurso: es (obligatorio)

rds: DescribeEvents

`DescribeEvents` devuelve eventos relacionados con las instancias de base de datos, grupos de seguridad de base de datos y grupos de parámetros de base de datos de los últimos 14 días.

Nivel de acceso: `List`.

Tipo de recurso: es (obligatorio)

Varillas: DB DescribeOrderable InstanceOptions

`DescribeOrderableDBInstanceOptions` devuelve una lista de opciones de instancia de base de datos ordenable para el motor especificado.

Nivel de acceso: `List`.

rds: DescribePending MaintenanceActions

`DescribePendingMaintenanceActions` devuelve una lista de recursos (por ejemplo, instancias de base de datos) que tienen al menos una acción de mantenimiento pendiente.

Nivel de acceso: `List`.

Tipo de recurso: db (obligatorio).

Varillas: DB DescribeValid InstanceModifications

`DescribeValidDBInstanceModifications` muestra las modificaciones disponibles que puede realizar en su instancia de base de datos.

Nivel de acceso: `List`.

Tipo de recurso: db (obligatorio).

rds:FailoverDBCluster

`FailoverDBCluster` fuerza una conmutación por error para un clúster de base de datos.

Nivel de acceso: `Write`.

Tipo de recurso: clúster (obligatorio).

rds: ListTags ForResource

`ListTagsForResource` muestra todas las etiquetas en un recurso de Neptune.

Nivel de acceso: `Read`.

Tipos de recursos:

- cluster-snapshot
- db
- es
- pg
- subgrp

rds:ModifyDBCluster

`ModifyDBCluster`

Modifica la configuración de un clúster de base de datos de Neptune.

Nivel de acceso: `Write`.

Acciones dependientes: `iam:PassRole`.

Tipos de recursos:

- cluster (obligatorio).
- cluster-pg (obligatorio).

ClusterParameterRDS: modificar el grupo de bases de datos

`ModifyDBClusterParameterGroup` modifica los parámetros de un grupo de parámetros del clúster de base de datos.

Nivel de acceso: `Write`.

Tipo de recurso: cluster-pg (obligatorio).

Atributo RDS:ModifyDB ClusterSnapshot

`ModifyDBClusterSnapshotAttribute` añade un atributo y valores a una instantánea manual del clúster de base de datos o los elimina.

Nivel de acceso: `Write`.

Tipo de recurso: cluster-snapshot (obligatorio).

rds:ModifyDBInstance

`ModifyDBInstance` modifica la configuración de una instancia de base de datos.

Nivel de acceso: `Write`.

Acciones dependientes: `iam:PassRole`.

Tipos de recursos:

- db (obligatorio).
- pg (obligatorio).

RDS: Modificar base de datos ParameterGroup

`ModifyDBParameterGroup` modifica los parámetros de un grupo de parámetros de base de datos.

Nivel de acceso: `Write`.

Tipo de recurso: pg (obligatorio).

RDS: modificar base de datos SubnetGroup

`ModifyDBSubnetGroup` modifica un grupo de subred de base de datos existente.

Nivel de acceso: `Write`.

Tipo de recurso: subgrp (obligatorio).

rds: Suscripción ModifyEvent

`ModifyEventSubscription` modifica una suscripción a notificaciones de eventos de Neptune existente.

Nivel de acceso: `Write`.

Tipo de recurso: es (obligatorio)

rds:RebootDBInstance

`RebootDBInstance` reinicia el servicio del motor de base de datos para la instancia.

Nivel de acceso: `Write`.

Tipo de recurso: db (obligatorio).

rds: desde DBCluster RemoveRole

`RemoveRoleFromDBCluster`disocia una función de AWS Identity and Access Management (IAM) de un clúster de base de datos de Amazon Neptune.

Nivel de acceso: `Write`.

Acciones dependientes: `iam:PassRole`.

Tipo de recurso: clúster (obligatorio).

rds: Suscripción RemoveSource IdentifierFrom

`RemoveSourceIdentifierFromSubscription` elimina un identificador de origen de una suscripción a notificaciones de eventos de Neptune existente.

Nivel de acceso: `Write`.

Tipo de recurso: es (obligatorio)

rds: RemoveTags FromResource

`RemoveTagsFromResource` elimina etiquetas de metadatos de un recurso de Neptune.

Nivel de acceso: `Tagging`.

Tipos de recursos:

- cluster-snapshot
- db
- es
- pg
- subgrp

Claves de condición:

- *aws:RequestTag//tag-key*
- leyes: TagKeys

RDSClusterParameter: ResetDB Group

ResetDBClusterParameterGroup modifica los parámetros de un grupo de parámetros del clúster de base de datos al valor predeterminado.

Nivel de acceso: `Write`.

Tipo de recurso: cluster-pg (obligatorio).

RDS: ResetDB ParameterGroup

ResetDBParameterGroup modifica los parámetros de un grupo de parámetros de base de datos al valor predeterminado del motor/sistema.

Nivel de acceso: `Write`.

Tipo de recurso: pg (obligatorio).

RDS: Instantánea de base de datos restaurada ClusterFrom

RestoreDBClusterFromSnapshot crea un nuevo clúster de base de datos desde una instantánea de clúster de base de datos.

Nivel de acceso: `Write`.

Acciones dependientes: `iam:PassRole`.

Tipos de recursos:

- cluster (obligatorio).
- cluster-snapshot (obligatorio).

Claves de condición:

- *aws:/tag-key RequestTag*
- leyes: TagKeys

RDS ClusterToPointIn: hora de base de datos restaurada

`RestoreDBClusterToPointInTime` restaura un clúster de base de datos a un punto arbitrario en el tiempo.

Nivel de acceso: `Write`.

Acciones dependientes: `iam:PassRole`.

Tipos de recursos:

- cluster (obligatorio).
- subgrp (obligatorio).

Claves de condición:

- *aws:/tag-key RequestTag*
- leyes: TagKeys

rds:StartDBCluster

`StartDBCluster` inicia el clúster de base de datos especificado.

Nivel de acceso: `Write`.

Tipo de recurso: clúster (obligatorio).

rds:StopDBCluster

`StopDBCluster` detiene el clúster de base de datos especificado.

Nivel de acceso: `Write`.

Tipo de recurso: clúster (obligatorio).

Tipos de recursos disponibles en las declaraciones de políticas administrativas de IAM de Neptune

Neptune admite los tipos de recursos de la siguiente tabla para su uso en el elemento `Resource` de las declaraciones de políticas de administración de IAM. Para obtener más información sobre el elemento `Resource`, consulte Elementos de la política de JSON de IAM: Resource.

La lista de acciones de administración de Neptune identifica los tipos de recursos que se pueden especificar con cada acción. Un tipo de recurso también determina qué claves de condición se pueden incluir en una política, tal y como se especifica en la última columna de la tabla de abajo.

En la columna ARN, se especifica el formato de nombre de recurso de Amazon (ARN) que debe utilizar para hacer referencia a los recursos de este tipo. Las partes precedidas por $ deben sustituirse por los valores reales de su escenario. Por ejemplo, si ve $user-name en un ARN, debe sustituir esa cadena por el nombre de usuario real de IAM o una variable de política que contenga un nombre de usuario de IAM. Para obtener más información sobre los ARN, consulte ARN de IAM y Uso de ARN administrativos en Amazon Neptune.

En la columna Condition Keys , se especifican las claves de contexto de condiciones que puede incluir en una declaración de política de IAM cuando se incluyen en la declaración este recurso y una acción de apoyo compatible.

Tipos de recursos	ARN	Claves de condición
cluster (un clúster de base de datos)	arn:*partition* :rds:*region*:*account-id* :cluster: *instance-name*	aws:ResourceTag/*tag-key* rds:cluster-tag/*tag-key*
cluster-pg (un grupo de parámetros de clúster de base de datos)	arn:*partition* :rds:*region*:*account-id* :cluster-pg: *neptune-DBClusterParameterGroupName*	*aws:/tag-key ResourceTag*
cluster-snapshot (una instantánea de clúster	arn:*partition* :rds:*region*:*account-id* :cluster-snapshot: *neptune-DBClusterSnapshotName*	*aws:/tag-key ResourceTag* rds:cluster-snapshot-tag/*tag-key*

Tipos de recursos	ARN	Claves de condición
de base de datos)		
db (una instancia de base de datos)	arn:*partition* :rds:*region*:*account-id* :db:*neptune-DbInstanceName*	*aws:/tag-key ResourceTag* Ards: DatabaseClass Varillas: DatabaseEngine rds:db-tag/*tag-key*
es (una suscripción a eventos)	arn:*partition* :rds:*region*:*account-id* :es:*neptune-CustSubscriptionId*	*aws:ResourceTag//tag-key* rds:es-tag/*tag-key*
pg (un grupo de parámetros de base de datos)	arn:*partition* :rds:*region*:*account-id* :pg:*neptune-ParameterGroupName*	*aws:/tag-key ResourceTag* rds:pg-tag/*tag-key*
subgrp (un grupo de subred de base de datos)	arn:*partition* :rds:*region*:*account-id* :subgrp:*neptune-DBSubnetGroupName* }	*aws:/tag-key ResourceTag* rds:subgrp-tag/*tag-key*

Las claves de condición están disponibles en las declaraciones de políticas administrativas de IAM de Neptune

Con las claves de condición, puede especificar condiciones en una declaración de política de IAM para que la declaración se aplique solo cuando se aplican las condiciones. Las claves de condición que puede utilizar en las declaraciones de políticas administrativas de Neptune se clasifican en las siguientes categorías:

- Claves de condición globales: se definen AWS para su uso general con AWS los servicios. La mayoría se puede utilizar en las declaraciones de políticas administrativas de Neptune.
- Claves de condición de las propiedades de recursos administrativos: estas claves, que se enumeran a continuación, se basan en las propiedades de los recursos administrativos.
- Claves de condición de acceso basadas en etiquetas: estas claves, que se enumeran a continuación, se basan en las etiquetas de AWS asociadas a los recursos administrativos.

Claves de condición de las propiedades de recursos administrativos de Neptune

Claves de condición	Descripción	Tipo
`rds:DatabaseClass`	Filtra el acceso por el tipo de clase de instancia de base de datos	Cadena
`rds:DatabaseEngine`	Filtra el acceso por el motor de la base de datos. Para obtener valores posibles, consulte el parámetro del motor en la API CreateDBInstance	Cadena
`rds:DatabaseName`	Filtra el acceso por el nombre definido por el usuario de la base de datos en la instancia de base de datos	Cadena
`rds:EndpointType`	Filtra el acceso según el tipo de punto de enlace Puede ser: READER, WRITER, CUSTOM (lectura, escritura, personalizado)	Cadena
`rds:Vpc`	Filtra el acceso por un valor que especifica si la instancia de base de datos se ejecuta en una Amazon Virtual Private Cloud (Amazon VPC) Para indicar que la	Booleano

Claves de condición	Descripción	Tipo
	instancia de base de datos se ejecuta en una Amazon VPC; especifique `true`.	

Claves de condición administrativas basadas en etiquetas

Amazon Neptune permite especificar condiciones en una política de IAM que utiliza etiquetas personalizadas para controlar el acceso a Neptune a través de Referencia de la API de administración.

Por ejemplo, si añade una etiqueta denominada `environment` a sus instancias de base de datos, con valores como `beta`, `staging` y `production`, puede crear una política que restrinja el acceso a las instancias en función del valor de esa etiqueta.

Important

Si administra el acceso a sus recursos de Neptune mediante el etiquetado, asegúrese de proteger el acceso a las etiquetas. Puede restringir el acceso a las etiquetas creando políticas para las acciones `AddTagsToResource` y `RemoveTagsFromResource`. Por ejemplo, podría utilizar la siguiente política para denegar a los usuarios la posibilidad de añadir o eliminar etiquetas para todos los recursos. A continuación, podría crear políticas para permitir que usuarios específicos añadan o quiten etiquetas.

```
{ "Version": "2012-10-17",
  "Statement":[
    { "Sid": "DenyTagUpdates",
      "Effect": "Deny",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource":"*"
    }
  ]
}
```

Las siguientes claves de condición basadas en etiquetas solo funcionan con los recursos administrativos en declaraciones de políticas administrativas.

Claves de condición administrativas basadas en etiquetas

Claves de condición	Descripción	Tipo
aws:RequestTag/${TagKey}	Filtra acciones en función de la presencia de pares de clave-valor de etiqueta en la solicitud.	Cadena
aws:ResourceTag/${TagKey}	Filtra el acceso en función de pares de clave-valor asociados al recurso.	Cadena
aws:TagKeys	Filtra el acceso en función de la presencia de claves de etiqueta en la solicitud.	Cadena
rds:cluster-pg-tag/${TagKey}	Filtra el acceso por la etiqueta asociada a un grupo de parámetros de clúster de base de datos.	Cadena
rds:cluster-snapshot-tag/${TagKey}	Filtra el acceso por la etiqueta asociada a una instantánea de clúster de base de datos.	Cadena
rds:cluster-tag/${TagKey}	Filtra el acceso por la etiqueta asociada a un clúster de base de datos.	Cadena
rds:db-tag/${TagKey}	Filtra el acceso por la etiqueta asociada a una instancia de base de datos.	Cadena
rds:es-tag/${TagKey}	Filtra el acceso por la etiqueta asociada a una suscripción de evento.	Cadena
rds:pg-tag/${TagKey}	Filtra el acceso por la etiqueta asociada a un grupo de parámetros de base de datos.	Cadena

Claves de condición	Descripción	Tipo
`rds:req-tag/${TagKey}`	Filtra el acceso por el conjunto de claves y valores de etiquetas que se pueden usar para etiquetar un recurso.	Cadena
`rds:secgrp-tag/${TagKey}`	Filtra el acceso por la etiqueta asociada a un grupo de seguridad de base de datos.	Cadena
`rds:snapshot-tag/${TagKey}`	Filtra el acceso por la etiqueta asociada a una instantánea de base de datos.	Cadena
`rds:subgrp-tag/${TagKey}`	Filtra el acceso por la etiqueta adjunta a un grupo de subred de base de datos	Cadena

Ejemplos de declaraciones de políticas administrativas de IAM para Neptune

Ejemplos de políticas administrativas generales

En los siguientes ejemplos, se muestra cómo crear políticas administrativas de Neptune que concedan permisos para realizar diversas acciones de administración en un clúster de base de datos.

Política que impide que un usuario de IAM elimine una instancia de base de datos específica

A continuación, se muestra un ejemplo de política que impide que un usuario de IAM elimine una instancia de base de datos de Neptune específica:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteOneInstance",
      "Effect": "Deny",
      "Action": "rds:DeleteDBInstance",
```

```
      "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-instance-name"
    }
  ]
}
```

La política concede permiso para crear nuevas instancias de base de datos

A continuación, se muestra un ejemplo de una política que permite a un usuario de IAM crear instancias de base de datos en un clúster de base de datos de Neptune específico:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstance",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster"
    }
  ]
}
```

Política que concede permiso para crear nuevas instancias de base de datos que usen un grupo de parámetros de base de datos específico

A continuación, tenemos un ejemplo de política que permite a un usuario de IAM crear instancias de base de datos en un clúster de base de datos específico (aquí `us-west-2`) en un clúster de base de datos de Neptune específico utilizando solo un grupo de parámetros de base de datos específico.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstanceWithPG",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": [
        "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
        "arn:aws:rds:us-west-2:123456789012:pg:my-instance-pg"
      ]
    }
  ]
```

```
}
```

Política que concede permiso para describir cualquier recurso

A continuación, tenemos un ejemplo de política que permite a un usuario de IAM describir cualquier recurso de Neptune.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDescribe",
      "Effect": "Allow",
      "Action": "rds:Describe*",
      "Resource": *
    }
  ]
}
```

Ejemplos de políticas administrativas basadas en etiquetas

En los siguientes ejemplos, se muestra cómo crear políticas administrativas de Neptune que se etiquetan para filtrar permisos para varias acciones de administración en un cluster de base de datos.

Ejemplo 1: conceder permiso para acciones en un recurso mediante una etiqueta personalizada que puede tomar varios valores

La política siguiente permite el uso de la API `ModifyDBInstance`, `CreateDBInstance` o `DeleteDBInstance` en cualquier instancia de base de datos que tenga la etiqueta `env` establecida en `dev` o `test`:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevTestAccess",
      "Effect": "Allow",
      "Action": [
        "rds:ModifyDBInstance",
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance"
      ],
```

```
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:db-tag/env": [
              "dev",
              "test"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Ejemplo 3: limitar el conjunto de claves y valores de etiquetas que se pueden usar para etiquetar un recurso

En esta política, se usa una clave `Condition` para permitir que una etiqueta que tiene la clave `env` y un valor de `test`, `qa` o `dev` se añada a un recurso:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagAccessForDevResources",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/env": [
            "test",
            "qa",
            "dev"
          ],
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Ejemplo 3: permitir el acceso completo a los recursos de Neptune en función de **`aws:ResourceTag`**

La siguiente política es similar al primer ejemplo anterior, pero utiliza en su lugar `aws:ResourceTag`.

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullAccessToDev",
      "Effect": "Allow",
      "Action": [
        "rds:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "dev",
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Declaraciones de políticas de acceso a datos de IAM personalizadas para Amazon Neptune

Las declaraciones de políticas de acceso a los datos de Neptune utilizan acciones de acceso a los datos, recursos y claves de condición de acceso a los datos, todos ellos precedidas por un prefijo `neptune-db:`.

Temas

- Uso de acciones de consulta en las declaraciones de políticas de acceso a datos de Neptune
- Acciones disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune
- Especificación de recursos en declaraciones de políticas de acceso a datos de IAM de Neptune
- Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune
- Ejemplos de políticas de acceso a datos de IAM de Neptune

Uso de acciones de consulta en las declaraciones de políticas de acceso a datos de Neptune

Hay tres acciones de consulta de Neptune que se pueden utilizar en las declaraciones de políticas de acceso a los datos: `ReadDataViaQuery`, `WriteDataViaQuery` y `DeleteDataViaQuery`. Es posible que una consulta en particular necesite permisos para realizar varias de estas acciones, y puede que no siempre sea evidente qué combinación de estas acciones debe permitirse para ejecutar una consulta.

Antes de ejecutar una consulta, Neptune determina los permisos necesarios para ejecutar cada paso de la consulta y los combina en el conjunto completo de permisos que requiere la consulta. Tenga en cuenta que este conjunto completo de permisos incluye todas las acciones que puede realizar la consulta, que no es necesariamente el conjunto de acciones que la consulta realizará realmente cuando se ejecute en sus datos.

Esto significa que, para permitir que se ejecute una consulta determinada, debe proporcionar permisos para todas las acciones que la consulta pueda realizar, tanto si las acaba realizando como si no.

Estos son algunos ejemplos de consultas de Gremlin en las que esto se explica con más detalle:

- ```
  g.V().count()
  ```

  `g.V()` y `count()` solo requieren acceso de lectura, por lo que la consulta en su conjunto solo requiere acceso `ReadDataViaQuery`.

- ```
  g.addV()
  ```

  `addV()` necesita comprobar si existe o no un vértice con un identificador determinado antes de insertar uno nuevo. Esto significa que requiere tanto acceso `ReadDataViaQuery` como `WriteDataViaQuery`.

- ```
  g.V('1').as('a').out('created').addE('createdBy').to('a')
  ```

  `g.V('1').as('a')` y `out('created')` solo requieren acceso de lectura, pero `addE().from('a')` requiere acceso de lectura y escritura, ya que `addE()` necesita leer los vértices `from` y `to` y comprobar si ya existe un borde con el mismo identificador antes de añadir uno nuevo. Por lo tanto, la consulta en su conjunto necesita tanto el acceso `ReadDataViaQuery` como `WriteDataViaQuery`.
  ```

- ```
  g.V().drop()
  ```

  `g.V()` solo requiere acceso de lectura. `drop()` necesita acceso de lectura y eliminación porque tiene que leer un vértice o borde antes de eliminarlo, por lo que la consulta en su conjunto requiere tanto acceso `ReadDataViaQuery` como `DeleteDataViaQuery`.

- ```
  g.V('1').property(single, 'key1', 'value1')
  ```

  `g.V('1')` solo requiere acceso de lectura, pero `property(single, 'key1', 'value1')` requiere acceso de lectura, escritura y eliminación. En este caso, en el paso `property()`, se inserta la clave y el valor si aún no existen en el vértice, pero si ya existen, se elimina el valor de la propiedad existente y se inserta un nuevo valor en su lugar. Por lo tanto, la consulta en su conjunto requiere acceso `ReadDataViaQuery`, `WriteDataViaQuery` y `DeleteDataViaQuery`.

  Cualquier consulta que contenga un paso `property()` necesitará permisos `ReadDataViaQuery`, `WriteDataViaQuery` y `DeleteDataViaQuery`.

Estos son algunos ejemplos de openCypher:

- ```
  MATCH (n)
  RETURN n
  ```

  Esta consulta lee todos los nodos de la base de datos y los devuelve, lo que solo requiere acceso `ReadDataViaQuery`.

- ```
  MATCH (n:Person)
  SET n.dept = 'AWS'
  ```

  Esta consulta requiere acceso `ReadDataViaQuery`, `WriteDataViaQuery` y `DeleteDataViaQuery`. Lee todos los nodos con la etiqueta 'Person' y añade una nueva propiedad con la clave `dept` y el valor `AWS` o, si la propiedad `dept` ya existe, elimina el valor anterior e inserta `AWS` en su lugar. Además, si el valor que se va a establecer es `null`, `SET` elimina la propiedad por completo.

  Como en algunos casos es posible que la cláusula `SET` necesite eliminar un valor existente, siempre necesitará permisos `DeleteDataViaQuery` y permisos `ReadDataViaQuery` y `WriteDataViaQuery`.

- ```
  MATCH (n:Person)
  DETACH DELETE n
  ```

  Esta consulta necesita permisos `ReadDataViaQuery` y `DeleteDataViaQuery`. Busca todos los nodos con la etiqueta `Person` y los elimina junto con los bordes conectados a esos nodos y las etiquetas y propiedades asociadas.

- ```
  MERGE (n:Person {name: 'John'})-[:knows]->(:Person {name: 'Peter'})
  RETURN n
  ```

  Esta consulta necesita permisos `ReadDataViaQuery` y `WriteDataViaQuery`. La cláusula `MERGE` coincide con un patrón específico o lo crea. Dado que puede producirse una escritura si el patrón no coincide, se necesitan permisos de escritura además de permisos de lectura.

Acciones disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune

Tenga en cuenta que las acciones de acceso a datos de Neptune tienen el prefijo `neptune-db:`, mientras que las acciones administrativas de Neptune tienen el prefijo `rds:`.

El nombre de recurso de Amazon (ARN) para un recurso de datos en IAM no es el mismo que el ARN asignado a un clúster en el momento de su creación. Debe crear el ARN como se muestra en Specifying data resources. Estos ARN de recursos de datos pueden utilizar comodines para incluir múltiples recursos.

Las declaraciones de política de acceso a los datos también pueden incluir la clave de QueryLanguage condición neptune-db: para restringir el acceso mediante el lenguaje de consulta.

A partir de Versión: 1.2.0.0 (21/07/2022), Neptune permite restringir los permisos a una o más acciones específicas de Neptune. Esto proporciona un control de acceso más detallado de lo que era posible anteriormente.

> **Important**
>
> - Los cambios realizados en una política de IAM pueden tardar hasta 10 minutos en aplicarse a los recursos de Neptune especificados.

- Las políticas de IAM que se aplican a un clúster de base de datos de Neptune se aplican a todas las instancias incluidas en dicho clúster.

Acciones de acceso a los datos basadas en consultas

> **Note**
>
> No siempre es obvio qué permisos se necesitan para ejecutar una consulta determinada, ya que las consultas pueden realizar más de una acción en función de los datos que procesen. Para obtener más información, consulte Uso de acciones de consulta.

`neptune-db:ReadDataViaQuery`

`ReadDataViaQuery` permite al usuario leer datos de la base de datos de Neptune mediante el envío de consultas.

Grupos de acciones: solo lectura, lectura-escritura.

Claves de contexto de acción: `neptune-db:QueryLanguage`.

Recursos necesarios: base de datos.

`neptune-db:WriteDataViaQuery`

`WriteDataViaQuery` permite al usuario escribir datos en la base de datos de Neptune mediante el envío de consultas.

Grupos de acciones: lectura-escritura.

Claves de contexto de acción: `neptune-db:QueryLanguage`.

Recursos necesarios: base de datos.

`neptune-db:DeleteDataViaQuery`

`DeleteDataViaQuery` permite al usuario eliminar datos de la base de datos de Neptune mediante el envío de consultas.

Grupos de acciones: lectura-escritura.

Claves de contexto de acción: `neptune-db:QueryLanguage`.

Recursos necesarios: base de datos.

`neptune-db:GetQueryStatus`

`GetQueryStatus` permite al usuario verificar el estado de todas las consultas activas.

Grupos de acciones: solo lectura, lectura-escritura.

Claves de contexto de acción: `neptune-db:QueryLanguage`.

Recursos necesarios: base de datos.

`neptune-db:GetStreamRecords`

`GetStreamRecords` permite al usuario obtener registros de flujo de Neptune.

Grupos de acciones: lectura-escritura.

Claves de contexto de acción: `neptune-db:QueryLanguage`.

Recursos necesarios: base de datos.

`neptune-db:CancelQuery`

`CancelQuery` permite al usuario cancelar una consulta.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

Acciones generales de acceso a los datos

`neptune-db:GetEngineStatus`

`GetEngineStatus` permite al usuario verificar el estado del motor de Neptune.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:GetStatisticsStatus

`GetStatisticsStatus` permite al usuario comprobar el estado de las estadísticas que se recopilan para la base de datos.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:GetGraphSummary

`GetGraphSummary` La API de resumen de gráficos le permite recuperar un resumen de solo lectura de su gráfico.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:ManageStatistics

`ManageStatistics` permite al usuario administrar la recopilación de estadísticas de la base de datos.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:DeleteStatistics

`DeleteStatistics` permite al usuario eliminar todas las estadísticas de la base de datos.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:ResetDatabase

`ResetDatabase` permite al usuario obtener el token necesario para el restablecimiento y restablecer la base de datos de Neptune.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

Acciones de acceso a datos del programa de carga masiva

neptune-db:StartLoaderJob

`StartLoaderJob` permite al usuario iniciar un trabajo del programa de carga masiva.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:GetLoaderJobStatus

`GetLoaderJobStatus` permite al usuario verificar el estado de un trabajo del programa de carga masiva.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:ListLoaderJobs

`ListLoaderJobs` permite al usuario enumerar todos los trabajos del programa de carga masiva.

Grupos de acciones: solo enumeración, solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:CancelLoaderJob

`CancelLoaderJob` permite al usuario cancelar un trabajo del programa de carga.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

Acciones de acceso a los datos mediante machine learning

neptune-db:StartMLDataProcessingJob

`StartMLDataProcessingJob` permite al usuario iniciar un trabajo de procesamiento de datos de Neptune ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:StartMLModelTrainingJob

`StartMLModelTrainingJob` permite a un usuario comenzar un trabajo de entrenamiento de modelos de ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:StartMLModelTransformJob

`StartMLModelTransformJob` permite que un usuario comience un trabajo de transformación de modelos de ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:CreateMLEndpoint

`CreateMLEndpoint` permite al usuario crear un punto de conexión de Neptune ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:GetMLDataProcessingJobStatus

`GetMLDataProcessingJobStatus` permite a un usuario comprobar el estado de un trabajo de procesamiento de datos de Neptune ML.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:GetMLModelTrainingJobStatus

`GetMLModelTrainingJobStatus` permite a un usuario comprobar el estado de un trabajo de entrenamiento de modelos de Neptune ML.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:GetMLModelTransformJobStatus

`GetMLModelTransformJobStatus` permite a un usuario comprobar el estado de un trabajo de transformación de modelos de Neptune ML.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:GetMLEndpointStatus

`GetMLEndpointStatus` permite a un usuario comprobar el estado de un punto de conexión de Neptune ML.

Grupos de acciones: solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:ListMLDataProcessingJobs

`ListMLDataProcessingJobs` permite a un usuario enumerar todos los trabajos de procesamiento de datos de Neptune ML.

Grupos de acciones: solo enumeración, solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:ListMLModelTrainingJobs

`ListMLModelTrainingJobs` permite a un usuario enumerar todos los trabajos de entrenamiento de modelos de Neptune ML.

Grupos de acciones: solo enumeración, solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:ListMLModelTransformJobs

`ListMLModelTransformJobs` permite a un usuario generar una lista de todos los trabajos de transformación de modelos de ML.

Grupos de acciones: solo enumeración, solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:ListMLEndpoints

`ListMLEndpoints` permite a un usuario enumerar todos los puntos de conexión de Neptune ML.

Grupos de acciones: solo enumeración, solo lectura, lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:CancelMLDataProcessingJob

`CancelMLDataProcessingJob` permite a un usuario cancelar un trabajo de procesamiento de datos de Neptune ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:CancelMLModelTrainingJob

`CancelMLModelTrainingJob` permite a un usuario cancelar un trabajo de entrenamiento de modelos de Neptune ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:CancelMLModelTransformJob

`CancelMLModelTransformJob` permite a un usuario cancelar un trabajo de transformación de modelos de Neptune ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

neptune-db:DeleteMLEndpoint

`DeleteMLEndpoint` permite al usuario eliminar un punto de conexión de Neptune ML.

Grupos de acciones: lectura-escritura.

Recursos necesarios: base de datos.

Especificación de recursos en declaraciones de políticas de acceso a datos de IAM de Neptune

Los recursos de datos, como las acciones de datos, tienen un prefijo `neptune-db:`.

En una política de acceso a datos de Neptune, se especifica el clúster de base de datos al que se va a dar acceso en un ARN con el siguiente formato:

```
arn:aws:neptune-db:region:account-id:cluster-resource-id/*
```

Dicho ARN de recurso contiene las siguientes partes:

- *`region`*es la AWS región del clúster de base de datos Amazon Neptune.
- *`account-id`* es el número de cuenta de AWS para el clúster de base de datos.
- *`cluster-resource-id`* es un ID de recurso del clúster de base de datos.

> **Important**
>
> El `cluster-resource-id` es diferente al identificador del clúster. Para encontrar un ID de recurso de clúster en Neptune AWS Management Console, busque el clúster de base de datos en cuestión en la sección Configuración.

Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune

Con las claves de condición, puede especificar condiciones en una declaración de política de IAM para que la declaración se aplique solo cuando se aplican las condiciones.

Las claves de condición que puede usar en las declaraciones de políticas de acceso a los datos de Neptune se clasifican en las siguientes categorías:

- Claves de condición globales: el subconjunto de claves de condición AWS globales que Neptune admite en las declaraciones de política de acceso a los datos se detalla a continuación.
- Claves de condición específicas del servicio: son claves definidas por Neptune específicamente para su uso en las declaraciones de políticas de acceso a los datos. En la actualidad, solo hay

una, neptune-db: QueryLanguage, que solo permite el acceso si se utiliza un lenguaje de consulta específico.

AWS claves de contexto de condiciones globales respaldadas por Neptune en las declaraciones de política de acceso a datos

En la siguiente tabla se muestra el subconjunto de claves de contexto de condición global de AWS que Amazon Neptune admite para su uso en las declaraciones de políticas de acceso a los datos:

Claves de condición globales que puede utilizar en las declaraciones de políticas de acceso a los datos

Claves de condición	Descripción	Tipo
aws:CurrentTime	Filtra el acceso por la fecha y hora actual de la solicitud.	String
aws:EpochTime	Filtra el acceso por fecha y hora de la solicitud expresadas como un valor de época de UNIX.	Numeric
aws:PrincipalAccount	Filtra el acceso por la cuenta a la que pertenece la entidad principal solicitante.	String
aws:PrincipalArn	Filtra el acceso por el ARN de la entidad principal que realizó la solicitud.	String
aws:PrincipalIsAWSService	Permite el acceso solo si la llamada la realiza directamente un director de AWS servicio.	Boolean
aws:PrincipalOrgID	Filtra el acceso por el identificador de la organización en AWS Organizations a la que pertenece el principal solicitante.	String
aws:PrincipalOrgPaths	Filtra el acceso por la AWS ruta Organizations para el director que realiza la solicitud.	String

Claves de condición	Descripción	Tipo
aws:PrincipalTag	Filtra el acceso por una etiqueta asociada a la entidad principal que realiza la solicitud.	String
aws:PrincipalType	Filtra el acceso por el tipo de la entidad principal que realiza la solicitud.	String
aws:RequestedRegion	Filtra el acceso por la AWS región a la que se llamó en la solicitud.	String
aws:SecureTransport	Permite el acceso solo si la solicitud se envió mediante SSL.	Boolean
aws:SourceIp	Filtra el acceso por la dirección IP del solicitante	String
aws:TokenIssueTime	Filtra el acceso por la fecha y hora en que se emitieron las credenciales de seguridad temporales.	String
aws:UserAgent	Filtra el acceso por aplicación cliente del solicitante.	String
aws:userid	Filtra el acceso mediante el identificador de la entidad principal del solicitante.	String
aws:ViaAWSService	Permite el acceso solo si un AWS servicio ha realizado la solicitud en tu nombre.	Boolean

Claves de condición específicas del servicio de Neptune

Neptune admite la siguiente clave de condición específica del servicio para las políticas de IAM:

Claves de condición específicas del servicio de Neptune

Claves de condición	Descripción	Tipo
`neptune-db:QueryLanguage`	Filtra el acceso a los datos según el lenguaje de consulta que se utilice. Los valores válidos son `Gremlin`, `OpenCypher` y `Sparql`. Las acciones admitidas son `ReadDataViaQuery`, `WriteDataViaQuery`, `DeleteDataViaQuery`, `GetQueryStatus` y `CancelQuery`.	`String`

Ejemplos de políticas de acceso a datos de IAM de Neptune

En los siguientes ejemplos, se muestra cómo crear políticas de IAM personalizadas que utilizan un control de acceso detallado de las API y acciones del plano de datos, introducidas en la versión 1.2.0.0 del motor de Neptune.

Ejemplo de política que permite el acceso sin restricciones a los datos de un clúster de base de datos de Neptune

En la siguiente política de ejemplo, se permite a un usuario de IAM conectarse a un clúster de base de datos de Neptune utilizando la autenticación de base de datos de IAM, y se utiliza el carácter "*" para hacer corresponder todas las acciones disponibles.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

La sección de recursos anterior incluye un ARN de recurso en un formato específico de la autenticación de IAM de Neptune. Para crear el ARN, consulte Specifying data resources. Tenga en cuenta que el ARN que se utiliza para una autorización de IAM `Resource` no es el mismo que el ARN asignado al clúster en el momento de su creación.

Ejemplo de política que permite acceso de solo lectura a un clúster de base de datos de Neptune

La siguiente política concede permiso para el acceso completo de solo lectura a los datos de un clúster de base de datos de Neptune:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:Read*",
        "neptune-db:Get*",
        "neptune-db:List*"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Ejemplo de política que deniega todo acceso a un clúster de base de datos de Neptune

La acción predeterminada de IAM es denegar el acceso a un clúster de base de datos, a menos que se conceda un efecto de `Allow`. Sin embargo, la siguiente política deniega todo acceso a un clúster de base de datos para una AWS cuenta y una región determinadas, por lo que prevalece sobre cualquier otro `Allow` efecto.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
```

```
  ]
}
```

Ejemplo de política que concede acceso de lectura a través de consultas

La siguiente política solo concede permiso para leer desde un clúster de base de datos de Neptune utilizando una consulta:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:ReadDataViaQuery",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Ejemplo de política que solo permite consultas de Gremlin

La siguiente política usa la clave de condición `neptune-db:QueryLanguage` para conceder permiso para consultar Neptune únicamente con el lenguaje de consulta Gremlin:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "neptune-db:QueryLanguage": "Gremlin"
        }
      }
    }
```

```
  ]
}
```

Ejemplo de política que permite todos los accesos excepto a la administración de modelos de Neptune ML

La siguiente política concede acceso completo a las operaciones de gráficos de Neptune, excepto a las características de administración de modelos de Neptune ML:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelLoaderJob",
        "neptune-db:CancelQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:DeleteStatistics",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetLoaderJobStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:GetStatisticsStatus",
        "neptune-db:GetStreamRecords",
        "neptune-db:ListLoaderJobs",
        "neptune-db:ManageStatistics",
        "neptune-db:ReadDataViaQuery",
        "neptune-db:ResetDatabase",
        "neptune-db:StartLoaderJob",
        "neptune-db:WriteDataViaQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Ejemplo de política que permite el acceso a la administración de modelos de Neptune ML

Esta política concede acceso a las características de administración de modelos de Neptune ML:

```
{
```

```
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:CancelMLDataProcessingJob",
        "neptune-db:CancelMLModelTrainingJob",
        "neptune-db:CancelMLModelTransformJob",
        "neptune-db:CreateMLEndpoint",
        "neptune-db:DeleteMLEndpoint",
        "neptune-db:GetMLDataProcessingJobStatus",
        "neptune-db:GetMLEndpointStatus",
        "neptune-db:GetMLModelTrainingJobStatus",
        "neptune-db:GetMLModelTransformJobStatus",
        "neptune-db:ListMLDataProcessingJobs",
        "neptune-db:ListMLEndpoints",
        "neptune-db:ListMLModelTrainingJobs",
        "neptune-db:ListMLModelTransformJobs",
        "neptune-db:StartMLDataProcessingJob",
        "neptune-db:StartMLModelTrainingJob",
        "neptune-db:StartMLModelTransformJob"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Ejemplo de política que otorga acceso completo a las consultas

La siguiente política concede acceso completo a las operaciones de consulta de gráficos de Neptune, pero no a características como el restablecimiento rápido, los flujos, el programa de carga masiva, la administración de modelos de Neptune ML, etc.:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
```

```
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Ejemplo de política que concede acceso completo únicamente a las consultas de Gremlin

La siguiente política concede acceso completo a las operaciones de consulta de gráficos de Neptune utilizando el lenguaje de consulta Gremlin, pero no a las consultas en otros lenguajes, ni a características como el restablecimiento rápido, los flujos, el programa de carga masiva, la administración de modelos de Neptune ML, etc.:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery",
        "neptune-db:GetEngineStatus",
        "neptune-db:GetQueryStatus",
        "neptune-db:CancelQuery"
      ],
      "Resource": [
        "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD1234EFGH5678IJKL90MNOP/
*"
      ],
      "Condition": {
        "StringEquals": {
           "neptune-db:QueryLanguage":"Gremlin"
        }
      }
    }
  ]
}
```

Ejemplo de política que permite el acceso total excepto el restablecimiento rápido

La siguiente política concede acceso completo a un clúster de base de datos de Neptune, excepto para el restablecimiento rápido:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    },
    {
      "Effect": "Deny",
      "Action": "neptune-db:ResetDatabase",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Uso de roles vinculados a servicios para Neptune

Amazon Neptune utiliza funciones vinculadas a AWS Identity and Access Management servicios (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM que está vinculado directamente a Neptune. Neptune predefine las funciones vinculadas al servicio e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en su nombre.

> **Important**
>
> Para ciertas características de administración, Amazon Neptune utiliza una tecnología operativa que comparte con Amazon RDS. Esto incluye la función vinculada a un servicio y los permisos de la API de administración.

Un rol vinculado a un servicio simplifica la configuración de Neptune porque ya no tendrá que añadir manualmente los permisos necesarios. Neptune define los permisos de sus roles vinculados a

servicios y, a menos que esté definido de otra manera, solo Neptune puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda asociar a ninguna otra entidad de IAM.

Las funciones se pueden eliminar únicamente después de eliminar primero sus recursos relacionados. De esta forma, se protegen los recursos de Neptune, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Para obtener información sobre otros servicios que son compatibles con los roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios que muestran Sí en la columna Rol vinculado a servicios. Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado a servicios en cuestión.

Permisos de roles vinculados a servicios de Neptune

Neptune usa la función `AWSServiceRoleForRDS` vinculada al servicio para permitir que Neptune y Amazon RDS AWS llamen a los servicios en nombre de sus instancias de base de datos. El rol vinculado a servicios `AWSServiceRoleForRDS` confía en el servicio `rds.amazonaws.com` para asumir el rol.

La política de permisos del rol permite que Neptune realice las siguientes acciones en los recursos especificados:

- Acciones en `ec2`:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`

  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Acciones en `sns`:
  - `ListTopic`
  - `Publish`
- Acciones en `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Podría encontrarse con el siguiente mensaje de error:
> Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.
> Si aparece este mensaje, asegúrese de que tiene los siguientes permisos habilitados:

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
    }
}
```

Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio de Neptune

No necesita crear manualmente un rol vinculado a servicios. Al crear una instancia o un clúster, Neptune se encarga de crear el rol vinculado al servicio.

> **Important**
>
> Para obtener más información, consulte Un nuevo rol ha aparecido en mi cuenta de IAM en la Guía del usuario de IAM.

Si elimina este rol vinculado a un servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Al crear una instancia o un clúster, Neptune se encarga de crear el rol vinculado al servicio de nuevo.

Modificación de un rol vinculado a un servicio de Neptune

Neptune no le permite editar el rol vinculado a servicios `AWSServiceRoleForRDS`. Después de crear un rol vinculado al servicio, no podrá cambiar el nombre del rol, ya que varias entidades podrían hacer referencia al rol. Sin embargo, sí puede editar la descripción del rol con IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio de Neptune

Si ya no necesita utilizar una característica o servicio que requiere un rol vinculado a servicios, recomendamos que elimine dicho rol. De esta forma no tiene una entidad no utilizada que no se monitoree ni mantenga de forma activa. Sin embargo, debe eliminar todas las instancias y clústeres para poder eliminar la función vinculada al servicio asociada.

Limpiar una función vinculada a un servicio antes de eliminar

Antes de poder utilizar IAM para eliminar un rol vinculado a un servicio, primero debe confirmar que dicho rol no tiene sesiones activas y eliminar los recursos que utiliza.

Para comprobar si el rol vinculado a un servicio tiene una sesión activa en la consola de IAM

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola de IAM, elija Roles (Roles). A continuación, seleccione el nombre (no la casilla de verificación) del rol de `AWSServiceRoleForRDS`.
3. En la página Resumen del rol seleccionado, seleccione la pestaña Asesor de acceso.
4. En la pestaña Asesor de acceso, revise la actividad reciente del rol vinculado a servicios.

> **Note**
>
> Si no está seguro de si Neptune utiliza el rol `AWSServiceRoleForRDS`, puede intentar eliminar el rol para comprobarlo. Si el servicio está utilizando el rol, este no podrá eliminarse y podrá ver las regiones en las que se está utilizando. Si el rol se está utilizando, debe esperar que la sesión finalice para poder eliminarlo. No se puede revocar la sesión de un rol vinculado a servicios.

Si desea quitar la función `AWSServiceRoleForRDS`, primero debe eliminar todos sus clústeres e instancias.

Eliminación de todas las instancias

Use alguno de estos procedimientos para eliminar cada una de sus instancias.

Para eliminar una instancia (consola)

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Instancias.
3. En la lista Instances, elija la instancia que desea eliminar.
4. Elija Instance actions y, a continuación, Delete.
5. Si aparece el mensaje Create final Snapshot? (¿Crear instantánea final?), elija Yes (Sí) o No.
6. Si eligió Yes (Sí) en el paso anterior, en Final snapshot name (Nombre de instantánea final) escriba el nombre de la instantánea final.
7. Elija Eliminar.

Para eliminar una instancia (AWS CLI)

Consulte `delete-db-instance` en la referencia de comandos de AWS CLI .

Para eliminar una instancia (API)

Consulte `DeleteDBInstance`.

Eliminación de todos los clústeres

Utilice uno de los siguientes procedimientos para eliminar un clúster y, a continuación, repita el procedimiento para cada uno de los clústeres.

Para eliminar un clúster (consola)

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En la lista Clusters, elija el clúster que desea eliminar.
3. Elija Cluster Actions y, a continuación, Delete.
4. Elija Eliminar.

Para eliminar un clúster (CLI)

Consulte `delete-db-cluster` en la referencia de comandos de AWS CLI .

Para eliminar un clúster (API)

Consulte `DeleteDBCluster`

Puede utilizar la consola de IAM, la CLI de IAM o la API de IAM para eliminar el rol vinculado a servicios `AWSServiceRoleForRDS`. Para más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Autenticación IAM mediante credenciales temporales

Amazon Neptune admite la autenticación de IAM con credenciales temporales.

Puede utilizar un rol asumido para la autenticación mediante una política de autenticación de IAM, como una de las políticas de ejemplo de las secciones anteriores.

Si está utilizando credenciales temporales, debe especificar `AWS_SESSION_TOKEN` además de `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `SERVICE_REGION`.

> **Note**
>
> Las credenciales temporales caducan después de un intervalo especificado, incluido el token de sesión.
> Tiene que actualizar el token de sesión cuando solicite nuevas credenciales. Para obtener más información, consulte Uso de credenciales de seguridad temporales para solicitar acceso a AWS los recursos.

En las siguientes secciones se describe cómo permitir el acceso y recuperar las credenciales temporales.

Para realizar la autenticación mediante credenciales temporales

1. Cree un rol de IAM con permiso para obtener acceso a un clúster de Neptune. Para obtener información acerca de la creación de esta función, consulte the section called “Tipos de políticas de IAM”.
2. Añada una relación de confianza al rol que permite el acceso a las credenciales.

   Recupere las credenciales temporales, incluidas las credenciales `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `AWS_SESSION_TOKEN`.
3. Conéctese al clúster de Neptune y firme las solicitudes con las credenciales temporales. Para obtener más información acerca de la conexión y la firma de solicitudes, consulte the section called “Conexión y firma”.

Hay varios métodos para recuperar credenciales temporales según el entorno.

Temas

- Obtención de credenciales temporales usando la AWS CLI
- Configuración de AWS Lambda para la autenticación IAM de Neptune
- Configuración de Amazon EC2 para la autenticación de IAM de Neptune

Obtención de credenciales temporales usando la AWS CLI

Para obtener las credenciales con AWS Command Line Interface (AWS CLI), primero debe agregar una relación de confianza que conceda permiso para asumir el rol al AWS usuario que ejecutará el AWS CLI comando.

Añada la siguiente relación de confianza al rol de autenticación de IAM de Neptune. Si no tiene un rol de autenticación de IAM de Neptune, consulte the section called “Tipos de políticas de IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Para obtener información sobre cómo añadir la relación de confianza a la función, consulte Edición de la relación de confianza para una función existente en la guía de administración de AWS Directory Service .

Si la política de Neptune aún no está asociada a un rol, cree un nuevo rol. Asocie la política de autenticación de IAM de Neptune y, a continuación, añada la política de confianza. Para obtener información sobre la creación de una función nueva, consulte Creación de una función nueva.

> **Note**
>
> En las siguientes secciones se presupone que lo tiene AWS CLI instalado.

Para ejecutar el AWS CLI manualmente

1. Escriba el comando siguiente para solicitar las credenciales usando la AWS CLI. Reemplace el ARN del rol, el nombre de la sesión y el perfil por sus propios valores.

```
aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/NeptuneIAMAuthRole
 --role-session-name test --profile testprofile
```

2. A continuación se muestra un ejemplo de salida del comando. La sección `Credentials` contiene los valores que necesita.

> **Note**
>
> Registre el valor de `Expiration` ya que lo necesitará para obtener nuevas credenciales transcurrido este tiempo.

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-access-
example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

3. Establezca las variables de entorno mediante las credenciales devueltas.

```
export AWS_ACCESS_KEY_ID=ASIAJEXAMPLEXEG2JICEA
export AWS_SECRET_ACCESS_KEY=9drTJvcXLB89EXAMPLELB8923FB892xMFI
export AWS_SESSION_TOKEN=AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/qwjzP2iEXAMPLEbw/
m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
```

```
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8BRi
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

4. Conéctese utilizando uno de los siguientes métodos.

   - the section called “Consola de Gremlin”
   - the section called “Java de Gremlin”
   - the section called “SPARQL Java (RDF4J y Jena)”
   - the section called “Ejemplo de Python”

Para usar un script para obtener las credenciales

1. Ejecute el siguiente comando para instalar el comando jq. El script usa este comando para analizar el resultado del AWS CLI comando.

   ```
   sudo yum -y install jq
   ```

2. Cree un archivo con el nombre `credentials.sh` en un editor de texto y añada el siguiente texto. Reemplace la región del servicio, el ARN del rol, el nombre de la sesión y el perfil por sus propios valores.

   ```
   #!/bin/bash

   creds_json=$(aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/
   NeptuneIAMAuthRole --role-session-name test --profile testprofile)

   export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .Credentials.AccessKeyId |tr -d
    '"')
   export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" |
    jq .Credentials.SecretAccessKey| tr -d '"')
   ```

```
export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Credentials.SessionToken|tr -d
 '"')

export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Conéctese utilizando uno de los siguientes métodos.

- the section called “Consola de Gremlin”
- the section called “Java de Gremlin”
- the section called “SPARQL Java (RDF4J y Jena)”
- the section called “Ejemplo de Python”

Configuración de AWS Lambda para la autenticación IAM de Neptune

AWS Lambda incluye las credenciales automáticamente cada vez que se ejecuta la función Lambda.

En primer lugar, añada una relación de confianza que conceda permiso para asumir el rol al servicio de Lambda.

Añada la siguiente relación de confianza al rol de autenticación de IAM de Neptune. Si no tiene un rol de autenticación de IAM de Neptune, consulte the section called “Tipos de políticas de IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
```

Para obtener información sobre cómo añadir la relación de confianza a la función, consulte Edición de la relación de confianza para una función existente en la guía de administración de AWS Directory Service.

Si la política de Neptune aún no está asociada a un rol, cree un nuevo rol. Asocie la política de autenticación de IAM de Neptune y, a continuación, añada la política de confianza. Para obtener información sobre la creación de una función nueva, consulte Creación de una función nueva en la guía de administración de AWS Directory Service .

Para obtener acceso a Neptune desde Lambda

1. Inicie sesión en la AWS Lambda consola AWS Management Console y ábrala en https://console.aws.amazon.com/lambda/.
2. Cree una nueva función Lambda para Python versión 3.6.
3. Asigne el rol `AWSLambdaVPCAccessExecutionRole` a la función Lambda. Esto es necesario para obtener acceso a los recursos de Neptune, que son solo VPC.
4. Asigne el rol de IAM de autenticación de Neptune a la función de Lambda.

   Para obtener más información, consulte Permisos de AWS Lambda en la Guía para desarrolladores de AWS Lambda .
5. Copie la muestra de Python de autenticación de IAM en el código de la función Lambda.

   Para obtener más información acerca de la muestra y el código de muestra, consulte the section called “Ejemplo de Python”.

Configuración de Amazon EC2 para la autenticación de IAM de Neptune

Amazon EC2 le permite utilizar perfiles de instancia para proporcionar credenciales automáticamente. Para obtener más información, consulte Uso de perfiles de instancias en la Guía del usuario de IAM.

En primer lugar, añada una relación de confianza que conceda permiso para asumir el rol al servicio de Amazon EC2.

Añada la siguiente relación de confianza al rol de autenticación de IAM de Neptune. Si no tiene un rol de autenticación de IAM de Neptune, consulte the section called “Tipos de políticas de IAM”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Para obtener información sobre cómo añadir la relación de confianza a la función, consulte Edición de la relación de confianza para una función existente en la guía de administración de AWS Directory Service .

Si la política de Neptune aún no está asociada a un rol, cree un nuevo rol. Asocie la política de autenticación de IAM de Neptune y, a continuación, añada la política de confianza. Para obtener información sobre la creación de una función nueva, consulte Creación de una función nueva en la guía de administración de AWS Directory Service .

Para usar un script para obtener las credenciales

1. Ejecute el siguiente comando para instalar el comando jq. El script utiliza este comando para analizar la salida del comando curl.

   ```
   sudo yum -y install jq
   ```

2. Cree un archivo con el nombre `credentials.sh` en un editor de texto y añada el siguiente texto. Reemplace la región de servicio por su propio valor.

   ```
   role_name=$( curl -s http://169.254.169.254/latest/meta-data/iam/security-
   credentials/ )
   creds_json=$(curl -s http://169.254.169.254/latest/meta-data/iam/security-
   credentials/${role_name})

   export AWS_ACCESS_KEY_ID=$(echo "$creds_json" | jq .AccessKeyId |tr -d '"')
   export AWS_SECRET_ACCESS_KEY=$(echo "$creds_json" | jq .SecretAccessKey| tr -d '"')
   export AWS_SESSION_TOKEN=$(echo "$creds_json" | jq .Token|tr -d '"')
   ```

```
export SERVICE_REGION=us-east-1 or us-east-2 or us-west-1 or us-west-2 or ca-
central-1 or
                      sa-east-1 or eu-north-1 or eu-west-1 or eu-west-2 or eu-
west-3 or eu-central-1 or me-south-1 or
                      me-central-1 or il-central-1 or af-south-1 or ap-east-1 or
 ap-northeast-1 or ap-northeast-2 or ap-southeast-1 or ap-southeast-2 or ap-south-1
 or
                      cn-north-1 or cn-northwest-1 or
                      us-gov-east-1 or us-gov-west-1
```

3. Ejecute el script en el shell `bash` con el comando `source`:

```
source credentials.sh
```

   Aún mejor es agregar los comandos de este script al archivo `.bashrc` de la instancia EC2 para que se invoquen automáticamente cuando inicie sesión, lo que hará que las credenciales temporales estén disponibles en la consola de Gremlin.

4. Conéctese utilizando uno de los siguientes métodos.

   - the section called “Consola de Gremlin”
   - the section called “Java de Gremlin”
   - the section called “SPARQL Java (RDF4J y Jena)”
   - the section called “Ejemplo de Python”

Registro y monitorización de recursos de Amazon Neptune

Amazon Neptune admite varios métodos para monitorizar el rendimiento y el uso:

- Estado del clúster: compruebe el estado del motor de base de datos de gráficos de un clúster de Neptune. Para obtener más información, consulte the section called “Estado de la instancia”.
- Amazon CloudWatch: Neptune envía automáticamente las métricas a las alarmas CloudWatch y también las admite CloudWatch . Para obtener más información, consulte the section called “Usando CloudWatch”.
- Archivos de registro de auditoría: consulte, descargue o vea los archivos de registro de base de datos con la consola de Neptune. Para obtener más información, consulte the section called “Registros de auditoría con Neptune”.

- Publicación de registros en Amazon CloudWatch Logs: puede configurar un clúster de base de datos de Neptune para publicar datos de registros de auditoría en un grupo de registros de Amazon CloudWatch Logs. Con CloudWatch Logs, puede realizar un análisis en tiempo real de los datos de registro, usarlos CloudWatch para crear alarmas y ver métricas, y usar CloudWatch Logs para almacenar sus registros de registro en un almacenamiento de alta durabilidad. Para obtener más información, consulte Registros de Neptune CloudWatch .
- AWS CloudTrail— Neptune admite el registro de API mediante. CloudTrail Para obtener más información, consulte the section called “Registro de llamadas a la API de Neptune con AWS CloudTrail”.
- Etiquetado: utilice etiquetas para añadir metadatos a sus recursos de Neptune y realizar un seguimiento del uso en función de las etiquetas. Para obtener más información, consulte the section called “Etiquetado de recursos de Neptune”.

Validación de la conformidad para Amazon Neptune

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Guías de inicio rápido sobre seguridad y cumplimiento: estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en AWS la seguridad y el cumplimiento.
- Diseño de arquitectura para garantizar la seguridad y el cumplimiento de la HIPAA en Amazon Web Services: en este documento técnico se describe cómo pueden utilizar AWS las empresas para crear aplicaciones aptas para la HIPAA.

> **Note**
>
> No Servicios de AWS todas cumplen los requisitos de la HIPAA. Para más información, consulte la Referencia de servicios compatibles con HIPAA.

- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.
- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Resiliencia en Amazon Neptune

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas

de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Un clúster de base de datos de Amazon Neptune solo se puede crear en una Amazon VPC que tenga un mínimo de dos subredes en al menos dos zonas de disponibilidad. Mediante la distribución de las instancias del clúster en al menos dos zonas de disponibilidad, Neptune contribuye a garantizar que haya instancias disponibles en el clúster de base de datos en el caso improbable de que se produzca un error en una zona de disponibilidad. El volumen de un clúster de base de datos de Neptune siempre abarca tres zonas de disponibilidad. De este modo, se proporciona un almacenamiento duradero y con un menor riesgo de pérdida de datos.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS

Migración de un gráfico existente a Amazon Neptune

Existen varias herramientas y técnicas que pueden ayudarle a migrar los datos de gráficos existentes a Amazon Neptune desde otro almacén de datos.

Un flujo de trabajo de migración sencillo implica los siguientes pasos:

- Exporte los datos de su almacén existente a Amazon Simple Storage Service (Amazon S3).
- Límpielos y formatéelos para importarlos.
- Cárguelos en un clúster de base de datos de Neptune mediante Programa de carga masiva de Neptune.
- Configure la aplicación de Gremlin o SPARQL para que utilice el punto de conexión correspondiente que proporciona Neptune.

> **Note**
>
> El clúster de Neptune debe estar ejecutándose en una VPC a la que pueda acceder la aplicación.

Existen formas de simplificar y automatizar algunos de estos pasos, en función del lugar donde se almacenen los datos:

Temas

- Migración de Neo4j a Amazon Neptune
- Migración de un gráfico existente de un servidor Apache TinkerPop Gremlin a Amazon Neptune
- Migración de un gráfico existente de un almacén triple de RDF a Amazon Neptune
- Uso de AWS Database Migration Service (AWS DMS) para migrar de una base de datos relacional o NoSQL a Amazon Neptune
- Migración de Blazegraph a Amazon Neptune

Migración de Neo4j a Amazon Neptune

Tanto Neo4j como Amazon Neptune son bases de datos de gráficos, diseñadas para cargas de trabajo de gráficos transaccionales en línea que admiten el modelo de datos de gráficos de propiedades etiquetadas. Estas similitudes hacen de Neptune una opción común para los clientes que buscan migrar sus aplicaciones de Neo4j actuales. Sin embargo, estas migraciones no son simples procesos de migrar mediante lift-and-shift, ya que existen diferencias en cuanto a la compatibilidad con los lenguajes y las características, las características operativas, la arquitectura de los servidores y las capacidades de almacenamiento entre las dos bases de datos.

En esta página se explica el proceso de migración y se presentan aspectos a tener en cuenta antes de migrar una aplicación de gráficos de Neo4j a Neptune. Estos aspectos suelen aplicarse a cualquier aplicación de gráficos de Neo4j, ya sea que funcione con una base de datos Community, Enterprise o Aura. Aunque cada solución sea única y pueda requerir procedimientos adicionales, todas las migraciones siguen el mismo patrón general.

Cada uno de los pasos descritos en las siguientes secciones incluye aspectos a tener en cuenta y recomendaciones para simplificar el proceso de migración. Además, hay herramientas de código abierto y publicaciones de blog que describen el proceso, así como una sección de compatibilidad de características con opciones arquitectónicas recomendadas.

Temas

Información general sobre la migración de Neo4j a Neptune

Con la compatibilidad de Neptune con el lenguaje de consulta de openCypher, puede mover la mayoría de las cargas de trabajo de Neo4j que utilizan el protocolo Bolt o HTTPS a Neptune.

Sin embargo, openCypher es una especificación de código abierto que incluye la mayoría de las funciones compatibles con otras bases de datos, como Neo4j, aunque no todas.

A pesar de ser compatible en muchos aspectos, Neptune no es un sustituto directo de Neo4j. Neptune es un servicio de base de datos de gráficos totalmente gestionado con características empresariales como alta disponibilidad y alta durabilidad que es arquitectónicamente diferente de Neo4j. Neptune se basa en instancias, con una única instancia de escritor principal y hasta 15 instancias de réplica de lectura que permiten escalar la capacidad de lectura horizontalmente. Con Neptune sin servidor, puede escalar y reducir verticalmente y de forma automática su capacidad informática en función del volumen de consultas. Esto es independiente del almacenamiento de Neptune, que se escala automáticamente a medida que se van añadiendo datos.

Neptune es compatible con la especificación estándar de openCypher de código abierto, versión 9. En AWS, creemos que el código abierto es la opción adecuada para todos y estamos comprometidos a llevar el valor del código abierto a nuestros clientes y la excelencia operativa de AWS a las comunidades de código abierto.

Sin embargo, muchas aplicaciones que se ejecutan en Neo4j también utilizan características propias que no son de código abierto y que Neptune no admite. Por ejemplo, Neptune no admite los procedimientos de APOC, algunas cláusulas y funciones específicas de Cypher ni tipos de datos `Char`, `Date` o `Duration`. Neptune convierte automáticamente los tipos de datos que faltan en tipos de datos compatibles.

Además de openCypher, Neptune también es compatible con el lenguaje de consulta Apache TinkerPop Gremlin para gráficos de propiedades (así como con SPARQL para datos RDF). Gremlin puede interoperar con openCypher en el mismo gráfico de propiedades y, en muchos casos, se puede usar Gremlin para proporcionar funciones que openCypher no proporciona. A continuación se muestra una comparación rápida de los dos lenguajes:

	openCypher	Gremlin
Style (Estilo)	Declarativo	Imperativo

Sintaxis	Coincidencia de patrones
	`Match p=(a)-[:route]->(d)`
	`WHERE a.code='ANC'`
	`RETURN p`
		Basado en la transversalidad
		`g.V().has('code', 'ANC').`
		`out('route').path().`
		`by(elementMap())`

	openCypher	Gremlin
Facilidad de uso	Inspirado en SQL, legible por personas que no son programadores	Curva de aprendizaje más pronunciada, similar a la de lenguajes de programación como Java
Flexibilidad	Baja	Alta
Soporte de consultas	Consultas basadas en cadenas	Consultas basadas en cadenas o código en línea compatibles con las bibliotec as cliente
Clientes	HTTPS y Bolt	HTTPS y Websockets

Por lo general, no es necesario cambiar el modelo de datos para migrar de Neo4j a Neptune, ya que tanto Neo4j como Neptune admiten datos de gráficos de propiedades etiquetadas (LPG). Sin embargo, Neptune presenta algunas diferencias en la arquitectura y el modelo de datos que puede sacar partido para optimizar el rendimiento. Por ejemplo:

- Los ID de Neptune se tratan como ciudadanos de primera clase.
- Neptune utiliza políticas de AWS Identity and Access Management (IAM) para proteger el acceso a los datos de sus gráficos de forma flexible y detallada.
- Neptune ofrece varias formas de utilizar los cuadernos de Jupyter para ejecutar consultas y visualizar los resultados. Neptune también funciona con herramientas de visualización de terceros.
- >Aunque Neptune no tiene un sustituto directo para la biblioteca Graph Data Science (GDS) de Neo4j, Neptune admite el análisis de gráficos en la actualidad a través de una variedad de soluciones. Por ejemplo, varios cuadernos de muestra muestran cómo sacar partido de la integración de Neptune con AWS SDK para Pandas en entornos de Python para ejecutar análisis en datos de gráficos.

Si tiene alguna pregunta, póngase en contacto con AWS Support o con el equipo de cuentas de AWS. Usamos sus comentarios para priorizar las nuevas características que se adapten a sus necesidades.

Preparación para migrar de Neo4j a Neptune

Métodos relacionados con la migración

Al migrar una aplicación de Neo4j a Neptune, recomendamos una de estas dos estrategias: redefinir la plataforma o refactorizar/rediseñar la arquitectura. Para obtener más información sobre las estrategias de migración, consulte Seis estrategias para migrar aplicaciones a la nube, una publicación del blog de Stephen Orban.

El método para redefinir la plataforma, que también se conoce como lift-tinker-and-shift, implica los siguientes pasos:

- Identificar los casos de uso para los que la aplicación está diseñada.
- Modificar el modelo de datos de gráficos y la arquitectura de la aplicación existentes para abordar mejor estas necesidades de carga de trabajo con las capacidades de Neptune.
- Determinar cómo migrar los datos, las consultas y otras partes de la aplicación de origen al modelo y la arquitectura de destino.

Este método de empezar por el final le permite migrar su aplicación al tipo de solución de Neptune que podría diseñar si se tratara de un proyecto completamente nuevo.

El método de refactorización, por el contrario, implica:

- Identificar los componentes de la implementación existente, incluidos la infraestructura, los datos, las consultas y las capacidades de las aplicaciones.
- Encontrar equivalentes en Neptune que puedan usarse para crear una implementación comparable.

Este método de empezar por el principio busca cambiar una implementación por otra.

En la práctica, es probable que adopte una combinación de estos dos métodos. Puede comenzar con un caso de uso, diseñar la arquitectura de Neptune de destino, pero luego pasar a la implementación existente de Neo4j para identificar las restricciones e invariables que tendrá que mantener. Por ejemplo, es posible que tenga que continuar con la integración con otros sistemas externos o seguir ofreciendo API específicas a los usuarios de la aplicación de gráficos. Con esta información, puede determinar qué datos ya existen para pasarlos al modelo de destino y cuáles deben proceder de otro lugar.

En otros casos, podría empezar por analizar una parte específica de la implementación de Neo4j como el mejor origen de información sobre el trabajo al que está destinada la aplicación. Ese tipo de arqueología en la aplicación existente puede ayudar a definir un caso de uso que luego se pueda diseñar con el fin de sacar partido de las capacidades de Neptune.

Ya sea que esté creando una nueva aplicación con Neptune o migrando una aplicación existente desde Neo4j, le recomendamos empezar por el final a partir de los casos de uso para diseñar un modelo de datos, un conjunto de consultas y una arquitectura de aplicaciones que aborde las necesidades de su empresa.

Diferencias arquitectónicas entre Neptune y Neo4j

Cuando los clientes se plantean por primera vez migrar una aplicación de Neo4j a Neptune, a menudo resulta tentador realizar una comparación similar en función del tamaño de la instancia. Sin embargo, las arquitecturas de Neo4j y Neptune tienen diferencias fundamentales. Neo4j se basa en un enfoque integral en el que la carga de datos, ETL de datos, las consultas a las aplicaciones, el almacenamiento de datos y las operaciones de administración se realizan en el mismo conjunto de recursos informáticos, como, por ejemplo, las instancias EC2.

Neptune, por el contrario, es una base de datos de gráficos centrada en OLTP en la que la arquitectura separa las responsabilidades y en la que los recursos se desacoplan para que puedan escalarse de forma dinámica e independiente.

Al migrar de Neo4j a Neptune, determine los requisitos de durabilidad, disponibilidad y escalabilidad de los datos de la aplicación. La arquitectura de clústeres de Neptune simplifica el diseño de aplicaciones que requieren una alta durabilidad, disponibilidad y escalabilidad. Una vez que comprenda la arquitectura de clústeres de Neptune, podrá diseñar una topología de clústeres de Neptune que haga frente a estos requisitos.

La arquitectura de clústeres de Neo4j

Muchas aplicaciones de producción utilizan la agrupación en clústeres causal de Neo4j para proporcionar durabilidad, alta disponibilidad y escalabilidad de los datos. La arquitectura de agrupación en clústeres de Neo4j utiliza instancias de servidor de núcleo e instancias de réplica de lectura:

- Los servidores de núcleo garantizan la durabilidad de los datos y la tolerancia a los errores al replicar los datos mediante el protocolo Raft.
- Las réplicas de lectura utilizan el envío de registros de transacciones para replicar datos de forma asíncrona para cargas de trabajo de alto rendimiento de lectura.

Cada instancia de un clúster, ya sea un servidor de núcleo o una réplica de lectura, incluye una copia completa de los datos del gráfico.

Arquitectura de clústeres de Neptune

Un clúster de Neptune consta de una instancia de escritor principal y hasta 15 instancias de réplica de lectura. Todas las instancias del clúster comparten el mismo servicio de almacenamiento distribuido subyacente que es independiente de las instancias.

- La instancia de escritor principal coordina todas las operaciones de escritura en la base de datos y se puede escalar verticalmente para ofrecer un soporte flexible a diferentes cargas de trabajo de escritura. También admite operaciones de lectura.
- Las instancias de réplica de lectura admiten operaciones de lectura del volumen de almacenamiento subyacente y permiten escalar horizontalmente para admitir cargas de trabajo de lectura elevadas. También proporcionan alta disponibilidad al servir como destinos de conmutación por error para la instancia principal.

  > **Note**
  >
  > En el caso de cargas de trabajo de escritura intensas, se recomienda escalar las instancias de réplica de lectura para que tengan el mismo tamaño que la instancia de escritor, con el fin de garantizar que los lectores puedan mantener la coherencia con los cambios realizados en los datos.

- El volumen de almacenamiento subyacente escala la capacidad de almacenamiento automáticamente a medida que aumentan los datos de la base de datos, hasta 128 tebibytes (TiB) de almacenamiento.

Los tamaños de las instancias son dinámicos e independientes. Se puede cambiar el tamaño de cada instancia mientras el clúster esté en ejecución, y las réplicas de lectura se pueden añadir o eliminar mientras el clúster esté en ejecución.

La característica Neptune sin servidor puede escalar y reducir verticalmente de forma automática su capacidad de cómputo a medida que la demanda aumenta y disminuye. Esto no solo reduce la sobrecarga administrativa, sino que también permite configurar la base de datos para gestionar los grandes picos de demanda sin que disminuya el rendimiento ni sea necesario un aprovisionamiento excesivo.

Puede detener un clúster de Neptune durante un máximo de siete días.

Neptune también admite el escalado automático para ajustar automáticamente los tamaños de las instancias de lector en función de la carga de trabajo.

Con la característica de base de datos global de Neptune, puede duplicar un clúster en otras cinco regiones como máximo.

Neptune también ofrece tolerancia a errores por diseño:

- El volumen del clúster que proporciona almacenamiento de datos a todas las instancias del clúster abarca varias zonas de disponibilidad (AZ) en una única Región de AWS. Cada zona de disponibilidad incluye una copia completa de los datos del clúster.
- Si la instancia principal deja de estar disponible, Neptune realiza automáticamente la conmutación por error a una réplica de lectura existente sin pérdida de datos, normalmente en menos de 30 segundos. Si no hay réplicas de lectura en el clúster, Neptune aprovisiona automáticamente una nueva instancia principal, una vez más, sin pérdida de datos.

Lo que todo esto significa es que, al migrar de un clúster causal de Neo4j a Neptune, no es necesario diseñar la topología del clúster de forma explícita para lograr alta disponibilidad y durabilidad de los datos. Esto le permite ajustar el tamaño del clúster a las cargas de trabajo de lectura y escritura previstas y a cualquier requisito de mayor disponibilidad que pueda tener de varias maneras:

- Para escalar las operaciones de lectura, añada instancias de réplica de lectura o habilite la funcionalidad Neptune sin servidor.
- Para mejorar la disponibilidad, distribuya las réplicas de lectura y la instancia principal del clúster entre varias de zonas de disponibilidad (AZ).
- Para reducir el tiempo de conmutación por error, aprovisione al menos una instancia de réplica de lectura que pueda servir como destino de conmutación por error para la principal. Puede determinar el orden en que se promueven las instancias de réplicas de lectura a la principal tras un error mediante la asignación de una prioridad a cada réplica. Una práctica recomendada es asegurarse que un destino de conmutación por error tenga una clase de instancia capaz de gestionar la carga de trabajo de escritura de la aplicación si se promociona a principal.

Diferencias de almacenamiento de datos entre Neptune y Neo4j

Neptune utiliza un modelo de datos de gráficos basado en un modelo cuádruple nativo. Al migrar los datos a Neptune, hay varias diferencias en la arquitectura del modelo de datos y la capa

de almacenamiento que debe tener en cuenta para hacer un uso óptimo del almacenamiento compartido distribuido y escalable que proporciona Neptune:

- Neptune no utiliza ningún esquema ni restricciones definidos de forma explícita. Permite añadir nodos, bordes y propiedades de forma dinámica sin tener que definir el esquema con antelación. Neptune no limita los valores ni los tipos de datos almacenados, salvo en los casos indicados en Límites de Neptune. Como parte de la arquitectura de almacenamiento de Neptune, los datos también se indexan automáticamente de forma que se gestionan muchos de los patrones de acceso más comunes. Esta arquitectura de almacenamiento elimina la sobrecarga operativa que supone la creación y administración del esquema de la base de datos y la optimización de índices.
- Neptune proporciona una arquitectura única de almacenamiento distribuido y compartido que se reduce horizontalmente de forma automática en fragmentos de 10 GB a medida que aumentan las necesidades de almacenamiento de la base de datos, hasta 128 tebibytes (TiB). Esta capa de almacenamiento es fiable, duradera y tolerante a errores, y los datos se copian seis veces, dos veces en cada una de las tres zonas de disponibilidad. Proporciona a todos los clústeres de Neptune una capa de almacenamiento de datos de alta disponibilidad y tolerante a errores de forma predeterminada. La arquitectura de almacenamiento de Neptune reduce los costos y elimina la necesidad de aprovisionar o sobreaprovisionar almacenamiento para gestionar el crecimiento futuro de los datos.

Antes de migrar los datos a Neptune, le recomendamos que se familiarice con el modelo de datos de gráficos de propiedades de Neptune y la semántica de transacciones.

Diferencias operativas entre Neptune y Neo4j

Neptune es un servicio totalmente gestionado que automatiza muchas de las tareas operativas normales que tendría que realizar al utilizar bases de datos autoadministradas o en las instalaciones, como Neo4j Enterprise o Community Edition:

- Copias de seguridad automatizadas: Neptune hace copias de seguridad del volumen del clúster automáticamente y las conserva durante el periodo de retención que se especifique (de 1 a 35 días). Estas copias de seguridad son continuas e incrementales para que se pueda restaurar con rapidez en cualquier momento durante el periodo de retención. No se produce ningún impacto en el desempeño ni ninguna interrupción del servicio de base de datos durante la escritura de los datos de copia de seguridad.
- Instantáneas manuales: Neptune le permite crear una instantánea del volumen de almacenamiento del clúster de base de datos para crear una copia de seguridad de todo el clúster de base de

datos. Este tipo de instantánea se puede usar luego para restaurar la base de datos, hacer una copia de la misma y compartirla entre cuentas.

- Clonación: Neptune admite una característica de clonación que le permite crear rápidamente clones rentables de una base de datos. Los clones utilizan un protocolo de copia en escritura que requiere un espacio adicional mínimo una vez creados. La clonación de bases de datos es una forma eficaz de probar nuevas características o actualizaciones de Neptune sin interrumpir el clúster de origen.
- Monitorización: Neptune proporciona varios métodos para monitorizar el rendimiento y el uso del clúster, entre los que se incluyen:
  - Estado de la instancia
  - Integración con Amazon CloudWatch y AWS CloudTrail
  - Capacidades de registro de auditoría
  - Notificaciones de eventos
  - Etiquetado
- Seguridad: Neptune proporciona un entorno seguro de forma predeterminada. Un clúster reside dentro de una VPC privada que aísla la red de otros recursos. Todo el tráfico se cifra mediante SSL y todos los datos se cifran en reposo mediante AWS KMS.

  Además, Neptune se integra con AWS Identity and Access Management (IAM) para proporcionar autenticación. Al especificar las claves de condición de IAM, puede utilizar las políticas de IAM para proporcionar un control de acceso detallado sobre las acciones de datos.

Diferencias de herramientas e integración entre Neptune y Neo4j

Neptune tiene una arquitectura de integraciones y herramientas diferente a la de Neo4j, lo que puede afectar a la arquitectura de la aplicación. Neptune usa los recursos informáticos del clúster para procesar las consultas, pero saca partido de otros servicios de AWS de primera clase para funciones como la búsqueda de texto completo (con OpenSearch), ETL (con Glue), etc. Para obtener una lista completa de estas integraciones, consulte Integraciones de Neptune.

Aprovisionamiento de la infraestructura al migrar de Neo4j a Neptune

Los clústeres de Amazon Neptune se han diseñado para reducirse horizontalmente en tres dimensiones: almacenamiento, capacidad de escritura y capacidad de lectura. En las siguientes secciones se analizan las opciones específicas que se deben tener en cuenta al llevar a cabo la migración.

Aprovisionamiento del almacenamiento

El almacenamiento de cualquier clúster de Neptune se aprovisiona automáticamente, sin ningún tipo de sobrecarga administrativa por su parte. Cambia el tamaño de forma dinámica en fragmentos de 10 GB a medida que aumentan las necesidades de almacenamiento del clúster. Como resultado, no hay necesidad de estimar, aprovisionar ni sobreaprovisionar el almacenamiento para gestionar el futuro crecimiento de los datos.

Aprovisionamiento de la capacidad de escritura

Neptune proporciona una instancia única de escritor que se puede escalar verticalmente a cualquier tamaño de instancia disponible en la página de precios de Neptune. Al leer y escribir los datos en una instancia de escritor, todas las transacciones cumplen con las normas ACID, con el aislamiento de datos, tal y como se define en Niveles de aislamiento de transacciones en Neptune.

Para elegir un tamaño óptimo para una instancia de escritor, es necesario ejecutar pruebas de carga para determinar el tamaño de instancia óptimo para la carga de trabajo. Para cambiar el tamaño de cualquier instancia de Neptune en cualquier momento, modifique la clase de instancia de base de datos. Puede estimar el tamaño de una instancia inicial en función de la simultaneidad y la latencia media de las consultas, tal y como se describe a continuación en Estimación del tamaño óptimo de la instancia al aprovisionar el clúster.

Aprovisionamiento de la capacidad de lectura

Neptune se ha diseñado para escalar las instancias de lectura de réplica tanto horizontalmente, añadiendo hasta 15 de ellas dentro de un clúster (o más en una base de datos global de Neptune), como verticalmente a cualquier tamaño de instancia disponible en lapágina de precios de Neptune. Todas las instancias de réplica de lectura de Neptune utilizan el mismo volumen de almacenamiento subyacente, lo que permite la replicación transparente de los datos con un retraso mínimo.

Además de permitir el escalado horizontal de las solicitudes de lectura dentro de un clúster de Neptune, las réplicas de lectura también actúan como objetivos de conmutación por error para

la instancia de escritor con el fin de que haya alta disponibilidad. Consulte Directrices operativas básicas de Amazon Neptune para obtener sugerencias sobre cómo determinar el número y la ubicación adecuados de las réplicas de lectura en el clúster.

Para las aplicaciones en las que la conectividad y la carga de trabajo son impredecibles, Neptune también admite una característica de escalado automático que puede ajustar automáticamente el número de réplicas de Neptune en función de los criterios que especifique.

Para determinar el tamaño y la cantidad óptimos de instancias de réplica de lectura, es necesario ejecutar pruebas de carga para determinar las características de la carga de trabajo de lectura que deben admitir. Para cambiar el tamaño de cualquier instancia de Neptune en cualquier momento, modifique la clase de instancia de base de datos. Puede estimar el tamaño de una instancia inicial en función de la simultaneidad y la latencia media de las consultas, tal y como se describe en la siguiente sección.

Uso de Neptune sin servidor para escalar las instancias de lector y escritor automáticamente según sea necesario

Si bien a menudo resulta útil poder estimar la capacidad de cómputo que requerirán las cargas de trabajo previstas, puede configurar la característica Neptune sin servidor para escalar y reducir verticalmente de forma automática la capacidad de lectura y escritura. Esto puede ayudarle a hacer frente a los requisitos de máxima demanda y, al mismo tiempo, reducirla automáticamente cuando la demanda disminuya.

Estimación del tamaño óptimo de la instancia al aprovisionar el clúster

La estimación del tamaño óptimo de la instancia requiere conocer la latencia media de consultas en Neptune, cuando se ejecuta la carga de trabajo, así como el número de consultas simultáneas que se estén procesando. Para calcular una estimación aproximada del tamaño de la instancia, multiplique la latencia media de las consultas por el número de consultas simultáneas. Esto le proporciona el número medio de subprocesos simultáneos necesarios para gestionar la carga de trabajo.

Cada vCPU de una instancia de Neptune puede admitir dos subprocesos de consulta simultáneos, por lo que dividir los subprocesos entre dos proporciona la cantidad de vCPU necesarias, que luego se puede correlacionar con el tamaño de instancia adecuado en la página de precios de Neptune. Por ejemplo:

```
Average Query Latency:         30ms (0.03s)
```

```
Number of concurrent queries:  1000/second

Number of threads needed:      0.03 x 1000 = 30 threads
Number of vCPUs needed:        30 / 2 = 15 vCPUs
```

Al correlacionar esto con la cantidad de vCPU en una instancia, obtenemos una estimación aproximada de que `r5.4xlarge` sería la instancia recomendada para intentar esta carga de trabajo. Esta estimación es aproximada y solo pretende proporcionar una orientación inicial sobre la selección del tamaño de la instancia. Cualquier solicitud debe someterse a un ejercicio de ajuste de tamaño adecuado para determinar el número y los tipos de instancias adecuados para la carga de trabajo.

También se deben tener en cuenta los requisitos de memoria, así como los requisitos de procesamiento. Neptune es más eficaz cuando los datos a los que acceden las consultas están disponibles en la caché del grupo de búferes de la memoria principal. El aprovisionamiento de suficiente memoria también puede reducir considerablemente los costos de E/S.

Puede encontrar información e instrucciones adicionales sobre el tamaño de las instancias en un clúster de Neptune en la página Dimensionamiento de las instancias de base de datos en un clúster de base de datos de Neptune.

Migración de datos de Neo4j a Neptune

Al realizar una migración de Neo4j a Amazon Neptune, la migración de los datos es un paso importante del proceso. Existen varios métodos para migrar datos. El método correcto viene determinado por las necesidades de la aplicación, el tamaño de los datos y el tipo de migración deseado. Sin embargo, muchas de estas migraciones requieren una evaluación de las mismas consideraciones, algunas de las cuales se destacan a continuación.

> **Note**
>
> Consulte Migración de una base de datos de gráficos de Neo4j a Neptune con una utilidad totalmente automatizada en el blog de bases de datos de AWS para obtener una explicación completa paso a paso de un ejemplo de cómo realizar una migración de datos sin conexión.

Evaluación de la migración de datos de Neo4j a Neptune

El primer paso a la hora de evaluar cualquier migración de datos es determinar cómo se van a migrar los datos. Las opciones dependen de la arquitectura de la aplicación que se va a migrar, del tamaño de los datos y de las necesidades de disponibilidad durante la migración. Por lo general, las migraciones suelen clasificarse en una de las siguientes dos categorías: en línea o sin conexión.

Las migraciones sin conexión suelen ser las más sencillas de realizar, ya que la aplicación no acepta tráfico de lectura o escritura durante la migración. Cuando la aplicación deja de aceptar tráfico, los datos se pueden exportar, optimizar e importar, y la aplicación se puede probar antes de volver a habilitarla.

Las migraciones en línea son más complejas, ya que la aplicación aún necesita aceptar el tráfico de lectura y escritura mientras se migran los datos. Las necesidades exactas de cada migración en línea pueden diferir, pero la arquitectura general suele ser similar a la siguiente:

- Es necesario habilitar en Neo4j la información sobre los cambios continuos realizados en la base de datos. Para ello, configure Neo4j Streams como fuente de un clúster de Kafka.
- Una vez hecho esto, se puede realizar una exportación del sistema en ejecución siguiendo las instrucciones de Exportación de datos de Neo4j al migrar a Neptune y el tiempo indicado para luego correlacionarlo con el tema de Kafka.
- A continuación, los datos exportados se importan a Neptune, siguiendo las instrucciones de Importación de datos de Neo4j al migrar a Neptune.

- Después, los datos modificados de la transmisión de Kafka se pueden copiar al clúster de Neptune mediante una arquitectura similar a la descrita en Escribir en Amazon Neptune desde Amazon Kinesis Data Streams. Tenga en cuenta que la replicación de los cambios se puede ejecutar en paralelo para validar la nueva arquitectura y el rendimiento de la aplicación.
- Una vez validada la migración de datos, el tráfico de la aplicación se puede redirigir al clúster de Neptune y se puede retirar la instancia de Neo4j.

Optimizaciones de modelos de datos para migrar de Neo4j a Neptune

Tanto Neptune como Neo4j admiten gráficos de propiedades etiquetadas (LPG). Sin embargo, Neptune presenta algunas diferencias en la arquitectura y el modelo de datos que puede sacar partido para optimizar el rendimiento:

Optimización de los identificadores de nodos y bordes

Neo4j genera automáticamente identificadores numéricos largos. Con Cypher, puede hacer referencia a los nodos según su ID, pero, en general, se recomienda buscar los nodos mediante una propiedad indexada.

Neptune le permite proporcionar sus propios identificadores basados en cadenas para vértices y bordes. Si no proporciona sus propios identificadores, Neptune genera automáticamente representaciones en cadena de los UUID para los nuevos vértices y bordes.

Si migra datos de Neo4j a Neptune exportándolos desde Neo4j y luego importándolos de forma masiva a Neptune, puede conservar los ID de Neo4j. Los valores numéricos generados por Neo4j pueden actuar como identificadores proporcionados por el usuario al importarlos a Neptune, donde se representan como cadenas en lugar de valores numéricos.

Sin embargo, hay circunstancias en las que es posible que desee promover una propiedad de vértice para que se convierta en un ID de vértice. Así como buscar un nodo con una propiedad indexada es la forma más rápida de encontrar un nodo en Neo4j, buscar un vértice según el ID es la forma más rápida de encontrar un vértice en Neptune. Por lo tanto, si puede identificar una propiedad de vértice adecuada que incluya valores únicos, debe plantearse reemplazar el valor ~id del vértice por el valor de propiedad designado en los archivos CSV de carga masiva. Si lo hace, también tendrá que volver a escribir los valores de las bordes ~from y ~to correspondientes en los archivos CSV.

Restricciones del esquema al migrar datos de Neo4j a Neptune

En Neptune, la única restricción de esquema disponible es la exclusividad del ID de un nodo o un borde. Se recomienda a las aplicaciones que necesiten sacar partido de una restricción de

exclusividad que utilicen este método para lograr una restricción de exclusividad especificando el identificador del nodo o del borde. Si la aplicación utilizó varias columnas como restricción de exclusividad, el ID puede configurarse en una combinación de estos valores. Por ejemplo, `id=123, code='SEA'` podría representarse como `ID='123_SEA'` para lograr una restricción de exclusividad compleja.

Optimización de la dirección de los bordes al migrar datos de Neo4j a Neptune

Cuando se añaden nodos, bordes o propiedades a Neptune, se indexan automáticamente de tres formas diferentes, con un cuarto índice opcional. Debido a la forma en que Neptune crea y usa los índices, las consultas que siguen los bordes salientes son más eficaces que las que usan los bordes entrantes. En cuanto al modelo de almacenamiento de datos de gráficos de Neptune, se trata de búsquedas basadas en temas que utilizan el índice SPOG.

Si, al migrar el modelo de datos y las consultas a Neptune, descubre que las consultas más importantes se basan en atravesar los bordes entrantes donde hay un alto grado de distribución, puede plantearse modificar el modelo para que estos recorridos sigan los bordes salientes, especialmente si no puede especificar las etiquetas de borde que desea atravesar. Para ello, invierta la dirección de los bordes relevantes y actualice las etiquetas de los bordes para que reflejen la semántica de este cambio de dirección. Por ejemplo, podría cambiar:

```
person_A — parent_of — person_B
   to:
person_B — child_of — person_A
```

Para realizar este cambio en un archivo CSV de borde de carga masiva, basta con cambiar los encabezados de las columnas `~from` y `~to` y actualizar los valores de la columna `~label`.

Como alternativa a la reversión de la dirección de los bordes, puede habilitar un cuarto índice de Neptune, el índice OSGP, que hace que el proceso de atravesar los bordes entrantes o las búsquedas basadas en objetos sean mucho más eficaces. Sin embargo, si se habilita este cuarto índice, se reducirán las tasas de inserción y se necesitará más espacio de almacenamiento.

Optimización del filtrado al migrar datos de Neo4j a Neptune

Neptune está optimizado para funcionar mejor cuando las propiedades se filtran a la propiedad más selectiva disponible. Cuando se utilizan varios filtros, se busca el conjunto de elementos que coincidan entre ellos y, a continuación, se calcula la superposición de todos estos conjuntos. Cuando es posible, la combinación de varias propiedades en una única propiedad minimiza el número de búsquedas en el índice y reduce la latencia de una consulta.

Por ejemplo, esta consulta utiliza dos búsquedas de índices y una combinación:

```
MATCH (n) WHERE n.first_name='John' AND n.last_name='Doe' RETURN n
```

Esta consulta recupera la misma información mediante una única búsqueda de índice:

```
MATCH (n) WHERE n.name='John Doe' RETURN n
```

Neptune admite tipos de datos diferentes a los de Neo4j.

Mapeos de tipos de datos de Neo4j a tipos de datos compatibles con Neptune

- Lógica: `Boolean`

  Asigne este valor en Neptune a `Bool` o `Boolean`.
- Numérica: `Number`

  Asigne este valor en Neptune al más estrecho de los siguientes tipos de openCypher de Neptune que puedan admitir todos los valores de la propiedad numérica en cuestión:

  ```
  Byte
  Short
  Integer
  Long
  Float
  Double
  ```
- Texto: `String`

  Asigne este valor en Neptune a `String`.
- Momento específico:

  ```
  Date
  Time
  LocalTime
  DateTime
  LocalDateTime
  ```

  Asigne estos valores en Neptune a `Date` como UTC mediante uno de los siguientes formatos ISO-8601 compatibles con Neptune:

```
yyyy-MM-dd
yyyy-MM-ddTHH:mm
yyyy-MM-ddTHH:mm:ss
yyyy-MM-ddTHH:mm:ssZ
```

- Duración: `Duration`

  Asigne este valor en Neptune a un valor numérico para la aritmética de fechas, si es necesario.

- Espacial: `Point`

  Asigne este valor en Neptune a los valores numéricos de los componentes, cada uno de los cuales se convierte en una propiedad independiente, o expréselo como un valor de cadena para que lo interprete la aplicación cliente. Tenga en cuenta que la integración de búsqueda de texto completo de Neptune mediante OpenSearch le permite indexar las propiedades de geolocalización.

Migración de propiedades multivalor de Neo4j a Neptune

Neo4j permite almacenar listas homogéneas de tipos sencillas como propiedades tanto de los nodos como de los bordes. Estas listas pueden incluir valores duplicados.

Sin embargo, Neptune solo permite una cardinalidad en conjuntos o única para las propiedades de los vértices y una cardinalidad única para las propiedades de los bordes en los datos de gráficos de propiedades. Como resultado, no existe una migración directa de las propiedades de la lista de nodos de Neo4j que incluyan valores duplicados en las propiedades de los vértices de Neptune, ni de las propiedades de la lista de relaciones de Neo4j en las propiedades de los bordes de Neptune.

Algunas posibles estrategias para migrar propiedades de nodos multivalor de Neo4j con valores duplicados a Neptune son las siguientes:

- Descarte los valores duplicados y convierta la propiedad del nodo multivalor de Neo4j en una propiedad de vértice de Neptune de cardinalidad en conjuntos. Tenga en cuenta que es posible que el conjunto de Neptune no refleje el orden de los elementos de la propiedad multivalor original de Neo4j.
- Convierta la propiedad de nodo multivalor de Neo4j en una representación de cadena de una lista con formato JSON en una propiedad de cadena de vértices de Neptune.
- Extraiga cada uno de los valores de propiedad multivalor en un vértice independiente con una propiedad de valor y conecte esos vértices al vértice principal mediante un borde etiquetado con el nombre de la propiedad.

Asimismo, algunas posibles estrategias para migrar propiedades de relaciones multivalor de Neo4j en Neptune son las siguientes:

- Convierta la propiedad de relación multivalor de Neo4j en una representación de cadena de una lista con formato JSON en una propiedad de cadena de bordes de Neptune.
- Refactorice la relación Neo4j en los bordes entrantes y salientes de Neptune asociados a un vértice intermedio. Extraiga cada uno de los valores de propiedad de relación multivalor en un vértice independiente con una propiedad de valor y conecte esos vértices al vértice intermedio mediante un borde etiquetado con el nombre de la propiedad.

Tenga en cuenta que la representación en cadena de una lista con formato JSON es opaca para el lenguaje de consultas de openCypher, aunque openCypher incluye un predicado `CONTAINS` que permite realizar búsquedas sencillas dentro de los valores de cadena.

Exportación de datos de Neo4j al migrar a Neptune

Al exportar datos de Neo4j, utilice los procedimientos de APOC para exportar a CSV o a GraphML. Aunque es posible exportar a otros formatos, existen herramientas de código abierto para convertir los datos CSV exportados desde Neo4j al formato de carga masiva Neptune, y también herramientas de código abierto para convertir los datos de GraphML exportados de Neo4j al formato de carga masiva de Neptune.

También puede exportar los datos directamente a Amazon S3 mediante los distintos procedimientos de APOC. La exportación a un bucket de Amazon S3 está deshabilitada de forma predeterminada, pero se puede habilitar mediante los procedimientos destacados en Exportación a Amazon S3 en la documentación sobre APOC de Neo4j.

Importación de datos de Neo4j al migrar a Neptune

Puede importar datos a Neptune mediante el programa de carga masiva de Neptune o mediante la lógica de la aplicación en un lenguaje de consulta compatible, como, por ejemplo, openCypher.

El programa de carga masiva de Neptune es el método preferido para importar grandes cantidades de datos, ya que proporciona un rendimiento de importación optimizado si se siguen las prácticas recomendadas. El programa de carga masiva admite dos formatos CSV diferentes, a los que los datos exportados desde Neo4j se pueden convertir con las utilidades de código abierto mencionadas anteriormente en la sección Exportación de datos.

También puede usar openCypher para importar datos con una lógica personalizada con el fin de analizarlos, transformarlos e importarlos. Puede enviar las consultas de openCypher a través del punto de conexión de HTTPS (lo que se recomienda) o mediante el controlador de Bolt.

Migración de aplicaciones de Neo4j a Neptune

Una vez que haya migrado los datos de Neo4j a Neptune, el siguiente paso es migrar la propia aplicación. Al igual que con los datos, existen varios métodos para migrar la aplicación en función de las herramientas que utilice, los requisitos, las diferencias arquitectónicas, etc. A continuación, se describen los aspectos que normalmente es necesario tener en cuenta en este proceso.

Migración de conexiones al migrar de Neo4j a Neptune

Si actualmente no utiliza los controladores Bolt, o si desea utilizar una alternativa, puede conectarse al punto de conexión de HTTPS, que proporciona acceso total a los datos devueltos.

Si tiene una aplicación que usa el protocolo Bolt, puede migrar estas conexiones a Neptune y permitir que las aplicaciones se conecten con los mismos controladores que utilizó en Neo4j. Para conectarse a Neptune, es posible que deba realizar uno o varios de estos cambios en la aplicación:

- Será necesario actualizar la URL y el puerto para usar los puntos de conexión y el puerto del clúster (el valor predeterminado es 8182).
- Neptune requiere que todas las conexiones usen SSL, por lo que debe especificar que cada conexión esté cifrada.
- Neptune administra la autenticación mediante la asignación de roles y políticas de IAM. Los roles y políticas de IAM proporcionan un nivel extremadamente flexible de administración de usuarios dentro de la aplicación, por lo que es importante leer y comprender la Información general de IAM antes de configurar el clúster.
- Las conexiones de Bolt se comportan de manera diferente en Neptune que en Neo4j de varias maneras, tal y como se explica en Comportamiento de conexión de Bolt en Neptune.
- Puede encontrar más información y sugerencias en Prácticas recomendadas de Neptune con openCypher y Bolt.

Hay ejemplos de código de lenguajes de uso común, como Java, Python, .NET y NodeJS, y para escenarios de conexión, como, por ejemplo, el uso de la autenticación de IAM, en Uso del protocolo Bolt para realizar consultas de openCypher a Neptune.

Enrutamiento de consultas a instancias de clúster al pasar de Neo4j a Neptune

Las aplicaciones cliente de Neo4j utilizan un controlador de enrutamiento y especifican un modo de acceso para enrutar las solicitudes de lectura y escritura a un servidor adecuado en un clúster causal.

Al migrar una aplicación cliente a Neptune, utilice los puntos de conexión de Neptune para enrutar las consultas de forma eficaz a una instancia adecuada del clúster:

- Todas las conexiones a Neptune deben utilizar `bolt://` en lugar de `bolt+routing://` o `neo4j://` en la URL.
- El punto de conexión del clúster se conecta a la instancia principal actual del clúster. Utilice el punto de conexión del clúster para dirigir las solicitudes de escritura al principal.
- El punto de conexión del lector distribuye las conexiones entre instancias de lectura de réplica del clúster. Si tiene un clúster de instancia única sin instancias de réplica de lectura, el punto de conexión del lector se conecta a la instancia principal, que admite operaciones de escritura. Si el clúster incluye una o varias instancias de réplica de lectura, el envío de una solicitud de escritura al punto de conexión del lector genera una excepción.
- Cada instancia del clúster también puede tener su propio punto de conexión de instancia. Utilice un punto de conexión de instancia si la aplicación cliente necesita enviar una solicitud a una instancia específica del clúster.

Para obtener más información, consulte Consideraciones sobre puntos de conexión de Neptune.

Coherencia de datos en Neptune

Cuando se utilizan clústeres causales de Neo4j, las réplicas de lectura acaban siendo coherentes con los servidores principales, pero las aplicaciones cliente pueden garantizar la coherencia causal mediante el encadenamiento causal. El encadenamiento causal implica pasar marcadores entre transacciones, lo que permite a una aplicación cliente escribir en un servidor principal y, a continuación, leer su propia escritura desde una réplica de lectura.

En Neptune, las instancias de lectura de réplica son finalmente coherentes con las del escritor, con un retraso de réplica que suele ser inferior a 100 milisegundos. Sin embargo, hasta que se haya replicado un cambio, las actualizaciones de los bordes y vértices existentes y las adiciones de bordes y vértices nuevos no estarán visibles en una instancia de réplica. Por lo tanto, si la aplicación necesita coherencia inmediata en Neptune al leer cada escritura, utilice el punto de conexión del clúster para la operación de lectura después de escritura. Esta es la única vez que se puede usar el punto de conexión del clúster para operaciones de lectura. En todas las demás circunstancias, utilice el punto de conexión del lector para las lecturas.

Migración de consultas de Neo4j a Neptune

Si bien la compatibilidad con openCypher de Neptune reduce drásticamente la cantidad de trabajo necesaria para migrar las consultas desde Neo4j, aún hay que evaluar algunas diferencias a la hora de migrar:

- Tal y como se ha mencionado anteriormente en Optimizaciones de modelos de datos, es posible que deba realizar modificaciones en el modelo de datos para crear un modelo de datos de gráficos optimizado para Neptune, lo que a su vez requerirá cambios en sus consultas y pruebas.
- Neo4j ofrece una variedad de extensiones de lenguaje específicas de Cypher que no se incluyen en la especificación de openCypher implementada por Neptune. Según el caso de uso y la característica utilizada, puede haber soluciones alternativas en el lenguaje de openCypher, en el lenguaje de Gremlin o mediante otros mecanismos, tal y como se describe en Reescritura de consultas de Cypher para ejecutarlas en openCypher en Neptune.
- Las aplicaciones suelen utilizar otros componentes de middleware para interactuar con la base de datos en lugar de los propios controladores Bolt. Eche un vistazo a Compatibilidad de Neptune con Neo4j para ver si las herramientas o el middleware que está utilizando son compatibles.
- En el caso de una conmutación por error, es posible que el controlador Bolt siga conectándose a la instancia anterior de escritor o lector, ya que el punto de conexión del clúster proporcionado a la conexión se ha resuelto en una dirección IP. Una gestión adecuada de los errores en la aplicación debería solucionar este problema, tal y como se describe en Cree una nueva conexión después de una conmutación por error.
- Cuando las transacciones se cancelan debido a conflictos que no se pueden resolver o a tiempos de espera de bloqueo, Neptune responde con un `ConcurrentModificationException`. Para obtener más información, consulte Códigos de error del motor. Como práctica recomendada, los clientes siempre deben detectar y gestionar estas excepciones.

  En ocasiones, se produce una `ConcurrentModificationException` cuando varios subprocesos o varias aplicaciones escriben en el sistema de forma simultánea. Dados los niveles de aislamiento de las transacciones, estos conflictos a veces pueden ser inevitables.
- Neptune permite ejecutar tanto consultas de Gremlin como de openCypher en los mismos datos. Esto significa que, en algunos casos, puede que tenga que plantearse utilizar Gremlin, con sus capacidades de consulta más potentes, para realizar algunas de las funciones de sus consultas.

Tal y como se ha explicado anteriormente en Aprovisionamiento de la infraestructura, cada aplicación debe realizar un ejercicio de dimensionamiento adecuado para garantizar que la cantidad de

instancias, los tamaños de las instancias y la topología del clúster estén optimizados para la carga de trabajo específica de la aplicación.

Los aspectos que se tratan aquí para migrar la aplicación son los más comunes, pero esta no es una lista exhaustiva. Cada aplicación es única. Si tiene más preguntas, póngase en contacto con AWS Support o con el equipo de cuentas.

Migración de características y herramientas específicas de Neo4j

Neo4j tiene una variedad de características y complementos personalizados con una funcionalidad en la que la aplicación puede confiar. Al evaluar la necesidad de migrar esta funcionalidad, suele ser útil investigar si existe un método mejor en AWS para lograr el mismo objetivo. Teniendo en cuenta las diferencias arquitectónicas entre Neo4j y Neptune, a menudo se pueden encontrar alternativas eficaces que saquen partido de otros servicios de AWS o integraciones.

Consulte Compatibilidad de Neptune con Neo4j para obtener una lista de las características específicas de Neo4j y las soluciones alternativas sugeridas.

Compatibilidad de Neptune con Neo4j

Neo4j se basa en un enfoque integral arquitectónico, en el que la carga de datos, ETL de datos, las consultas a las aplicaciones, el almacenamiento de datos y las operaciones de administración se realizan en el mismo conjunto de recursos informáticos, como, por ejemplo, las instancias EC2. Amazon Neptune es una base de datos de gráficos de especificaciones abiertas y centrada en OLTP en la que la arquitectura separa las operaciones y desacopla los recursos para que puedan escalarse de forma dinámica.

Neo4j incluye diversas características y herramientas, incluidas herramientas de terceros, que no forman parte de la especificación de openCypher, son incompatibles con openCypher o son incompatibles con la implementación de openCypher por Neptune. A continuación, veremos algunas de las más comunes.

Características específicas de Neo4j que no están presentes en Neptune

- **`LOAD CSV`**: Neptune tiene un enfoque arquitectónico diferente al de Neo4j para cargar datos. Para permitir una mejor escalabilidad y la optimización de los costos, Neptune implementa una separación de las preocupaciones en torno a los recursos y recomienda utilizar una de las integraciones de servicios de AWS, como, por ejemplo, AWS Glue para realizar los procesos de ETL necesarios para preparar los datos en un formato compatible con el programa de carga masiva Neptune.

  Otra opción es hacer lo mismo con código de aplicación que se ejecute en recursos informáticos de AWS, como instancias de Amazon EC2, funciones de Lambda, Amazon Elastic Container Service, tareas AWS Batch, etc. El código puede usar tanto el punto de conexión de HTTPS de Neptune como el punto de conexión de Bolt.
- Control de acceso detallado: Neptune admite un control de acceso granular sobre las acciones de acceso a los datos mediante claves de condición de IAM. Se puede implementar un control de acceso más detallado en la capa de aplicación.
- Neo4j Fabric: Neptune admite la federación de consultas entre bases de datos para cargas de trabajo de RDF mediante la palabra clave `SERVICE` de SPARQL. Dado que actualmente no existe un estándar o especificación abiertos para la federación de consultas para las cargas de trabajo de gráficos de propiedades, sería necesario implementar esa funcionalidad en la capa de aplicación.
- Control de acceso basado en roles (RBAC): Neptune administra la autenticación mediante la asignación de roles y políticas de IAM. Los roles y políticas de IAM proporcionan un nivel

extremadamente flexible de administración de usuarios dentro de la aplicación, por lo que es importante leer y comprender la Información general de IAM antes de configurar el clúster.

- Marcadores: los clústeres de Neptune constan de una sola instancia de escritor y hasta 15 instancias de réplica de lectura. Los datos escritos en la instancia de escritos son compatibles con ACID y ofrecen una sólida garantía de coherencia en las lecturas posteriores. Las réplicas de lectura utilizan el mismo volumen de almacenamiento que la instancia de escritor y, en última instancia, son coherentes, normalmente en menos de 100 ms desde el momento en que se escriben los datos. Si su caso de uso requiere garantizar inmediatamente la coherencia de lectura de las nuevas escrituras, estas lecturas deben dirigirse al punto de conexión del clúster en lugar de al punto de conexión del lector.
- Procedimientos de APOC: dado que los procedimientos de APOC no están incluidos en la especificación de openCypher, Neptune no proporciona soporte directo para procedimientos externos. En cambio, Neptune se basa en integraciones con otros servicios de AWS para lograr una funcionalidad similar para el usuario final de una manera escalable, segura y sólida. A veces, los procedimientos de APOC se pueden volver a escribir en openCypher o Gremlin, y algunos no son relevantes para las aplicaciones de Neptune.

  Por lo general, los procedimientos de APOC se clasifican en las siguientes categorías:

  - Importación: Neptune admite la importación de datos mediante una variedad de formatos mediante lenguajes de consulta, el programa de carga masiva de Neptune o como destino de AWS Database Migration Service. Las operaciones de ETL en los datos se pueden realizar mediante AWS Glue y el paquete de código abierto de `neptune-python-utils`.
  - Exportación: Neptune admite la exportación de datos mediante la utilidad `neptune-export`, que admite una variedad de formatos y métodos de exportación comunes.
  - Integración de bases de datos: Neptune admite la integración con otras bases de datos mediante herramientas de ETL, tales como AWS Glue o herramientas de migración como AWS Database Migration Service.
  - Actualizaciones de gráficos: Neptune admite un amplio conjunto de características para actualizar los datos de los gráficos de propiedades mediante su compatibilidad con los lenguajes de consulta de openCypher y Gremlin. Consulte Reescrituras de Cypher para ver algunos ejemplos de reescrituras de procedimientos que se usan habitualmente.
  - Estructuras de datos: Neptune admite un amplio conjunto de características para actualizar los datos de los gráficos de propiedades mediante su compatibilidad con los lenguajes de consulta de openCypher y Gremlin. Consulte Reescrituras de Cypher para ver algunos ejemplos de reescrituras de procedimientos que se usan habitualmente.

  - Temporal (fecha y hora): Neptune admite un amplio conjunto de características para actualizar los datos de los gráficos de propiedades mediante su compatibilidad con los lenguajes de consulta de openCypher y Gremlin. Consulte Reescrituras de Cypher para ver algunos ejemplos de reescrituras de procedimientos que se usan habitualmente.
  - Matemática: Neptune admite un amplio conjunto de características para actualizar los datos de los gráficos de propiedades mediante su compatibilidad con los lenguajes de consulta de openCypher y Gremlin. Consulte Reescrituras de Cypher para ver algunos ejemplos de reescrituras de procedimientos que se usan habitualmente.
  - Consultas avanzadas de gráficos: Neptune admite un amplio conjunto de características para actualizar los datos de los gráficos de propiedades mediante su compatibilidad con los lenguajes de consulta de openCypher y Gremlin. Consulte Reescrituras de Cypher para ver algunos ejemplos de reescrituras de procedimientos que se usan habitualmente.
  - Comparación de gráficos: Neptune admite un amplio conjunto de características para actualizar los datos de los gráficos de propiedades mediante su compatibilidad con los lenguajes de consulta de openCypher y Gremlin. Consulte Reescrituras de Cypher para ver algunos ejemplos de reescrituras de procedimientos que se usan habitualmente.
  - Ejecución de cifrado: Neptune admite un amplio conjunto de características para actualizar los datos de los gráficos de propiedades mediante su compatibilidad con los lenguajes de consulta de openCypher y Gremlin. Consulte Reescrituras de Cypher para ver algunos ejemplos de reescrituras de procedimientos que se usan habitualmente.
- Procedimientos personalizados: Neptune no admite procedimientos personalizados creados por los usuarios. Esta funcionalidad tendría que implementarse en la capa de aplicación.
- Geoespacial: aunque Neptune no ofrezca soporte nativo para las características geoespaciales, se puede lograr una funcionalidad similar mediante la integración con otros servicios de AWS, tal y como se muestra en esta publicación del blog: Combine Amazon Neptune and Amazon OpenSearch Service for geospatial queries (Combinación de Amazon Neptune y Amazon OpenSearch Service para consultas geoespaciales) de Ross Gabay y Abhilash Vinod (1 de febrero de 2022).
- Ciencia de datos de gráficos: Neptune actualmente admite el análisis de gráficos a través de Neptune Analytics, un motor con optimización de memoria que admite una biblioteca de algoritmos de análisis de gráficos.

  Neptune también proporciona una integración con el AWS SDK para Pandas y varios cuadernos de ejemplo que muestran cómo sacar partido de esta integración en los entornos de Python para ejecutar análisis en los datos de gráficos.

- Restricciones de esquema: en Neptune, la única restricción de esquema disponible es la exclusividad del identificador de un nodo o un borde. No hay ninguna característica que especifique alguna restricción de esquema adicional ni ninguna restricción adicional de exclusividad o valor en un elemento del gráfico. Los valores de ID en Neptune son cadenas y se pueden configurar con Gremlin, de la siguiente manera:

  ```
  g.addV('person').property(id, '1') )
  ```

  Se recomienda a las aplicaciones que necesiten sacar partido del ID como una restricción de exclusividad que utilicen este método para lograr una restricción de exclusividad. Si la aplicación utilizó varias columnas como restricción de exclusividad, el ID puede configurarse en una combinación de estos valores. Por ejemplo, `id=123, code='SEA'` podría representarse como `ID='123_SEA'` para lograr una restricción de exclusividad compleja.
- Varios inquilinos: Neptune solo admite un gráfico por clúster. Para crear un sistema de varios inquilinos con Neptune, utilice varios clústeres o divida los inquilinos de forma lógica en un único gráfico y utilice la lógica del lado de la aplicación para reforzar la separación. Por ejemplo, añada una propiedad `tenantId` e inclúyala en cada consulta, como, por ejemplo:

  ```
  MATCH p=(n {tenantId:1})-[]->({tenantId:1}) RETURN p LIMIT 5)
  ```

  Neptune sin servidor hace que sea relativamente sencillo implementar la opción de varios inquilinos mediante varios clústeres de bases de datos, cada uno de los cuales se escala de forma independiente y automática según sea necesario.

Soporte de Neptune para las herramientas de Neo4j

Neptune ofrece las siguientes alternativas a las herramientas de Neo4j:

- Navegador de Neo4j: Neptune proporciona cuadernos de gráficos de código abierto que proporcionan un IDE centrado en el desarrollador para ejecutar consultas y visualizar los resultados.
- Bloom de Neo4j: Neptune admite visualizaciones detalladas de gráficos mediante soluciones de visualización de terceros, como Graph-explorer, Tom Sawyer, Cambridge Intelligence, Graphistry, metaphacts y G.V().
- GraphQL: Neptune actualmente admite GraphQL a través de integraciones de AWS AppSync personalizadas. Consulte la publicación del blog Build a graph application with Amazon Neptune

and AWS Amplify y el proyecto de ejemplo Building Serverless Calorie tracker application with AWS AppSync and Amazon Neptune.

- NeoSemantics: Neptune admite de forma nativa el modelo de datos RDF, por lo que se recomienda a los clientes que deseen ejecutar cargas de trabajo de RDF que utilicen el soporte de modelos RDF de Neptune.
- Arrows.app: el cifrado creado al exportar el modelo mediante el comando export es compatible con Neptune.
- Linkurious Ogma: una integración de ejemplo con Linkurious Ogma está disponible aquí.
- Spring Data Neo4j: por el momento, esta opción no es compatible con Neptune.
- Conector de Spark de Neo4j: el conector de Spark de Neo4j se puede utilizar en un trabajo de Spark para conectarse a Neptune con openCypher. Estos son algunos ejemplos de configuración de códigos y aplicaciones:

  Código de muestra:

```
SparkSession spark = SparkSession
            .builder()
            .config("encryption.enabled", "true")
            .appName("Simple Application").config("spark.master",
 "local").getOrCreate();

Dataset<Row> df = spark.read().format("org.neo4j.spark.DataSource")
            .option("url", "bolt://(your cluster endpoint):8182")
            .option("encryption.enabled", "true")
            .option("query", "MATCH (n:airport) RETURN n")
            .load();

System.out.println("TOTAL RECORD COUNT: " + df.count());
spark.stop();
```

  Configuración de aplicaciones:

```
<dependency>
    <groupId>org.neo4j</groupId>
    <artifactId>neo4j-connector-apache-spark_2.12-4.1.0</artifactId>
    <version>4.0.1_for_spark_3</version>
</dependency>
```

Características y herramientas de Neo4j que no se incluyen aquí

Si utiliza una herramienta o característica que no se incluye aquí, no tenemos la certeza de su compatibilidad con Neptune u otros servicios incluidos en AWS. Si tiene más preguntas, póngase en contacto con AWS Support o con el equipo de cuentas.

Reescritura de consultas de Cypher para ejecutarlas en openCypher en Neptune

El lenguaje de openCypher es un lenguaje de consulta declarativo para gráficos de propiedades que desarrolló originalmente Neo4j y que pasó a ser de código abierto en 2015. Además, contribuyó al proyecto openCypher en virtud de una licencia de código abierto Apache 2. En AWS, creemos que el código abierto es la opción adecuada para todos y estamos comprometidos a llevar el valor del código abierto a nuestros clientes y la excelencia operativa de AWS a las comunidades de código abierto.

La sintaxis de openCypher está documentada en Referencia del lenguaje de consultas (versión 9).

Dado que openCypher incluye un subconjunto de la sintaxis y las características del lenguaje de consultas de Cypher, algunos escenarios de migración requieren reescribir las consultas en formatos compatibles con openCypher o examinar métodos alternativos para lograr la funcionalidad deseada.

En esta sección se incluyen recomendaciones para tratar las diferencias más comunes, pero no es en absoluto exhaustiva. Debe probar minuciosamente cualquier aplicación que utilice estas reescrituras para asegurarse de que los resultados son los previstos.

Reescritura de funciones de predicado **`None`**, **`All`** y **`Any`**

Estas funciones no forman parte de la especificación de openCypher. Se pueden lograr resultados comparables en openCypher con Comprensión de listas.

Por ejemplo, busque todas las rutas que van del nodo `Start` al nodo `End`, pero no se permite que ningún recorrido atraviese un nodo con una propiedad de clase de `D`:

```
# Neo4J Cypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where none(node IN nodes(p) where node.class ='D')
return p

# Neptune openCypher code
match p=(a:Start)-[:HOP*1..]->(z:End)
where size([node IN nodes(p) where node.class = 'D']) = 0
return p
```

La opción Comprensión de listas puede lograr los siguientes resultados:

```
all  => size(list_comprehension(list)) = size(list)
```

```
any  => size(list_comprehension(list)) >= 1
none => size(list_comprehension(list)) = 0
```

Reescritura de la función **`reduce()`** de Cypher en openCypher

La función `reduce()` no forma parte de la especificación de openCypher. Suele utilizarse para crear una agregación de datos a partir de elementos de una lista. En muchos casos, puede utilizar una combinación de Comprensión de listas y la cláusula `UNWIND` para obtener resultados similares.

Por ejemplo, la siguiente consulta de Cypher busca todos los aeropuertos en las rutas que tengan de una a tres paradas entre Anchorage (ANC) y Austin (AUS) y devuelve la distancia total de cada ruta:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
RETURN p, reduce(totalDist=0, r in relationships(p) | totalDist + r.dist) AS totalDist
ORDER BY totalDist LIMIT 5
```

Puede escribir la misma consulta en openCypher para Neptune, como, por ejemplo:

```
MATCH p=(a:airport {code: 'ANC'})-[r:route*1..3]->(z:airport {code: 'AUS'})
UNWIND [i in relationships(p) | i.dist] AS di
RETURN p, sum(di) AS totalDist
ORDER BY totalDist
LIMIT 5
```

Reescritura de la cláusula FOREACH de Cypher en openCypher

La cláusula FOREACH no forma parte de la especificación de openCypher. Suele utilizarse para actualizar los datos en mitad de una consulta, a menudo a partir de agregaciones o elementos de una ruta.

Como ejemplo de ruta, busque todos los aeropuertos de una ruta con no más de dos paradas entre Anchorage (ANC) y Austin (AUS) y establezca una propiedad de visitado en cada una de ellos:

```
# Neo4J Example
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
FOREACH (n IN nodes(p) | SET n.visited = true)

# Neptune openCypher
MATCH p=(:airport {code: 'ANC'})-[*1..2]->({code: 'AUS'})
WITH nodes(p) as airports
UNWIND airports as a
```

```
SET a.visited=true
```

Otro ejemplo:

```
# Neo4J Example
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
FOREACH (n IN nodes(p) | SET n.marked = true)

# Neptune openCypher
MATCH p=(start)-[*]->(finish)
WHERE start.name = 'A' AND finish.name = 'D'
UNWIND nodes(p) AS n
SET n.marked = true
```

Reescritura de los procedimientos de APOC de Neo4j en Neptune

Los ejemplos siguientes utilizan openCypher para reemplazar algunos de los procedimientos de APOC más utilizados. Estos ejemplos son solo de referencia y están destinados a proporcionar algunas sugerencias sobre cómo gestionar situaciones comunes. En la práctica, cada aplicación es diferente y tendrá que idear sus propias estrategias para proporcionar toda la funcionalidad que necesite.

Reescritura de los procedimientos de **`apoc.export`**

Neptune ofrece una variedad de opciones para exportaciones completas basadas en consultas y gráficos en varios formatos de salida, como CSV y JSON, mediante la utilidad neptune-export (consulte Exportación de datos desde un clúster de base de datos de Neptune).

Reescritura de los procedimientos de **`apoc.schema`**

Neptune no tiene esquemas, índices ni restricciones definidos de forma explícita, por lo que muchos procedimientos de `apoc.schema` ya no son necesarios. Algunos ejemplos son:

- `apoc.schema.assert`
- `apoc.schema.node.constraintExists`
- `apoc.schema.node.indexExists`,
- `apoc.schema.relationship.constraintExists`
- `apoc.schema.relationship.indexExists`
- `apoc.schema.nodes`

- `apoc.schema.relationships`

openCypher de Neptune admite la recuperación de valores similares a los de los procedimientos, tal y como se muestra a continuación, pero puede tener problemas de rendimiento en gráficos de mayor tamaño, ya que para ello es necesario escanear una gran parte del gráfico con el fin de obtener la respuesta.

```
# openCypher replacement for apoc.schema.properties.distinct
MATCH (n:airport)
RETURN DISTINCT n.runways
```

```
# openCypher replacement for apoc.schema.properties.distinctCount
MATCH (n:airport)
RETURN DISTINCT n.runways, count(n.runways)
```

Alternativas a los procedimientos de **`apoc.do`**

Estos procedimientos se utilizan para proporcionar una ejecución de consultas condicional que sea fácil de implementar con otras cláusulas de openCypher. En Neptune hay al menos dos formas de lograr un comportamiento similar:

- Una forma es combinar las capacidades de Comprensión de listas de openCypher con la cláusula `UNWIND`.
- Otra forma es usar los pasos choose() y coalesce() de Gremlin.

A continuación se muestran ejemplos de estos enfoques.

Alternativas a apoc.do.when

```
# Neo4J Example
MATCH (n:airport {region: 'US-AK'})
CALL apoc.do.when(
 n.runways>=3,
 'SET n.is_large_airport=true RETURN n',
 'SET n.is_large_airport=false RETURN n',
 {n:n}
) YIELD value
WITH collect(value.n) as airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count
```

```
# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways >= 3] as large_airports,
[a IN airports where a.runways < 3] as small_airports, airports
UNWIND large_airports as la
SET la.is_large_airport=true
WITH DISTINCT small_airports, airports
UNWIND small_airports as la
    SET la.small_airports=true
WITH DISTINCT airports
RETURN size([a in airports where a.is_large_airport]) as large_airport_count,
size([a in airports where NOT a.is_large_airport]) as small_airport_count

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  choose(
    values('runways').is(lt(3)),
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())

 #Neptune Gremlin using coalesce()
g.V().
  has('airport', 'region', 'US-AK').
  coalesce(
    where(values('runways').is(lt(3))).
    property(single, 'is_large_airport', false),
    property(single, 'is_large_airport', true)).
  fold().
  project('large_airport_count', 'small_airport_count').
    by(unfold().has('is_large_airport', true).count()).
    by(unfold().has('is_large_airport', false).count())
```

Alternativas a apoc.do.case

```
# Neo4J Example
```

```
MATCH (n:airport {region: 'US-AK'})
CALL apoc.case([
 n.runways=1, 'RETURN "Has one runway" as b',
 n.runways=2, 'RETURN "Has two runways" as b'
 ],
 'RETURN "Has more than 2 runways" as b'
) YIELD value
RETURN {type: value.b,airport: n}

# Neptune openCypher
MATCH (n:airport {region: 'US-AK'})
WITH n.region as region, collect(n) as airports
WITH [a IN airports where a.runways =1] as single_runway,
[a IN airports where a.runways =2] as double_runway,
[a IN airports where a.runways >2] as many_runway
UNWIND single_runway as sr
    WITH {type: "Has one runway",airport: sr} as res, double_runway, many_runway
WITH DISTINCT double_runway as double_runway, collect(res) as res, many_runway
UNWIND double_runway as dr
    WITH {type: "Has two runways",airport: dr} as two_runways, res, many_runway
WITH collect(two_runways)+res as res, many_runway
UNWIND many_runway as mr
    WITH {type: "Has more than 2 runways",airport: mr} as res2, res, many_runway
WITH collect(res2)+res as res
UNWIND res as r
RETURN r

#Neptune Gremlin using choose()
g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      choose(values('runways')).
        option(1, constant("Has one runway")).
        option(2, constant("Has two runways")).
        option(none, constant("Has more than 2 runways"))).
    by(elementMap())

 #Neptune Gremlin using coalesce()
 g.V().
  has('airport', 'region', 'US-AK').
  project('type', 'airport').
    by(
      coalesce(
```

```
        has('runways', 1).constant("Has one runway"),
        has('runways', 2).constant("Has two runways"),
        constant("Has more than 2 runways"))).
    by(elementMap())
```

Alternativas a las propiedades basadas en listas

Neptune no admite actualmente el almacenamiento de propiedades basadas en listas. Sin embargo, se pueden obtener resultados similares si se almacenan los valores de la lista como una cadena separada por comas y, a continuación, se utilizan las funciones `join()` y `split()` para construir y deconstruir la propiedad de la lista.

Por ejemplo, si quisiéramos guardar una lista de etiquetas como una propiedad, podríamos usar el ejemplo de reescritura, que muestra cómo recuperar una propiedad separada por comas, y luego usar las funciones `split()` y `join()` con Compresión de listas para lograr resultados comparables:

```
# Neo4j Example (In this example, tags is a durable list of string.
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in person.tags WHERE NOT (tag IN ['test1', 'test2', 'test3'])] AS
 newTags
SET person.tags = newTags
RETURN person

# Neptune openCypher
MATCH (person:person {name: "TeeMan"})
WITH person, [tag in split(person.tags, ',') WHERE NOT (tag IN ['test1', 'test2',
 'test3'])] AS newTags
SET person.tags = join(newTags,',')
RETURN person
```

Recursos para migrar de Neo4j a Neptune

Neptune proporciona varias herramientas y recursos que pueden ayudar en el proceso de migración.

Herramientas para ayudar a migrar de Neo4j a Neptune

- Hoja informativa de openCypher.
- neo4j-to-neptune: una utilidad de línea de comandos para migrar datos de Neo4j a Neptune.
- fully-automated-neo4j-to-neptune: una aplicación de AWS CDK que le muestra cómo migrar bases de datos sencillas de Neo4j a Amazon Neptune.
- csv-to-neptune-bulk-format: esta herramienta adopta un método basado en la configuración para reformatear uno o varios archivos CSV en un formato de carga masiva de Neptune compatible.

Publicaciones de blog

- Cambiar captura de datos de Neo4j a Amazon Neptune mediante Amazon Managed Streaming for Apache Kafka de Sanjeet Sahay (22 de junio de 2020)
- Migración de una base de datos de gráficos Neo4j a Amazon Neptune con una utilidad totalmente automatizada de Sanjeet Sahay (13 de abril de 2020)

Migración de un gráfico existente de un servidor Apache TinkerPop Gremlin a Amazon Neptune

Si tiene datos gráficos en un servidor Apache TinkerPop Gremlin que desea migrar a Amazon Neptune, debe seguir los siguientes pasos:

1. Exporte los datos del servidor Gremlin a Amazon Simple Storage Service (Amazon S3).
2. Convierta los datos exportados a un formato CSV que el programa de carga masiva de Neptune pueda importar.
3. Con Programa de carga masiva de Neptune, importe los datos a un clúster de base de datos de Neptune que haya preparado.
4. Modifique la aplicación existente para conectarla al punto de conexión de Gremlin de Neptune y realice los cambios necesarios para adaptarla a las diferencias de implementación de Gremlin en Neptune.

Migración de un gráfico existente de un almacén triple de RDF a Amazon Neptune

Si tiene datos de gráficos en un RDF/SPARQL para migrar a Amazon Neptune, debe realizar los siguientes pasos:

1. Exporte los datos del almacén triple de RDF.
2. Convierta los datos exportados a un formato que el programa de carga masiva de Neptune pueda importar.
3. Guarde los datos que se van a importar en Amazon Simple Storage Service (Amazon S3).
4. Con Programa de carga masiva de Neptune, importe los datos de Amazon S3 a un clúster de base de datos de Neptune que haya preparado.
5. Modifique la aplicación existente para conectarla al punto de conexión de SPARQL de Neptune.

Si quiere intentar migrar datos CSV de gráficos de propiedades a RDF, puede utilizar el conversor CSV a RDF de Amazon Neptune.

Uso de AWS Database Migration Service (AWS DMS) para migrar de una base de datos relacional o NoSQL a Amazon Neptune

AWS Database Migration Service (AWS DMS) es un servicio en la nube que facilita la migración de bases de datos relacionales, almacenes de datos, bases de datos NoSQL y otros tipos de almacenes de datos. Si tiene datos de gráficos almacenados en una de las bases de datos relacionales o NoSQL compatibles con AWS DMS, AWS DMS puede ayudarle a migrar a Neptune de forma rápida y segura, sin que se produzca ningún tiempo de inactividad en su base de datos actual. Para obtener más información, consulte AWS Database Migration Service Utilización para cargar datos en Amazon Neptune desde un almacén de datos diferente.

El flujo de datos de migración con AWS DMS es el siguiente:

- Cree un objeto de mapeo de tablas de AWS DMS. Este objeto de JSON especifica qué tablas se deben leer en la base de datos de origen y en qué orden y cómo se denominan sus columnas. También puede filtrar las filas que se copian y proporcionar transformaciones de valor simples como convertir a minúsculas o redondear.
- Cree una `GraphMappingConfig` de Neptune para especificar cómo deben cargarse los datos extraídos de la base de datos de origen en Neptune.
  - Para los datos RDF (consultados mediante SPARQL), el `GraphMappingConfig` se escribe en el lenguaje de asignación R2RML estándar de W3.
  - Para los datos del gráfico de propiedades (consultados con Gremlin), `GraphMappingConfig` es un objeto de JSON, descrito en GraphMappingConfig Diseño para datos de Property-Graph/Gremlin.
- Cree una instancia de replicación de AWS DMS en la misma VPC que el clúster de base de datos de Neptune para realizar la migración.
- Cree un bucket de Amazon S3 para utilizarlo como almacenamiento intermedio para organizar los datos que se van a migrar.
- Ejecute la tarea de migración de AWS DMS.

Consulte AWS Database Migration Service Utilización para cargar datos en Amazon Neptune desde un almacén de datos diferente para obtener información y también la publicación del blog de cuatro artículos de Chris Smith, “Populating your graph in Amazon Neptune from a relational database using AWS Database Migration Service (DMS)” (Cómo rellenar un gráfico en Amazon Neptune a partir de una base de datos relacional con AWS Database Migration Service DMS):

- Part 1: Setting the stage (Parte 1: Preparación del escenario)
- Part 2: Designing the property graph model (Parte 2: Diseño del modelo de gráficos de propiedades)
- Part 3: Designing the RDF Model (Parte 3: Diseño del modelo RDF)
- Part 4: Putting it all together (Parte 4: Resumen global)

Migración de Blazegraph a Amazon Neptune

Si tiene un gráfico en el almacén triple de RDF de Blazegraph de código abierto, puede migrar los datos del gráfico a Amazon Neptune siguiendo estos pasos:

- Aprovisionamiento de la infraestructura de AWS. Comience por aprovisionar la infraestructura de Neptune necesaria mediante una plantilla de AWS CloudFormation (consulte Creación de un clúster de base de datos).
- Exportación de datos desde Blazegraph. Existen dos métodos principales para exportar datos desde Blazegraph: utilizar las consultas de SPARQL CONSTRUCT o utilizar la utilidad de Exportación de Blazegraph.
- Importación de los datos en Neptune. A continuación, puede cargar los archivos de datos exportados en Neptune con Neptune Workbench y Programa de carga masiva de Neptune.

Este método también suele aplicarse a la migración desde otras bases de datos de almacén triple de RDF.

Compatibilidad entre Blazegraph y Neptune

Antes de migrar los datos de los gráficos a Neptune, hay varias diferencias importantes entre Blazegraph y Neptune que debe tener en cuenta. Estas diferencias pueden requerir cambios en las consultas, en la arquitectura de la aplicación o en ambas, o incluso hacer que la migración no sea práctica:

- **`Full-text search`**: en Blazegraph, puede utilizar las funciones de búsqueda de texto completo interna o externa mediante una integración con Apache Solr. Si utiliza alguna de estas características, manténgase informado sobre las últimas actualizaciones de las características de búsqueda de texto completo compatibles con Neptune. Consulte Búsqueda de texto completo de Neptune.
- **`Query hints`**: tanto Blazegraph como Neptune amplían SPARQL con el concepto de sugerencias de consulta. Durante una migración, debe migrar todas las sugerencias de consulta que utilice. Para obtener información sobre las sugerencias de consulta más recientes que admite Neptune, consulte Sugerencias de consulta SPARQL.
- Inferencia: Blazegraph admite la inferencia como opción configurable en el modo triples, pero no en el modo cuádruple. Neptune aún no admite la inferencia.

- Búsqueda geoespacial: Blazegraph admite la configuración de espacios de nombres que permiten el soporte geoespacial. Esta característica todavía no está disponible en Neptune.
- Varios inquilinos: Blazegraph admite varios inquilinos en una única base de datos. En Neptune, se admiten varios inquilinos mediante el almacenamiento de datos en gráficos con nombre y el uso de las cláusulas USING NAMED para las consultas de SPARQL, o bien mediante la creación de un clúster de base de datos independiente para cada inquilino.
- Federación: Neptune actualmente admite la federación de SPARQL 1.1 en ubicaciones accesibles para la instancia de Neptune, como dentro de la VPC privada, entre VPC o para puntos de conexión de Internet externos. En función de la configuración específica y de los puntos de conexión de federación necesarios, es posible que necesite alguna configuración de red adicional.
- Extensiones de estándares de Blazegraph: Blazegraph incluye varias extensiones de los estándares API de REST y SPARQL, mientras que Neptune solo es compatible con las propias especificaciones de los estándares. Esto puede requerir cambios en la aplicación o dificultar la migración.

Aprovisionamiento de la infraestructura de AWS para Neptune

Aunque puede crear la infraestructura de AWS necesaria manualmente mediante la AWS Management Console o AWS CLI, suele ser mejor utilizar una plantilla de CloudFormation en su lugar, tal y como se describe a continuación:

Aprovisionamiento de Neptune mediante una plantilla de CloudFormation:

1. Vaya a Uso de una AWS CloudFormation pila para crear un clúster de base de datos de Neptune.
2. Elija Lanzar pila en la región que prefiera.
3. Establezca los parámetros necesarios (nombre de la pila y `EC2SSHKeyPairName`). Configure también los siguientes parámetros opcionales para facilitar el proceso de migración:
   - Establezca `AttachBulkloadIAMRoleToNeptuneCluster` en true. Este parámetro permite crear y asociar el rol de IAM adecuado al clúster para permitir la carga masiva de datos.
   - Establezca `NotebookInstanceType` en el tipo de instancia que prefiera. Este parámetro crea un libro de Neptune que se utiliza para ejecutar la carga masiva en Neptune y validar la migración.
4. Elija Next (Siguiente).
5. Establezca cualquier otra opción de pila que desee.

6. Elija Next (Siguiente).
7. Revise sus opciones y seleccione ambas casillas de verificación para reconocer que AWS CloudFormation puede requerir capacidades adicionales.
8. Elija Crear pila.

El proceso de creación de pilas puede tardar unos minutos.

Exportación de datos desde Blazegraph

El siguiente paso es exportar los datos de Blazegraph en un formato que sea compatible con el programa de carga masiva de Neptune.

En función de cómo se almacenen los datos en Blazegraph (triples o cuádruples) y del número de gráficos con nombre que se utilicen, Blazegraph puede requerir que realice el proceso de exportación varias veces y genere varios archivos de datos:

- Si los datos se almacenan como triples, debe ejecutar una exportación para cada gráfico con nombre.
- Si los datos se almacenan como cuádruples, puede optar por exportar los datos en formato N-Quads o exportar cada gráfico con nombre asignado en formato triples.

A continuación, supongamos que exporta un único espacio de nombres como N-Quads, pero puede repetir el proceso para obtener más espacios de nombres o los formatos de exportación que desee.

Si necesita que Blazegraph esté en línea y disponible durante la migración, utilice las consultas SPARQL CONSTRUCT. Esto requiere que instale, configure y ejecute una instancia de Blazegraph con un punto de conexión de SPARQL accesible.

Si no necesita que Blazegraph esté en línea, utilice la utilidad de exportación de BlazeGraph. Para ello, debe descargar Blazegraph y ser capaz de acceder al archivo de datos y a los archivos de configuración, pero no es necesario que el servidor esté funcionando.

Exportación de datos de Blazegraph mediante SPARQL CONSTRUCT

SPARQL CONSTRUCT es una característica de SPARQL que devuelve un gráfico RDF que coincide con la plantilla de consulta especificada. En este caso de uso, se utiliza para exportar los datos de un espacio de nombres a la vez, mediante una consulta como la siguiente:

```
CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }
```

Aunque existen otras herramientas RDF para exportar estos datos, la forma más sencilla de ejecutar esta consulta es mediante el punto de conexión de la API de REST que proporciona Blazegraph. En el siguiente script se muestra cómo usar el script de Python (3.6+) para exportar datos como N-Quads:

```
import requests

# Configure the URL here: e.g. http://localhost:9999/sparql
url = "http://localhost:9999/sparql"
payload = {'query': 'CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }'}
# Set the export format to be n-quads
headers = {
'Accept': 'text/x-nquads'
}
# Run the http request
response = requests.request("POST", url, headers=headers, data = payload, files = [])
#open the file in write mode, write the results, and close the file handler
f = open("export.nq", "w")
f.write(response.text)
f.close()
```

Si los datos se almacenan como triples, debe cambiar el parámetro del encabezado de `Accept` para exportar los datos en un formato adecuado (N-Triples, RDF/XML o Turtle) mediante los valores especificados en el repositorio de Blazegraph GitHub.

Uso de la utilidad de exportación de Blazegraph para exportar datos

Blazegraph incluye un método de utilidad para exportar datos, es decir, la clase `ExportKB`. `ExportKB` facilita la exportación de datos desde Blazegraph, pero a diferencia del método anterior, requiere que el servidor no esté conectado mientras se ejecuta la exportación. Esto lo convierte en el método ideal para usar cuando se puede desconectar Blazegraph durante la migración o si la migración se puede realizar a partir de una copia de seguridad de los datos.

La utilidad se ejecuta desde una línea de comandos de Java en una máquina que tenga Blazegraph instalado pero no en ejecución. La forma más sencilla de ejecutar este comando es descargar la

versión más reciente de blazegraph.jar que se encuentra en GitHub. La ejecución de este comando requiere varios parámetros:

- **`log4j.primary.configuration`**: la ubicación del archivo de propiedades de log4j.
- **`log4j.configuration`**: la ubicación del archivo de propiedades de log4j.
- **`output`**: el directorio de salida de los datos exportados. Los archivos se encuentran ubicados como `tar.gz` en un subdirectorio denominado según lo indicado en la base de conocimientos.
- **`format`**: el formato de salida deseado seguido de la ubicación del archivo `RWStore.properties`. Si trabaja con triples, debe cambiar el parámetro `-format` a `N-Triples`, `Turtle` o `RDF/XML`.

Por ejemplo, si tiene el archivo de diario y los archivos de propiedades de Blazegraph, exporte los datos como N-Quads con el siguiente código:

```
java -cp blazegraph.jar \
    com.bigdata.rdf.sail.ExportKB \
    -outdir ~/temp/ \
    -format N-Quads \
    ./RWStore.properties
```

Si la exportación se realiza de forma correcta, verá el siguiente resultado:

```
Exporting kb as N-Quads on /home/ec2-user/temp/kb
Effective output directory: /home/ec2-user/temp/kb
Writing /home/ec2-user/temp/kb/kb.properties
Writing /home/ec2-user/temp/kb/data.nq.gz
Done
```

Creación de un bucket de Amazon Simple Storage Service (Amazon S3) y copia de los datos exportados en él

Una vez que haya exportado los datos de Blazegraph, cree un bucket de Amazon Simple Storage Service (Amazon S3) en la misma región que el clúster de base de datos de Neptune de destino para que el programa de carga masiva Neptune lo utilice para importar los datos.

Para obtener instrucciones sobre cómo crear un bucket de Amazon S3, consulte ¿Cómo puedo crear un bucket de S3? en la Guía del usuario de Amazon Simple Storage Service y Ejemplos de cómo crear un bucket en la Guía del usuario de Amazon Simple Storage Service.

Para obtener instrucciones sobre cómo copiar los archivos de datos que ha exportado al nuevo bucket de Amazon S3, consulte Cargar un objeto en un bucket en la Guía del usuario de Amazon Simple Storage Service o Uso de comandos de alto nivel (s3) con la AWS CLI. También puede usar código de Python como el siguiente para copiar los archivos uno por uno:

```
import boto3

region = 'region name'
bucket_name = 'bucket name'
s3 = boto3.resource('s3')
s3.meta.client.upload_file('export.nq', bucket_name, 'export.nq')
```

Uso del programa de carga masiva de Neptune para importar los datos en Neptune

Tras exportar los datos de Blazegraph y copiarlos en un bucket de Amazon S3, estará listo para importar los datos en Neptune. Neptune tiene un programa de carga masiva que carga los datos de forma más rápida y con menos sobrecarga que al realizar operaciones de carga con SPARQL. El proceso del programa de carga masiva se inicia con una llamada a la API de punto de conexión del programa de carga para cargar los datos almacenados en el bucket S3 identificado en Neptune.

Aunque puede hacerlo con una llamada directa al punto de conexión REST del programa de carga, debe tener acceso a la VPC privada en la que se ejecuta la instancia de Neptune de destino. Puede configurar un host bastión, usar SSH en esa máquina y ejecutar el comando cURL, pero el uso de Neptune Workbench es más sencillo.

Neptune Workbench es un cuaderno de Jupyter preconfigurado que funciona como un cuaderno de Amazon SageMaker, con varios comandos mágicos de cuaderno específicos de Neptune instalados. Estos comandos mágicos simplifican las operaciones habituales de Neptune, como comprobar el estado del clúster, ejecutar recorridos de SPARQL y Gremlin y ejecutar una operación de carga masiva.

Para iniciar el proceso de carga masiva, utilice el comando mágico `%load`, que proporciona una interfaz para ejecutar la Comando del programa de carga de Neptune:

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Seleccione aws-neptune-blazegraph-to-neptune.

3. Seleccione Abrir cuaderno.
4. En la instancia en ejecución de Jupyter, selecciona un cuaderno existente o cree uno nuevo con el núcleo de Python 3.
5. En el cuaderno, abra una celda, introduzca `%load` y ejecute la celda.
6. Establezca los parámetros para el programa de carga masiva:

    a. En Origen, introduzca la ubicación del archivo de origen que desee importar: `s3://{bucket_name}/{file_name}`.

    b. En Formato, elija el formato adecuado, que en este ejemplo es `nquads`.

    c. En Cargar ARN, introduzca el ARN del rol `IAMBulkLoad` (esta información se encuentra en la consola de IAM, en la opción Roles).
7. Elija Submit (Enviar).

El resultado incluye el estado de la solicitud. Las cargas masivas suelen ser procesos de larga duración, por lo que la respuesta no significa que la carga se haya completado, solo que ha comenzado. Esta información de estado se actualiza periódicamente hasta que se informa de que el trabajo se ha completado.

> **Note**
>
> Esta información también está disponible en la publicación del blog Moving to the cloud: Migrating Blazegraph to Amazon Neptune (Cambiar a la nube: migración de Blazegraph a Amazon Neptune).

Carga de datos en Amazon Neptune

Hay varias formas diferentes de cargar datos de gráficos en Amazon Neptune:

- Si solo tiene que cargar una cantidad relativamente pequeña de datos, puede usar consultas como instrucciones `INSERT` de SPARQL o pasos `addV` y `addE` de Gremlin.
- Puede aprovechar Programa de carga masiva de Neptune para adquirir grandes cantidades de datos que residan en archivos externos. El comando de carga masiva es más rápido y tiene menos sobrecarga que los comandos de lenguaje de consultas. Está optimizado para conjuntos de datos grandes y admite tanto datos RDF (Resource Description Framework) como datos de Gremlin.
- Puede usar AWS Database Migration Service (AWS DMS) para importar datos de otros almacenes de datos (consulte AWS Database Migration Service Utilización para cargar datos en Amazon Neptune desde un almacén de datos diferente la Guía AWS Database Migration Service del usuario).
- Por último, puede usar el paso `g.io(URL).read()` de Gremlin para leer archivos de datos en GraphML (un formato XML), GraphSon (un formato JSON) y otros formatos. Consulte TinkerPopla documentación para obtener más información.

Temas

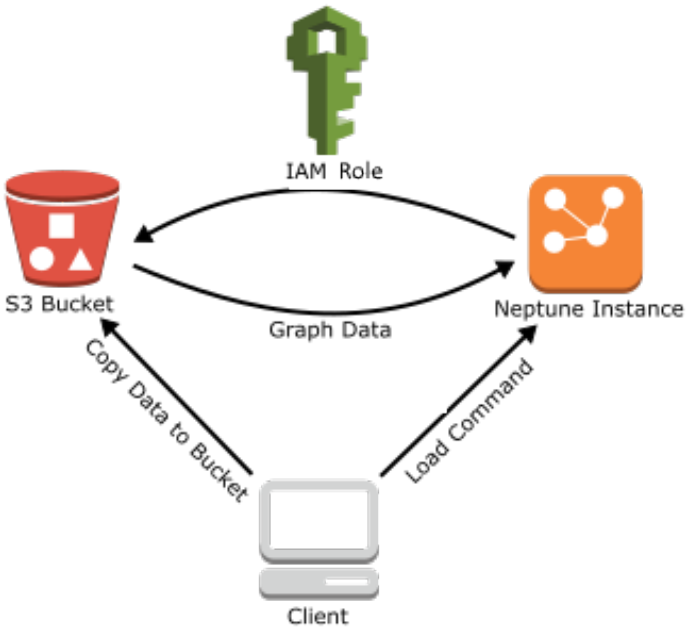
- Uso del programa de carga masiva de Amazon Neptune para adquirir datos
- AWS Database Migration Service Utilización para cargar datos en Amazon Neptune desde un almacén de datos diferente

Uso del programa de carga masiva de Amazon Neptune para adquirir datos

Amazon Neptune ofrece un comando `Loader` para cargar datos de archivos externos directamente en un clúster de base de datos de Neptune. Puede utilizar este comando en lugar de ejecutar un gran número de instrucciones `INSERT`, pasos `addV` y `addE`, u otras llamadas a la API.

El comando Loader de Neptune es más rápido, implica una menor sobrecarga, está optimizado para conjuntos de datos grandes y es compatible tanto con los datos de Gremlin como los datos RDF (marco de descripción de recursos) que utiliza SPARQL.

En el diagrama siguiente se muestra información general del proceso de carga:



A continuación se muestran los pasos del proceso de carga:

1. Copie los archivos de datos en un bucket de Amazon Simple Storage Service (Amazon S3).
2. Crear una función de IAM con acceso de lectura y lista al bucket.
3. Cree punto de conexión de VPC de Amazon S3
4. Inicie el programa de carga de Neptune enviando una solicitud mediante HTTP a la instancia de base de datos de Neptune.
5. La instancia de base de datos de Neptune asume el rol de IAM para cargar los datos del bucket.

> **Note**
>
> Puede cargar datos cifrados de Amazon S3 si se cifraron mediante `SSE-S3` de Amazon S3 o el modo `SSE-KMS`, siempre que el rol que utilice para la carga masiva tenga acceso al objeto de Amazon S3 y, en el caso de SSE-KMS, a `kms:decrypt`. En ese caso, Neptune puede suplantar sus credenciales y emitir llamadas `s3:getObject` en su nombre.
> Sin embargo, Neptune actualmente no admite la carga de datos cifrados con el modo `SSE-C`.

En las secciones siguientes, encontrará instrucciones para preparar y cargar datos en Neptune.

Temas

Requisitos previos: rol de IAM y acceso a Amazon S3

La carga de datos desde un bucket de Amazon Simple Storage Service (Amazon S3) requiere AWS Identity and Access Management un rol (IAM) que tenga acceso al bucket. Amazon Neptune asume este rol para cargar los datos.

> **Note**
>
> Puede cargar datos cifrados desde Amazon S3 si se cifran mediante el modo `SSE-S3` de Amazon S3. En ese caso, Neptune es capaz de suplantar sus credenciales y emitir llamadas de `s3:getObject` en su nombre.
> También puede cargar datos cifrados desde Amazon S3 que se hayan cifrado mediante el modo `SSE-KMS`, siempre que el rol de IAM incluya los permisos necesarios para obtener acceso a AWS KMS. Sin AWS KMS los permisos adecuados, la operación de carga masiva falla y devuelve una `LOAD_FAILED` respuesta.
> Actualmente Neptune no admite la carga de datos cifrados de Amazon S3 con el modo `SSE-C`.

En las siguientes secciones, se muestra cómo utilizar una política de IAM administrada para crear un rol de IAM para obtener acceso a los recursos de Amazon S3 y, luego, asociarla a su clúster de Neptune.

Temas

> **Note**
>
> En estas instrucciones, se requiere que tenga acceso a la consola de IAM y permisos para administrar las políticas y los roles de IAM. Para obtener más información, consulte Permisos para trabajar en la consola AWS de administración en la Guía del usuario de IAM.
> La consola de Amazon Neptune requiere que el usuario tenga los siguientes permisos de IAM para asociar el rol al clúster de Neptune:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Creación de un rol de IAM que permita a Amazon Neptune acceder a los recursos de Amazon S3.

Utilice la política de IAM administrada `AmazonS3ReadOnlyAccess` para crear un nuevo rol de IAM que permita a Amazon Neptune acceder a los recursos de Amazon S3.

Para crear un nuevo rol de IAM que permita que Neptune acceda a Amazon S3

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles en el panel de navegación.
3. Elija Crear rol.
4. En Servicio de AWS , elija S3.
5. Elija Siguiente: permisos.
6. Utilice la casilla de filtro para filtrar por el término S3 y marque la casilla situada junto a AmazonS3 Access ReadOnly.

> **Note**
>
> Esta política concede los permisos `s3:Get*` y `s3:List*` a todos los buckets. Los pasos posteriores restringen el acceso al rol mediante la política de confianza.
> El programa de carga solo requiere los permisos `s3:Get*` y `s3:List*` para el bucket desde el que realiza la carga, por lo que también puede restringir estos permisos según el recurso de Amazon S3.

Si su bucket de S3 está cifrado, debe añadir permisos `kms:Decrypt`

7. Elija Siguiente: Revisar.
8. En Nombre de rol, escriba un nombre para el rol de IAM, por ejemplo, `NeptuneLoadFromS3`. También puede añadir un valor opcional en Descripción del rol, como: "Permite a Neptune obtener acceso a los recursos de Amazon S3 en su nombre".
9. Seleccione Crear rol.
10. Seleccione Roles en el panel de navegación.
11. En el campo Search (Buscar), introduzca el nombre de la función creada y elija la función cuando aparezca en la lista.
12. En la pestaña Trust Relationships (Relaciones de confianza), elija Edit trust relationship (Editar relación de confianza).
13. Copie la siguiente política de confianza en el campo de texto:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

14. Elija Actualizar política de confianza.
15. Realice los pasos que se indican en Adición del rol de IAM a un clúster de Amazon Neptune.

Adición del rol de IAM a un clúster de Amazon Neptune

Utilice la consola para añadir el rol de IAM a un clúster de Amazon Neptune. De este modo, cualquier instancia de base de datos de Neptune del clúster puede asumir el rol y cargarlo desde Amazon S3.

> **Note**
>
> La consola de Amazon Neptune requiere que el usuario tenga los siguientes permisos de IAM para asociar el rol al clúster de Neptune:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Para añadir un rol de IAM a un clúster de Amazon Neptune

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el identificador del clúster que desee modificar.
4. Seleccione la pestaña Conectividad y seguridad.
5. En la sección Funciones de IAM, elija la función que creó en la sección anterior.
6. Seleccione Add role (Añadir rol).
7. Espere hasta que el clúster tenga acceso al rol de IAM antes de utilizarlo.

Creación de un punto de conexión de VPC de Amazon S3

El programa de carga de Neptune requiere un punto de conexión de VPC de tipo Puerta de enlace para Amazon S3.

Para configurar el acceso a Amazon S3

1. Inicie sesión en la consola de Amazon VPC AWS Management Console y ábrala en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación, elija Puntos de conexión.
3. Elija Crear punto de conexión.
4. Elija el Nombre del servicio `com.amazonaws.`*`region`*`.s3` para el punto de conexión de tipo Puerta de enlace.

> **Note**
>
> Si la región indicada es incorrecta, asegúrese de que la región de la consola es correcta.

5. Elija la VPC que contiene su instancia de base de datos de Neptune (aparece para su instancia de base de datos en la consola de Neptune).
6. Seleccione la casilla de verificación situada junto a las tablas de ruteo asociadas a las subredes relacionadas con el clúster. Si solo tiene una tabla de ruteo, debe seleccionar esa casilla.
7. Seleccione Crear punto de conexión.

Para obtener más información acerca de la creación del punto de conexión, consulte Puntos de conexión de la VPC en la Guía del usuario de Amazon VPC. Para obtener información acerca de las limitaciones de los puntos de conexión de VPC, consulte VPC Endpoints for Amazon S3.

Siguientes pasos

Ahora que ha concedido acceso al bucket de Amazon S3, puede prepararse para la carga de datos. Para obtener información acerca de los formatos admitidos, consulte Formatos de los datos de carga.

Encadenamiento de roles de IAM en Amazon Neptune

> **Important**
>
> La nueva característica multicuenta de carga masiva introducida en la versión 1.2.1.0.R3 del motor, que aprovecha la posibilidad de encadenar roles de IAM, en algunos casos, puede provocar una degradación del rendimiento de la carga masiva. En consecuencia, las actualizaciones de las versiones del motor que admiten esta característica se han suspendido temporalmente hasta que se resuelva el problema.

Cuando asocia un rol a su clúster, este puede asumir dicho rol para obtener acceso a los datos almacenados en Amazon S3. A partir de la versión 1.2.1.0.R3 del motor, si ese rol no tiene acceso a todos los recursos que necesita, puede encadenar uno o más roles adicionales que el clúster puede asumir para acceder a otros recursos. Cada rol de la cadena asume el siguiente rol de la cadena, hasta que su clúster haya asumido el rol al final de la cadena.

Para encadenar roles, debe establecer una relación de confianza entre ellos. Por ejemplo, para encadenar `RoleB` a `RoleA`, `RoleA` debe tener una política de permisos que le permita asumir `RoleB` y `RoleB` debe tener una política de confianza que le permita volver a transferir sus permisos a `RoleA`. Para obtener más información, consulte Uso de roles de IAM.

El primer rol de una cadena debe estar asociado al clúster que está cargando los datos.

El primer rol, y cada rol subsiguiente que asuma el siguiente rol en la cadena, debe tener:

- Una política que incluye una instrucción específica con el efecto `Allow` sobre la acción `sts:AssumeRole`.
- El nombre de recurso de Amazon (ARN) del siguiente rol de un elemento `Resource`.

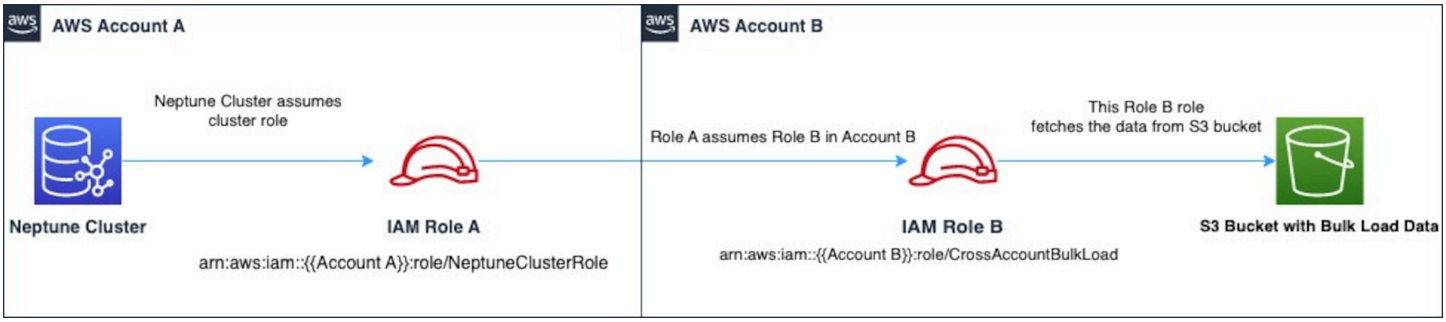
> **Note**
>
> El bucket de Amazon S3 de destino debe estar en la misma AWS región que el clúster.

Acceso entre cuentas mediante roles encadenados

Puede conceder el acceso entre cuentas encadenando un rol o roles que pertenezcan a otra cuenta. Cuando su clúster asuma temporalmente un rol que pertenece a otra cuenta, puede acceder a los recursos de esa cuenta.

Por ejemplo, imagine que la cuenta A desea obtener acceso a los datos de un bucket de Amazon S3 que pertenece a la cuenta B:

- La cuenta A crea un rol de AWS servicio para Neptune denominado `RoleA` y lo adjunta a un clúster.
- La cuenta B crea un rol denominado `RoleB` que está autorizado para obtener acceso a los datos de un bucket de la cuenta B.
- La cuenta A asocia una política de permisos a `RoleA` que le permite asumir `RoleB`.
- La cuenta B asocia una política de confianza a `RoleB`, que le permite transferir sus permisos a `RoleA`.
- Para obtener acceso a los datos del bucket de la cuenta B, la cuenta A ejecuta un comando del programa de carga utilizando un parámetro `iamRoleArn` que encadena `RoleA` y `RoleB`. Durante la operación del programa de carga, `RoleA` asume de forma temporal el `RoleB` para obtener acceso al bucket de Amazon S3 en la cuenta B.



Por ejemplo, `RoleA` tendría una política de confianza que establece una relación de confianza con Neptune:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

`RoleA` también tendría una política de permisos que le permitiría asumir el `RoleB`, que es propiedad de la cuenta B:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1487639602000",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::(Account B ID):role/RoleB"
    }
  ]
}
```

Por el contrario, `RoleB` tendría una política de confianza para establecer una relación de confianza con`RoleA`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::(Account A ID):role/RoleA"
      }
    }
  ]
}
```

`RoleB` también necesitaría permiso para acceder a los datos del bucket de Amazon S3 ubicado en la cuenta B.

Creación de un punto AWS Security Token Service final de VPC (STS)

El cargador de Neptune requiere un punto final de VPC para encadenar funciones AWS STS de IAM a API de acceso AWS STS privado a través de direcciones IP privadas. Puede conectarse directamente desde una Amazon VPC a AWS STS través de un punto final de VPC de forma segura y escalable. Cuando utiliza un punto de conexión de VPC de interfaz, ofrece una mejor postura de seguridad, ya que no necesita abrir firewalls de tráfico saliente. También ofrece las demás ventajas del uso de puntos de conexión de Amazon VPC.

Cuando se utiliza un punto final de VPC, el tráfico AWS STS no se transmite a través de Internet y nunca sale de la red de Amazon. Su VPC está conectada de forma segura AWS STS sin riesgos de disponibilidad ni restricciones de ancho de banda en el tráfico de su red. Para obtener más información, consulte Uso de puntos de conexión de VPC de interfaz en AWS STS.

Para configurar el acceso para AWS Security Token Service (STS)

1. Inicie sesión en la consola de Amazon VPC AWS Management Console y ábrala en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación, elija Puntos de conexión.
3. Elija Crear punto de conexión.

4. Elija el Nombre del servicio: `com.amazonaws.region.sts` para el punto de conexión de tipo Interfaz.
5. Elija la VPC que contiene la instancia de base de datos de Neptune y la instancia EC2.
6. Seleccione la casilla de verificación junto a la subred en la que se encuentra su instancia EC2. No puede seleccionar varias subredes de la misma zona de disponibilidad.
7. En IP address type (Tipo de dirección IP), elija entre las siguientes opciones:

   - IPv4: se asignan direcciones IPv4 a las interfaces de red del punto de conexión. Esta opción solo se admite si todas las subredes seleccionadas tienen rangos de direcciones IPv4.
   - IPv6: se asignan direcciones IPv6 a las interfaces de red del punto de conexión. Esta opción solo se admite si todas las subredes seleccionadas son subredes solo IPv6.
   - Dualstack: se asignan direcciones IPv4 e IPv6 a las interfaces de red del punto de conexión. Esta opción solo se admite si todas las subredes seleccionadas tienen rangos de direcciones IPv4 e IPv6.
8. En Security groups (Grupos de seguridad), seleccione los grupos de seguridad para asociarlas a las interfaces de red del punto de conexión para el punto de conexión de VPC. Debería seleccionar todos los grupos de seguridad que están asociados a la instancia de base de datos de Neptune y a la instancia EC2.
9. En Política, seleccione Acceso completo para permitir todas las operaciones de todas las entidades principales en todos los recursos del punto de conexión de VPC. De lo contrario, seleccione Personalizar para adjuntar una política de punto de conexión de VPC que controle los permisos que tienen las entidades principales para realizar acciones en los recursos a través del punto de conexión de VPC. Esta opción solo está disponible si el servicio admite las políticas de punto de conexión de VPC. Para obtener más información, consulte Políticas de puntos de conexión.
10. (Opcional) Para añadir una etiqueta, elija Agregar etiqueta nueva e introduzca la clave y el valor de la etiqueta.
11. Seleccione Crear punto de conexión.

Para obtener más información acerca de la creación del punto de conexión, consulte Puntos de conexión de la VPC en la Guía del usuario de Amazon VPC. Tenga en cuenta que el punto de conexión de VPC de Amazon STS es un requisito previo obligatorio para el encadenamiento de roles de IAM.

Ahora que ha concedido el acceso al AWS STS punto final, puede prepararse para cargar los datos. Para obtener más información acerca de los formatos admitidos, consulte Load Data Formats.

Encadenamiento de roles en un comando del programa de carga

Para especificar el encadenamiento de roles cuando ejecute un comando del programa de carga, incluya en el parámetro `iamRoleArn` una lista separada por comas de los ARN de los roles.

Aunque la mayoría de las veces solo necesitará tener dos roles en una cadena, es posible encadenar tres o más. Por ejemplo, este comando del programa de carga encadena tres roles:

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A ID):role/(RoleA),arn:aws:iam::(Account
 B ID):role/(RoleB),arn:aws:iam::(Account C ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

Formatos de los datos de carga

La API `Load` de Amazon Neptune admite la carga de datos en una gran variedad de formatos.

Formatos de carga de gráficos de propiedades

A continuación, se pueden consultar los datos cargados en uno de los siguientes formatos de gráficos de propiedades mediante Gremlin y openCypher:

- Formato de datos de carga de Gremlin (`csv`): formato de valores separados por comas (CSV).
- Formato de carga de datos openCypher: (`opencypher`) formato de valores separados por comas (CSV).

Formatos de carga RDF

Para cargar datos del marco de descripción de recursos (RDF) que consulte mediante SPARQL, puede utilizar uno de los siguientes formatos estándar especificados por el World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) de la especificación en https://www.w3.org/TR/n-triples/.

- N-Quads (`nquads`) de la especificación en https://www.w3.org/TR/n-quads/.
- RDF/XML (`rdfxml`) de la especificación en https://www.w3.org/TR/rdf-syntax-grammar/.
- Turtle (`turtle`) de la especificación en https://www.w3.org/TR/turtle/.

Los datos de carga deben utilizar la codificación UTF-8

> **Important**
>
> Todos los archivos de datos de carga deben estar codificados en formato UTF-8. Si un archivo no tiene formato UTF-8, Neptune intenta cargarlo de todos modos como UTF-8.

Para los datos N-Quads y N-triples que incluyen caracteres Unicode, se admiten las secuencias de escape `\u`*`xxxxx`*. Sin embargo, Neptune no admite la normalización. Si hay un valor que requiere normalización, no coincidirá byte-to-byte durante la consulta. Para obtener más información acerca de la normalización, consulte la página Normalization en Unicode.org.

Si los datos no están en un formato compatible, debe convertirlos antes de cargarlos.

Una herramienta para convertir GraphML al formato CSV de Neptune está disponible en el proyecto GraphML2CSV en. GitHub

Compatibilidad con compresión para archivos de datos de carga

Neptune admite la compresión de archivos individuales en formato `gzip` o `bzip2`.

El archivo comprimido debe tener una extensión `.gz` o `.bz2` y debe ser un archivo de texto único codificado en formato UTF-8. Puede cargar varios archivos, pero cada uno debe ser un archivo `.gz`, `.bz2` o un archivo de texto sin comprimir independiente. No es posible archivar archivos con extensiones como `.tar`, `.tar.gz` y `.tgz`.

En las siguientes secciones se describen los formatos de manera más detallada.

Temas

- Formato de datos de carga de Gremlin
- Formato de carga para los datos de openCypher
- Formatos de los datos de carga de RDF

Formato de datos de carga de Gremlin

Para cargar datos de Apache TinkerPop Gremlin en formato CSV, debe especificar los vértices y los bordes en archivos separados.

El cargador puede cargar desde varios archivos de vértice y varios archivos de borde en una única tarea de carga.

Para cada comando de carga, el conjunto de archivos que se va a cargar debe estar en la misma carpeta del bucket de Amazon S3 y se debe especificar el nombre de la carpeta para el parámetro `source`. Los nombres de archivo y las extensiones de estos no son importantes.

El formato CSV de Amazon Neptune se ajusta a la especificación RFC 4180 para este tipo de formato. Para obtener más información, consulte la sección sobre formato común y tipos MIME para archivos CSV en el sitio web de Internet Engineering Task Force (IETF).

> **Note**
>
> Todos los archivos deben estar codificados en formato UTF-8.

Cada archivo tiene una fila de encabezado con elementos separados por comas. La fila de encabezado se compone tanto de encabezados de columnas del sistema como encabezados de columnas de propiedades.

Encabezados de columnas del sistema

Los encabezados de columnas del sistema necesarios y permitidos son distintos para los archivos de vértices y para los archivos de bordes.

Cada columna del sistema solo puede aparecer una vez en un encabezado.

Todas las etiquetas distinguen entre mayúsculas y minúsculas.

Encabezados de vértices

- `~id`: obligatorio

  ID del vértice.
- `~label`

Etiqueta para el vértice. Se permiten varios valores de etiqueta, separados por punto y coma (`;`).

Si no `~label` está presente, TinkerPop proporciona una etiqueta con el valor`vertex`, ya que cada vértice debe tener al menos una etiqueta.

Encabezados de borde

- `~id`: obligatorio

  ID del borde.
- `~from`: obligatorio

  ID del vértice desde.
- `~to`: obligatorio

  ID del vértice hasta.
- `~label`

  Etiqueta para el borde. Los bordes solo pueden tener una etiqueta.

  Si no `~label` está presente, TinkerPop proporciona una etiqueta con el valor`edge`, ya que cada borde debe tener una etiqueta.

Encabezados de columnas de propiedades

Para especificar una columna (`:`) para una propiedad, use la sintaxis siguiente. Los nombres de tipos no distinguen entre mayúsculas y minúsculas. Sin embargo, tenga en cuenta que si aparecen dos puntos dentro del nombre de una propiedad, deben ir precedidos de una barra invertida: `\:`.

```
propertyname:type
```

> **Note**
>
> En los encabezados de las columnas no se admiten espacios, comas, volantes ni líneas nuevas, por lo que los nombres de las propiedades no pueden incluir estos caracteres.

Para especificar una columna para un tipo de matriz, añada `[]` al tipo:

```
propertyname:type[]
```

> **Note**
>
> Las propiedades de borde solo pueden tener un único valor y provocarán un error si se especifica un tipo de matriz o si se especifica un segundo valor.

En el siguiente ejemplo, se muestra el encabezado de columna para una propiedad llamada `age` de tipo `Int`.

```
age:Int
```

Cada fila del archivo debe tener un entero en esa posición o bien dejarse en blanco.

Se permiten matrices de cadenas, pero las cadenas de una matriz no pueden incluir el carácter de punto y coma (`;`) a menos que se escapen mediante una barra invertida (como esta: `\;`).

Especificación de la cardinalidad de una columna

A partir de Versión 1.0.1.0.200366.0 (26/07/2019), el encabezado de columna se pueden utilizar para especificar la cardinalidad de la propiedad identificada por la columna. Esto permite que el programa de carga masivo respete la cardinalidad de manera similar a como lo hace con las consultas Gremlin.

Especifique la cardinalidad de una columna como se indica a continuación:

```
propertyname:type(cardinality)
```

El valor de la *`cardinalidad`* puede ser `single` o `set`. El valor predeterminado se supone que es `set`, lo que significa que la columna puede aceptar varios valores. En el caso de los archivos de borde, la cardinalidad es siempre única y la especificación de otra cardinalidad provoca que el programa de carga genere una excepción.

Si la cardinalidad es `single`, el programa de carga genera un error si ya hay presente un valor anterior cuando se cargan uno o varios valores. Este comportamiento se puede anular, de forma que un valor existente se sustituye cuando un nuevo valor se carga mediante el uso del indicador `updateSingleCardinalityProperties`. Consulte Comando Loader.

Es posible utilizar una configuración de cardinalidad con un tipo de matriz, aunque esto, por lo general, no es es necesario. A continuación se muestran las posibles combinaciones:

- `name:type`: la cardinalidad es `set` y el contenido es de un solo valor.
- `name:type[]`: la cardinalidad es `set` y el contenido es de varios valores.
- `name:type(single)`: la cardinalidad es `single` y el contenido es de un solo valor.
- `name:type(set)`— la cardinalidad es `set`, que es la misma que la predeterminada, y el contenido es de un solo valor.
- `name:type(set)[]`: la cardinalidad es `set` y el contenido es de varios valores.
- `name:type(single)[]`: esto es contradictorio y produce un error.

En la siguiente sección se enumeran todos los tipos de datos Gremlin disponibles.

Tipos de datos de Gremlin

Esta es una lista de los tipos de propiedades permitidos, con una descripción de cada tipo.

Bool (o Booleano)

Indica un campo booleano. Valores permitidos: `false`, `true`

> **Note**
>
> Cualquier valor distinto de `true` se tratará como falso.

Tipos de número entero

Los valores fuera de los intervalos definidos generarán un error.

Tipo	Range
Byte	De -128 a 127
Short	De -32768 a 32767
Int	-2^31 a 2^31-1

Long	-2^63 a 2^63-1

Tipos de número decimal

Admite la notación decimal y la notación científica. También permite símbolos, como (+/-), Infinity o NaN. INF no se admite.

Tipo	Rango
Flotante	Punto flotante IEEE 754 de 32 bits
Doble	Punto flotante IEEE 754 de 64 bits

Los valores float y double que son demasiado largos, se cargan y se redondean al valor más cercano con una precisión de 24 bits (float) y de 53 bits (double). Un valor intermedio se redondea a 0 hasta el último dígito restante en el nivel de bit.

Cadena

El uso de las comillas es opcional. A los caracteres de coma, nueva línea y retorno de carro se les aplica escape automáticamente si están incluidos en una cadena entre comillas dobles (`"`). Ejemplo: `"Hello, World"`

Para incluir comillas en una cadena con comillas, use dos entradas de comillas en una fila para aplicar escape: Ejemplo: `"Hello ""World"""`

Se permiten matrices de cadenas, pero las cadenas de una matriz no pueden incluir el carácter de punto y coma (`;`) a menos que se escapen mediante una barra invertida (como esta: `\;`).

Si desea rodear las cadenas de una matriz con comillas, debe rodear toda la matriz con un conjunto de comillas. Ejemplo: `"String one; String 2; String 3"`

Date

Fecha de Java en formato ISO-8601. Admite los siguientes formatos: `yyyy-MM-dd`, `yyyy-MM-ddTHH:mm`, `yyyy-MM-ddTHH:mm:ss`, `yyyy-MM-ddTHH:mm:ssZ`

Formato de fila de Gremlin

Delimitadores

Los campos de una fila se separan por una coma. Los registros se separan por una nueva línea o por una nueva línea seguida de un retorno de carro.

Campos en blanco

Se permiten los campos en blanco para las columnas que no son obligatorias (por ejemplo, las propiedades que define el usuario). Un campo en blanco también requiere el separador de coma. Los campos en blanco en las columnas obligatorias provocarán un error de análisis. Los valores de cadena vacíos se interpretan como un valor de cadena vacío para el campo, no como un campo en blanco. El ejemplo de la siguiente sección tiene un campo en blanco en cada vértice de ejemplo.

ID de vértice

Los valores `~id` deben ser únicos para todos los vértices en cada archivo de vértices. Varias filas de vértice que tengan valores `~id` idénticos se aplican a un mismo vértice del gráfico. La cadena vacía (`""`) es un identificador válido y el vértice se crea con una cadena vacía como identificador.

ID de borde

Los valores `~id` también deben ser únicos para todos los bordes en cada archivo de bordes. Varias filas de borde que tengan valores `~id` idénticos se aplican a un mismo borde del gráfico. La cadena vacía (`""`) es un identificador válido y el borde se crea con una cadena vacía como identificador.

Etiquetas

Las etiquetas distinguen mayúsculas de minúsculas y no pueden estar vacías. Un valor de `""` provocará un error.

Valores de cadena

El uso de las comillas es opcional. A los caracteres de coma, nueva línea y retorno de carro se les aplica escape automáticamente si están incluidos en una cadena entre comillas dobles (`"`). Los valores de cadena vacíos (`""`) se interpretan como un valor de cadena vacío para el campo, no como un campo en blanco.

Especificación del formato CSV

El formato CSV de Neptune se ajusta a la especificación RFC 4180 correspondiente a este tipo de formato, que incluye los siguientes requisitos:

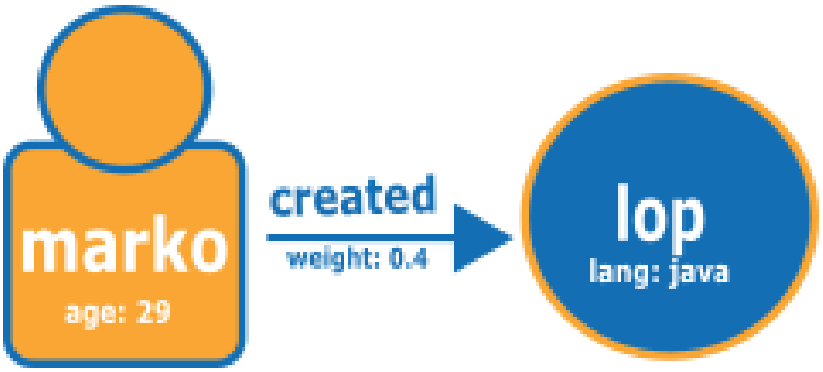
- Se admiten finales de línea tanto de estilo Unix como Windows (\n o \r\n).

- Cualquier campo se puede incluir entre comillas (usando comillas dobles).
- Los campos que contienen un salto de línea, comillas dobles o comas deben incluirse entre comillas. (De lo contrario, la carga se anula inmediatamente).
- Un carácter de comillas dobles (") en un campo debe representarse con dos caracteres de comillas (dobles). Por ejemplo, una cadena `Hello "World"` debe aparecer como `"Hello ""World"""` en los datos.
- Los espacios que hay entre los delimitadores se omiten. Si una fila está presente como`value1, value2`, se almacenan como `"value1"` y`"value2"`.
- Cualquier otro carácter de escape se almacena literalmente. Por ejemplo, `"data1\tdata2"` se almacena como `"data1\tdata2"`. No es necesario seguir aplicando el escape, siempre que estos caracteres estén encerrados entre comillas.
- Se permiten campos en blanco. Un campo en blanco se considera un valor vacío.
- Si un campo tiene varios valores, estos se separan por punto y coma (`;`).

Para obtener más información, consulte la sección sobre formato común y tipos MIME para archivos CSV en el sitio web de Internet Engineering Task Force (IETF).

Ejemplo de Gremlin

El siguiente diagrama muestra un ejemplo de dos vértices y una arista tomados del gráfico TinkerPop moderno.



A continuación se muestra el gráfico con formato de carga CSV de Neptune.

Archivo de vértice:

```
~id,name:String,age:Int,lang:String,interests:String[],~label
v1,"marko",29,,"sailing;graphs",person
```

```
v2,"lop",,"java",,software
```

Vista tabular del archivo de vértice:

~id	name:String	age:Int	lang:String	Intereses: cadena []	~label
v1	"marko"	29		["navegación», «gráficos"]	person
v2	"lop"		"java"		software

Archivo de borde:

```
~id,~from,~to,~label,weight:Double
e1,v1,v2,created,0.4
```

Vista tabular del archivo de borde:

~id	~from	~a	~label	weight:Double
e1	v1	v2	created	0.4

Siguientes pasos

Ahora que conoce mejor los formatos de carga, consulte Ejemplo: carga de datos en una instancia de base de datos de Neptune.

Formato de carga para los datos de openCypher

Para cargar datos de openCypher con el formato CSV de OpenCypher, debe especificar los nodos y las relaciones en archivos independientes. El programa de carga puede cargar desde varios de estos archivos de nodos y archivos de relaciones en un solo trabajo de carga.

Para cada comando de carga, el conjunto de archivos que se va a cargar debe tener el mismo prefijo de ruta en un bucket de Amazon Simple Storage Service. Debe especificar ese prefijo en el parámetro de origen. Los nombres de archivo y sus extensiones no son importantes.

En Amazon Neptune, el formato CSV de openCypher cumple la especificación CSV de la RFC 4180. Para obtener más información, consulte la sección Common Format and MIME Type for CSV Files (https://tools.ietf.org/html/rfc4180) en el sitio web de Internet Engineering Task Force (IETF).

> **Note**
>
> Estos archivos deben estar codificados en formato UTF-8.

Cada archivo tiene una fila de encabezados separada por comas que contiene los encabezados de las columnas del sistema y los encabezados de las columnas de propiedades.

Encabezados de columnas del sistema en archivos de carga de datos de openCypher

Una columna del sistema determinada solo puede aparecer una vez en cada archivo. Todas las etiquetas de encabezado de columna del sistema distinguen entre mayúsculas y minúsculas.

Los encabezados de columna del sistema que son obligatorios y están permitidos son diferentes para los archivos de carga de nodos de openCypher y los archivos de carga de relaciones:

Encabezados de columnas del sistema en los archivos de nodos

- **`:ID`**: (obligatorio) Un identificador para el nodo.

  Se puede añadir un espacio de identificación opcional al encabezado de la columna `:ID` del nodo de la siguiente manera: `:ID(`*`ID Space`*`)`. Un ejemplo es `:ID(movies)`.

  Al cargar las relaciones que conectan los nodos de este archivo, utilice los mismos espacios de identificador en las columnas de relaciones `:START_ID` y/o `:END_ID` de los archivos de relaciones.

  La columna `:ID` de los nodos se puede almacenar opcionalmente como una propiedad en el formato *`property name`*`:ID`. Un ejemplo es `name:ID`.

  Los identificadores de nodo deben ser únicos en todos los archivos de nodos de las cargas actuales y anteriores. Si se usa un espacio de identificador, los identificadores de nodo deben ser únicos en todos los archivos de nodo que usen el mismo espacio de identificador en las cargas actuales y anteriores.
- **`:LABEL`**: etiqueta del nodo.

  Se permiten varios valores de etiqueta, separados por punto y coma (`;`).

Encabezados de columnas del sistema en archivos de relaciones

- **`:ID`**: identificador de la relación. Es obligatorio cuando `userProvidedEdgeIds` es true (el valor predeterminado), pero no es válido cuando `userProvidedEdgeIds` es`false`.

  Los identificadores de relaciones deben ser únicos en todos los archivos de relaciones de las cargas actuales y anteriores.
- **`:START_ID`**: (obligatorio) el identificador de nodo del nodo desde el que se inicia esta relación.

  Opcionalmente, se puede asociar un espacio de identificador a la columna del identificador inicial en el formato `:START_ID(`*`ID Space`*`)`. El espacio de identificador asignado al identificador del nodo inicial debe coincidir con el espacio de identificador asignado al nodo en su archivo de nodos.
- **`:END_ID`**: (obligatorio) el identificador de nodo del nodo en el que termina esta relación.

  Opcionalmente, se puede asociar un espacio de identificador a la columna del identificador final en el formato `:END_ID(`*`ID Space`*`)`. El espacio de identificador asignado al identificador del nodo final debe coincidir con el espacio de identificador asignado al nodo en su archivo de nodos.
- **`:TYPE`**: tipo de la relación. Las relaciones solo pueden tener un tipo.

> **Note**
>
> Consulte Carga de datos de openCypher para obtener información sobre cómo se gestionan los identificadores de nodos o relaciones duplicados en el proceso de carga masiva.

Encabezados de columnas de propiedades en archivos de carga de datos de openCypher

Puede especificar que una columna contenga los valores de una propiedad concreta mediante un encabezado de columna de propiedades con el siguiente formato:

```
propertyname:type
```

En los encabezados de las columnas no se admiten espacios, comas, vagones de ida y vuelta ni líneas nuevas, por lo que los nombres de las propiedades no pueden incluir estos caracteres. A continuación, se muestra un ejemplo de encabezado de columna para una propiedad denominada `age` del tipo `Int`:

```
age:Int
```

La columna con `age:Int` como encabezado de columna tendría entonces que contener un número entero o un valor vacío en cada fila.

Tipos de datos en archivos de carga de datos de openCypher de Neptune

- **`Bool`** o **`Boolean`**: campo booleano. Los valores permitidos son `true` y `false`.

  Cualquier valor que no sea `true` se trata como `false`.
- **`Byte`**: número entero comprendido en el rango de `-128` a `127`.
- **`Short`**: número entero comprendido en el rango de `-32,768` a `32,767`.
- **`Int`**: número entero comprendido en el rango de `-2^31` a `2^31 - 1`.
- **`Long`**: número entero comprendido en el rango de `-2^63` a `2^63 - 1`.
- **`Float`**: número de coma flotante IEEE 754 de 32 bits. Se admiten tanto la notación decimal como la notación científica. `Infinity`, `-Infinity` y `NaN` se reconocen, pero no `INF`.

  Los valores que tienen demasiados dígitos y no caben se redondean al valor más cercano (un valor intermedio se redondea a 0 para el último dígito restante en el nivel de bits).
- **`Double`**: número de coma flotante IEEE 754 de 64 bits. Se admiten tanto la notación decimal como la notación científica. `Infinity`, `-Infinity` y `NaN` se reconocen, pero no `INF`.

  Los valores que tienen demasiados dígitos y no caben se redondean al valor más cercano (un valor intermedio se redondea a 0 para el último dígito restante en el nivel de bits).
- **`String`**: el uso de las comillas es opcional. Los caracteres de coma, nueva línea y retorno de carro se escapan automáticamente si se incluyen en una cadena rodeada de comillas dobles (`"`) como `"Hello, World"`.

  Puede incluir comillas en una cadena entre comillas usando dos seguidas; por ejemplo `"Hello ""World"""`.
- **`DateTime`**: fecha Java en uno de los siguientes formatos ISO-8601:
  - `yyyy-MM-dd`
  - `yyyy-MM-ddTHH:mm`
  - `yyyy-MM-ddTHH:mm:ss`
  - `yyyy-MM-ddTHH:mm:ssZ`

Tipos de datos de conversión automática en archivos de carga de datos de openCypher de Neptune

Los tipos de datos de conversión automática sirven para cargar tipos de datos que Neptune no admite actualmente de forma nativa. Los datos de estas columnas se almacenan en forma de cadenas, textualmente, sin necesidad de compararlos con los formatos previstos. Se permiten los siguientes tipos de datos de conversión automática:

- **`Char`**: campo `Char`. Almacenado como una cadena.
- **`Date`**, **`LocalDate`** y **`LocalDateTime`**: consulte Neo4j Temporal Instants para ver una descripción de los tipos `date`, `localdate` y `localdatetime`. Los valores se cargan textualmente como cadenas, sin validación.
- **`Duration`**: consulte Neo4j Duration format. Los valores se cargan textualmente como cadenas, sin validación.
- Punto: campo de puntos para almacenar datos espaciales. Consulte Spatial instants. Los valores se cargan textualmente como cadenas, sin validación.

Ejemplo del formato de carga de openCypher

El siguiente diagrama, tomado del gráfico TinkerPop moderno, muestra un ejemplo de dos nodos y una relación:

A continuación, se muestra el gráfico en el formato de carga de openCypher normal de Neptune.

Archivo de nodos:

```
:ID,name:String,age:Int,lang:String,:LABEL
v1,"marko",29,,person
v2,"lop",,"java",software
```

Archivo de relaciones:

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,v1,v2,created,0.4
```

Como alternativa, puede utilizar los espacios de identificador y el identificador como una propiedad, de la siguiente manera:

Primer archivo de nodo:

```
name:ID(person),age:Int,lang:String,:LABEL
"marko",29,,person
```

Segundo archivo de nodo:

```
name:ID(software),age:Int,lang:String,:LABEL
"lop",,"java",software
```

Archivo de relaciones:

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,"marko","lop",created,0.4
```

Formatos de los datos de carga de RDF

Para cargar datos del marco de descripción de recursos (RDF), puede utilizar uno de los formatos estándar siguientes de la forma especificada por el World Wide Web Consortium (W3C):

- N-Triples (`ntriples`) de la especificación en https://www.w3.org/TR/n-triples/
- N-Quads (`nquads`) de la especificación en https://www.w3.org/TR/n-quads/
- RDF/XML (`rdfxml`) de la especificación en https://www.w3.org/TR/rdf-syntax-grammar/
- Turtle (`turtle`) de la especificación en https://www.w3.org/TR/turtle/

> **Important**
>
> Todos los archivos deben estar codificados en formato UTF-8.

Para los datos N-Quads y N-triples que incluyen caracteres Unicode, se admiten las secuencias de escape \u*xxxxx*. Sin embargo, Neptune no admite la normalización. Si hay un valor que requiere normalización, no coincidirá byte-to-byte durante la consulta. Para obtener más información acerca de la normalización, consulte la página Normalization en Unicode.org.

Siguientes pasos

Ahora que conoce mejor los formatos de carga, consulte Ejemplo: carga de datos en una instancia de base de datos de Neptune.

Ejemplo: carga de datos en una instancia de base de datos de Neptune

Este ejemplo muestra cómo cargar datos en Amazon Neptune. A menos que se indique lo contrario, debe seguir estos pasos desde una instancia de Amazon Elastic Compute Cloud (Amazon EC2) en la misma instancia de la nube privada virtual (VPC) de Amazon que su instancia de base de datos de Neptune.

Ejemplo de requisitos previos de carga de datos

Antes de comenzar, debe disponer de lo siguiente:

- Una instancia de base de datos de Neptune.

  Para obtener información acerca del inicio de una instancia de base de datos de Neptune, consulte Creación de un nuevo clúster de base de datos de Neptune.
- Un bucket de Amazon Simple Storage Service (Amazon S3) en el que colocar los datos.

  Es posible utilizar un bucket existente. Si no tiene un bucket de S3, consulte Crear un bucket en la Guía de introducción de Amazon S3.
- Datos del gráfico que se van a cargar, en uno de los formatos admitidos por el programa de carga de Neptune:

  Si utiliza Gremlin para consultar el gráfico, Neptune puede cargar datos en formato comma-separated-values (CSV), como se describe en. Formato de datos de carga de Gremlin

  Si utiliza openCypher para consultar el gráfico, Neptune también puede cargar datos en un formato CSV específico de openCypher, como se describe en Formato de carga para los datos de openCypher.

Si utiliza SPARQL, Neptune puede cargar los datos en una serie de formatos RDF, tal y como se describe en Formatos de los datos de carga de RDF.

- Un rol de IAM para que la instancia de base de datos de Neptune asuma que tiene una política de IAM que permite el acceso a los archivos de datos en el bucket de S3. La política debe conceder permisos de lectura y lista.

  Para obtener información acerca de cómo crear un rol con acceso a Amazon S3 y, después, asociarlo a un clúster de Neptune, consulte Requisitos previos: rol de IAM y acceso a Amazon S3.

  > **Note**
  >
  > La API `Load` de Neptune necesita acceso de lectura solo para los archivos de datos. No es necesario que la política de IAM conceda acceso de escritura o acceso al bucket completo.

- Un punto de conexión de VPC de Amazon S3. Para obtener más información, consulte la sección Creación de un punto de conexión de VPC de Amazon S3.

Creación de un punto de conexión de VPC de Amazon S3

El programa de carga de Neptune requiere un punto de conexión de VPC para Amazon S3.

Para configurar el acceso a Amazon S3

1. Inicie sesión en la consola de Amazon VPC AWS Management Console y ábrala en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación izquierdo, seleccione Puntos de conexión.
3. Seleccione Crear punto de conexión.
4. Elija el Service Name (Nombre del servicio) `com.amazonaws.`*`region`*`.s3`.

   > **Note**
   >
   > Si la región indicada es incorrecta, asegúrese de que la región de la consola es correcta.

5. Elija la VPC que contiene la instancia de base de datos de Neptune.
6. Seleccione la casilla de verificación situada junto a las tablas de ruteo asociadas a las subredes relacionadas con el clúster. Si solo tiene una tabla de ruteo, debe seleccionar esa casilla.
7. Seleccione Crear punto de conexión.

Para obtener más información acerca de la creación del punto de conexión, consulte Puntos de conexión de la VPC en la Guía del usuario de Amazon VPC. Para obtener información acerca de las limitaciones de los puntos de conexión de VPC, consulte VPC Endpoints for Amazon S3.

Para cargar datos en una instancia de base de datos de Neptune

1. Copie los archivos de datos en un bucket de Amazon S3. El bucket de S3 debe estar en la misma AWS región que el clúster que carga los datos.

   Puede usar el siguiente AWS CLI comando para copiar los archivos al bucket.

   > **Note**
   >
   > No es necesario ejecutar este comando desde la instancia de Amazon EC2.

   ```
   aws s3 cp data-file-name s3://bucket-name/object-key-name
   ```

   > **Note**
   >
   > En Amazon S3, un nombre de una clave de objeto es la ruta completa de un archivo, incluido el nombre de este.
   > Ejemplo: en el comando `aws s3 cp datafile.txt s3://examplebucket/mydirectory/datafile.txt`, el nombre de la clave de objeto es **`mydirectory/datafile.txt`**.

   Como alternativa, puede usar el AWS Management Console para cargar archivos en el bucket de S3. Abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/ y elija un bucket. En la esquina superior izquierda, elija Upload (Cargar) para cargar los archivos.

2. Desde una ventana de la línea de comandos, introduzca lo siguiente para ejecutar el programa de carga de Neptune con los valores correctos para su punto de conexión, la ruta de Amazon S3, el formato y el ARN del rol de IAM.

   El parámetro `format` puede tener cualquiera de los siguientes valores: `csv` para Gremlin, `opencypher` para openCypher o `ntriples`, `nquads`, `turtle` y `rdfxml` para RDF. Para

obtener información acerca del resto de parámetros, consulte Comando del programa de carga de Neptune.

Para obtener información acerca de cómo encontrar el nombre de host de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

El parámetro de región debe coincidir con la región del clúster y del bucket de S3.

Amazon Neptune está disponible en las siguientes regiones: AWS

- Este de EE. UU. (Norte de Virginia): `us-east-1`
- Este de EE. UU. (Ohio): `us-east-2`
- Oeste de EE. UU. (Norte de California): `us-west-1`
- Oeste de EE. UU. (Oregón): `us-west-2`
- Canadá (centro): `ca-central-1`
- América del Sur (São Paulo): `sa-east-1`
- Europa (Estocolmo): `eu-north-1`
- Europa (Irlanda): `eu-west-1`
- Europa (Londres): `eu-west-2`
- Europa (París): `eu-west-3`
- Europa (Fráncfort): `eu-central-1`
- Medio Oriente (Baréin): `me-south-1`
- Medio Oriente (EAU): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- África (Ciudad del Cabo): `af-south-1`
- Asia Pacífico (Hong Kong): `ap-east-1`
- Asia-Pacífico (Tokio): `ap-northeast-1`
- Asia-Pacífico (Seúl): `ap-northeast-2`
- Asia-Pacífico (Osaka): `ap-northeast-3`
- Asia-Pacífico (Singapur): `ap-southeast-1`
- Asia-Pacífico (Sídney): `ap-southeast-2`
- Asia-Pacífico (Bombay): `ap-south-1`

- China (Pekín): `cn-north-1`
- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (EE. UU.-Oeste): `us-gov-west-1`
- AWS GovCloud (EE. UU.-Este): `us-gov-east-1`

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "format",
      "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "TRUE",
      "dependencies" : ["load_A_id", "load_B_id"]
    }'
```

Para obtener información acerca de cómo crear y asociar un rol de IAM a un clúster de Neptune, consulte Requisitos previos: rol de IAM y acceso a Amazon S3.

> **Note**
>
> Consulte Parámetros de solicitudes del programa de carga de Neptune para obtener información detallada sobre los parámetros de solicitud de carga. En resumen:
> El parámetro `source` acepta un URI de Amazon S3 que apunta a un archivo único o a una carpeta. Si especifica una carpeta, Neptune carga todos los archivos de datos en esta.
> La carpeta puede contener varios archivos de vértice y varios archivos de borde.
> El URI puede tener cualquiera de los siguientes formatos:
>
> - `s3://bucket_name/object-key-name`
> - `https://s3.amazonaws.com/bucket_name/object-key-name`
> - `https://s3-us-east-1.amazonaws.com/bucket_name/object-key-name`

Este parámetro `format` puede ser uno de los siguientes:

- Formato CSV de Gremlin (`csv`) para gráfico de propiedades de Gremlin
- Formato CSV de openCypher (`opencypher`) para gráficos de propiedades de openCypher
- Formato N -Triples (`ntriples`) para RDF/SPARQL
- Formato N-Quads (`nquads`) para RDF/SPARQL
- Formato RDF/XML (`rdfxml`) para RDF/SPARQL
- Formato Turtle (`turtle`) para RDF/SPARQL

El parámetro opcional `parallelism` le permite restringir el número de subprocesos utilizados en el proceso de carga masiva. Se puede establecer en `LOW`, `MEDIUM`, `HIGH` o `OVERSUBSCRIBE`.
Cuando `updateSingleCardinalityProperties` se establece en `"FALSE"`, el cargador devuelve un error si se proporciona más de un valor en un archivo de origen que se está cargando para una propiedad de borde o de vértice de cardinalidad única.
Si `queueRequest` se establece en `"TRUE"`, la solicitud de carga se colocará en una cola si ya se está ejecutando un trabajo de carga.
El parámetro `dependencies` hace que la ejecución de la solicitud de carga dependa de la finalización correcta de uno o más trabajos de carga que ya se han colocado en la cola.

3. El programa de carga de Neptune devuelve un `id` de tarea que le permite comprobar el estado o cancelar el proceso de carga; por ejemplo:

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

4. Escriba lo siguiente para obtener el estado de la carga con el `loadId` del paso 3:

```
curl -G 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Si el estado de la carga muestra un error, puede solicitar un estado más detallado y una lista de los errores. Para obtener más información y ejemplos, consulte API de obtención de estado del programa de carga de Neptune.

5. (Opcional) Cancele la tarea `Load`.

   Escriba lo siguiente `Delete` la tarea del programa de carga con el `id` de tarea del paso 3:

   ```
   curl -X DELETE 'https://your-neptune-endpoint:port/loader/ef478d76-
   d9da-4d94-8ff1-08d9d4863aa5'
   ```

   El comando `DELETE` devuelve el código HTTP `200 OK` si la cancelación se realiza correctamente.

   Los datos de los archivos de la tarea de carga que han terminado de cargarse no se revierten, Los datos permanecen en la instancia de base de datos de Neptune.

Optimización de una carga masiva de Amazon Neptune

Utilice las siguientes estrategias para reducir al mínimo el tiempo de carga de una carga masiva de Neptune:

- Limpie sus datos:
  - Asegúrese de convertir los datos a un formato de datos compatible antes de cargarlos.
  - Elimine cualquier duplicado o error conocido.
  - Reduzca el número de predicados únicos (por ejemplo, propiedades de bordes y vértices) en la medida de lo posible.
- Optimice sus archivos:
  - Si carga archivos grandes, como archivos CSV, desde un bucket de Amazon S3, el programa de carga administra automáticamente la simultaneidad analizándolos en fragmentos que puede cargar en paralelo. El uso de una gran cantidad de archivos pequeños puede ralentizar este proceso.

  - Si carga varios archivos de una carpeta de Amazon S3, el programa de carga carga automáticamente primero los archivos de vértices y, después, los archivos de bordes.
  - La compresión de los archivos reduce los tiempos de transferencia. El programa de carga admite la compresión `gzip` de archivos de origen.
- Compruebe la configuración del programa de carga:
  - Si no necesita realizar ninguna otra operación durante la carga, utilice el parámetro `OVERSUBSCRIBEparallelism`. Esta configuración de parámetros hace que el programa de carga masiva utilice todos los recursos de CPU disponibles cuando se ejecuta. Por lo general, se necesita entre un 60 % y un 70 % de la capacidad de la CPU para que la operación se ejecute tan rápido como lo permitan las restricciones de E/S.

    > **Note**
    >
    > Cuando `parallelism` se establece en `OVERSUBSCRIBE` o `HIGH` (configuración predeterminada), al cargar datos de openCypher, existe el riesgo de que los subprocesos se encuentren en una condición de carrera y se bloqueen, lo que provoque un error `LOAD_DATA_DEADLOCK`. En este caso, ajuste `parallelism` a un valor inferior y vuelva a intentar realizar la carga.

  - Si su trabajo de carga incluye varias solicitudes de carga, utilice el parámetro `queueRequest`. Si configura `queueRequest` en `TRUE`, Neptune pone en cola sus solicitudes para que no tenga que esperar a que termine una para emitir otra.
  - Si las solicitudes de carga están en cola, puede configurar niveles de dependencia mediante el parámetro `dependencies`, de modo que si falla un trabajo, los trabajos dependientes también fallan. Esto puede evitar incoherencias en los datos cargados.
  - Si un trabajo de carga va a implicar la actualización de valores cargados anteriormente, asegúrese de establecer el parámetro `updateSingleCardinalityProperties` en `TRUE`. Si no lo hace, el programa de carga considerará un error el intento de actualizar un valor de cardinalidad única existente. En el caso de los datos de Gremlin, la cardinalidad también se especifica en los encabezados de las columnas de propiedades (consulte Encabezados de columnas de propiedades).

> **Note**
>
> El parámetro `updateSingleCardinalityProperties` no está disponible para los datos del marco de descripción de recursos (RDF).

- Puede usar el parámetro `failOnError` para determinar si las operaciones de carga masiva deben fallar o continuar cuando se detecta un error. Además, puede usar el parámetro `mode` para asegurarse de que un trabajo de carga reanude la carga desde el punto en el que falló un trabajo anterior, en lugar de volver a cargar los datos que ya se habían cargado.

- Escalamiento vertical: defina la instancia del escritor del clúster de base de datos en el tamaño máximo antes de cargarla de forma masiva. Tenga en cuenta que, si lo hace, también debe escalar verticalmente las instancias de réplica y lectura del clúster de base de datos o eliminarlas hasta que haya terminado de cargar los datos.

  Cuando se complete la carga masiva, asegúrese de volver a reducir verticalmente la instancia del escritor.

> **Important**
>
> Si experimenta un ciclo de reinicios repetidos de réplicas de lectura debido a un retardo en la replicación durante una carga masiva, es probable que sus réplicas no puedan mantener el ritmo del escritor en el clúster de base de datos. Escale los lectores a un tamaño superior al del escritor o quítelos temporalmente durante la carga masiva y vuelva a crearlos una vez finalizada la operación.

Consulte Parámetros de la solicitud para obtener más información sobre cómo configurar los parámetros de las solicitudes de carga.

Referencia del programa de carga de Neptune

En esta sección, se describen las API de `Loader` de Amazon Neptune que están disponibles desde el punto de conexión HTTP de una instancia de base de datos de Neptune.

> **Note**
>
> Consulte Mensajes de errores y fuente del programa de carga de Neptune para obtener una lista de los mensajes de error y de fuentes devueltos por el programa de carga en caso de errores.

Contenido

Comando del programa de carga de Neptune

Carga datos desde un bucket de Amazon S3 en una instancia de base de datos de Neptune.

Para cargar los datos, debe enviar una solicitud HTTP `POST` al punto de enlace `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. Los parámetros de la solicitud `loader` se pueden enviar en el cuerpo de `POST` o como parámetros codificados en URL.

> **Important**
>
> El tipo MIME debe ser `application/json`.

El depósito S3 debe estar en la misma AWS región que el clúster.

> **Note**
>
> Puede cargar datos cifrados desde Amazon S3 si se cifran mediante el modo `SSE-S3` de Amazon S3. En ese caso, Neptune es capaz de suplantar sus credenciales y emitir llamadas de `s3:getObject` en su nombre.
> También puede cargar datos cifrados desde Amazon S3 que se hayan cifrado mediante el modo `SSE-KMS`, siempre que el rol de IAM incluya los permisos necesarios para obtener acceso a AWS KMS. Sin AWS KMS los permisos adecuados, la operación de carga masiva falla y devuelve una `LOAD_FAILED` respuesta.
> Actualmente Neptune no admite la carga de datos cifrados de Amazon S3 con el modo `SSE-C`.

No tiene que esperar a que finalice un trabajo de carga para iniciar otro. Neptune puede poner en cola hasta 64 solicitudes de trabajo a la vez, siempre que todos sus parámetros `queueRequest` estén configurados como `"TRUE"`. El orden de espera de los trabajos será first-in-first-out (FIFO). Por otro lado, si no desea que un trabajo de carga esté en cola, puede establecer su parámetro `queueRequest` en `"FALSE"` (valor predeterminado), de modo que se producirá un error en el trabajo de carga si otro ya está en curso.

Puede utilizar el parámetro `dependencies` para poner en cola un trabajo que solo debe ejecutarse después de que los trabajos anteriores especificados en la cola se hayan completado correctamente. Si lo hace y se produce un error en cualquiera de esos trabajos especificados, su trabajo no se ejecutará y el estado se establecerá en `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Sintaxis de las solicitudes del programa de carga de Neptune

```
{
  "source" : "string",
  "format" : "string",
  "iamRoleArn" : "string",
  "mode": "NEW|RESUME|AUTO",
  "region" : "us-east-1",
  "failOnError" : "string",
  "parallelism" : "string",
  "parserConfiguration" : {
    "baseUri" : "http://base-uri-string",
    "namedGraphUri" : "http://named-graph-string"
  },
  "updateSingleCardinalityProperties" : "string",
  "queueRequest" : "TRUE",
  "dependencies" : ["load_A_id", "load_B_id"]
}
```

Parámetros de solicitudes del programa de carga de Neptune

- **`source`**: un URI de Amazon S3.

  El parámetro `SOURCE` acepta un URI de Amazon S3 que identifica un solo archivo, varios archivos, una carpeta o varias carpetas. Neptune carga todos los archivos de datos de cualquier carpeta especificada.

  El URI puede tener cualquiera de los siguientes formatos:

  - `s3://bucket_name/object-key-name`

- `https://s3.amazonaws.com/bucket_name/object-key-name`
- `https://s3.us-east-1.amazonaws.com/bucket_name/object-key-name`

El `object-key-name` elemento del URI equivale al parámetro de prefijo en una llamada a la ListObjectsAPI de Amazon S3. Identifica todos los objetos del bucket de Amazon S3 especificado cuyos nombres comienzan con ese prefijo. Puede ser un único archivo o carpeta o varios archivos o carpetas.

La carpeta o carpetas especificadas pueden contener varios archivos de vértice y varios archivos de borde.

Por ejemplo, si tuviera la siguiente estructura de carpetas y archivos en un bucket de Amazon S3 denominado`bucket-name`:

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
s3://bucket-name/bcd
```

Si el parámetro de origen se especifica como`s3://bucket-name/a`, se cargarán los tres primeros archivos.

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
```

- **`format`**: el formato de los datos. Para obtener más información acerca de los formatos de los datos para el comando `Loader` de Neptune, consulte Uso del programa de carga masiva de Amazon Neptune para adquirir datos.

  Valores permitidos

  - **`csv`** para el formato de datos CSV de Gremlin.
  - **`opencypher`** para el formato de datos CSV de openCypher.
  - **`ntriples`** para el formato de datos RDF N-Triples.
  - **`nquads`** para el formato de datos RDF N-Quads.
  - **`rdfxml`** para el formato de datos RDF RDF\XML.
  - **`turtle`** para el formato de datos RDF de Turtle.

- **`iamRoleArn`**: nombre de recurso de Amazon (ARN) para que la instancia de base de datos de Neptune asuma el rol de IAM para obtener acceso al bucket de S3. Para obtener información acerca de cómo crear un rol con acceso a Amazon S3 y, después, asociarlo a un clúster de Neptune, consulte Requisitos previos: rol de IAM y acceso a Amazon S3.

  A partir de la versión 1.2.1.0.R3 del motor, también puede encadenar varias funciones de IAM si la instancia de base de datos Neptune y el bucket de Amazon S3 están ubicados en cuentas diferentes. AWS En este caso, `iamRoleArn` contiene una lista de ARN de roles separados por comas, tal y como se describe en Encadenamiento de roles de IAM en Amazon Neptune. Por ejemplo:

```
curl -X POST https://localhost:8182/loader \
  -H 'Content-Type: application/json' \
  -d '{
        "source" : "s3://(the target bucket name)/(the target date file name)",
        "iamRoleArn" : "arn:aws:iam::(Account A
 ID):role/(RoleA),arn:aws:iam::(Account B ID):role/(RoleB),arn:aws:iam::(Account C
 ID):role/(RoleC)",
        "format" : "csv",
        "region" : "us-east-1"
      }'
```

- **`region`**— El `region` parámetro debe coincidir con la AWS región del clúster y el bucket de S3.

  Amazon Neptune está disponible en las siguientes regiones de :

  - Este de EE. UU. (Norte de Virginia): `us-east-1`
  - Este de EE. UU. (Ohio): `us-east-2`
  - Oeste de EE. UU. (Norte de California): `us-west-1`
  - Oeste de EE. UU. (Oregón): `us-west-2`
  - Canadá (centro): `ca-central-1`
  - América del Sur (São Paulo): `sa-east-1`
  - Europa (Estocolmo): `eu-north-1`
  - Europa (Irlanda): `eu-west-1`
  - Europa (Londres): `eu-west-2`
  - Europa (París): `eu-west-3`
  - Europa (Fráncfort): `eu-central-1`
  - Medio Oriente (Baréin): `me-south-1`

- Medio Oriente (EAU): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- África (Ciudad del Cabo): `af-south-1`
- Asia Pacífico (Hong Kong): `ap-east-1`
- Asia-Pacífico (Tokio): `ap-northeast-1`
- Asia-Pacífico (Seúl): `ap-northeast-2`
- Asia-Pacífico (Osaka): `ap-northeast-3`
- Asia-Pacífico (Singapur): `ap-southeast-1`
- Asia-Pacífico (Sídney): `ap-southeast-2`
- Asia-Pacífico (Bombay): `ap-south-1`
- China (Pekín): `cn-north-1`
- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (EE. UU.-Oeste): `us-gov-west-1`
- AWS GovCloud (EE. UU.-Este): `us-gov-east-1`

- **`mode`**: el modo de trabajo de carga.

  Valores permitidos: `RESUME`, `NEW`, `AUTO`.

  Valor predeterminado: `AUTO`

  - `RESUME`: en el modo RESUME, el programa de carga busca una carga anterior de este origen y, si encuentra una, reanuda ese trabajo de carga. Si no se encuentra ningún trabajo de carga anterior, el programa de carga se detiene.

    El programa de carga evita la recarga de archivos cargados correctamente en un trabajo anterior. Solo intenta procesar los archivos con errores. Si ha eliminado los datos cargados anteriormente del clúster de Neptune, esos datos no se vuelven a cargar en este modo. Si un trabajo de carga anterior ha cargado todos los archivos del mismo origen correctamente, no se vuelve a cargar nada y el programa de carga devuelve una operación correcta.
  - `NEW`: en el modo NEW, crea una solicitud de carga, independientemente de cualquier carga anterior. Puede utilizar este modo para volver a cargar todos los datos de un origen después de descartar los datos cargados anteriormente desde el clúster de Neptune o bien para cargar nuevos datos disponibles en el mismo origen.

- `AUTO`: en el modo AUTO, el programa de carga busca un trabajo de carga anterior del mismo origen y, si encuentra uno, lo reanuda, igual que en el modo `RESUME`.

  Si el programa de carga no encuentra un trabajo de carga anterior del mismo origen, carga todos los datos del origen, al igual que en el modo `NEW`.

- **`failOnError`**: un indicador para activar la detención total al encontrar un error.

  Valores permitidos: `"TRUE"` y `"FALSE"`.

  Valor predeterminado: `"TRUE"`.

  Cuando este parámetro se establece en `"FALSE"`, el programa de carga intenta cargar todos los datos de la ubicación especificada, omitiendo cualquier entrada con errores.

  Cuando este parámetro se establece en `"TRUE"`, el programa de carga se detiene en cuanto encuentra un error. Los datos cargados hasta ese momento persisten.

- **`parallelism`**: es un parámetro opcional que se puede establecer para reducir el número de subprocesos utilizados por el proceso de carga masiva.

  Valores permitidos:

  - `LOW`: el número de subprocesos utilizados es el número de vCPU disponibles dividido entre 8.
  - `MEDIUM`: el número de subprocesos utilizados es el número de vCPU disponibles dividido entre 2.
  - `HIGH`: el número de subprocesos utilizados es el mismo número de vCPU disponibles.
  - `OVERSUBSCRIBE`: el número de subprocesos utilizados es el número de vCPU disponibles multiplicado por 2. Si se utiliza este valor, el programa de carga masiva absorbe todos los recursos disponibles.

    Sin embargo, esto no significa que el ajuste de `OVERSUBSCRIBE` dé como resultado un uso del 100 % de la CPU. Dado que la operación de carga está vinculada a la E/S, la máxima utilización de la CPU que cabe esperar se sitúa entre el 60 y el 70 %.

  Valor predeterminado: `HIGH`

  En ocasiones, este ajuste de `parallelism` puede provocar un bloqueo entre los subprocesos al cargar datos de openCypher. Cuando esto ocurre, Neptune devuelve el error `LOAD_DATA_DEADLOCK`. Por lo general, puede solucionar el problema configurando `parallelism` en un ajuste inferior y volviendo a intentar ejecutar el comando de carga.

- **`parserConfiguration`**: objeto opcional con valores de configuración de analizador adicionales. Cada uno de los parámetros secundarios también es opcional:

Nombre	Ejemplo de valor	Descripción
`namedGraphUri`	*`http://aws.amazon.com/neptune/vocab/v01/ GráficoDefaultNamed`*	El gráfico predeterminado para todos los formatos RDF cuando no se especific a ningún gráfico (para formatos no QUAD y entradas NQUAD sin gráfico). El valor predeterminado es `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`
`baseUri`	*`http://aws.amazon.com/neptune/default`*	El URI base para los formatos RDF/XML y Turtle. El valor predeterminado es `http://aws.amazon.com/neptune/default` .

`allowEmptyStrings`	*true*	Los usuarios de Gremlin deben poder pasar valores de cadenas vacías (“”) como propiedades de nodo y borde al cargar datos CSV. Si `allowEmptyStrings` se establece en `false` (el valor predeterminado), estas cadenas vacías se tratan como nulas y no se cargan. Si `allowEmptyStrings` se establece en `true`, el programa de carga trata las cadenas vacías como valores de propiedad válidos y las carga en consecuencia.

Para obtener más información, consulte Gráfico predeterminado SPARQL y gráficos con nombre.

- **`updateSingleCardinalityProperties`**: es un parámetro opcional que controla cómo el programa de carga masiva trata un nuevo valor para las propiedades de vértice o borde de cardinalidad única. Esto no se admite para cargar datos de openCypher (consulte Carga de datos de openCypher).

  Valores permitidos: `"TRUE"` y `"FALSE"`.

  Valor predeterminado: `"FALSE"`.

  De forma predeterminada, o cuando `updateSingleCardinalityProperties` está configurado explícitamente en `"FALSE"`, el programa de carga trata un nuevo valor como un error, porque infringe la cardinalidad única.

  Por el contrario, cuando `updateSingleCardinalityProperties` está configurado en `"TRUE"`, el programa de carga en bloque reemplaza el valor existente por el nuevo. Si se proporcionan varios valores de propiedades de vértices de borde o de cardinalidad única en los archivos origen que se están cargando, el valor final al terminar la carga masiva podría ser cualquiera de esos

nuevos valores. El programa de carga solo garantiza que el valor existente se ha reemplazado por uno de los nuevos.

- **`queueRequest`**: es un parámetro de indicador opcional que indica si la solicitud de carga se puede poner en cola o no.

  No tiene que esperar a que se complete un trabajo de carga antes de emitir el siguiente, porque Neptune puede poner en cola hasta 64 trabajos a la vez, siempre que sus parámetros `queueRequest` estén configurados en `"TRUE"`. El orden de espera de los trabajos será first-in-first-out (FIFO).

  Si el parámetro `queueRequest` se omite o se establece en `"FALSE"`, se producirá un error en la solicitud de carga si ya se está ejecutando otro trabajo de carga.

  Valores permitidos: `"TRUE"` y `"FALSE"`.

  Valor predeterminado: `"FALSE"`.

- **`dependencies`**: se trata de un parámetro opcional que puede supeditar una solicitud de carga en cola a la finalización satisfactoria de uno o más trabajos anteriores de la cola.

  Neptune puede poner en cola hasta 64 solicitudes de carga a la vez, si sus parámetros `queueRequest` están configurados en `"TRUE"`. El parámetro `dependencies` le permite hacer que la ejecución de dicha solicitud en cola dependa de la finalización correcta de una o más solicitudes anteriores especificadas en la cola.

  Por ejemplo, si las cargas `Job-A` y `Job-B` son independientes entre sí, pero la carga `Job-C` necesita `Job-A` y `Job-B` debe terminar antes de que comience, proceda de la siguiente manera:

  1. Envíe `load-job-A` y `load-job-B` uno tras otro en cualquier orden, y guarde sus identificadores de carga.
  2. Envíe `load-job-C` con los identificadores de carga de los dos trabajos en su campo `dependencies`:

  ```
  "dependencies" : ["job_A_load_id", "job_B_load_id"]
  ```

  Debido al parámetro `dependencies`, el programa de carga en bloque no iniciará `Job-C` hasta que `Job-A` y `Job-B` se hayan completado correctamente. Si se produce un error en alguno de ellos, Job-C no se ejecutará y su estado se establecerá en `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Puede configurar varios niveles de dependencia de esta manera, de modo que el error de un trabajo provoque la cancelación de todas las solicitudes que dependen directa o indirectamente de él.

- **`userProvidedEdgeIds`**: este parámetro solo es necesario cuando se cargan datos de openCypher que contienen identificadores de relación. Debe incluirse y configurarse en `True` cuando los identificadores de relación de openCypher se proporcionen de forma explícita en los datos de carga (recomendado).

  Si `userProvidedEdgeIds` está ausente o se establece en `True`, debe haber una columna `:ID` en todos los archivos de relaciones de la carga.

  Cuando `userProvidedEdgeIds` está presente y se establece en `False`, los archivos de relaciones de la carga no deben contener ninguna columna `:ID`. En su lugar, el programa de carga de Neptune genera automáticamente un identificador para cada relación.

  Resulta útil proporcionar los identificadores de relación de forma explícita para que el programa de carga pueda reanudar la carga una vez que se haya corregido un error en los datos CSV, sin tener que volver a cargar ninguna relación que ya se haya cargado. Si los identificadores de relación no se han asignado de forma explícita, el programa de carga no puede reanudar una carga fallida si se ha tenido que corregir algún archivo de relación y, en su lugar, debe volver a cargar todas las relaciones.
- `accessKey`: [en desuso] un identificador de clave de acceso de un rol de IAM con acceso a los archivos de datos y al bucket de S3.

  En su lugar, se recomienda el parámetro `iamRoleArn`. Para obtener información acerca de cómo crear un rol con acceso a Amazon S3 y, después, asociarlo a un clúster de Neptune, consulte Requisitos previos: rol de IAM y acceso a Amazon S3.

  Para obtener más información, consulte Claves de acceso (ID de clave de acceso y clave de acceso secreta).
- `secretKey`: [en desuso] en su lugar, se recomienda el parámetro `iamRoleArn`. Para obtener información acerca de cómo crear un rol con acceso a Amazon S3 y, después, asociarlo a un clúster de Neptune, consulte Requisitos previos: rol de IAM y acceso a Amazon S3.

  Para obtener más información, consulte Claves de acceso (ID de clave de acceso y clave de acceso secreta).

Consideraciones especiales para cargar datos de openCypher

- Al cargar datos de openCypher en formato CSV, el parámetro de formato debe estar establecido en `opencypher`.
- El parámetro `updateSingleCardinalityProperties` no es compatible con las cargas de openCypher, porque todas las propiedades de openCypher tienen una cardinalidad única. El formato de carga de openCypher no admite matrices y, si un valor de identificador aparece más de una vez, se trata como un duplicado o un error de inserción (véase más abajo).
- El programa de carga de Neptune gestiona los duplicados que encuentra en los datos de openCypher de la siguiente manera:
  - Si el programa de carga encuentra varias filas con el mismo identificador de nodo, se fusionan según la siguiente regla:
    - Todas las etiquetas de las filas se añaden al nodo.
    - Para cada propiedad, solo se carga uno de los valores de la propiedad. La selección del que se va a cargar es no determinista.
  - Si el programa de carga encuentra varias filas con el mismo identificador de relación, solo se carga una de ellas. La selección del que se va a cargar es no determinista.
  - El programa de carga nunca actualiza los valores de las propiedades de un nodo o relación existente en la base de datos si encuentra datos de carga con el identificador del nodo o la relación existente. Sin embargo, carga etiquetas y propiedades de los nodos que no están presentes en el nodo o la relación existentes.
- Aunque no es necesario asignar identificadores a las relaciones, suele ser una buena idea (consulte el parámetro `userProvidedEdgeIds` anterior). Sin identificadores de relación explícitos, el programa de carga debe volver a cargar todas las relaciones en caso de que se produzca un error en un archivo de relaciones, en lugar de reanudar la carga desde donde se produjo el error.

  Además, si los datos de carga no contienen identificadores de relación explícitos, el programa de carga no tiene forma de detectar relaciones duplicadas.

A continuación se ofrece un ejemplo de un comando de carga de openCypher:

```
curl -X POST https://your-neptune-endpoint:port/loader \
     -H 'Content-Type: application/json' \
     -d '
```

```
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "opencypher",
      "userProvidedEdgeIds": "TRUE",
      "iamRoleArn" : "arn:aws:iam::account-id:role/role-name",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
    }'
```

La respuesta del programa de carga es la misma que la normal. Por ejemplo:

```
{
  "status" : "200 OK",
  "payload" : {
    "loadId" : "guid_as_string"
  }
}
```

Sintaxis de respuestas del programa de carga de Neptune

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "guid_as_string"
    }
}
```

200 OK

Una tarea de carga que se inició correctamente devuelve el código `200`.

Errores del programa de carga de Neptune

Si se produce un error, se devuelve un objeto JSON en el elemento `BODY` de la respuesta. El objeto `message` contiene una descripción del error.

Categorías de errores

- `Error 400`: los errores de sintaxis devuelven un error de solicitud incorrecta HTTP `400`. El mensaje describe el error.

- `Error 500`: una solicitud válida que no se puede procesar devuelve un error interno del servidor HTTP `500`. El mensaje describe el error.

A continuación se muestran los posibles mensajes de error del programa de carga con la descripción correspondiente.

Mensajes de error del programa de carga

- `Couldn't find the AWS credential for iam_role_arn` (HTTP 400)

  No se encontraron las credenciales. Compruebe las credenciales suministradas con la consola o la salida de IAM. AWS CLI Asegúrese de haber añadido el rol de IAM especificado en `iamRoleArn` al clúster.
- `S3 bucket not found for source` (HTTP 400)

  El bucket de S3 no existe. Compruebe el nombre del bucket.
- `The source` *`source-uri`* `does not exist/not reachable` (HTTP 400)

  No se encontraron archivos coincidentes en el bucket de S3.
- `Unable to connect to S3 endpoint. Provided source =` *`source-uri`* `and region =` *`aws-region`* (HTTP 500)

  No es posible conectarse a Amazon S3. La región debe coincidir con la región del clúster. Asegúrese de que tiene un punto de enlace de la VPC. Para obtener información acerca de cómo crear un punto de enlace de la VPC, consulte Creación de un punto de conexión de VPC de Amazon S3.
- `Bucket is not in provided Region (`*`aws-region`*`)` (HTTP 400)

  El bucket debe estar en la misma AWS región que la instancia de base de datos de Neptune.
- `Unable to perform S3 list operation` (HTTP 400)

  La función o el usuario de IAM que se ha proporcionado no tiene permisos `List` en el bucket o en la carpeta. Compruebe la política o la lista de control de acceso (ACL) en el bucket.
- `Start new load operation not permitted on a read replica instance` (HTTP 405)

  La carga es una operación de escritura. Pruebe a volver a cargar en el punto de enlace del clúster de lectura/escritura.

- `Failed to start load because of unknown error from S3` (HTTP 500)

  Amazon S3 ha devuelto un error desconocido. Póngase en contacto con AWS Support.

- `Invalid S3 access key` (HTTP 400)

  La clave de acceso no es válida. Compruebe las credenciales proporcionadas.

- `Invalid S3 secret key` (HTTP 400)

  La clave secreta no es válida. Compruebe las credenciales proporcionadas.

- `Max concurrent load limit breached` (HTTP 400)

  Si una solicitud de carga se envía sin `"queueRequest" : "TRUE"` y un trabajo de carga se está ejecutando actualmente, se producirá este error en la solicitud.

- `Failed to start new load for the source "source name". Max load task queue size limit breached. Limit is 64` (HTTP 400)

  Neptune permite poner en cola hasta 64 trabajos de carga a la vez. Si se envía una solicitud de carga adicional a la cola cuando ya contiene 64 trabajos, se producirá un error en la solicitud con este mensaje.

Ejemplos del programa de carga de Neptune

Example Solicitud

A continuación, se muestra una solicitud enviada mediante HTTP POST con el comando `curl` Carga un archivo con el formato CSV de Neptune. Para obtener más información, consulte Formato de datos de carga de Gremlin.

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your-neptune-endpoint:port/loader -d '
    {
      "source" : "s3://bucket-name/object-key-name",
      "format" : "csv",
      "iamRoleArn" : "ARN for the IAM role you are using",
      "region" : "region",
      "failOnError" : "FALSE",
      "parallelism" : "MEDIUM",
      "updateSingleCardinalityProperties" : "FALSE",
      "queueRequest" : "FALSE"
```

```
    }'
```

Example Respuesta

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

API de obtención de estado del programa de carga de Neptune

Obtiene el estado de una tarea `loader`.

Para obtener el estado de carga, debe enviar una solicitud HTTP `GET` al punto de enlace `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`. Para obtener el estado de una solicitud de carga concreta, debe incluir `loadId` como parámetro de la URL o añadir `loadId` a la ruta de la dirección URL.

Neptune solo realiza un seguimiento de los 1024 trabajos de carga masiva más recientes y solo almacena los últimos 10 000 detalles de error por trabajo.

Consulte Mensajes de errores y fuente del programa de carga de Neptune para obtener una lista de los mensajes de error y de fuentes devueltos por el programa de carga en caso de errores.

Contenido

Solicitudes de obtención de estado del programa de carga de Neptune

Sintaxis de solicitudes de obtención de estado del programa de carga

```
GET https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
GET https://your-neptune-endpoint:port/loader/loadId
```

```
GET https://your-neptune-endpoint:port/loader
```

Parámetros de las solicitudes de obtención de estado del programa de carga de Neptune

- **`loadId`**: el identificador del trabajo de carga. Si no se especifica el valor `loadId`, se devuelve una lista de los ID de carga.
- **`details`**: incluye otros detalles aparte del estado general.

  Valores permitidos: `TRUE` y `FALSE`.

  Valor predeterminado: `FALSE`.
- **`errors`**: incluye la lista de errores.

  Valores permitidos: `TRUE` y `FALSE`.

  Valor predeterminado: `FALSE`.

  Dicha lista está paginada. Los parámetros `page` y `errorsPerPage` le permiten desplazarse por todos los errores.
- **`page`**: el número de la página de error. Solo es válido si el conjunto de parámetros `errors` está establecido en `TRUE`.

  Valores permitidos: números enteros positivos.

Valor predeterminado: 1.

- **`errorsPerPage`**: el número de errores por cada página. Solo es válido si el conjunto de parámetros `errors` está establecido en `TRUE`.

  Valores permitidos: números enteros positivos.

  Valor predeterminado: 10.

- **`limit`**: el número de identificadores de carga que se va a incluir en la lista. Solo es válido cuando se solicita una lista de los ID de carga mediante el envío de una solicitud `GET` sin ningún `loadId` especificado.

  Valores permitidos: números enteros positivos del 1 al 100.

  Valor predeterminado: 100.

- **`includeQueuedLoads`**: un parámetro opcional que se puede utilizar para excluir los identificadores de carga de las solicitudes de carga en cola cuando se solicita una lista de identificadores de carga.

  > **Note**
  >
  > Este parámetro está disponible a partir de la versión 1.0.4.0 del motor de Neptune.

  De forma predeterminada, los ID de carga de todos los trabajos de carga con el estado `LOAD_IN_QUEUE` se incluyen en dicha lista. Aparecen delante de los ID de carga de otros trabajos, ordenados por el momento en que se agregaron a la cola de más reciente a más temprano.

  Valores permitidos: `TRUE` y `FALSE`.

  Valor predeterminado: `TRUE`.

Respuestas de obtención de estado del programa de carga de Neptune

Diseño JSON de respuestas de obtención de estado del programa de carga de Neptune

El diseño general de la respuesta de estado de un programa de carga es el siguiente:

```
{
```

```
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : number
            }
        ],
        "overallStatus" : {
            "fullUri" : "s3://bucket/key",
            "runNumber" : number,
            "retryNumber" : number,
            "status" : "string",
            "totalTimeSpent" : number,
            "startTime" : number,
            "totalRecords" : number,
            "totalDuplicates" : number,
            "parsingErrors" : number,
            "datatypeMismatchErrors" : number,
            "insertErrors" : number,
        },
        "failedFeeds" : [
            {
                "fullUri" : "s3://bucket/key",
                "runNumber" : number,
                "retryNumber" : number,
                "status" : "string",
                "totalTimeSpent" : number,
                "startTime" : number,
                "totalRecords" : number,
                "totalDuplicates" : number,
                "parsingErrors" : number,
                "datatypeMismatchErrors" : number,
                "insertErrors" : number,
            }
        ],
        "errors" : {
            "startIndex" : number,
            "endIndex" : number,
            "loadId" : "string",
            "errorLogs" : [ ]
        }
    }
}
```

Objetos de respuesta **`overallStatus`** y **`failedFeeds`** de obtención de estado del programa de carga de Neptune

Las posibles respuestas devueltas para cada fuente fallida, incluidas las descripciones de los errores, son las mismas que para el objeto `overallStatus` de una respuesta `Get-Status`.

Los siguientes campos aparecen en el objeto `overallStatus` de todas las cargas y en el objeto `failedFeeds` de cada fuente fallida:

- **`fullUri`**: el URI del archivo o archivos que se van a cargar.

  Tipo: string

  Formato: `s3://`*`bucket`*`/`*`key`*.
- **`runNumber`**: el número de ejecución de la carga o fuente. Aumenta cuando se reinicia la carga.

  Tipo: entero largo sin signo.
- **`retryNumber`**: el número de reintento de la carga o fuente. Aumenta cuando el programa de carga reintenta automáticamente realizar una carga o procesar una fuente.

  Tipo: entero largo sin signo.
- **`status`**: el estado que devuelve la carga o la fuente. `LOAD_COMPLETED` indica una carga correcta sin problemas. Para ver una lista de otros mensajes de estado de carga, consulte Mensajes de errores y fuente del programa de carga de Neptune.

  Tipo: string.
- **`totalTimeSpent`**: el tiempo, en segundos, dedicado a analizar e insertar datos para la carga o la fuente. No incluye el tiempo dedicado a obtener la lista de archivos de origen.

  Tipo: entero largo sin signo.
- **`totalRecords`**: número total de registros que se han cargado o que se ha intentado cargar.

  Tipo: entero largo sin signo.

  Tenga en cuenta que al cargar desde un archivo CSV, el recuento de registros no se refiere al número de líneas cargadas, sino al número de registros individuales en esas líneas. Tomemos como ejemplo un archivo CSV pequeño como este:

```
~id,~label,name,team
```

```
'P-1','Player','Stokes','England'
```

Neptune consideraría que este archivo contiene 3 registros:

```
P-1  label Player
P-1  name  Stokes
P-1  team  England
```

- **`totalDuplicates`**: el número de registros duplicados que se ha encontrado.

  Tipo: entero largo sin signo.

  Como en el caso del recuento de `totalRecords`, este valor contiene el número de registros duplicados individuales de un archivo CSV, no el número de líneas duplicadas. Tomemos como ejemplo este archivo CSV pequeño:

```
~id,~label,name,team
P-2,Player,Kohli,India
P-2,Player,Kohli,India
```

  El estado devuelto después de cargarlo tendría este aspecto, con un total de 6 registros, de los cuales 3 son duplicados:

```
{
  "status": "200 OK",
  "payload": {
    "feedCount": [
      {
        "LOAD_COMPLETED": 1
      }
    ],
    "overallStatus": {
      "fullUri": "(the URI of the CSV file)",
      "runNumber": 1,
      "retryNumber": 0,
      "status": "LOAD_COMPLETED",
      "totalTimeSpent": 3,
      "startTime": 1662131463,
      "totalRecords": 6,
      "totalDuplicates": 3,
      "parsingErrors": 0,
      "datatypeMismatchErrors": 0,
```

```
      "insertErrors": 0
    }
  }
}
```

En el caso de las cargas de openCypher, se cuenta un duplicado cuando:

- El programa de carga detecta que una fila de un archivo de nodo tiene un identificador sin un espacio de identificador que es igual a otro valor de identificador sin espacio de identificador, ya sea en otra fila o que pertenezca a un nodo existente.
- El programa de carga detecta que una fila de un archivo de nodo tiene un identificador con un espacio de identificador que es igual a otro valor de identificador con espacio de identificador, ya sea en otra fila o que pertenezca a un nodo existente.

Consulte Consideraciones especiales para cargar datos de openCypher.

- **`parsingErrors`**: el número de errores de análisis encontrados.

  Tipo: entero largo sin signo.
- **`datatypeMismatchErrors`**: el número de registros con un tipo de datos que no coincide con los datos proporcionados.

  Tipo: entero largo sin signo.
- **`insertErrors`**: el número de registros que no se han podido insertar debido a errores.

  Tipo: entero largo sin signo.

Objeto de respuesta **`errors`** de obtención de estado del programa de carga de Neptune

Los errores se dividen en las categorías siguientes:

- **`Error 400`**: un `loadId` no válido devuelve un error de solicitud incorrecta HTTP `400`. El mensaje describe el error.
- **`Error 500`**: una solicitud válida que no se puede procesar devuelve un error interno del servidor HTTP `500`. El mensaje describe el error.

Consulte Mensajes de errores y fuente del programa de carga de Neptune para obtener una lista de los mensajes de error y de fuentes devueltos por el programa de carga en caso de errores.

Si se produce un error, se devuelve un objeto JSON `errors` en el elemento `BODY` de la respuesta con los siguientes campos:

- **`startIndex`**: el índice del primer error incluido.

  Tipo: entero largo sin signo.
- **`endIndex`**: el índice del último error incluido.

  Tipo: entero largo sin signo.
- **`loadId`**: el identificador de la carga. Puede utilizar este ID para imprimir los errores de la carga si establece el parámetro `errors` en `TRUE`.

  Tipo: string.
- **`errorLogs`**: una lista de los errores.

  Tipo: lista.

Objeto de respuesta **`errorLogs`** de obtención de estado del programa de carga de Neptune

El objeto `errorLogs` que aparece en `errors` en la respuesta del estado de obtención del programa de carga contiene un objeto que describe cada error mediante los siguientes campos:

- **`errorCode`**: identifica la naturaleza del error.

  Puede tener uno de los siguientes valores:
  - `PARSING_ERROR`
  - `S3_ACCESS_DENIED_ERROR`
  - `FROM_OR_TO_VERTEX_ARE_MISSING`
  - `ID_ASSIGNED_TO_MULTIPLE_EDGES`
  - `SINGLE_CARDINALITY_VIOLATION`
  - `FILE_MODIFICATION_OR_DELETION_ERROR`
  - `OUT_OF_MEMORY_ERROR`
  - `INTERNAL_ERROR` (se devuelve cuando el programa de carga masiva no puede determinar el tipo de error).
- **`errorMessage`**: mensaje que describe el error.

Puede ser un mensaje genérico asociado al código de error o un mensaje específico que contenga detalles, por ejemplo, sobre la falta de un vértice de origen o destino o sobre un error de análisis.

- **`fileName`**: el nombre de la fuente.
- **`recordNum`**: en el caso de un error de análisis, se trata del número de registro del archivo del registro que no se ha podido analizar. Se establece en cero si el número de registro no es aplicable al error o si no se ha podido determinar.

Por ejemplo, el programa de carga masiva generaría un error de análisis si encontrara una fila fallida como la siguiente en un archivo RDF `nquads`:

```
<http://base#subject> |http://base#predicate> <http://base#true> .
```

Como puede ver, el segundo `http` de la fila anterior debería ir precedido de < en lugar de `|`. El objeto de error resultante en `errorLogs` en una respuesta de estado tendría el siguiente aspecto:

```
{
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Expected '<', found: |",
    "fileName" : "s3://bucket/key",
    "recordNum" : 12345
},
```

Ejemplos de obtención de estado del programa de carga de Neptune

Ejemplo de solicitud del estado de carga

A continuación, se muestra una solicitud enviada mediante HTTP `GET` con el comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)'
```

Example Respuesta

```
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
```

```
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Ejemplo de solicitud de loadIds

A continuación, se muestra una solicitud enviada mediante HTTP `GET` con el comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader?limit=3'
```

Example Respuesta

```
{
    "status" : "200 OK",
    "payload" : {
         "loadIds" : [
            "a2c0ce44-a44b-4517-8cd4-1dc144a8e5b5",
            "09683a01-6f37-4774-bb1b-5620d87f1931",
            "58085eb8-ceb4-4029-a3dc-3840969826b9"
        ]
    }
}
```

Ejemplo de solicitud del estado detallado

A continuación, se muestra una solicitud enviada mediante HTTP `GET` con el comando `curl`.

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true'
```

Example Respuesta


```
{
    "status" : "200 OK",
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        }
    }
}
```

Ejemplos de **`errorLogs`** de obtención de estado del programa de carga de Neptune

Ejemplo de respuesta de estado detallada cuando se produjeron errores

Se trata de una solicitud enviada a través de HTTP `GET` mediante `curl`:

```
curl -X GET 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3'
```

Example de una respuesta detallada cuando se produjeron errores

Este es un ejemplo de la respuesta que podría obtener de la consulta anterior, con un objeto `errorLogs` que muestre los errores de carga detectados:

```
{
    "status" : "200 OK",
    "payload" : {
        "failedFeeds" : [
            {
                "datatypeMismatchErrors" : 0,
                "fullUri" : "s3://bucket/key",
                "insertErrors" : 0,
                "parsingErrors" : 5,
                "retryNumber" : 0,
                "runNumber" : 1,
                "status" : "LOAD_FAILED",
                "totalDuplicates" : 0,
                "totalRecords" : 5,
                "totalTimeSpent" : 3.0
            }
        ],
        "feedCount" : [
            {
                "LOAD_FAILED" : 1
            }
        ],
        "overallStatus" : {
            "datatypeMismatchErrors" : 0,
            "fullUri" : "s3://bucket/key",
            "insertErrors" : 0,
            "parsingErrors" : 5,
            "retryNumber" : 0,
            "runNumber" : 1,
```

```
            "status" : "LOAD_FAILED",
            "totalDuplicates" : 0,
            "totalRecords" : 5,
            "totalTimeSpent" : 3.0
        },
        "errors" : {
            "endIndex" : 3,
            "errorLogs" : [
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 1
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 2
                },
                {
                    "errorCode" : "PARSING_ERROR",
                    "errorMessage" : "Expected '<', found: |",
                    "fileName" : "s3://bucket/key",
                    "recordNum" : 3
                }
            ],
            "loadId" : "0a237328-afd5-4574-a0bc-c29ce5f54802",
            "startIndex" : 1
        }
    }
}
```

Ejemplo de un error **`Data prefetch task interrupted`**

Ocasionalmente, al obtener un estado `LOAD_FAILED` y solicitar a continuación información más detallada, el error devuelto puede ser un `PARSING_ERROR` con un mensaje `Data prefetch task interrupted`, como este:

```
"errorLogs" : [
    {
        "errorCode" : "PARSING_ERROR",
```

```
        "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
        "fileName" : "s3://some-source-bucket/some-source-file",
        "recordNum" : 0
    }
]
```

Este error se produce si ha habido una interrupción temporal en el proceso de carga de datos no provocada normalmente por su solicitud o sus datos. En general, puede resolverse simplemente ejecutando de nuevo la solicitud de carga masiva. En caso de que esté usando la configuración predeterminada, es decir, `"mode":"AUTO"` y `"failOnError":"TRUE"`, el programa de carga omite los archivos que ya ha cargado correctamente y reanuda la carga de los archivos que aún no había cargado en el momento de la interrupción.

Cancelación de trabajo del programa de carga de Neptune

Cancela un trabajo de carga.

Para cancelar un trabajo, debe enviar una solicitud HTTP `DELETE` al punto de enlace `https://your-neptune-endpoint:port/loader`. El valor `loadId` se puede añadir a la ruta de la dirección URL `/loader` o bien incluirse como variable en la URL.

Sintaxis de solicitud de cancelación de trabajo

```
DELETE https://your-neptune-endpoint:port/loader?loadId=loadId
```

```
DELETE https://your-neptune-endpoint:port/loader/loadId
```

Parámetros de solicitud de cancelación de tarea

loadId

El ID de la tarea de carga.

Sintaxis de respuesta de cancelación de trabajo

```
no response body
```

200 OK

Una tarea de carga eliminada correctamente devuelve el código `200`.

Errores de cancelación de tarea

Si se produce un error, se devuelve un objeto JSON en el elemento `BODY` de la respuesta. El objeto `message` contiene una descripción del error.

Categorías de errores

- **`Error 400`**: un `loadId` no válido devuelve un error de solicitud incorrecta HTTP `400`. El mensaje describe el error.
- **`Error 500`**: una solicitud válida que no se puede procesar devuelve un error interno del servidor HTTP `500`. El mensaje describe el error.

Mensajes de error de cancelación de tarea

A continuación se muestran los posibles mensajes de error de la API de cancelación con la descripción correspondiente.

- `The load with id = load_id does not exist or not active` (HTTP 404): no se ha encontrado la carga. Compruebe el valor del parámetro `id`.
- `Load cancellation is not permitted on a read replica instance.` (HTTP 405): la carga es una operación de escritura. Pruebe a volver a cargar en el punto de enlace del clúster de lectura/escritura.

Ejemplos de cancelación de tarea

Example Solicitud

A continuación, se muestra una solicitud enviada mediante HTTP `DELETE` con el comando `curl`.

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802'
```

AWS Database Migration Service Utilización para cargar datos en Amazon Neptune desde un almacén de datos diferente

AWS Database Migration Service (AWS DMS) puede cargar datos en Neptune desde bases de datos fuente compatibles de forma rápida y segura. La base de datos de origen permanece totalmente

operativa durante la migración, lo que minimiza el tiempo de inactividad de las aplicaciones que dependen de ella.

Encontrará información detallada al respecto AWS DMS en la Guía del AWS Database Migration Service usuario y en la Referencia de la AWS Database Migration Service API. En concreto, puede encontrar información sobre cómo configurar un clúster de Neptune como objetivo para la migración en Using Amazon Neptune as a Target for AWS Database Migration Service.

Estos son algunos requisitos previos para importar datos en Neptune mediante AWS DMS:

- Deberá crear un objeto de mapeo de AWS DMS tablas para definir cómo deben extraerse los datos de la base de datos de origen (consulte Especificar la selección y las transformaciones de las tablas mediante el mapeo de tablas mediante JSON en la AWS DMS guía del usuario para obtener más información). Este objeto de configuración de asignación de tablas especifica qué tablas se deben leer y en qué orden y cómo se denominan sus columnas. También puede filtrar las filas que se copian y proporcionar transformaciones de valor simples como convertir a minúsculas o redondear.
- Tendrá que crear una `GraphMappingConfig` de Neptune para especificar cómo deben cargarse los datos extraídos de la base de datos de origen en Neptune. Para los datos RDF (consultados mediante SPARQL), el `GraphMappingConfig` se escribe en el lenguaje de asignación R2RML estándar de W3. Para los datos del gráfico de propiedades (consultados con Gremlin), el `GraphMappingConfig` es un objeto JSON, descrito en GraphMappingConfig Diseño para datos de Property-Graph/Gremlin.
- Debe utilizarla AWS DMS para crear una instancia de replicación en la misma VPC que su clúster de base de datos de Neptune, para mediar en la transferencia de datos.
- También necesitará un bucket de Amazon S3 para que se utilice como almacenamiento intermedio para el almacenamiento provisional de los datos de migración.

Creando un Neptuno GraphMappingConfig

El `GraphMappingConfig` que crea especifica cómo se deben cargar los datos extraídos de un almacén de datos de origen en un clúster de base de datos de Neptune. Su formato varía en función de si está destinado a cargar datos RDF o para cargar datos de gráficos de propiedades.

Para los datos de RDF, puede utilizar el lenguaje R2RML de W3 para asignar datos relacionales a RDF.

Si está cargando datos de gráficos de propiedades para consultarlos mediante Gremlin, se crea un objeto JSON para `GraphMappingConfig`.

GraphMappingConfig Diseño para datos RDF/SPARQL

Si está cargando datos RDF para consultarlos mediante SPARQL, escriba `GraphMappingConfig` en R2RML. `R2RML` es un lenguaje W3 estándar para asignación de datos relacionales a RDF. Aquí tiene un ejemplo:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/ns#> .

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "nodes" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{id}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "label" ];
    ] .
```

Este es otro ejemplo:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#TriplesMap2>
    rr:logicalTable [ rr:tableName "Student" ];
    rr:subjectMap   [ rr:template "http://example.com/{ID}{Name}";
                      rr:class foaf:Person ];
    rr:predicateObjectMap [
        rr:predicate ex:id ;
        rr:objectMap  [ rr:column "ID";
                        rr:datatype xsd:integer ]
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name ;
        rr:objectMap  [ rr:column "Name" ]
```

```
    ] .
```

La recomendación de W3 en R2RML: RDB to RDF Mapping Language proporciona detalles sobre el lenguaje.

GraphMappingConfig Diseño para datos de Property-Graph/Gremlin

Un `GraphMappingConfig` comparable para datos de Property-Graph es un objeto JSON que proporciona una regla de asignación para cada entidad de gráfico que se generará a partir de los datos de origen. La siguiente plantilla muestra el aspecto de cada regla en este objeto:

```
{
  "rules": [
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "vertex_definitions": [
        {
          "vertex_id_template": "{col1}",
          "vertex_label": "(the vertex to create)",
          "vertex_definition_id": "(an identifier for this vertex)",
          "vertex_properties": [
            {
              "property_name": "(name of the property)",
              "property_value_template": "{col2} or text",
              "property_value_type": "(data type of the property)"
            }
          ]
        }
      ]
    },
    {
      "rule_id": "(an identifier for this rule)",
      "rule_name": "(a name for this rule)",
      "table_name": "(the name of the table or view being loaded)",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{col1}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "to_vertex": {
```

```
            "vertex_id_template": "{col3}",
            "vertex_definition_id": "(an identifier for the vertex referenced above)"
          },
          "edge_id_template": {
            "label": "(the edge label to add)",
            "template": "{col1}_{col3}"
          },
          "edge_properties":[
            {
              "property_name": "(the property to add)",
              "property_value_template": "{col4} or text",
              "property_value_type": "(data type like String, int, double)"
            }
          ]
        }
      ]
    }
  ]
}
```

Tenga en cuenta que la presencia de una etiqueta de vértice implica que el vértice se está creando aquí, mientras que su ausencia implica que el vértice es creado por un origen distinto y esta definición solo agrega propiedades de vértice.

Aquí hay una regla de ejemplo para un registro de empleado:

```
{
  "rules": [
    {
      "rule_id": "1",
      "rule_name": "vertex_mapping_rule_from_nodes",
      "table_name": "nodes",
      "vertex_definitions": [
        {
          "vertex_id_template": "{emp_id}",
          "vertex_label": "employee",
          "vertex_definition_id": "1",
          "vertex_properties": [
            {
              "property_name": "name",
              "property_value_template": "{emp_name}",
              "property_value_type": "String"
            }
```

```
          ]
        }
      ]
    },
    {
      "rule_id": "2",
      "rule_name": "edge_mapping_rule_from_emp",
      "table_name": "nodes",
      "edge_definitions": [
        {
          "from_vertex": {
            "vertex_id_template": "{emp_id}",
            "vertex_definition_id": "1"
          },
          "to_vertex": {
            "vertex_id_template": "{mgr_id}",
            "vertex_definition_id": "1"
          },
          "edge_id_template": {
            "label": "reportsTo",
            "template": "{emp_id}_{mgr_id}"
          },
          "edge_properties":[
            {
              "property_name": "team",
              "property_value_template": "{team}",
              "property_value_type": "String"
            }
          ]
        }
      ]
    }
  ]
}
```

Creación de una tarea de AWS DMS replicación con Neptune como objetivo

Una vez que haya creado las configuraciones de mapeo de tablas y mapeo de gráficos, utilice el siguiente proceso para cargar datos desde el almacén de origen en Neptune. Consulte la AWS DMS documentación para obtener más información sobre las API en cuestión.

Paso 1: Crear una instancia AWS DMS de replicación

Cree una instancia de AWS DMS replicación en la VPC en la que se ejecuta el clúster de base de datos de Neptune (consulte Trabajo con una instancia y una instancia de replicación de AWS DMS en la Guía del CreateReplicationusuario). AWS DMS Para ello, puede utilizar un AWS CLI comando como el siguiente:

```
aws dms create-replication-instance \
    --replication-instance-identifier (the replication instance identifier) \
    --replication-instance-class (the size and capacity of the instance, like
 'dms.t2.medium') \
    --allocated-storage (the number of gigabytes to allocate for the instance
 initially) \
    --engine-version (the DMS engine version that the instance should use) \
    --vpc-security-group-ids (the security group to be used with the instance)
```

Paso 2. Cree un AWS DMS punto final para la base de datos de origen

El siguiente paso es crear un AWS DMS punto final para el almacén de datos de origen. Puedes usar la AWS DMS CreateEndpointAPI de la siguiente AWS CLI manera:

```
aws dms create-endpoint \
    --endpoint-identifier (source endpoint identifier) \
    --endpoint-type source \
    --engine-name (name of source database engine) \
    --username (user name for database login) \
    --password (password for login) \
    --server-name (name of the server) \
    --port (port number) \
    --database-name (database name)
```

Paso 3. Configure un bucket de Amazon S3 para que Neptune almacene provisionalmente los datos.

Si no tiene un bucket de Amazon S3 que pueda utilizar para almacenar provisionalmente los datos, cree uno como se explica en Creación de un bucket en la Guía de introducción de Amazon S3 o ¿Cómo se puede crear un bucket de S3? en la Guía de usuario de la consola.

Tendrá que crear un política de IAM que conceda permisos `GetObject`, `PutObject`, `DeleteObject` y `ListObject` al bucket si no dispone de uno aún:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::(bucket-name)"
      ]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:ListObject"
       ],
      "Resource": [
        "arn:aws:s3:::(bucket-name)/*"
      ]
    }
  ]
}
```

Si su clúster de base de datos de Neptune tiene habilitada la autenticación de IAM, también deberá incluir la siguiente política:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "(the ARN of your Neptune DB cluster resource)"
    }
  ]
```

```
}
```

Cree un rol de IAM como documento de confianza al que asociar la política:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "dms.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Sid": "neptune",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Después de asociar la política al rol, asocie el rol al clúster de base de datos de Neptune. Esto permitirá AWS DMS usar el depósito para organizar los datos que se están cargando.

Paso 4. Cree punto de conexión de Amazon S3 en la VPC de Neptune

Ahora cree un punto de conexión de la puerta de enlace de VPC para su bucket de Amazon S3 intermediario, en la VPC donde se encuentra su clúster de Neptune. AWS CLI Para ello, puede utilizar el AWS Management Console o el, tal y como se describe en Creación de un punto final de puerta de enlace.

Paso 5. Crear un punto final de AWS DMS destino para Neptune

Cree un AWS DMS punto final para el clúster de base de datos Neptune de destino. Puede usar la AWS DMS CreateEndpointAPI con el `NeptuneSettings` parámetro de la siguiente manera:

```
aws dms create-endpoint \
```

```
    --endpoint-identifier (target endpoint identifier) \
    --endpoint-type target \
    --engine-name neptune \
    --server-name (name of the server) \
    --port (port number) \
    --neptune-settings '{ \
      "ServiceAccessRoleArn": "(ARN of the service access role)", \
      "S3BucketName": "(name of S3 bucket to use for staging files when migrating)", \
      "S3BucketFolder": "(name of the folder to use in that S3 bucket)", \
      "ErrorRetryDuration": (number of milliseconds to wait between bulk-load retries),
\
      "MaxRetryCount": (the maximum number of times to retry a failing bulk-load job),
\
      "MaxFileSize": (maximum file size, in bytes, of the staging files written to S3),
\
      "isIamAuthEnabled": (set to true if IAM authentication is enabled on the Neptune
cluster) }'
```

El objeto JSON que se pasa a la AWS DMS `CreateEndpoint` API en su `NeptuneSettings` parámetro tiene los siguientes campos:

- **`ServiceAccessRoleArn`**: (obligatorio) el ARN de un rol de IAM que permite acceso detallado al bucket de S3 utilizado para el almacenamiento provisional de la migración de los datos a Neptune. Este rol también debe tener permisos para acceder al clúster de base de datos de Neptune si la autorización de IAM está habilitada en él.
- **`S3BucketName`**: (obligatorio) para la migración de carga completa, la instancia de replicación convierte todos los datos RDS en archivos CSV quad, los carga en este bucket de almacenamiento provisional en S3 y, a continuación, realiza una carga masiva de los mismos en Neptune.
- **`S3BucketFolder`**: (obligatorio) la carpeta que se va a utilizar en el bucket de almacenamiento provisional de S3.
- **`ErrorRetryDuration`**: (opcional) el número de milisegundos que hay que esperar después de que la solicitud de Neptune devuelva un error antes de realizar una solicitud de reintento. El valor predeterminado es 250.
- **`MaxRetryCount`**— (opcional) El número máximo de solicitudes de reintento que se AWS DMS deben realizar tras un error que se pueda volver a intentar. El valor predeterminado es 5.
- **`MaxFileSize`**: (opcional) el tamaño máximo en bytes de cada archivo de almacenamiento provisional guardado en S3 durante la migración. El valor predeterminado es 1.048.576 KB (1 GB).

- **`IsIAMAuthEnabled`**: (opcional) establézcalo en `true` si la autenticación de IAM está habilitada en el clúster de base de datos de Neptune o en `false` si no es así. El valor predeterminado es `false`.

Paso 6. Probar conexiones a los nuevos puntos de enlace

Puedes probar la conexión a cada uno de estos nuevos puntos de conexión mediante la API de la AWS DMS TestConnectionsiguiente manera:

```
aws dms test-connection \
    --replication-instance-arn (the ARN of the replication instance) \
    --endpoint-arn (the ARN of the endpoint you are testing)
```

Paso 7. Cree una tarea de AWS DMS replicación

Una vez que haya completado correctamente los pasos anteriores, cree una tarea de replicación para migrar los datos del almacén de datos de origen a Neptune, utilizando AWS DMS CreateReplicationla API de tareas de la siguiente manera:

```
aws dms create-replication-task \
    --replication-task-identifier (name for the replication task) \
    --source-endpoint-arn (ARN of the source endpoint) \
    --target-endpoint-arn (ARN of the target endpoint) \
    --replication-instance-arn (ARN of the replication instance) \
    --migration-type full-load \
    --table-mappings (table-mapping JSON object or URI like 'file:///tmp/table-
mappings,json') \
    --task-data (a GraphMappingConfig object or URI like 'file:///tmp/graph-mapping-
config.json')
```

El parámetro `TaskData` proporciona el GraphMappingConfig que especifica cómo deben almacenarse en Neptune los datos que se copian.

Paso 8. Inicie la tarea de AWS DMS replicación

Ahora puede iniciar la tarea de replicación:

```
aws dms start-replication-task
    --replication-task-arn (ARN of the replication task started in the previous step)
```

```
    --start-replication-task-type start-replication
```

Consulta de un gráfico de Neptune

Neptune admite los siguientes lenguajes de consulta de gráficos para obtener acceso a un gráfico:

- Gremlin, definido por Apache TinkerPop para crear y consultar gráficos de propiedades.

  En Gremlin, una consulta es un recorrido compuesto por pasos discretos, cada uno de los cuales sigue un borde hasta un nodo.

  Consulte Acceso al gráfico de Neptune con Gremlin para obtener información sobre el uso de Gremlin en Neptune y Conformidad con los estándares de Gremlin en Amazon Neptune para encontrar detalles específicos sobre la implementación de Gremlin en Neptune.
- openCypher es un lenguaje de consulta declarativo para gráficos de propiedades que desarrolló originalmente Neo4j, luego de código abierto en 2015, y que contribuyó al proyecto openCypher en virtud de una licencia de código abierto Apache 2. Su sintaxis está documentada en la especificación de OpenCypher.
- SPARQL es un lenguaje declarativo basado en la coincidencia de patrones de gráficos para consultar datos RDF. Cuenta con el apoyo del World Wide Web Consortium.

  Consulte Acceso al gráfico de Neptune con SPARQL para obtener información sobre el uso de SPARQL en Neptune y Conformidad con los estándares de SPARQL en Amazon Neptune para encontrar detalles específicos sobre la implementación de SPARQL en Neptune.

> **Note**
>
> Tanto Gremlin como openCypher se pueden usar para consultar cualquier dato de gráfico de propiedades almacenado en Neptune, independientemente de cómo se haya cargado.

Temas

Colas de consultas en Amazon Neptune

Al desarrollar y ajustar aplicaciones gráficas, puede ser útil conocer las implicaciones de cómo la base de datos está colocando en cola las consultas. En Amazon Neptune, la cola de consultas se produce de la siguiente manera:

- El número máximo de consultas que se pueden colocar en cola por instancia, con independencia del tamaño de instancia, es 8.192. Cualquier consulta superior a este número se rechaza y se genera un error con una `ThrottlingException`.
- El número máximo de consultas que se pueden ejecutar a la vez viene determinado por el número de subprocesos de trabajo asignados, que generalmente se establece en el doble del número de núcleos de CPU virtuales (vCPU) disponibles.
- La latencia de consulta incluye el tiempo que una consulta pasa en la cola, así como los viajes de ida y vuelta a la red y el tiempo que realmente tarda en ejecutarse.

Determinación del número de consultas en la cola en un momento dado

La `MainRequestQueuePendingRequests` CloudWatch métrica registra el número de solicitudes en espera en la cola de entrada con una precisión de cinco minutos (consulte). Métricas de Neptune CloudWatch

Para Gremlin, puede obtener un recuento actual de consultas en la cola utilizando el valor `acceptedQueryCount` devuelto por el API del estado de la consulta de Gremlin. Tenga en cuenta, sin embargo, que el valor `acceptedQueryCount` devuelto por el API de estado de la consulta SPARQL incluye todas las consultas aceptadas desde que se inició el servidor, incluidas las consultas completadas.

Cómo puede afectar a los tiempos de espera la cola de consultas

Como se señaló anteriormente, la latencia de consulta incluye el tiempo que pasa una consulta en la cola, así como el tiempo que tarda en ejecutarse.

Dado que el período de tiempo de espera de una consulta generalmente se mide a partir del momento en que entra en la cola, una cola con movimiento lento puede hacer que muchas consultas agoten el tiempo de espera en cuanto se eliminan de la cola. Obviamente, esto no es deseable, por lo que es bueno evitar poner en cola un gran número de consultas a menos que se puedan ejecutar rápidamente.

Acceso al gráfico de Neptune con Gremlin

Amazon Neptune es compatible con Apache TinkerPop 3 y Gremlin. Esto significa que puede conectarse a una instancia de base de datos de Neptune y utilizar el lenguaje transversal Gremlin para consultar el gráfico (consulte The Graph en la documentación de Apache 3). TinkerPop Para conocer las diferencias en la implementación de Gremlin, consulte Conformidad con los estándares de Gremlin.

Las diferentes versiones del motor de Neptune admiten diferentes versiones de Gremlin. Consulte la página de versiones del motor de la versión de Neptune que está utilizando para determinar qué versión de Gremlin es compatible.

Un recorrido en Gremlin es una serie de pasos encadenados. Comienza en un vértice (o borde). Recorre el gráfico siguiendo el borde de salida de cada vértice y, después, los bordes resultantes de todos ellos. Cada paso es una operación del recorrido. Para obtener más información, consulte The Traversal en la documentación de 3. TinkerPop

Existen variantes del lenguaje Gremlin y compatibilidad con el acceso a Gremlin en varios lenguajes de programación. Para obtener más información, consulte Sobre las variantes del lenguaje gremlin en la TinkerPop documentación 3.

En esta documentación, se describe cómo obtener acceso a Neptune con las variantes y los lenguajes de programación siguientes.

Como se explica en Cifrado en tránsito: conexión con Neptune mediante SSL/HTTPS, puede conectarse a Neptune mediante Transport Layer Security/capa de conexión segura (TLS/SSL) en todas las regiones de AWS .

Gremlin-Groovy

En los ejemplos de la consola de Gremlin y HTTP REST de esta sección se utiliza la variante Gremlin-Groovy. Para obtener más información sobre la consola de Gremlin y Amazon Neptune, consulte la sección the section called “Utilice Gremlin” del inicio rápido.

Gremlin-Java

El ejemplo de Java está escrito con la implementación oficial de Java TinkerPop 3.0 y usa la variante Gremlin-Java.

Gremlin-Python

El ejemplo de Python está escrito con la implementación oficial de Python TinkerPop 3 y usa la variante Gremlin-Python.

En las secciones siguientes, se muestra cómo utilizar la consola de Gremlin, REST sobre HTTPS y varios lenguajes de programación para conectarse a una instancia de base de datos de Neptune.

Antes de comenzar, debe disponer de lo siguiente:

- Una instancia de base de datos de Neptune. Para obtener información acerca de la creación de una instancia de base de datos de Neptune, consulte Creación de un nuevo clúster de base de datos de Neptune.
- Una instancia de Amazon EC2 en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Para obtener más información acerca de cómo cargar datos en Neptune, incluidos los requisitos previos, los formatos de carga y los parámetros de carga, consulte Carga de datos en Amazon Neptune.

Temas

- Realización de actualizaciones o inserciones eficientes con los pasos mergeV() y mergeE() de Gremlin
- Realización de actualizaciones o inserciones de Gremlin eficientes con fold()/coalesce()/unfold()
- Análisis de la ejecución de las consultas de Neptune con explain de Gremlin
- Uso de Gremlin con el motor de consultas DFE de Neptune

Configure la consola de Gremlin para conectarse a una instancia de base de datos de Neptune

La consola Gremlin le permite experimentar con TinkerPop gráficos y consultas en un entorno REPL (bucle). read-eval-print

Instalación de la consola de Gremlin y conexión a ella de la forma habitual

Puede utilizar dicha consola para conectarse a una base de datos de gráficos remota. En la siguiente sección, se explica la instalación y la configuración de la consola de Gremlin para conectarse a una instancia de base de datos de Neptune de forma remota. Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Para obtener ayuda para conectarse a Neptune con SSL/TLS (obligatorio), consulte Configuración de SSL/TLS.

> **Note**
>
> Si tiene habilitada la autenticación de IAM en su clúster de base de datos de Neptune, siga en Conexión con Neptune mediante la consola de Gremlin con firma de Signature Version 4 las instrucciones para instalar la consola de Gremlin en lugar de las instrucciones que aparecen aquí.

Para instalar la consola de Gremlin y conectarse a Neptune

1. Los archivos binarios de la consola de Gremlin requieren Java 8 o Java 11. En estas instrucciones se presupone el uso de Java 11. Puede instalar Java 11 en su instancia de EC2 de la siguiente manera:

- Si usa Amazon Linux 2 (AL2):

```
sudo amazon-linux-extras install java-openjdk11
```

- Si usa Amazon Linux 2023 (AL2023):

```
sudo yum install java-11-amazon-corretto-devel
```

- Para otras distribuciones, utilice la que sea adecuada de las siguientes opciones:

```
sudo yum install java-11-openjdk-devel
```

o bien:

```
sudo apt-get install openjdk-11-jdk
```

2. Escriba lo siguiente para establecer Java 11 como tiempo de ejecución predeterminado en la instancia EC2.

```
sudo /usr/sbin/alternatives --config java
```

Cuando se le solicite, escriba el número para Java 11.

3. Descargue la versión adecuada de la consola de Gremlin del sitio web de Apache. Consulte la página de versiones del motor de la versión del motor de Neptune que está utilizando para determinar qué versión de Gremlin es compatible. Por ejemplo, para la versión 3.6.5, puede descargar la consola de Gremlin del sitio web de Apache Tinkerpop3 a su instancia de EC2 de la siguiente manera:

```
wget https://archive.apache.org/dist/tinkerpop/3.6.5/apache-tinkerpop-gremlin-
console-3.6.5-bin.zip
```

4. Descomprima el archivo .zip de la consola de Gremlin.

```
unzip apache-tinkerpop-gremlin-console-3.6.5-bin.zip
```

5. Cambie al directorio del directorio descomprimido.

```
cd apache-tinkerpop-gremlin-console-3.6.5
```

6. En el subdirectorio `conf` del directorio extraído, cree un archivo llamado `neptune-remote.yaml` con el texto siguiente. Sustituya *`your-neptune-endpoint`* por el nombre de host o la dirección IP de su instancia de base de datos de Neptune. Los corchetes (`[ ]`) son obligatorios.

   > **Note**
   >
   > Para obtener información acerca de cómo encontrar el nombre de host de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

   ```
   hosts: [your-neptune-endpoint]
   port: 8182
   connectionPool: { enableSsl: true }
   serializer: { className:
    org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
                 config: { serializeResultToString: true }}
   ```

   > **Note**
   >
   > Los serializadores se trasladaron del módulo al nuevo `gremlin-driver` `gremlin-util` módulo en la versión 3.7.0. El paquete cambió de org.apache.tinkerpop.gremlin.driver.ser a org.apache.tinkerpop.gremlin.util.ser.

7. En un terminal, vaya al directorio de la consola de Gremlin (`apache-tinkerpop-gremlin-console-3.6.5`) y, a continuación, escriba el siguiente comando para ejecutarla.

   ```
   bin/gremlin.sh
   ```

   Debería ver los siguientes datos de salida:

   ```
           \,,,/
           (o o)
   -----oOOo-(3)-oOOo-----
   plugin activated: tinkerpop.server
   plugin activated: tinkerpop.utilities
   plugin activated: tinkerpop.tinkergraph
   ```

```
gremlin>
```

Ahora se encuentra en `gremlin>`. Escriba los pasos restantes en este punto.

8. En el símbolo del sistema `gremlin>`, escriba lo siguiente para conectarse a la instancia de base de datos de Neptune.

```
:remote connect tinkerpop.server conf/neptune-remote.yaml
```

9. En la entrada `gremlin>`, escriba lo siguiente para cambiar al modo remoto. Esto envía todas las consultas de Gremlin a la conexión remota.

```
:remote console
```

10. Escriba lo siguiente para enviar una consulta al gráfico de Gremlin.

```
g.V().limit(1)
```

11. Cuando haya terminado, escriba lo siguiente para salir de la consola de Gremlin.

```
:exit
```

> **Note**
>
> Utilice punto y coma (`;`) o un carácter de nueva línea (\n) para separar las instrucciones. Cada recorrido anterior al final debe terminar en `next()` para ejecutarse. Solo se devuelven los datos del recorrido final.

Para obtener más información sobre la implementación de Gremlin en Neptune, consulte the section called “Conformidad con los estándares de Gremlin”.

Una forma alternativa de conectarse a la consola de Gremlin

Inconvenientes del enfoque de conexión normal

La forma más común de conectarse a la consola de Gremlin es la que hemos explicado anteriormente, utilizando comandos como estos en el símbolo del sistema de `gremlin>`:

```
gremlin> :remote connect tinkerpop.server conf/(file name).yaml
```

```
gremlin> :remote console
```

Esto funciona bien y permite enviar consultas a Neptune. Sin embargo, saca del bucle al motor de scripts Groovy, por lo que Neptune trata todas las consultas como puras de Gremlin. Esto significa que los siguientes formularios de consulta fallan:

```
gremlin> 1 + 1
gremlin> x = g.V().count()
```

Lo más parecido a utilizar una variable cuando se está conectado de esta forma es utilizar la variable `result` que mantiene la consola y enviar la consulta utilizando `:>`, de esta forma:

```
gremlin> :remote console
==>All scripts will now be evaluated locally - type ':remote console' to return
 to remote mode for Gremlin Server - [krl-1-cluster.cluster-ro-cm9t6tfwbtsr.us-
east-1.neptune.amazonaws.com/172.31.19.217:8182]
gremlin> :> g.V().count()
==>4249

gremlin> println(result)
[result{object=4249 class=java.lang.Long}]

gremlin> println(result['object'])
[4249]
```

Una forma diferente de conectarse

También puede conectarse a la consola de Gremlin de una forma diferente, que tal vez le parezca más agradable, como esta:

```
gremlin> g = traversal().withRemote('conf/neptune.properties')
```

Aquí `neptune.properties` toma esta forma:

```
gremlin.remote.remoteConnectionClass=org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteCon
gremlin.remote.driver.clusterFile=conf/my-cluster.yaml
gremlin.remote.driver.sourceName=g
```

El archivo `my-cluster.yaml` debería ser similar a esto:

```
hosts: [my-cluster-abcdefghijk.us-east-1.neptune.amazonaws.com]
port: 8182
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: false } }
connectionPool: { enableSsl: true }
```

> **Note**
>
> Los serializadores se movieron del módulo al nuevo módulo en la versión `gremlin-driver` 3.7.0. `gremlin-util` El paquete cambió de org.apache.tinkerpop.gremlin.driver.ser a org.apache.tinkerpop.gremlin.util.ser.

Configurar la conexión de la consola de Gremlin de esta manera le permite realizar correctamente los siguientes tipos de consultas:

```
gremlin> 1+1
==>2

gremlin> x=g.V().count().next()
==>4249

gremlin> println("The answer was ${x}")
The answer was 4249
```

Puede evitar que se muestre el resultado, de esta manera:

```
gremlin> x=g.V().count().next();[]
gremlin> println(x)
4249
```

Todas las formas habituales de realizar consultas (sin el paso del terminal) siguen funcionando. Por ejemplo:

```
gremlin> g.V().count()
==>4249
```

Incluso puede usar el paso `g.io().read()` para cargar un archivo con este tipo de conexión.

Uso del punto de conexión HTTP REST para conectarse a una instancia de base de datos de Neptune

Amazon Neptune proporciona un punto de conexión HTTP para las consultas de Gremlin. La interfaz REST es compatible con cualquier versión de Gremlin que utilice su clúster de base de datos (consulte la página de versiones del motor de Neptune que esté utilizando para determinar qué versión de Gremlin admite).

> **Note**
>
> Como se explica en Cifrado en tránsito: conexión con Neptune mediante SSL/HTTPS, Neptune ahora requiere que se conecte mediante HTTPS en lugar de HTTP.

Las siguientes instrucciones le ayudarán a conectar con el punto de enlace de Gremlin mediante el comando `curl` y HTTPS. Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

El punto de conexión HTTPS para las consultas de Gremlin a una instancia de base de datos de Neptune es `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`.

> **Note**
>
> Para obtener información acerca de cómo encontrar el nombre de host de la instancia de base de datos de Neptune, consulte Conexión a los puntos de conexión de Amazon Neptune.

Para conectarse a Neptune mediante el punto de conexión HTTP REST

En el siguiente ejemplo se utiliza curl para enviar una consulta de Gremlin a través de HTTP POST. La consulta se envía en formato JSON en el cuerpo de la publicación como propiedad `gremlin`.

```
curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://your-neptune-endpoint:port/
gremlin
```

Este ejemplo devuelve el primer vértice del gráfico utilizando el recorrido `g.V().limit(1)`. Puede consultar otra cosa sustituyéndolo por otro recorrido de Gremlin.

> **Important**
>
> De manera predeterminada, el punto de conexión REST devuelve todos los resultados en un único conjunto de resultados JSON. Si este conjunto de resultados es demasiado grande, se puede producir una excepción `OutOfMemoryError` en la instancia de base de datos de Neptune.
> Para evitarlo, habilite las respuestas fragmentadas (los resultados se devuelven en una serie de respuestas independientes). Consulte Use encabezados finales HTTP opcionales para habilitar las respuestas de Gremlin compuestas por varias partes.

Aunque se recomiendan las solicitudes HTTP POST para enviar consultas de Gremlin, también es posible utilizar solicitudes HTTP GET:

```
curl -G "https://your-neptune-endpoint:port?gremlin=g.V().count()"
```

> **Note**
>
> Neptune no admite la propiedad `bindings`.

Use encabezados finales HTTP opcionales para habilitar las respuestas de Gremlin compuestas por varias partes

De forma predeterminada, la respuesta HTTP a las consultas de Gremlin se devuelve en un único conjunto de resultados JSON. En el caso de un conjunto de resultados muy grande, esto puede provocar una excepción `OutOfMemoryError` en la instancia de base de datos.

Sin embargo, puede habilitar las respuestas fragmentadas (respuestas que se devuelven en varias partes separadas). Para ello, incluya un encabezado final (`te: trailers`) con codificación de transferencia (TE) en su solicitud. Consulte la página de MDN sobre los encabezados de las solicitudes TE para obtener más información sobre los encabezados TE.

Cuando se devuelve una respuesta en varias partes, puede resultar difícil diagnosticar un problema que se produce después de recibir la primera parte, ya que la primera parte llega con un código de

estado HTTP de `200` (OK). Un error posterior suele provocar que el cuerpo del mensaje contenga una respuesta dañada, al final de la cual Neptune añade un mensaje de error.

Para facilitar la detección y el diagnóstico de este tipo de errores, Neptune también incluye dos nuevos campos de encabezado dentro de los encabezados finales de cada fragmento de respuesta:

- `X-Neptune-Status`: contiene el código de respuesta seguido de un nombre abreviado. Por ejemplo, en caso de que se realizara correctamente, el encabezado final sería: `X-Neptune-Status: 200 OK`. En caso de fallo, el código de respuesta sería uno de los códigos de error del motor de Neptune, como `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail`: está vacío si las solicitudes se han realizado correctamente. En caso de errores, contiene el mensaje de error JSON. Como solo se permiten caracteres ASCII en los valores de los encabezados HTTP, la cadena JSON está codificada en URL.

> **Note**
>
> Neptune no admite actualmente la compresión `gzip` de respuestas fragmentadas. Si el cliente solicita codificación y compresión fragmentadas al mismo tiempo, Neptune omite la compresión.

Clientes Gremlin basados en Java que se utilizan con Amazon Neptune

Puede usar cualquiera de los dos clientes Gremlin de código abierto basados en Java con Amazon Neptune: el cliente Apache TinkerPop Java Gremlin o el cliente Gremlin para Amazon Neptune.

El cliente Apache Java Gremlin TinkerPop

Si puede, utilice siempre la última versión del cliente Apache TinkerPop Java Gremlin compatible con la versión de su motor. Las versiones más recientes contienen numerosas correcciones de errores que pueden mejorar la estabilidad, el rendimiento y la facilidad de uso del cliente.

En la siguiente tabla se enumeran las versiones más antiguas y más recientes del TinkerPop cliente compatibles con las distintas versiones del motor Neptune:

Versión del motor de Neptune	Versión mínima TinkerPop	TinkerPop Versión máxima
`1.3.2.0`	`3.6.2`	`3.7.1`

Versión del motor de Neptune	Versión mínima TinkerPop	TinkerPop Versión máxima
1.3.1.0	3.6.2	3.6.5
1.3.0.0	3.6.2	3.6.4
1.2.1.1	3.6.2	3.6.2
1.2.1.0	3.6.2	3.6.2
1.2.0.2	3.5.2	3.5.6
1.2.0.1	3.5.2	3.5.6
1.2.0.0	3.5.2	3.5.6
1.1.1.0	3.5.2	3.5.6
1.1.0.0	3.4.0	3.4.13
1.0.5.1 y más antiguas	(en desuso)	(en desuso)

TinkerPop los clientes suelen ser compatibles con versiones anteriores dentro de una serie (3.3.xpor ejemplo, o3.4.x). Hay casos excepcionales en los que se debe interrumpir la compatibilidad con versiones anteriores, por lo que es mejor comprobar la recomendación de TinkerPop actualización antes de actualizar a una nueva versión del cliente.

Es posible que el cliente no pueda seguir los nuevos pasos o las nuevas características introducidas en versiones posteriores a las que admite el servidor, pero es de esperar que las consultas y características existentes funcionen a menos que la recomendación de actualización requiera un cambio radical.

> **Note**
>
> A partir de la versión 1.1.1.0 del motor de Neptune, no utilice una TinkerPop versión inferior a. 3.5.2
>
> Los usuarios de Python deben evitar usar la TinkerPop versión 3.4.9 debido a una configuración de tiempo de espera predeterminada que requiere una configuración directa (consulte TINKERPOP-2505).

Cliente Java de Gremlin para Amazon Neptune

El cliente Gremlin para Amazon Neptune es un cliente Gremlin de código abierto basado en Java que sustituye directamente al cliente Java estándar. TinkerPop

El cliente de Gremlin para Neptune está optimizado para los clústeres de Neptune. Le permite administrar la distribución del tráfico entre varias instancias de un clúster y se adapta a los cambios en la topología del clúster al añadir o eliminar una réplica. Incluso puede configurar el cliente para que distribuya las solicitudes entre un subconjunto de instancias del clúster, en función del rol, el tipo de instancia, la zona de disponibilidad (AZ) o las etiquetas asociadas a las instancias.

La versión más reciente del cliente Java de Gremlin para Neptune está disponible en Maven Central.

Para obtener más información acerca del cliente Java de Gremlin para Neptune, consulte esta publicación del blog. Para ver ejemplos de código y demostraciones, consulte el proyecto del cliente. GitHub

Uso de un cliente Java para conectarse a una instancia de base de datos de Neptune

En la siguiente sección, se explica cómo ejecutar un ejemplo completo de Java que se conecta a una instancia de base de datos de Neptune y realiza un recorrido por Gremlin mediante el cliente Apache Gremlin. TinkerPop

Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Para conectarse a Neptune mediante Java

1. Instale Apache Maven en la instancia EC2. En primer lugar, escriba lo siguiente para añadir un repositorio con un paquete de Maven:

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Escriba lo siguiente para establecer el número de versión de los paquetes:

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   A continuación, use el comando yum para instalar Maven:

```
sudo yum install -y apache-maven
```

2. Instale Java. Las bibliotecas de Gremlin necesitan Java 8 u 11. Puede instalar Java 11 como se indica a continuación:

   - Si usa Amazon Linux 2 (AL2):

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Si usa Amazon Linux 2023 (AL2023):

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

   - Para otras distribuciones, utilice la que sea adecuada de las siguientes opciones:

     ```
     sudo yum install java-11-openjdk-devel
     ```

     o bien:

     ```
     sudo apt-get install openjdk-11-jdk
     ```

3. Defina Java 11 como tiempo de ejecución predeterminado en su instancia EC2: introduzca lo siguiente para establecer Java 8 como tiempo de ejecución predeterminado en su instancia EC2:

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Cuando se le solicite, escriba el número para Java 11.

4. Cree un nuevo directorio llamado **`gremlinjava`**:

   ```
   mkdir gremlinjava
   cd gremlinjava
   ```

5. En el directorio `gremlinjava`, cree un archivo `pom.xml` y, a continuación, ábralo en un editor de texto:

   ```
   nano pom.xml
   ```

6. Copie lo siguiente en el archivo `pom.xml` y guárdelo:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://
maven.apache.org/maven-v4_0_0.xsd">
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>GremlinExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>GremlinExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-driver</artifactId>
      <version>3.6.5</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.tinkerpop/gremlin-groovy
      (Not needed for TinkerPop version 3.5.2 and up)
    <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>gremlin-groovy</artifactId>
      <version>3.6.5</version>
    </dependency> -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jdk14</artifactId>
      <version>1.7.25</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.5.1</version>
        <configuration>
          <source>11</source>
          <target>11</target>
        </configuration>
```

```
      </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.3</version>
          <configuration>
            <executable>java</executable>
            <arguments>
              <argument>-classpath</argument>
              <classpath/>
              <argument>com.amazonaws.App</argument>
            </arguments>
            <mainClass>com.amazonaws.App</mainClass>
            <complianceLevel>1.11</complianceLevel>
            <killAfter>-1</killAfter>
          </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Si está modificando un proyecto de Maven ya existente, la dependencia requerida aparece resaltada en el código anterior.

7. Escriba lo siguiente en la línea de comandos a fin de crear subdirectorios para el código fuente de ejemplo (`src/main/java/com/amazonaws/`):

```
mkdir -p src/main/java/com/amazonaws/
```

8. En el directorio `src/main/java/com/amazonaws/`, cree un archivo llamado `App.java` y, a continuación, ábralo en un editor de texto.

```
nano src/main/java/com/amazonaws/App.java
```

9. Copie lo siguiente en el archivo `App.java`. Sustituya *`your-neptune-endpoint`* por la dirección de su instancia de base de datos de Neptune. No incluya el prefijo `https://` en el método `addContactPoint`.

> **Note**
>
> Para obtener información acerca de cómo encontrar el nombre de host de la instancia de base de datos de Neptune, consulte Conexión a los puntos de conexión de Amazon Neptune.

```
package com.amazonaws;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import org.apache.tinkerpop.gremlin.driver.Client;
import
 org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversal;
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.structure.T;

public class App
{
  public static void main( String[] args )
  {
    Cluster.Builder builder = Cluster.build();
    builder.addContactPoint("your-neptune-endpoint");
    builder.port(8182);
    builder.enableSsl(true);

    Cluster cluster = builder.create();

    GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));

    // Add a vertex.
    // Note that a Gremlin terminal step, e.g. iterate(), is required to make a
 request to the remote server.
    // The full list of Gremlin terminal steps is at https://tinkerpop.apache.org/
docs/current/reference/#terminal-steps
    g.addV("Person").property("Name", "Justin").iterate();

    // Add a vertex with a user-supplied ID.
```

```
    g.addV("Custom Label").property(T.id, "CustomId1").property("name", "Custom id
 vertex 1").iterate();
    g.addV("Custom Label").property(T.id, "CustomId2").property("name", "Custom id
 vertex 2").iterate();

    g.addE("Edge Label").from(__.V("CustomId1")).to(__.V("CustomId2")).iterate();

    // This gets the vertices, only.
    GraphTraversal t = g.V().limit(3).elementMap();

    t.forEachRemaining(
      e ->  System.out.println(t.toList())
    );

    cluster.close();
  }
}
```

Para obtener ayuda para conectarse a Neptune con SSL/TLS (obligatorio), consulte Configuración de SSL/TLS.

10. Compile y ejecute el ejemplo usando el siguiente comando Maven:

```
mvn compile exec:exec
```

El ejemplo anterior devuelve un mapa de las claves y los valores de cada propiedad para los dos primeros índices del gráfico utilizando el recorrido `g.V().limit(3).elementMap()`. Para otras consultas, sustitúyalo por otro recorrido de Gremlin con uno de los métodos de finalización adecuados.

> **Note**
>
> La parte final de la consulta de Gremlin, `.toList()`, es necesaria para enviar el recorrido al servidor para su evaluación. Si no incluye ese método u otro equivalente, la consulta no se envía a la instancia de base de datos de Neptune.
> También debe añadir una terminación adecuada al añadir un vértice o borde, como cuando utiliza el paso `addV( )`.

Los siguientes métodos envían la consulta a la instancia de base de datos de Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Configuración de SSL/TLS para el cliente Java de Gremlin

Neptune requiere que SSL/TLS esté habilitado de forma predeterminada. Normalmente, si el controlador Java está configurado con `enableSsl(true)`, se puede conectar a Neptune sin tener que configurar un `trustStore()` o `keyStore()` con una copia local de un certificado. En las versiones anteriores se TinkerPop recomendaba el uso de `keyCertChainFile()` para configurar un `.pem` archivo almacenado localmente, pero ha quedado obsoleto y ya no está disponible después de la versión 3.5.x. Si utilizaba esa configuración con un certificado público, ahora puede eliminar la copia local con `SFSRootCAG2.pem`.

Sin embargo, si la instancia con la que se está conectando no tiene una conexión a Internet a través de la cual verificar un certificado público, o si el certificado que está usando no es público, puede seguir estos pasos para configurar una copia del certificado local:

Configuración de una copia del certificado local para habilitar SSL/TLS

1. Descargue e instale keytool de Oracle. Esto facilitará mucho la configuración del almacén de claves local.
2. Descargue el certificado `SFSRootCAG2.pem`CA (el SDK de Java de Gremlin necesita un certificado para verificar el certificado remoto):

   ```
   wget https://www.amazontrust.com/repository/SFSRootCAG2.pem
   ```

3. Cree un almacén de claves en formato JKS o PKCS12. En este ejemplo se utiliza JKS. Responda las preguntas que aparecen a continuación cuando se le pida. La contraseña que cree aquí la necesitará más adelante:

   ```
   keytool -genkey -alias (host name) -keyalg RSA -keystore server.jks
   ```

4. Importe el archivo `SFSRootCAG2.pem` que descargó al almacén de claves recién creado:

```
keytool -import -keystore server.jks -file .pem
```

5. Configure el objeto `Cluster` mediante programación:

```
Cluster cluster = Cluster.build("(your neptune endpoint)")
                         .port(8182)
                         .enableSSL(true)
                         .keyStore('server.jks')
                         .keyStorePassword("(the password from step 2)")
                         .create();
```

Si lo desea, puede hacer lo mismo en un archivo de configuración, y también con la consola de Gremlin:

```
hosts: [(your neptune endpoint)]
port: 8182
connectionPool: { enableSsl: true, keyStore: server.jks, keyStorePassword: (the
 password from step 2) }
serializer: { className:
 org.apache.tinkerpop.gremlin.driver.ser.GraphBinaryMessageSerializerV1, config:
 { serializeResultToString: true }}
```

Ejemplo en Java de conexión a una instancia de base de datos de Neptune con lógica de reconexión

El siguiente ejemplo de Java muestra cómo conectarse al cliente de Gremlin con una lógica de reconexión para recuperarse de una desconexión inesperada.

Tiene las siguientes dependencias:

```
<dependency>
    <groupId>org.apache.tinkerpop</groupId>
    <artifactId>gremlin-driver</artifactId>
    <version>${gremlin.version}</version>
</dependency>

<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>amazon-neptune-sigv4-signer</artifactId>
```

```
    <version>${sig4.signer.version}</version>
</dependency>

<dependency>
    <groupId>com.evanlennick</groupId>
    <artifactId>retry4j</artifactId>
    <version>0.15.0</version>
</dependency>
```

Este es el código de ejemplo:

```
public static void main(String args[]) {
  boolean useIam = true;

  // Create Gremlin cluster and traversal source
  Cluster.Builder builder = Cluster.build()
         .addContactPoint(System.getenv("neptuneEndpoint"))
         .port(Integer.parseInt(System.getenv("neptunePort")))
         .enableSsl(true)
         .minConnectionPoolSize(1)
         .maxConnectionPoolSize(1)
         .serializer(Serializers.GRAPHBINARY_V1D0)
         .reconnectInterval(2000);

  if (useIam) {
      builder.handshakeInterceptor( r -> {
         try {
            NeptuneNettyHttpSigV4Signer sigV4Signer =
                        new NeptuneNettyHttpSigV4Signer("(your region)", new
 DefaultAWSCredentialsProviderChain());
            sigV4Signer.signRequest(r);
         } catch (NeptuneSigV4SignerException e) {
            throw new RuntimeException("Exception occurred while signing the request",
 e);
         }
         return r;
      });
   }

  Cluster cluster = builder.create();

  GraphTraversalSource g = AnonymousTraversalSource
      .traversal()
```

```
      .withRemote(DriverRemoteConnection.using(cluster));

  // Configure retries
  RetryConfig retryConfig = new RetryConfigBuilder()
      .retryOnCustomExceptionLogic(getRetryLogic())
      .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
      .withMaxNumberOfTries(5)
      .withFixedBackoff()
      .build();

  @SuppressWarnings("unchecked")
  CallExecutor<Object> retryExecutor = new CallExecutorBuilder<Object>()
      .config(retryConfig)
      .build();

  // Do lots of queries
  for (int i = 0; i < 100; i++){
    String id = String.valueOf(i);

    @SuppressWarnings("unchecked")
    Callable<Object> query = () -> g.V(id)
        .fold()
        .coalesce(
            unfold(),
            addV("Person").property(T.id, id))
        .id().next();

    // Retry query
    // If there are connection failures, the Java Gremlin client will automatically
    // attempt to reconnect in the background, so all we have to do is wait and retry.
    Status<Object> status = retryExecutor.execute(query);

    System.out.println(status.getResult().toString());
  }

  cluster.close();
}

private static Function<Exception, Boolean> getRetryLogic() {

  return e -> {

    Class<? extends Exception> exceptionClass = e.getClass();
```

```
    StringWriter stringWriter = new StringWriter();
    String message = stringWriter.toString();


    if (RemoteConnectionException.class.isAssignableFrom(exceptionClass)){
      System.out.println("Retrying because RemoteConnectionException");
      return true;
    }

    // Check for connection issues
    if (message.contains("Timed out while waiting for an available host") ||
        message.contains("Timed-out") && message.contains("waiting for connection on
Host") ||
        message.contains("Connection to server is no longer active") ||
        message.contains("Connection reset by peer") ||
        message.contains("SSLEngine closed already") ||
        message.contains("Pool is shutdown") ||
        message.contains("ExtendedClosedChannelException") ||
        message.contains("Broken pipe") ||
        message.contains(System.getenv("neptuneEndpoint")))
    {
      System.out.println("Retrying because connection issue");
      return true;
    };

    // Concurrent writes can sometimes trigger a ConcurrentModificationException.
    // In these circumstances you may want to backoff and retry.
    if (message.contains("ConcurrentModificationException")) {
      System.out.println("Retrying because ConcurrentModificationException");
      return true;
    }

    // If the primary fails over to a new instance, existing connections to the old
primary will
    // throw a ReadOnlyViolationException. You may want to back and retry.
    if (message.contains("ReadOnlyViolationException")) {
      System.out.println("Retrying because ReadOnlyViolationException");
      return true;
    }

    System.out.println("Not a retriable error");
    return false;
  };
```

```
}
```

Uso de Python para conectarse a una instancia de base de datos de Neptune

Si puede, utilice siempre la última versión del cliente Apache TinkerPop Python Gremlin, gremlinpython, compatible con la versión de su motor. Las versiones más recientes contienen numerosas correcciones de errores que mejoran la estabilidad, el rendimiento y la facilidad de uso del cliente. La `gremlinpython` versión que se utilizará normalmente se alineará con TinkerPop las versiones descritas en la tabla del cliente Java Gremlin.

> **Note**
>
> Las versiones `gremlinpython` 3.5.x son compatibles con las versiones TinkerPop 3.4.x siempre que solo utilices las funciones 3.4.x en las consultas de Gremlin que escribas.

En la siguiente sección se indica cómo ejecutar una muestra de Python que se conecta a una instancia de base de datos de Amazon Neptune y realiza un recorrido de Gremlin.

Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Antes de comenzar, haga lo siguiente:

- Descargue e instale Python 3.6 o una versión posterior desde el sitio web de Python.org.
- Compruebe que tiene pip instalado. Si no tiene pip o no está seguro, consulte Do I need to install pip? en la documentación de pip.
- Si su instalación de Python aún no lo tiene, descargue `futures` tal y como se indica a continuación: `pip install futures`

Para conectarse a Neptune mediante Python

1. Escriba lo siguiente para instalar el paquete `gremlinpython`:

   ```
   pip install --user gremlinpython
   ```

2. Cree un archivo con el nombre `gremlinexample.py` y, a continuación, ábralo en un editor de texto.
3. Copie lo siguiente en el archivo `gremlinexample.py`. Sustituya *`your-neptune-endpoint`* por la dirección de su instancia de base de datos de Neptune.

   Para obtener información acerca de cómo encontrar la dirección de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

```
from __future__  import print_function  # Python 2/3 compatibility

from gremlin_python import statics
from gremlin_python.structure.graph import Graph
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection

graph = Graph()

remoteConn = DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin','g')
g = graph.traversal().withRemote(remoteConn)

print(g.V().limit(2).toList())
remoteConn.close()
```

4. Escriba el comando siguiente para ejecutar el ejemplo:

```
python gremlinexample.py
```

   La consulta de Gremlin al final de este ejemplo devuelve los vértices (`g.V().limit(2)`) en una lista. A continuación, la lista se imprime con la función `print` de Python estándar.

> **Note**
>
> La parte final de la consulta de Gremlin, `toList()`, es necesaria para enviar el recorrido al servidor para su evaluación. Si no incluye ese método u otro equivalente, la consulta no se envía a la instancia de base de datos de Neptune.

   Los siguientes métodos envían la consulta a la instancia de base de datos de Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

El ejemplo anterior devuelve los dos primeros vértices del gráfico utilizando el recorrido `g.V().limit(2).toList()`. Para otras consultas, sustitúyalo por otro recorrido de Gremlin con uno de los métodos de finalización adecuados.

Uso de .NET para conectarse a una instancia de base de datos de Neptune

Si puede, utilice siempre la última versión del cliente Gremlin de Apache TinkerPop .NET, compatible con la versión de su motor. Las versiones más recientes contienen numerosas correcciones de errores que mejoran la estabilidad, el rendimiento y la facilidad de uso del cliente. La `Gremlin.Net` versión que se utilizará normalmente se alineará con TinkerPop las versiones descritas en la tabla del cliente Java Gremlin.

La siguiente sección contiene un ejemplo de código escrito en C# que se conecta a una instancia de base de datos de Neptune y realiza un recorrido de Gremlin.

La conexiones a Amazon Neptune deben realizarse desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune. Este código de muestra se ha probado en una instancia de Amazon EC2 que ejecuta Ubuntu.

Antes de comenzar, haga lo siguiente:

- Instale .NET en la instancia de Amazon EC2. Para obtener instrucciones de instalación de .NET en varios sistemas operativos, incluidos Windows, Linux y macOS, consulte el artículo de introducción a .NET.
- Instale Gremlin.NET ejecutando `dotnet add package gremlin.net` para su paquete. Para obtener más información, consulte Gremlin.NET en la documentación. TinkerPop

Para conectarse a Neptune mediante Gremlin.NET

1. Creación de un nuevo proyecto de .NET.

```
dotnet new console -o gremlinExample
```

2. Cambie al directorio nuevo del proyecto.

```
cd gremlinExample
```

3. Copie lo siguiente en el archivo `Program.cs`. Sustituya *`your-neptune-endpoint`* por la dirección de su instancia de base de datos de Neptune.

   Para obtener información acerca de cómo encontrar la dirección de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

```
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using Gremlin.Net;
using Gremlin.Net.Driver;
using Gremlin.Net.Driver.Remote;
using Gremlin.Net.Structure;
using static Gremlin.Net.Process.Traversal.AnonymousTraversalSource;
namespace gremlinExample
{
  class Program
  {
    static void Main(string[] args)
    {
      try
      {
        var endpoint = "your-neptune-endpoint";
        // This uses the default Neptune and Gremlin port, 8182
        var gremlinServer = new GremlinServer(endpoint, 8182, enableSsl: true );
        var gremlinClient = new GremlinClient(gremlinServer);
        var remoteConnection = new DriverRemoteConnection(gremlinClient, "g");
        var g = Traversal().WithRemote(remoteConnection);
        g.AddV("Person").Property("Name", "Justin").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 1").Iterate();
        g.AddV("Custom Label").Property("name", "Custom id vertex 2").Iterate();
        var output = g.V().Limit<Vertex>(3).ToList();
        foreach(var item in output) {
```

```
            Console.WriteLine(item);
        }
      }
      catch (Exception e)
      {
          Console.WriteLine("{0}", e);
      }
    }
  }
}
```

4. Escriba el comando siguiente para ejecutar el ejemplo:

```
dotnet run
```

La consulta de Gremlin al final de este ejemplo devuelve un recuento de un solo vértice con fines de prueba. Después, se imprime en la consola.

> **Note**
>
> La parte final de la consulta de Gremlin, `Next()`, es necesaria para enviar el recorrido al servidor para su evaluación. Si no incluye ese método u otro equivalente, la consulta no se envía a la instancia de base de datos de Neptune.

Los siguientes métodos envían la consulta a la instancia de base de datos de Neptune:

- `ToList()`
- `ToSet()`
- `Next()`
- `NextTraverser()`
- `Iterate()`

Utilice `Next()` si necesita que los resultados de la consulta se serialicen y devuelvan, o `Iterate()` si no es así.

En el ejemplo anterior se devuelve una lista utilizando el recorrido `g.V().Limit(3).ToList()`. Para otras consultas, sustitúyalo por otro recorrido de Gremlin con uno de los métodos de finalización adecuados.

Uso de Node.js para conectarse a una instancia de base de datos de Neptune

Si puede, utilice siempre la última versión del cliente Apache TinkerPop JavaScript Gremlin, compatible con la versión de su motor. Las versiones más recientes contienen numerosas correcciones de errores que mejoran la estabilidad, el rendimiento y la facilidad de uso del cliente. La versión `gremlin` que se utilizará normalmente coincidirá con las TinkerPop versiones descritas en la tabla del cliente Java Gremlin.

En la siguiente sección se indica cómo ejecutar una muestra de Node.js que se conecta a una instancia de base de datos de Amazon Neptune y realiza un recorrido de Gremlin.

Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Antes de comenzar, haga lo siguiente:

- Compruebe que esté instalado Node.js versión 8.11 o superior. Si no lo está, descargue e instale Node.js desde el sitio web de Nodejs.org.

Para conectarse a Neptune mediante Node.js

1. Escriba lo siguiente para instalar el paquete `gremlin-javascript`:

   ```
   npm install gremlin
   ```

2. Cree un archivo con el nombre `gremlinexample.js` y ábralo en un editor de texto.
3. Copie lo siguiente en el archivo `gremlinexample.js`. Sustituya *`your-neptune-endpoint`* por la dirección de su instancia de base de datos de Neptune.

   Para obtener información acerca de cómo encontrar la dirección de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

   ```
   const gremlin = require('gremlin');
   ```

```
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const Graph = gremlin.structure.Graph;

dc = new DriverRemoteConnection('wss://your-neptune-endpoint:8182/gremlin',{});

const graph = new Graph();
const g = graph.traversal().withRemote(dc);

g.V().limit(1).count().next().
    then(data => {
        console.log(data);
        dc.close();
    }).catch(error => {
        console.log('ERROR', error);
        dc.close();
    });
```

4. Escriba el comando siguiente para ejecutar el ejemplo:

```
node gremlinexample.js
```

El ejemplo anterior devuelve el recuento de un único vértice en el gráfico utilizando el recorrido `g.V().limit(1).count().next()`. Para otras consultas, sustitúyalo por otro recorrido de Gremlin con uno de los métodos de finalización adecuados.

> **Note**
>
> La parte final de la consulta de Gremlin, `next()`, es necesaria para enviar el recorrido al servidor para su evaluación. Si no incluye ese método u otro equivalente, la consulta no se envía a la instancia de base de datos de Neptune.

Los siguientes métodos envían la consulta a la instancia de base de datos de Neptune:

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Utilice `next()` si necesita que los resultados de la consulta se serialicen y devuelvan, o `iterate()` si no es así.

> **Important**
>
> Se trata de un ejemplo aislado de Node.js. Si planea ejecutar código como este en una AWS Lambda función, consulte Ejemplos de funciones de Lambda para obtener más información sobre el uso JavaScript eficiente de una función de Neptune Lambda.

Uso de Go para conectarse a una instancia de base de datos de Neptune

Si puede, utilice siempre la versión más reciente del cliente Apache TinkerPop Go Gremlin, compatible con la versión de su motor. Las versiones más recientes contienen numerosas correcciones de errores que mejoran la estabilidad, el rendimiento y la facilidad de uso del cliente.

La `gremlingo` versión que se utilizará normalmente se alineará con TinkerPop las versiones descritas en la tabla del cliente Java Gremlin.

> **Note**
>
> Las versiones 3.5.x de gremlingo son retrocompatibles con las versiones 3.4.x siempre y cuando solo utilices las funciones de la versión TinkerPop 3.4.x en las consultas de Gremlin que escribas.

En la siguiente sección se indica cómo ejecutar una muestra de Go que se conecta a una instancia de base de datos de Amazon Neptune y realiza un recorrido Gremlin.

Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Antes de comenzar, haga lo siguiente:

- Descargue e instale Go 1.17 o una versión posterior desde el sitio web go.dev.

Para conectarse a Neptune mediante Go

1. A partir de un directorio vacío, inicialice un nuevo módulo de Go:

```
go mod init example.com/gremlinExample
```

2. Añada gremlin-go como dependencia de su nuevo módulo:

```
go get github.com/apache/tinkerpop/gremlin-go/v3/driver
```

3. Cree un archivo con el nombre `gremlinExample.go` y, a continuación, ábralo en un editor de texto.

4. Copie lo siguiente en el archivo `gremlinExample.go` y sustituya *`(your neptune endpoint)`* por la dirección de su instancia de base de datos de Neptune:

```
package main

import (
  "fmt"
  gremlingo "github.com/apache/tinkerpop/gremlin-go/v3/driver"
)

func main() {
  // Creating the connection to the server.
  driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://(your
 neptune endpoint):8182/gremlin",
    func(settings *gremlingo.DriverRemoteConnectionSettings) {
      settings.TraversalSource = "g"
    })
  if err != nil {
    fmt.Println(err)
    return
  }
  // Cleanup
  defer driverRemoteConnection.Close()

  // Creating graph traversal
  g := gremlingo.Traversal_().WithRemote(driverRemoteConnection)

  // Perform traversal
  results, err := g.V().Limit(2).ToList()
  if err != nil {
    fmt.Println(err)
    return
  }
```

```
  // Print results
  for _, r := range results {
    fmt.Println(r.GetString())
  }
}
```

> **Note**
>
> El formato del certificado TLS de Neptune no es compatible actualmente en Go 1.18+ con macOS y puede generar un error 509 al intentar iniciar una conexión. En el caso de las pruebas locales, puede omitirlo añadiendo “crypto/tls” a las importaciones y modificando la configuración de `DriverRemoteConnection` de la siguiente manera:
>
> ```
> // Creating the connection to the server.
> driverRemoteConnection, err := gremlingo.NewDriverRemoteConnection("wss://
> your-neptune-endpoint:8182/gremlin",
>   func(settings *gremlingo.DriverRemoteConnectionSettings) {
>       settings.TraversalSource = "g"
>       settings.TlsConfig = &tls.Config{InsecureSkipVerify: true}
>   })
> ```

5. Escriba el comando siguiente para ejecutar el ejemplo:

```
go run gremlinExample.go
```

La consulta de Gremlin al final de este ejemplo devuelve los vértices (`g.V().Limit(2)`) en un sector. A continuación, este segmento se itera y se imprime con la función `fmt.Println` estándar.

> **Note**
>
> La parte final de la consulta de Gremlin, `ToList()`, es necesaria para enviar el recorrido al servidor para su evaluación. Si no incluye ese método u otro equivalente, la consulta no se envía a la instancia de base de datos de Neptune.

Los siguientes métodos envían la consulta a la instancia de base de datos de Neptune:

- `ToList()`

- `ToSet()`
- `Next()`
- `GetResultSet()`
- `Iterate()`

El ejemplo anterior devuelve los dos primeros vértices del gráfico utilizando el recorrido `g.V().Limit(2).ToList()`. Para otras consultas, sustitúyalo por otro recorrido de Gremlin con uno de los métodos de finalización adecuados.

Sugerencias de consulta de Gremlin

Puede utilizar sugerencias de consulta para especificar estrategias de optimización y evaluación para una consulta de Gremlin concreta en Amazon Neptune.

Las sugerencias de consulta se especifican añadiendo un paso `withSideEffect` a la consulta con la siguiente sintaxis.

```
g.withSideEffect(hint, value)
```

- hint: identifica el tipo de sugerencia que se va a aplicar.
- value: determina el comportamiento del aspecto del sistema considerado.

Por ejemplo, a continuación se muestra cómo incluir una sugerencia `repeatMode` en un recorrido Gremlin.

> **Note**
>
> Todos los efectos secundarios de las sugerencias de consulta Gremlin están precedidos por `Neptune#`.

```
g.withSideEffect('Neptune#repeatMode',
 'DFS').V("3").repeat(out()).times(10).limit(1).path()
```

La consulta anterior indica al motor de Neptune que recorra el gráfico Primero por profundidad (DFS) en vez del gráfico predeterminado de Neptune Primero por amplitud (BFS).

En las siguientes secciones se proporciona más información sobre las sugerencias de consulta disponibles y su uso.

Temas

Sugerencia de consulta repeatMode de Gremlin

La sugerencia de consulta `repeatMode` de Neptune muestra cómo el motor de Neptune evalúa el paso `repeat()` en un recorrido de Gremlin: primero por amplitud, primero por profundidad o primero por profundidad fragmentada.

El modo de evaluación del paso `repeat()` es importante cuando se utiliza para encontrar o seguir una ruta, en lugar de simplemente repetir un paso un número limitado de veces.

Sintaxis

La sugerencia de consulta `repeatMode` se especifica mediante la incorporación de un paso `withSideEffect` a la consulta.

```
g.withSideEffect('Neptune#repeatMode', 'mode').gremlin-traversal
```

> **Note**
>
> Todos los efectos secundarios de las sugerencias de consulta Gremlin están precedidos por `Neptune#`.

Modos disponibles

- `BFS`

  Búsqueda primero por amplitud.

Modo de ejecución predeterminado para el paso `repeat()`. Obtiene todos los nodos del mismo nivel antes de profundizar en la ruta.

Esta versión tiene un uso intensivo de la memoria y los límites pueden ser muy grandes. Existe un mayor riesgo de que la consulta se quede sin memoria y la cancele el motor de Neptune. Es lo que más se acerca a otras implementaciones de Gremlin.

- `DFS`

  Búsqueda primero por profundidad.

  Sigue cada ruta hasta la profundidad máxima antes de pasar a la siguiente solución.

  Utiliza menos memoria. Puede proporcionar mejor rendimiento en situaciones como la búsqueda de una sola ruta desde un punto de partida con múltiples saltos.

- `CHUNKED_DFS`

  Búsqueda primero por profundidad fragmentada.

  Un planteamiento híbrido que explora primero la profundidad del gráfico en fragmentos de 1.000 nodos, en lugar de 1 nodo (`DFS`) o todos los nodos (`BFS`).

  El motor de Neptune obtendrá hasta 1000 nodos en cada nivel antes de profundizar más en la ruta.

  Se trata de un enfoque equilibrado entre la velocidad y el uso de la memoria.

  También es útil si desea usar `BFS`, pero la consulta usa demasiada memoria.

Ejemplo

En la siguiente sección se describe el efecto del modo de repetición en un recorrido Gremlin.

En Neptune, el modo predeterminado para el paso `repeat()` es realizar una estrategia de ejecución primero por amplitud (`BFS`) para todos los recorridos.

En la mayoría de los casos, la TinkerGraph implementación usa la misma estrategia de ejecución, pero en algunos casos altera la ejecución de un recorrido.

Por ejemplo, la TinkerGraph implementación modifica la siguiente consulta.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

El paso `repeat()` de este recorrido se "despliega" en el siguiente recorrido, lo que resulta una estrategia de primero por profundidad (DFS).

```
g.V(<id>).out().out().out().out().out().out().out().out().out().out().limit(1).path()
```

> **Important**
>
> El motor de consulta de Neptune no lo hace automáticamente.

Breadth-first (BFS) es la estrategia de ejecución predeterminada y es similar a la que se utiliza TinkerGraph en la mayoría de los casos. Sin embargo, existen ciertos casos donde se prefieren las estrategias primero por profundidad (DFS).

BFS (predeterminado)

La estrategia de ejecución predeterminada es primero por amplitud (BFS) para el operador de `repeat()`.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

El motor de Neptune explora a fondo las primeras fronteras de nueve saltos antes de buscar una solución de diez saltos. Esto es efectivo en muchos casos, como la consulta de la ruta más corta.

Sin embargo, en el ejemplo anterior, el recorrido sería mucho más rápido utilizando el modo de primero por profundidad (DFS) para el operador (`repeat()`).

DFS

La siguiente consulta utiliza el modo de primero por profundidad (DFS) para el operador `repeat()`.

```
g.withSideEffect("Neptune#repeatMode", "DFS").V("3").repeat(out()).times(10).limit(1)
```

Sigue cada solución individual hasta la máxima profundidad antes de explorar la siguiente solución.

Sugerencia de consulta noReordering de Gremlin

Cuando envía un recorrido de Gremlin, el motor de consulta de Neptune investiga la estructura del recorrido y reordena las partes de la consulta, intentando minimizar la cantidad de trabajo necesaria para la evaluación y el tiempo de respuesta de la consulta. Por ejemplo, un recorrido con múltiples restricciones, como múltiples pasos `has()`, no suele evaluarse en un orden determinado. En su lugar, se reordena después de que la consulta se compruebe con el análisis estático.

El motor de consulta de Neptune intenta identificar qué restricción es más selectiva y la ejecuta primero. Esto a menudo produce un mejor rendimiento, pero el orden en el que Neptune decide evaluar la consulta podría no ser siempre el óptimo.

Si conoce las características exactas de los datos y desea dictar manualmente el orden de ejecución de la consulta, puede utilizar la sugerencia de consulta `noReordering` de Neptune para especificar que el recorrido se evalúe en el orden indicado.

Sintaxis

La sugerencia de consulta `noReordering` se especifica mediante la incorporación de un paso `withSideEffect` a la consulta.

```
g.withSideEffect('Neptune#noReordering', true or false).gremlin-traversal
```

> **Note**
>
> Todos los efectos secundarios de las sugerencias de consulta Gremlin están precedidos por `Neptune#`.

Valores disponibles

- `true`
- `false`

Sugerencia de consulta typePromotion de Gremlin

Cuando envía un recorrido de Gremlin que filtra por un valor o rango numérico, el motor de consultas de Neptune debe usar normalmente la promoción de tipos cuando ejecuta la consulta. Esto significa

que tiene que examinar los valores de todos los tipos que puedan contener el valor por el que se está filtrando.

Por ejemplo, si está filtrando valores iguales a 55, el motor debe buscar números enteros iguales a 55, enteros largos iguales a 55L, flotantes iguales a 55,0, etc. Cada promoción de tipo requiere una búsqueda adicional en el almacenamiento, lo que puede provocar que una consulta aparentemente simple tarde un tiempo inesperadamente largo en completarse.

Supongamos que está buscando todos los vértices con una propiedad de edad del cliente superior a 5:

```
g.V().has('customerAge', gt(5))
```

Para ejecutar ese recorrido bien, Neptune debe expandir la consulta para examinar todos los tipos numéricos a los que se pueda promover el valor que está consultando. En este caso, el filtro `gt` debe aplicarse a cualquier número entero superior a 5, a cualquier largo superior a 5L, a cualquier flotante superior a 5,0 y a cualquier doble superior a 5,0. Como cada una de estas promociones de tipos requiere una búsqueda adicional en el almacenamiento, verá varios filtros por filtro numérico cuando ejecute la consulta API `profile` de Gremlin y tardará mucho más en completarse de lo que cabría esperar.

A menudo, la promoción de tipos no es necesaria porque ya sabe de antemano que solo necesita encontrar valores de un tipo específico. Cuando sea así, puede acelerar considerablemente sus consultas utilizando la sugerencia de consulta `typePromotion` para desactivar la promoción de tipos.

Sintaxis

La sugerencia de consulta `typePromotion` se especifica mediante la incorporación de un paso `withSideEffect` a la consulta.

```
g.withSideEffect('Neptune#typePromotion', true or false).gremlin-traversal
```

> **Note**
>
> Todos los efectos secundarios de las sugerencias de consulta Gremlin están precedidos por `Neptune#`.

Valores disponibles

- `true`
- `false`

Para desactivar la promoción de tipos en la consulta anterior, utilizaría lo sigiuiente:

```
g.withSideEffect('Neptune#typePromotion', false).V().has('customerAge', gt(5))
```

Sugerencia de consulta useDFE de Gremlin

Utilice esta sugerencia de consulta para permitir el uso del DFE para ejecutar la consulta. De forma predeterminada, Neptune no usa el DFE sin que esta sugerencia de consulta esté establecida en `true`, ya que el parámetro de instancia neptune_dfe_query_engine tiene el valor predeterminado `viaQueryHint`. Si establece ese parámetro de instancia en `enabled`, el motor DFE se utiliza para todas las consultas excepto para las que la sugerencia de consulta `useDFE` está establecida en `false`.

Ejemplo de cómo habilitar el DFE para una consulta:

```
g.withSideEffect('Neptune#useDFE', true).V().out()
```

Sugerencias de consulta de Gremlin para usar la caché de resultados

Las siguientes sugerencias de consulta se pueden utilizar cuando la caché de resultados de consultas está habilitada.

Sugerencia de consulta **`enableResultCache`** de Gremlin

La sugerencia de consulta `enableResultCache` con un valor de `true` hace que los resultados de la consulta se devuelvan desde la caché si ya se han almacenado en caché. De lo contrario, devuelve nuevos resultados y los almacena en caché hasta que se borren de la caché. Por ejemplo:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Luego, puede acceder a los resultados en caché emitiendo exactamente la misma consulta de nuevo:

Si el valor de esta sugerencia de consulta es `false`, o si no está presente, los resultados de la consulta no se almacenan en caché. Sin embargo, si se establece en `false`, no se borran los resultados en caché existentes. Para borrar los resultados en caché, use la sugerencia `invalidateResultCache` o `invalidateResultCachekey`.

Sugerencia de consulta **`enableResultCacheWithTTL`** de Gremlin

La sugerencia de consulta `enableResultCacheWithTTL` también devuelve los resultados en caché, si los hay, sin que ello afecte al TTL de los resultados que ya están en la caché. Si actualmente no hay resultados en caché, la consulta devuelve nuevos resultados y los guarda en caché durante el tiempo de vida (TTL) especificado en la sugerencia de consulta `enableResultCacheWithTTL`. Ese tiempo de vida se especifica en segundos. La consulta siguiente especifica un tiempo de vida de sesenta segundos:

```
g.with('Neptune#enableResultCacheWithTTL', 60)
 .V().has('genre','drama').in('likes')
```

Antes de que pasen los 60 segundos time-to-live , puedes usar la misma consulta (aquí,`g.V().has('genre','drama').in('likes')`) con la sugerencia de consulta `enableResultCache` o con la sugerencia de `enableResultCacheWithTTL` consulta para acceder a los resultados almacenados en caché.

> **Note**
>
> El tiempo de vida especificado con `enableResultCacheWithTTL` no afecta a los resultados que ya se hayan almacenado en caché.
>
> - Si los resultados se almacenaron previamente en caché utilizando `enableResultCache`, primero se debe borrar explícitamente la caché antes de que `enableResultCacheWithTTL` genere nuevos resultados y los almacene en caché durante el TTL que especifique.
> - Si los resultados se almacenaron previamente en caché utilizando `enableResultCachewithTTL`, ese TTL anterior debe vencer primero antes de que `enableResultCacheWithTTL` genere nuevos resultados y los almacene en caché durante el TTL que especifique.

Una vez transcurrido el tiempo de vida, se borran los resultados de la consulta en caché y, a continuación, una instancia posterior de la misma consulta devuelve nuevos resultados. Si `enableResultCacheWithTTL` se asocia a la consulta posterior, los nuevos resultados se almacenan en caché con el TTL que especifique.

Sugerencia de consulta **`invalidateResultCacheKey`** de Gremlin

La sugerencia de consulta `invalidateResultCacheKey` puede tomar un valor `true` o `false`. Un valor `true` hace que se borren los resultados en caché de la consulta a la que `invalidateResultCacheKey` está asociada. Por ejemplo, el siguiente ejemplo hace que se borren los resultados almacenados en caché para la clave de consulta `g.V().has('genre','drama').in('likes')`:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

El ejemplo de consulta anterior no hace que sus nuevos resultados se almacenen en caché. Puede incluir `enableResultCache` (o`enableResultCacheWithTTL`) en la misma consulta si desea almacenar en caché los nuevos resultados después de borrar los existentes:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Sugerencia de consulta **`invalidateResultCache`** de Gremlin

La sugerencia de consulta `invalidateResultCache` puede tomar un valor `true` o `false`. Un valor `true` hace que se borren todos los resultados de la caché de resultados. Por ejemplo:

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

El ejemplo de consulta anterior no hace que sus nuevos resultados se almacenen en caché. Puede incluir `enableResultCache` (o`enableResultCacheWithTTL`) en la misma consulta si desea almacenar en caché los nuevos resultados después de borrar por completo la caché existente:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
```

```
  .V().has('genre','drama').in('likes')
```

Sugerencia de consulta **`numResultsCached`** de Gremlin

La sugerencia de consulta `numResultsCached` solo se puede usar con consultas que contengan `iterate()` y especifica el número máximo de resultados que se guardarán en caché para la consulta a la que está asociada. Tenga en cuenta que los resultados almacenados en caché cuando `numResultsCached` está presente no se devuelven, solo se almacenan en caché.

Por ejemplo, la siguiente consulta especifica que se deben almacenar en caché hasta 100 de sus resultados, pero no se debe devolver ninguno de esos resultados en caché:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').iterate()
```

A continuación, puede utilizar una consulta como la siguiente para recuperar un rango de los resultados en caché (en este caso, los diez primeros):

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Sugerencia de consulta **`noCacheExceptions`** de Gremlin

La sugerencia de consulta `noCacheExceptions` puede tomar un valor `true` o `false`. Un valor `true` hace que se suprima cualquier excepción relacionada con la caché de resultados. Por ejemplo:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

En concreto, esto suprime la `QueryLimitExceededException`, que se activa si los resultados de una consulta son demasiado grandes para la caché de resultados.

API del estado de la consulta de Gremlin

Para obtener el estado de las consultas Gremlin, utilice HTTP `GET` o `POST` para realizar una solicitud al punto de enlace `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`.

Parámetros de solicitud de estado de consulta de Gremlin

- queryID (opcional): el identificador de una consulta de Gremlin en ejecución. Solo muestra el estado de la consulta especificada.
- includeWaiting (opcional): devuelve el estado de todas las consultas en espera.

  Normalmente, en la respuesta solo se incluyen las consultas en ejecución, pero cuando se especifica el parámetro `includeWaiting`, también se devuelve el estado de todas las consultas en espera.

Sintaxis de respuesta de estado de la consulta de Gremlin

```
{
  "acceptedQueryCount": integer,
  "runningQueryCount": integer,
  "queries": [
    {
      "queryId":"guid",
      "queryEvalStats":
        {
          "waited": integer,
          "elapsed": integer,
          "cancelled": boolean
        },
      "queryString": "string"
    }
  ]
}
```

Valores de respuesta de estado de la consulta de Gremlin

- aceptadas QueryCount: el número de consultas que se han aceptado pero que aún no se han completado, incluidas las consultas de la cola.
- en ejecución QueryCount: el número de consultas de Gremlin que se están ejecutando actualmente.
- queries: una lista de las consultas de Gremlin actuales.
- queryId: un identificador GUID de la consulta. Neptune asigna automáticamente este valor de identificador a cada consulta o también puede asignar su propio identificador (consulte Inserte un identificador personalizado en una consulta de Neptune Gremlin o SPARQL).

- consulta EvalStats: estadísticas de esta consulta.
- subqueries: el número de subconsultas de esta consulta.
- elapsed: el número de milisegundos que la consulta lleva en ejecución.
- cancelled: true indica que se canceló la consulta.
- queryString: la consulta enviada. Esta se trunca en 1024 caracteres, si supera este número.
- waited: indica cuánto tiempo esperó la consulta, en milisegundos.

Ejemplo de estado de la consulta de Gremlin

A continuación se muestra un ejemplo de comando de estado que utiliza `curl` y HTTP `GET`.

```
curl https://your-neptune-endpoint:port/gremlin/status
```

Esta salida muestra una única consulta en ejecución.

```
{
  "acceptedQueryCount":9,
  "runningQueryCount":1,
  "queries": [
    {
      "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
      "queryEvalStats":
        {
          "waited": 0,
          "elapsed": 23,
          "cancelled": false
        },
      "queryString": "g.V().out().count()"
    }
  ]
}
```

Cancelación de consultas de Gremlin

Para obtener el estado de las consultas Gremlin, utilice HTTP `GET` o `POST` para realizar una solicitud al punto de enlace `https://your-neptune-endpoint:port/gremlin/status`.

Parámetros de solicitud de cancelación de la consulta de Gremlin

- cancelQuery: obligatorio para la cancelación. Este parámetro no tiene un valor correspondiente.
- queryId: el identificador de la consulta de Gremlin en ejecución que se va a cancelar.

Ejemplo de cancelación de la consulta de Gremlin

A continuación se muestra un ejemplo del comando que utiliza `curl` para cancelar una consulta.

```
curl https://your-neptune-endpoint:port/gremlin/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Si la cancelación se lleva a cabo correctamente, se devuelve HTTP `200` OK.

Compatibilidad con sesiones basadas en scripts de Gremlin

Puede utilizar sesiones de Gremlin con transacciones implícitas en Amazon Neptune. Para obtener información sobre las sesiones de Gremlin, consulte Considerar las sesiones en la documentación de Apache TinkerPop . En las siguientes secciones, se describe cómo utilizar las sesiones de Gremlin con Java.

> **Note**
>
> Esta característica está disponible a partir de la versión 1.0.1.0.200463.0 del motor de Neptune.
> A partir de las versiones 1.1.1.0 y TinkerPop 3.5.2 del motor Neptune, también puede utilizar. Transacciones de Gremlin

> **Important**
>
> Actualmente, el tiempo máximo durante el que Neptune puede mantener una sesión abierta es de 10 minutos. Si no cierra una sesión antes de que se agote el tiempo, la sesión expirará y todo lo que se haya hecho volverá a su estado inicial.

Temas

- Sesiones de Gremlin en la consola de Gremlin
- Sesiones de Gremlin en la variante de lenguaje Gremlin

Sesiones de Gremlin en la consola de Gremlin

Si crea una conexión remota en la consola de Gremlin sin el parámetro `session`, la conexión remota se crea en el modo sin sesión . En este modo, cada solicitud que se envía al servidor se trata como una transacción completa en sí misma y no se guarda ningún estado entre las solicitudes. Si se produce un error en la solicitud, solo se revierte esta.

Si crea una conexión remota que emplea el parámetro `session`, crea una sesión basada en script que dura hasta que se cierre la conexión remota. Cada sesión se identifica mediante un UUID único, el cual crea la consola y le devuelve.

A continuación se muestra un ejemplo de una llamada a la consola que crea una sesión. Una vez enviadas las consultas, otra llamada cierra la sesión y confirma las consultas.

> **Note**
>
> El cliente de Gremlin siempre debe estar cerrado para liberar los recursos del servidor.

```
gremlin> :remote connect tinkerpop.server conf/neptune-remote.yaml session
  . . .
  . . .
gremlin> :remote close
```

Para obtener más información y ejemplos, consulte las sesiones en la documentación. TinkerPop

Todas las consultas que ejecuta durante una sesión forman una única transacción que no se confirma hasta que todas las consultas se completan correctamente y se cierra la conexión remota. Si se produce un error con una solicitud o si no cierra la conexión en el plazo máximo de duración de una sesión que Neptune admite, la transacción de la sesión no se confirmará y todas las solicitudes de la misma se revertirán.

Sesiones de Gremlin en la variante de lenguaje Gremlin

En la variante de lenguaje Gremlin (GLV), tiene que crear un objeto `SessionedClient` para emitir varias consultas en una única transacción como se muestra en el siguiente ejemplo.

```
try {                                // line 1
  Cluster cluster = Cluster.open();                    // line 2
  Client client = cluster.connect("sessionName");      // line 3
   ...
   ...
} finally {
  // Always close. If there are no errors, the transaction is committed; otherwise,
 it's rolled back.
  client.close();
}
```

La línea 3 del ejemplo anterior crea el objeto `SessionedClient` en función de las opciones de configuración establecidas para el clúster en cuestión. La cadena *`sessionname`* que ha pasado al método de conexión pasa a ser el único nombre de la sesión. Para evitar las colisiones, utilice un UUID para el nombre.

El cliente inicia una transacción de sesión cuando se inicializa. Todas las consultas que ejecuta durante el formulario de sesión se confirman solo cuando llama a `client.close( )`. Si se produce un error de nuevo con una única solicitud o si no cierra la conexión en el plazo máximo de duración de una sesión que Neptune admite, la transacción de la sesión no se confirmará y todas las solicitudes de la misma se revertirán.

> **Note**
>
> El cliente de Gremlin siempre debe estar cerrado para liberar los recursos del servidor.

```
GraphTraversalSource g = traversal().withRemote(conn);

Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
  gtx.addV('person').iterate();
  gtx.addV('software').iterate();

  tx.commit();
} finally {
```

```
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Transacciones de Gremlin en Neptune

Hay varios contextos en los que se ejecutan las transacciones de Gremlin. Al trabajar con Gremlin, es importante entender el contexto en el que se trabaja y cuáles son sus implicaciones:

- **`Script-based`**: las solicitudes se realizan mediante cadenas de Gremlin basadas en texto, como esta:
  - Con el controlador Java y `Client.submit(`*`string`*`)`.
  - Con la consola de Gremlin y `:remote connect`.
  - Con la API HTTP.
- **`Bytecode-based`**: las solicitudes se realizan utilizando el código de bytes de Gremlin serializado típico de las variantes del lenguaje Gremlin (GLV).

  Por ejemplo, utilizando el controlador de Java, `g = traversal().withRemote(...)`.

Para cualquiera de los contextos anteriores, existe el contexto adicional de la solicitud que se envía sin sesión o vinculada a una sesión.

> **Note**
>
> Las transacciones de Gremlin siempre deben confirmarse o revertirse, de modo que se puedan liberar los recursos del lado del servidor.

Solicitudes sin sesión

Cuando no hay sesión, una solicitud equivale a una sola transacción.

En el caso de los scripts, esto implica que una o más instrucciones de Gremlin enviadas en una sola solicitud se confirmarán o anularán como una sola transacción. Por ejemplo:

```
Cluster cluster = Cluster.open();
```

```
Client client = cluster.connect(); // sessionless
// 3 vertex additions in one request/transaction:
client.submit("g.addV();g.addV();g.addV()").all().get();
```

En el caso del código de bytes, se realiza una solicitud sin sesión para cada recorrido que se genera y ejecuta desde g:

```
GraphTraversalSource g = traversal().withRemote(...);

// 3 vertex additions in three individual requests/transactions:
g.addV().iterate();
g.addV().iterate();
g.addV().iterate();

// 3 vertex additions in one single request/transaction:
g.addV().addV().addV().iterate();
```

Solicitudes vinculadas a una sesión

Cuando están vinculadas a una sesión, se pueden aplicar varias solicitudes en el contexto de una sola transacción.

En el caso de los scripts, esto implica que no es necesario concatenar todas las operaciones gráficas en un único valor de cadena integrado:

```
Cluster cluster = Cluster.open();
Client client = cluster.connect(sessionName); // session
try {
    // 3 vertex additions in one request/transaction:
    client.submit("g.addV();g.addV();g.addV()").all().get();
} finally {
    client.close();
}

try {
    // 3 vertex additions in three requests, but one transaction:
    client.submit("g.addV()").all().get(); // starts a new transaction with the same
 sessionName
    client.submit("g.addV()").all().get();
    client.submit("g.addV()").all().get();
} finally {
    client.close();
```

```
}
```

En el caso del código de bytes, después TinkerPop `3.5.x`, la transacción se puede controlar de forma explícita y la sesión se puede gestionar de forma transparente. Las variantes del lenguaje Gremlin (GLV) admiten la sintaxis `tx()` de Gremlin para utilizar `commit()` o `rollback()` en una transacción de la siguiente manera:

```
GraphTraversalSource g = traversal().withRemote(conn);

Transaction tx = g.tx();

// Spawn a GraphTraversalSource from the Transaction.
// Traversals spawned from gtx are executed within a single transaction.
GraphTraversalSource gtx = tx.begin();
try {
    gtx.addV('person').iterate();
    gtx.addV('software').iterate();

    tx.commit();
} finally {
    if (tx.isOpen()) {
        tx.rollback();
    }
}
```

Aunque el ejemplo anterior está escrito en Java, también puede utilizar esta sintaxis de `tx()` en Python, Javascript y .NET.

> **Warning**
>
> Las consultas de solo lectura sin sesión se ejecutan bajo el aislamiento SNAPSHOT, pero las consultas de solo lectura que se ejecutan dentro de una transacción explícita se ejecutan bajo el aislamiento SERIALIZABLE. Las consultas de solo lectura que se ejecutan bajo el aislamiento `SERIALIZABLE` generan una mayor sobrecarga y pueden bloquearse o quedar bloqueadas por escrituras simultáneas, a diferencia de las que se ejecutan bajo el aislamiento `SNAPSHOT`.

Uso de la API de Gremlin con Amazon Neptune

> **Note**
>
> Amazon Neptune no admite la propiedad `bindings`.

Todas las solicitudes HTTPS de Gremlin utilizan un único punto de enlace: `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`. Todas las conexiones de Neptune deben usar HTTPS.

Puedes conectar la consola Gremlin a un gráfico de Neptuno directamente a través de ella. WebSockets

Para obtener más información acerca de la conexión al punto de enlace de Gremlin, consulte Acceso al gráfico de Neptune con Gremlin.

La implementación de Gremlin en Amazon Neptune presenta detalles y diferencias específicas que debe tener en cuenta. Para obtener más información, consulte Conformidad con los estándares de Gremlin en Amazon Neptune.

Para obtener información sobre el lenguaje Gremlin y los recorridos, consulte The Traversal en la documentación de Apache. TinkerPop

Almacenamiento en caché de los resultados de las consultas en Gremlin de Amazon Neptune

A partir de la versión 1.0.5.1 del motor, Amazon Neptune admite una caché de resultados para las consultas de Gremlin.

Puede habilitar la caché de resultados de las consultas y, a continuación, utilizar una sugerencia de consulta para almacenar en caché los resultados de una consulta de solo lectura de Gremlin.

Al volver a ejecutar la consulta, se recuperan los resultados en caché con una latencia baja y sin costos de E/S, siempre y cuando sigan en la memoria caché. Esto funciona para las consultas que se envían tanto en un punto de conexión HTTP como mediante Websockets, ya sea como código de bytes o en forma de cadena.

> **Note**
>
> Las consultas que se envían al punto de conexión del perfil no se almacenan en caché ni siquiera cuando la caché de consultas está habilitada.

Hay varias maneras de controlar el comportamiento de la caché de resultados de consultas de Neptune. Por ejemplo:

- Puede paginar los resultados almacenados en caché en bloques.
- Puede especificar el time-to-live (TTL) para consultas específicas.
- Puede borrar la caché para consultas específicas.
- Puede borrar toda la caché.
- Puede configurarlo para que se le notifique si los resultados superan el tamaño de la caché.

La caché se mantiene mediante una política least-recently-used (LRU), lo que significa que, una vez que se llena el espacio asignado a la caché, se eliminan los least-recently-used resultados para dejar espacio cuando se almacenen nuevos resultados en la memoria caché.

> **Important**
>
> La caché de resultados de consultas no está disponible en los tipos de instancias `t3.medium` y `t4.medium`.

Habilitación de la caché de resultados de consultas en Neptune

Para habilitar la caché de resultados de consultas en Neptune, utilice la consola para configurar el parámetro de instancia de base de datos `neptune_result_cache` en `1` (habilitado).

Una vez que se habilita la caché de resultados, Neptune reserva una parte de la memoria actual para almacenar en caché los resultados de las consultas. Cuanto más grande sea el tipo de instancia que utilice y más memoria haya disponible, más memoria reservará Neptune para la caché.

Si la memoria caché de resultados se llena, Neptune descarta automáticamente los resultados en caché least-recently-used (LRU) para dar paso a otros nuevos.

Puede comprobar el estado actual de la caché de resultados mediante el comando Estado de la instancia.

Uso de sugerencias para almacenar en caché los resultados de las consultas

Una vez que se habilita la caché de resultados de consultas, las sugerencias de consulta se utilizan para controlar el almacenamiento en caché de las consultas. Todos los ejemplos siguientes se aplican al mismo recorrido de consultas:

```
g.V().has('genre','drama').in('likes')
```

Uso de **`enableResultCache`**

Con la caché de resultados de consultas habilitada, puede almacenar en caché los resultados de una consulta de Gremlin utilizando la sugerencia de consulta `enableResultCache`, de la siguiente manera:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

A continuación, Neptune le devuelve los resultados de la consulta y también los almacena en caché. Luego, puede acceder a los resultados en caché emitiendo exactamente la misma consulta de nuevo:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

La clave de caché que identifica los resultados en caché es la propia cadena de consulta:

```
g.V().has('genre','drama').in('likes')
```

Uso de **`enableResultCacheWithTTL`**

Puede especificar durante cuánto tiempo se deben almacenar en caché los resultados de la consulta mediante la sugerencia de consulta `enableResultCacheWithTTL`. Por ejemplo, la siguiente consulta especifica que los resultados de la consulta deben caducar después de 120 segundos:

```
g.with('Neptune#enableResultCacheWithTTL', 120)
```

```
  .V().has('genre','drama').in('likes')
```

De nuevo, la clave de caché que identifica los resultados almacenados en caché es la cadena de consulta base:

```
g.V().has('genre','drama').in('likes')
```

Y, de nuevo, puede acceder a los resultados en caché utilizando esa cadena de consulta con la sugerencia de consulta `enableResultCache`:

```
g.with('Neptune#enableResultCache', true)
  .V().has('genre','drama').in('likes')
```

Si han transcurrido 120 segundos o más desde que los resultados se almacenaron en caché, la consulta devolverá nuevos resultados y los guardará en la memoria caché, sin ninguno. time-to-live

También puede acceder a los resultados almacenados en caché si vuelve a ejecutar la misma consulta con la sugerencia de consulta `enableResultCacheWithTTL`. Por ejemplo:

```
g.with('Neptune#enableResultCacheWithTTL', 140)
  .V().has('genre','drama').in('likes')
```

Hasta que hayan transcurrido 120 segundos (es decir, el TTL actualmente en vigor), esta nueva consulta que utilice la sugerencia de consulta `enableResultCacheWithTTL` devolverá los resultados almacenados en caché. Transcurridos 120 segundos, devolverá nuevos resultados y los almacenará en la memoria caché durante 140 segundos. time-to-live

> **Note**
>
> Si los resultados de una clave de consulta ya están en caché, la misma clave de consulta `enableResultCacheWithTTL` no genera nuevos resultados y no tiene ningún efecto en time-to-live los resultados actualmente almacenados en caché.
>
> - Si los resultados se almacenaron previamente en caché utilizando `enableResultCache`, primero se debe borrar la caché antes de que `enableResultCacheWithTTL` genere nuevos resultados y los almacene en caché durante el TTL que especifique.
> - Si los resultados se almacenaron previamente en caché utilizando `enableResultCachewithTTL`, ese TTL anterior debe vencer primero antes de que

`enableResultCacheWithTTL` genere nuevos resultados y los almacene en caché durante el TTL que especifique.

Uso de **`invalidateResultCacheKey`**

Puede utilizar la sugerencia de consulta `invalidateResultCacheKey` para borrar los resultados en caché de una consulta concreta. Por ejemplo:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Esa consulta borra la caché de la clave de consulta, `g.V().has('genre','drama').in('likes')`, y devuelve nuevos resultados para esa consulta.

También se puede combinar `invalidateResultCacheKey` con `enableResultCache` o `enableResultCacheWithTTL`. Por ejemplo, la siguiente consulta borra los resultados actuales almacenados en caché, almacena en caché los nuevos resultados y los devuelve:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Uso de **`invalidateResultCache`**

Puede usar la sugerencia de consulta `invalidateResultCache` para borrar todos los resultados en caché de la caché de resultados de consultas. Por ejemplo:

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Esa consulta borra la caché de resultados completa y devuelve nuevos resultados para la consulta.

También se puede combinar `invalidateResultCache` con `enableResultCache` o `enableResultCacheWithTTL`. Por ejemplo, la siguiente consulta borra toda la caché de resultados, almacena en caché los nuevos resultados de esta consulta y los devuelve:

```
g.with('Neptune#enableResultCache', true)
```

```
  .with('Neptune#invalidateResultCache', true)
  .V().has('genre','drama').in('likes')
```

Paginación de los resultados de consultas en caché

Supongamos que ya ha almacenado en caché un gran número de resultados como este:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Ahora supongamos que ejecuta la siguiente consulta de rango:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes').range(0,10)
```

Neptune busca primero la clave de caché completa, es decir, `g.V().has('genre','drama').in('likes').range(0,10)`. Si esa clave no existe, Neptune busca si hay una clave para esa cadena de consulta sin el rango (es decir, `g.V().has('genre','drama').in('likes')`). Cuando encuentra esa clave, Neptune obtiene los diez primeros resultados de su caché, según lo que especifique el rango.

> **Note**
>
> Si usa la sugerencia de consulta `invalidateResultCacheKey` con una consulta que tiene un rango al final, Neptune borra la caché de una consulta sin el rango si no encuentra una coincidencia exacta para la consulta con el rango.

Uso de **`numResultsCached`** con **`.iterate()`**

Con la sugerencia de consulta `numResultsCached`, puede rellenar la caché de resultados sin devolver todos los resultados almacenados en la caché, lo que puede resultar útil cuando prefiere paginar un gran número de resultados.

La sugerencia de consulta `numResultsCached` solo funciona con las consultas que terminan en `iterate()`.

Por ejemplo, si quiere almacenar en caché los primeros 50 resultados de la consulta de ejemplo:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

En este caso, la clave de consulta de la caché es `g.with("Neptune#numResultsCached", 50).V().has('genre','drama').in('likes')`. Ahora puede recuperar los diez primeros resultados en caché con esta consulta:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Además, puede recuperar los diez resultados siguientes de la consulta de la siguiente manera:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(10, 20)
```

¡No se olvide incluir la sugerencia `numResultsCached`! Es una parte esencial de la clave de consulta y, por lo tanto, debe estar presente para poder acceder a los resultados almacenados en caché.

Algunas cosas a tener en cuenta al usar **`numResultsCached`**

- El número que proporcione con **`numResultsCached`** se aplica al final de la consulta. Esto significa, por ejemplo, que la siguiente consulta almacena en caché los resultados del rango`(1000, 1500)`:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 500)
   .V().range(1000, 2000).iterate()
  ```

- El número que proporcione con **`numResultsCached`** especifica el número máximo de resultados que se van a almacenar en caché. Esto significa, por ejemplo, que la siguiente consulta almacena en caché los resultados del rango`(1000, 2000)`:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 100000)
   .V().range(1000, 2000).iterate()
  ```

- Los resultados que almacenan en caché las consultas que terminan con **`.range().iterate()`** tienen su propio rango. Por ejemplo, supongamos que se almacenan en caché los resultados mediante una consulta como esta:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

Para recuperar los primeros 100 resultados de la caché, escribiría una consulta como esta:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).range(0, 100)
```

Esos cien resultados equivaldrían a los resultados de la consulta base del rango `(1000, 1100)`.

Las claves de caché de consultas que se utilizan para localizar los resultados en caché

Una vez guardados en caché los resultados de una consulta, las consultas posteriores con la misma clave de caché de consultas recuperan los resultados de la caché en lugar de generar otros nuevos. La clave de caché de consultas de una consulta se evalúa de la siguiente manera:

1. Se omiten todas las sugerencias de consulta relacionadas con la caché, excepto `numResultsCached`.
2. Se omite un último paso `iterate()`.
3. El resto de la consulta se ordena según su representación en código de bytes.

La cadena resultante se compara con un índice de los resultados de la consulta que ya están en la caché para determinar si hay un acierto de caché para la consulta.

Por ejemplo, vamos a tomar como ejemplo esta consulta:

```
g.withSideEffect('Neptune#typePromotion', false).with("Neptune#enableResultCache",
 true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Se almacenará como la versión de código de bytes de esto:

```
g.withSideEffect('Neptune#typePromotion', false)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes')
```

Excepciones relacionadas con la caché de resultados

Si los resultados de una consulta que está intentando almacenar en caché son demasiado grandes para caber en la memoria caché incluso después de eliminar todo lo que estaba almacenado en caché anteriormente, Neptune genera un error `QueryLimitExceededException`. No se devuelve ningún resultado y la excepción genera el siguiente mensaje de error:

```
The result size is larger than the allocated cache,
      please refer to results cache best practices for options to rerun the query.
```

Puede suprimir este mensaje mediante la sugerencia de consulta no`CacheExceptions`, de la siguiente manera:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Realización de actualizaciones o inserciones eficientes con los pasos **`mergeV()`** y **`mergeE()`** de Gremlin

Una actualización o inserción (o inserción condicional) reutiliza un vértice o un borde si ya existe, o lo crea si no existe. Las actualizaciones o inserciones eficientes pueden marcar una diferencia significativa en el rendimiento de las consultas de Gremlin.

Las actualizaciones o inserciones permiten escribir operaciones de inserción de idempotencia; independientemente del número de veces que se ejecute una operación de este tipo, el resultado general es el mismo. Esto resulta útil en situaciones de escritura muy simultáneas, en las que las modificaciones simultáneas en la misma parte del gráfico pueden obligar a una o más transacciones a revertirse con una `ConcurrentModificationException`, por lo que es necesario volver a intentarlas.

Por ejemplo, la siguiente consulta desplaza realiza actualizaciones o inserciones en un vértice utilizando el `Map` proporcionado para intentar buscar primero un vértice con un `T.id` de `"v-1"`.

Si se encuentra ese vértice, se devuelve. Si no se encuentra, se crea un vértice con ese `id` y una propiedad a través de la cláusula `onCreate`.

```
g.mergeV([(id):'v-1']).
  option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org'])
```

Realización de actualizaciones o inserciones por lotes para mejorar el rendimiento

En escenarios de escritura de alto rendimiento, puede encadenar los pasos `mergeV()` y `mergeE()` para realizar actualizaciones o inserciones en vértices y bordes por lotes. El procesamiento por lotes reduce la sobrecarga transaccional que supone realizar actualizaciones o inserciones en un gran número de vértices y bordes. A continuación, puede mejorar aún más el rendimiento realizando actualizaciones o inserciones en solicitudes por lotes en paralelo con varios clientes.

Como regla general, recomendamos modificar aproximadamente 200 registros por solicitud en lote. Un registro es una propiedad o etiqueta de un solo vértice o borde. Un vértice con una sola etiqueta y 4 propiedades, por ejemplo, crea 5 registros. Un borde con una etiqueta y una sola propiedad crea 2 registros. Si desea realizar actualizaciones o inserciones en lotes de vértices, cada uno con una sola etiqueta y 4 propiedades, debería empezar con un tamaño de lote de 40, ya que `200 / (1 + 4) = 40`.

Puede experimentar con el tamaño del lote. Un buen punto de partida son 200 registros por lote, pero el tamaño de lote ideal puede ser mayor o menor en función de la carga de trabajo. Sin embargo, tenga en cuenta que Neptune puede limitar el número total de pasos de Gremlin por solicitud. Este límite no está documentado, pero por si acaso, intente asegurarse de que sus solicitudes no contengan más de 1500 pasos de Gremlin. Neptune puede rechazar solicitudes por lotes grandes con más de 1500 pasos.

Para aumentar el rendimiento, puede realizar actualizaciones o inserciones en los lotes en paralelo utilizando varios clientes (consulte Creación de escrituras de Gremlin eficientes de múltiples subprocesos). La cantidad de clientes debe ser la misma que la cantidad de subprocesos de trabajadores de la instancia de escritor de Neptune, que normalmente es 2 veces la cantidad de vCPU del servidor. Por ejemplo, una instancia de `r5.8xlarge` tiene 32 vCPU y 64 subprocesos de trabajadores. En escenarios de escritura de alto rendimiento en los que se usa un `r5.8xlarge`, utilizaría 64 clientes que escriben actualizaciones o inserciones por lotes en Neptune en paralelo.

Cada cliente debe enviar una solicitud por lotes y esperar a que se complete antes de enviar otra. Aunque los múltiples clientes se ejecutan en paralelo, cada cliente individual envía las solicitudes en serie. Esto garantiza que el servidor reciba un flujo constante de solicitudes que ocupen todos

los subprocesos de trabajadores sin saturar la cola de solicitudes del lado del servidor (consulte Dimensionamiento de las instancias de base de datos en un clúster de base de datos de Neptune).

Intente evitar pasos que generen múltiples recorridos

Cuando se ejecuta un paso de Gremlin, toma un recorrido entrante y emite uno o más recorridos de salida. El número de entrante que emite un paso determina el número de veces que se ejecuta el siguiente paso.

Por lo general, cuando se realizan operaciones por lotes, se desea que cada operación, como el vértice de actualización o inserción A, se ejecute una vez, de modo que la secuencia de operaciones tenga el siguiente aspecto: vértice de actualización o inserción A, vértice de actualización o inserción B, vértice de actualización o inserción C, etc. Mientras que un paso cree o modifique solo un elemento, solo emite un recorrido y los pasos que representan la siguiente operación se ejecutan solo una vez. Si, por otro lado, una operación crea o modifica más de un elemento, emite varios recorridos, lo que a su vez provoca que los pasos siguientes se ejecuten varias veces, una vez por recorrido emitido. Esto puede provocar que la base de datos realice un trabajo adicional innecesario y, en algunos casos, que se creen vértices, bordes o valores de propiedad adicionales no deseados.

Un ejemplo de cómo esto puede ir mal es con una consulta como `g.V().addV()`. Esta sencilla consulta añade un vértice por cada vértice que se encuentre en el gráfico, ya que `V()` emite un recorrido para cada vértice del gráfico y cada uno de esos recorridos activa una llamada a `addV()`.

Consulte en Combinación de actualizaciones o inserciones e inserciones las formas de gestionar las operaciones que pueden emitir varios recorridos.

Actualizaciones o inserciones en vértices

El paso `mergeV()` está diseñado específicamente para realizar opciones actualizaciones o inserciones en vértices. Toma como argumento un `Map` que representa los elementos que deben coincidir con los vértices existentes en el gráfico y, si no se encuentra un elemento, usa ese `Map` para crear un nuevo vértice. El paso también le permite alterar el comportamiento en caso de una creación o una coincidencia, donde el modulador `option()` se puede aplicar con los tokens `Merge.onCreate` y `Merge.onMatch` para controlar esos comportamientos respectivos. Consulte la documentación de TinkerPop referencia para obtener más información sobre cómo utilizar este paso.

Puede usar un identificador de vértice para determinar si existe un vértice específico. Este es el enfoque preferido, ya que Neptune optimiza las actualizaciones o inserciones para casos de uso con una simultaneidad muy alta en torno a los identificadores. Por ejemplo, la siguiente consulta crea un vértice con un identificador de vértice determinado si aún no existe, o lo reutiliza si ya existe:

```
g.mergeV([(T.id): 'v-1']).
    option(onCreate, [(T.label): 'PERSON', email: 'person-1@example.org', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Tenga en cuenta que esta consulta termina con un paso `id()`. Si bien no es estrictamente necesario para realizar una actualización o inserción en el vértice, un paso `id()` hasta el final de una consulta de actualización o inserción garantiza que el servidor no vuelva a serializar todas las propiedades del vértice para el cliente, lo que ayuda a reducir el costo de bloqueo de la consulta.

También puede utilizar una propiedad de vértice para identificar un vértice:

```
g.mergeV([email: 'person-1@example.org']).
    option(onCreate, [(T.label): 'PERSON', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Si es posible, utilice sus propios identificadores proporcionados por el usuario para crear vértices y utilice estos identificadores para determinar si existe un vértice durante una operación de actualización o inserción. Esto permite a Neptune optimizar las actualizaciones o inserciones. Una actualización o inserción basada en identificador puede ser considerablemente más eficiente que una actualización o inserción basada en propiedades cuando las modificaciones simultáneas son frecuentes.

Encadenamiento de actualizaciones o inserciones de vértices

Puede encadenar actualizaciones o inserciones de vértices para insertarlos en un lote:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
```

```
          addV('Person').property(id, 'v-3')
                        .property('email', 'person-3@example.org'))
  .id()
```

También puede utilizar esta sintaxis de `mergeV()`:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org'])
```

Sin embargo, dado que esta forma de consulta incluye elementos en los criterios de búsqueda que son superfluos para la búsqueda básica de `id`, no es tan eficaz como la consulta anterior.

Actualizaciones o inserciones de bordes

El paso `mergeE()` está diseñado específicamente para realizar actualizaciones o inserciones de bordes. Toma como argumento un `Map` que representa los elementos que deben coincidir con los bordes existentes en el gráfico y, si no se encuentra un elemento, usa ese `Map` para crear un nuevo borde. El paso también le permite alterar el comportamiento en caso de una creación o una coincidencia, donde el modulador `option()` se puede aplicar con los tokens `Merge.onCreate` y `Merge.onMatch` para controlar esos comportamientos respectivos. Consulte la documentación de TinkerPop referencia para obtener más información sobre cómo utilizar este paso.

Puede usar los identificadores de borde para actualizaciones o inserciones de bordes de la misma manera que realiza actualizaciones o inserciones de vértices con identificadores de vértices personalizados. De nuevo, este es el enfoque preferido porque permite a Neptune optimizar la consulta. Por ejemplo, la siguiente consulta crea un borde en función de su identificador de borde si aún no existe, o lo reutiliza si ya existe. La consulta también utiliza los identificadores de los vértices `Direction.from` y `Direction.to` si necesita crear un borde nuevo:

```
g.mergeE([(T.id): 'e-1']).
    option(onCreate, [(from): 'v-1', (to): 'v-2', weight: 1.0]).
    option(onMatch, [weight: 0.5]).
  id()
```

Tenga en cuenta que esta consulta termina con un paso `id()`. Si bien no es estrictamente necesario para realizar una actualización o inserción del borde, un paso `id()` hasta el final de una consulta de actualización o inserción garantiza que el servidor no vuelva a serializar todas las propiedades del borde para el cliente, lo que ayuda a reducir el costo de bloqueo de la consulta.

Muchas aplicaciones utilizan identificadores de vértices personalizados, pero dejan que Neptune genere identificadores de borde. Si no conoce el identificador de un borde, pero sí los identificadores de vértice `from` y `to`, puede utilizar este tipo de consulta para realizar una actualización o inserción en un borde:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  id()
```

Todos los vértices a los que hace referencia `mergeE()` deben existir para que el paso cree el borde.

Encadenamiento de actualizaciones o inserciones de bordes

Al igual que con las actualizaciones o inserciones de vértices, es sencillo encadenar los pasos `mergeE()` para las solicitudes por lotes:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
  mergeE([(from): 'v-3', (to): 'v-4', (T.label): 'KNOWS']).
  id()
```

Combinación de actualizaciones o inserciones de vértices y bordes

A veces, es posible que desee realizar actualizaciones o inserciones en ambos vértices y los bordes que los conectan. Puede combinar los ejemplos de lotes que se presentan aquí. En el siguiente ejemplo, se realizan actualizaciones o inserciones de 3 vértices y 2 bordes:

```
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  mergeE([(from): 'v-2', (to): 'v-3', (T.label): 'KNOWS']).
 id()
```

Combinación de actualizaciones o inserciones e inserciones

A veces, es posible que desee realizar actualizaciones o inserciones en ambos vértices y los bordes que los conectan. Puede combinar los ejemplos de lotes que se presentan aquí. En el siguiente ejemplo, se realizan actualizaciones o inserciones de 3 vértices y 2 bordes:

Por lo general, las actualizaciones o inserciones se realizan con un elemento a la vez. Si sigue los patrones de actualización o inserción que se presentan aquí, cada operación de actualización o inserción emite un único recorrido, lo que hace que la siguiente operación se ejecute solo una vez.

Sin embargo, a veces es posible que desee mezclar actualizaciones o inserciones con inserciones. Podría ser así, por ejemplo, si utiliza bordes para representar instancias de acciones o eventos. Una solicitud puede usar actualizaciones o inserciones para garantizar que existan todos los vértices necesarios y, a continuación, usar inserciones para añadir bordes. En el caso de solicitudes de este tipo, preste atención a la cantidad potencial de recorridos que emite cada operación.

Examine el siguiente ejemplo, en el que se combinan actualizaciones o inserciones e inserciones para añadir bordes que representen eventos en el gráfico:

```
// Fully optimized, but inserts too many edges
g.mergeV([(id):'v-1']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org']).
  mergeV([(id):'v-2']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-2@example.org']).
  mergeV([(id):'v-3']).
    option(onCreate, [(label): 'PERSON', 'email': 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').to(V('p-1')).
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').to(V('c-1')).
  id()
```

La consulta debe insertar 5 bordes: 2 bordes FOLLOWED y 3 bordes VISITED. Sin embargo, la consulta tal como está escrita inserta 8 bordes: 2 FOLLOWED y 6 VISITED. Esto se debe a que la operación que inserta los 2 bordes FOLLOWED emite 2 recorridos, lo que provoca que la siguiente operación de inserción, que inserta 3 bordes, se ejecute dos veces.

La solución consiste en añadir un paso `fold()` después de cada operación que pueda emitir más de un recorrido:

```
g.mergeV([(T.id): 'v-1', (T.label): 'PERSON', email: 'person-1@example.org']).
  mergeV([(T.id): 'v-2', (T.label): 'PERSON', email: 'person-2@example.org']).
  mergeV([(T.id): 'v-3', (T.label): 'PERSON', email: 'person-3@example.org']).
  mergeV([(T.id): 'c-1', (T.label): 'CITY', name: 'city-1']).
  V('p-1', 'p-2').
  addE('FOLLOWED').
```

```
    to(V('p-1')).
  fold().
  V('p-1', 'p-2', 'p-3').
  addE('VISITED').
    to(V('c-1')).
  id()
```

Aquí hemos insertado un paso `fold()` después de la operación que inserta los bordes FOLLOWED. Esto da como resultado un único recorrido, lo que hace que la siguiente operación se ejecute solo una vez.

La desventaja de este enfoque es que la consulta ahora no se optimiza por completo, porque `fold()` no se optimiza. La siguiente operación de inserción que sigue `fold()` ahora tampoco se optimizará.

Si necesita usar `fold()` para reducir el número de recorridos para los pasos posteriores, intente ordenar las operaciones de manera que las menos costosas ocupen la parte no optimizada de la consulta.

Establecer la cardinalidad

La cardinalidad predeterminada para las propiedades de los vértices en Neptune está establecida, lo que significa que al usar mergeV (), a todos los valores proporcionados en el mapa se les asignará esa cardinalidad. Para usar una cardinalidad única, debe ser explícito en su uso. A partir de la TinkerPop versión 3.7.0, hay una nueva sintaxis que permite proporcionar la cardinalidad como parte del mapa, como se muestra en el siguiente ejemplo:

```
g.mergeV([(T.id): 1234]).
  option(onMatch, ['age': single(20), 'name': single('alice'), 'city': set('miami')])
```

Como alternativa, puede establecer la cardinalidad por defecto de la siguiente manera`option`:

```
// age and name are set to single cardinality by default
g.mergeV([(T.id): 1234]).
  option(onMatch, ['age': 22, 'name': 'alice', 'city': set('boston')], single)
```

En las versiones `mergeV()` anteriores a la 3.7.0, había menos opciones para establecer la cardinalidad. El enfoque general consiste en volver al `property()` paso siguiente:

```
g.mergeV([(T.id): '1234']).
  option(onMatch, sideEffect(property(single,'age', 20).
```

```
  property(set,'city','miami')).constant([:]))
```

> **Note**
>
> Este enfoque solo funcionará `mergeV()` cuando se utilice con un paso inicial. Por lo tanto, no podría encadenarse `mergeV()` dentro de un único recorrido, ya que el primero `mergeV()` después del paso de inicio que utilice esta sintaxis producirá un error si el travesaño entrante es un elemento gráfico. En este caso, querrás dividir las `mergeV()` llamadas en varias solicitudes, cada una de las cuales puede ser un paso inicial.

Realización de actualizaciones o inserciones de Gremlin eficientes con `fold()/coalesce()/unfold()`

Una actualización o inserción (o inserción condicional) reutiliza un vértice o un borde si ya existe, o lo crea si no existe. Las actualizaciones o inserciones eficientes pueden marcar una diferencia significativa en el rendimiento de las consultas de Gremlin.

En esta página se muestra cómo usar el patrón de Gremlin `fold()/coalesce()/unfold()` para crear actualizaciones o inserciones eficientes. Sin embargo, con el lanzamiento de la TinkerPop versión 3.6.x introducido en Neptune en la versión 1.2.1.0 del motor, en la mayoría de los casos son preferibles lo nuevo `mergeV()` y los `mergeE()` escalones. El patrón `fold()/coalesce()/unfold()` descrito aquí podría seguir siendo útil en algunas situaciones complejas, pero en general, utilice `mergeV()` y `mergeE()` si es posible, como se describe en Realización de actualizaciones o inserciones eficientes con los pasos `mergeV()` y `mergeE()` de Gremlin.

Las actualizaciones o inserciones permiten escribir operaciones de inserción de idempotencia; independientemente del número de veces que se ejecute una operación de este tipo, el resultado general es el mismo. Esto resulta útil en situaciones de escritura muy simultáneas, en las que las modificaciones simultáneas en la misma parte del gráfico pueden obligar a una o más transacciones a revertirse con una `ConcurrentModificationException`, por lo que es necesario volver a intentarlas.

Por ejemplo, la siguiente consulta realiza actualizaciones o inserciones en un vértice. Para ello, busca primero el vértice especificado en el conjunto de datos y, a continuación, pliega los resultados en una lista. En el primer recorrido proporcionado al paso `coalesce()`, la consulta despliega esta lista. Si la lista desplegada no está vacía, los resultados se emiten desde `coalesce()`. Sin embargo, si `unfold()` devuelve una colección vacía porque el vértice no existe actualmente,

`coalesce()` pasa a evaluar el segundo recorrido con el que se ha proporcionado y, en este segundo recorrido , la consulta crea el vértice que falta.

```
g.V('v-1').fold()
          .coalesce(
             unfold(),
             addV('Person').property(id, 'v-1')
                           .property('email', 'person-1@example.org')
           )
```

Utilice una forma optimizada de **`coalesce()`** para las actualizaciones o inserciones

Neptune puede optimizar la expresión `fold().coalesce(unfold(), ...)` para realizar actualizaciones de alto rendimiento, pero esta optimización solo funciona si ambas partes de `coalesce()` devuelven un vértice o un borde, pero nada más. Si intenta devolver algo diferente, como una propiedad, desde cualquier parte del `coalesce()`, no se produce la optimización de Neptune. Es posible que la consulta se realice correctamente, pero no funcionará tan bien como en una versión optimizada, especialmente en conjuntos de datos de gran tamaño.

Dado que las consultas de actualización o inserción no optimizadas aumentan los tiempos de ejecución y reducen el rendimiento, vale la pena utilizar el punto de conexión `explain` de Gremlin para determinar si una consulta de actualización o inserción está totalmente optimizada. Al revisar los planes `explain`, busque líneas que comiencen por `+ not converted into Neptune steps` y `WARNING: >>`. Por ejemplo:

```
+ not converted into Neptune steps: [FoldStep, CoalesceStep([[UnfoldStep],
 [AddEdgeSte...
WARNING: >> FoldStep << is not supported natively yet
```

Estas advertencias pueden ayudarle a identificar las partes de una consulta que impiden que se optimice por completo.

A veces, no es posible optimizar una consulta por completo. En estas situaciones, debe intentar colocar los pasos que no se pueden optimizar al final de la consulta, lo que permitirá que el motor optimice tantos pasos como sea posible. Esta técnica se utiliza en algunos de los ejemplos de actualizaciones o inserciones por lotes, en los que todas las actualizaciones o inserciones optimizadas para un conjunto de vértices o bordes se realizan antes de aplicar cualquier modificación adicional, potencialmente no optimizada, a los mismos vértices o bordes.

Realización de actualizaciones o inserciones por lotes para mejorar el rendimiento

En escenarios de escritura de alto rendimiento, puede encadenar pasos de actualizaciones o inserciones para realizar actualizaciones o inserciones en vértices y bordes por lotes. El procesamiento por lotes reduce la sobrecarga transaccional que supone realizar actualizaciones o inserciones en un gran número de vértices y bordes. A continuación, puede mejorar aún más el rendimiento realizando actualizaciones o inserciones en solicitudes por lotes en paralelo con varios clientes.

Como regla general, recomendamos modificar aproximadamente 200 registros por solicitud en lote. Un registro es una propiedad o etiqueta de un solo vértice o borde. Un vértice con una sola etiqueta y 4 propiedades, por ejemplo, crea 5 registros. Un borde con una etiqueta y una sola propiedad crea 2 registros. Si desea realizar actualizaciones o inserciones en lotes de vértices, cada uno con una sola etiqueta y 4 propiedades, debería empezar con un tamaño de lote de 40, ya que `200 / (1 + 4) = 40`.

Puede experimentar con el tamaño del lote. Un buen punto de partida son 200 registros por lote, pero el tamaño de lote ideal puede ser mayor o menor en función de la carga de trabajo. Sin embargo, tenga en cuenta que Neptune puede limitar el número total de pasos de Gremlin por solicitud. Este límite no está documentado, pero por si acaso, intente asegurarse de que sus solicitudes no contengan más de 1500 pasos de Gremlin. Neptune puede rechazar solicitudes por lotes grandes con más de 1500 pasos.

Para aumentar el rendimiento, puede realizar actualizaciones o inserciones en los lotes en paralelo utilizando varios clientes (consulte Creación de escrituras de Gremlin eficientes de múltiples subprocesos). La cantidad de clientes debe ser la misma que la cantidad de subprocesos de trabajadores de la instancia de escritor de Neptune, que normalmente es 2 veces la cantidad de vCPU del servidor. Por ejemplo, una instancia de `r5.8xlarge` tiene 32 vCPU y 64 subprocesos de trabajadores. En escenarios de escritura de alto rendimiento en los que se usa un `r5.8xlarge`, utilizaría 64 clientes que escriben actualizaciones o inserciones por lotes en Neptune en paralelo.

Cada cliente debe enviar una solicitud por lotes y esperar a que se complete antes de enviar otra. Aunque los múltiples clientes se ejecutan en paralelo, cada cliente individual envía las solicitudes en serie. Esto garantiza que el servidor reciba un flujo constante de solicitudes que ocupen todos los subprocesos de trabajadores sin saturar la cola de solicitudes del lado del servidor (consulte Dimensionamiento de las instancias de base de datos en un clúster de base de datos de Neptune).

Intente evitar pasos que generen múltiples recorridos

Cuando se ejecuta un paso de Gremlin, toma un recorrido entrante y emite uno o más recorridos de salida. El número de entrante que emite un paso determina el número de veces que se ejecuta el siguiente paso.

Por lo general, cuando se realizan operaciones por lotes, se desea que cada operación, como el vértice de actualización o inserción A, se ejecute una vez, de modo que la secuencia de operaciones tenga el siguiente aspecto: vértice de actualización o inserción A, vértice de actualización o inserción B, vértice de actualización o inserción C, etc. Mientras que un paso cree o modifique solo un elemento, solo emite un recorrido y los pasos que representan la siguiente operación se ejecutan solo una vez. Si, por otro lado, una operación crea o modifica más de un elemento, emite varios recorridos, lo que a su vez provoca que los pasos siguientes se ejecuten varias veces, una vez por recorrido emitido. Esto puede provocar que la base de datos realice un trabajo adicional innecesario y, en algunos casos, que se creen vértices, bordes o valores de propiedad adicionales no deseados.

Un ejemplo de cómo esto puede ir mal es con una consulta como `g.V().addV()`. Esta sencilla consulta añade un vértice por cada vértice que se encuentre en el gráfico, ya que `V()` emite un recorrido para cada vértice del gráfico y cada uno de esos recorridos activa una llamada a `addV()`.

Consulte en Combinación de actualizaciones o inserciones e inserciones las formas de gestionar las operaciones que pueden emitir varios recorridos.

Actualizaciones o inserciones en vértices

Puede usar un identificador de vértice para determinar si existe un vértice correspondiente. Este es el enfoque preferido, ya que Neptune optimiza las actualizaciones o inserciones para casos de uso con una simultaneidad muy alta en torno a los identificadores. Por ejemplo, la siguiente consulta crea un vértice con un identificador de vértice determinado si aún no existe, o lo reutiliza si ya existe:

```
g.V('v-1')
 .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .id()
```

Tenga en cuenta que esta consulta termina con un paso `id()`. Si bien no es estrictamente necesario para realizar una actualización o inserción en el vértice, añadir un paso `id()` hasta el final de

una consulta de actualización o inserción garantiza que el servidor no vuelva a serializar todas las propiedades del vértice para el cliente, lo que ayuda a reducir el costo de bloqueo de la consulta.

También puede utilizar una propiedad de vértice para determinar si el vértice existe:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .fold()
 .coalesce(unfold(),
           addV('Person').property('email', 'person-1@example.org'))
 .id()
```

Si es posible, utilice sus propios identificadores proporcionados por el usuario para crear vértices y utilice estos identificadores para determinar si existe un vértice durante una operación de actualización o inserción. Esto permite a Neptune optimizar las actualizaciones o inserciones en torno a los identificadores. Una actualización o inserción basada en identificador puede ser considerablemente más eficiente que una actualización o inserción basada en propiedades en escenarios con una gran simultaneidad en las modificaciones.

Encadenamiento de actualizaciones o inserciones de vértices

Puede encadenar actualizaciones o inserciones de vértices para insertarlos en un lote:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .id()
```

Actualizaciones o inserciones de bordes

Puede usar los identificadores de borde para actualizaciones o inserciones de bordes de la misma manera que realiza actualizaciones o inserciones de vértices con identificadores de vértices personalizados. De nuevo, este es el enfoque preferido porque permite a Neptune optimizar la consulta. Por ejemplo, la siguiente consulta crea un borde en función de su identificador de borde si aún no existe, o lo reutiliza si ya existe. La consulta también utiliza los identificadores de los vértices `from` y `to` si necesita crear un borde nuevo.

```
g.E('e-1')
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2'))
                        .property(id, 'e-1'))
 .id()
```

Muchas aplicaciones utilizan identificadores de vértices personalizados, pero dejan que Neptune genere identificadores de borde. Si no conoce el identificador de un borde, pero sí los identificadores de vértice `from` y `to`, puede utilizar esta fórmula para realizar una actualización o inserción en un borde:

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2')))
 .id()
```

Tenga en cuenta que el paso de vértice de la cláusula `where()` debería ser `inV()` (o `outV()` si ha utilizado `inE()` para buscar el borde), no `otherV()`. No utilice `otherV()` aquí o la consulta no se optimizará y el rendimiento se verá afectado. Por ejemplo, Neptune no optimizaría la siguiente consulta:

```
// Unoptimized upsert, because of otherV()
g.V('v-1')
 .outE('KNOWS')
 .where(otherV().hasId('v-2'))
```

```
  .fold()
  .coalesce(unfold(),
            addE('KNOWS').from(V('v-1'))
                         .to(V('v-2')))
  .id()
```

Si no conoce los identificadores de borde o vértice desde el principio, puede realizar una actualización o inserción mediante las propiedades de los vértices:

```
g.V()
  .hasLabel('Person')
  .has('name', 'person-1')
  .outE('LIVES_IN')
  .where(inV().hasLabel('City').has('name', 'city-1'))
  .fold()
  .coalesce(unfold(),
            addE('LIVES_IN').from(V().hasLabel('Person')
                                     .has('name', 'person-1'))
                            .to(V().hasLabel('City')
                                   .has('name', 'city-1')))
  .id()
```

Al igual que con las actualizaciones o inserciones de vértices, es preferible utilizar actualizaciones o inserciones de bordes basadas en identificadores utilizando un identificador de borde o identificadores de vértice `from` y `to`, en lugar de actualizaciones o inserciones basadas en propiedades, de modo que Neptune pueda optimizar completamente la actualización o inserción.

Comprobación de la existencia de vértices **`from`** y **`to`**

Observe la construcción de los pasos que permiten crear un nuevo borde: `addE().from().to()`. Esta construcción garantiza que la consulta compruebe la existencia tanto del vértice `from` como `to`. Si alguna de estas opciones no existe, la consulta devuelve el siguiente error:

```
{
  "detailedMessage": "Encountered a traverser that does not map to a value for child...
  "code": "IllegalArgumentException",
  "requestId": "..."
}
```

Si es posible que el vértice `from` o `to` no existan, debería intentar realizar actualizaciones o inserciones en ellos antes de realizar actualizaciones o inserciones en el borde entre ellos. Consulte Combinación de actualizaciones o inserciones de vértices y bordes.

Existe una construcción alternativa para crear un borde que no debería usar: `V().addE().to()`. Solo añade un borde si existe el vértice `from`. Si el vértice `to` no existe, la consulta genera un error, como se ha descrito anteriormente, pero si el vértice `from` no existe, falla de forma silenciosa al insertar un borde, sin generar ningún error. Por ejemplo, si el vértice `from` no existe, la siguiente actualización o inserción se completa sin realizar actualizaciones o inserciones en un borde:

```
// Will not insert edge if from vertex does not exist
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2')))
 .id()
```

Encadenamiento de actualizaciones o inserciones de bordes

Si desea encadenar actualizaciones o inserciones de bordes para crear una solicitud en lote, debe comenzar cada actualización o inserción con una búsqueda de vértices, incluso si ya conoce los identificadores de los bordes.

Si ya conoce los identificadores de los bordes en los que desea realizar la actualización o inserción y los identificadores de los vértices `from` y `to`, puede utilizar esta fórmula:

```
g.V('v-1')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V('v-1').addE('KNOWS')
                   .to(V('v-2'))
                   .property(id, 'e-1'))
 .V('v-3')
 .outE('KNOWS')
 .hasId('e-2').fold()
 .coalesce(unfold(),
           V('v-3').addE('KNOWS')
```

```
                  .to(V('v-4'))
                  .property(id, 'e-2'))
 .V('v-5')
 .outE('KNOWS')
 .hasId('e-3')
 .fold()
 .coalesce(unfold(),
          V('v-5').addE('KNOWS')
                  .to(V('v-6'))
                  .property(id, 'e-3'))
 .id()
```

Quizás el escenario más común de actualización o inserción de bordes por lotes sea que conozca los identificadores de los vértices `from` y `to`, pero que desconozca los identificadores de los bordes en los que desea realizar la actualización o inserción. En ese caso, utilice la siguiente fórmula:

```
g.V('v-1')
 .outE('KNOWS')
 .where(inV().hasId('v-2'))
 .fold()
 .coalesce(unfold(),
          V('v-1').addE('KNOWS')
                  .to(V('v-2')))

 .V('v-3')
 .outE('KNOWS')
 .where(inV().hasId('v-4'))
 .fold()
 .coalesce(unfold(),
          V('v-3').addE('KNOWS')
                  .to(V('v-4')))
 .V('v-5')
 .outE('KNOWS')
 .where(inV().hasId('v-6'))
 .fold()
 .coalesce(unfold(),
          V('v-5').addE('KNOWS').to(V('v-6')))
 .id()
```

Si conoce los identificadores de los bordes en los que desea realizar una actualización o inserción, pero no conoce los identificadores de los vértices `from` y `to` (aunque no es habitual), puede utilizar esta fórmula:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-1@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-2@example.org'))
                     .property(id, 'e-1'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-3@example.org')
 .outE('KNOWS')
 .hasId('e-2')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-3@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-4@example.org'))
              .property(id, 'e-2'))
 .V()
 .hasLabel('Person')
 .has('email', 'person-5@example.org')
 .outE('KNOWS')
 .hasId('e-1')
 .fold()
 .coalesce(unfold(),
           V().hasLabel('Person')
              .has('email', 'person-5@example.org')
              .addE('KNOWS')
              .to(V().hasLabel('Person')
                     .has('email', 'person-6@example.org'))
                     .property(id, 'e-3'))
 .id()
```

Combinación de actualizaciones o inserciones de vértices y bordes

A veces, es posible que desee realizar actualizaciones o inserciones en ambos vértices y los bordes que los conectan. Puede combinar los ejemplos de lotes que se presentan aquí. En el siguiente ejemplo, se realizan actualizaciones o inserciones de 3 vértices y 2 bordes:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2')
                         .property('name', 'person-2@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
           addV('City').property(id, 'c-1')
                       .property('name', 'city-1'))
 .V('p-1')
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
 .fold()
 .coalesce(unfold(),
           V('p-1').addE('LIVES_IN')
                   .to(V('c-1')))
 .V('p-2')
 .outE('LIVES_IN')
 .where(inV().hasId('c-1'))
 .fold()
 .coalesce(unfold(),
           V('p-2').addE('LIVES_IN')
                   .to(V('c-1')))
 .id()
```

Combinación de actualizaciones o inserciones e inserciones

A veces, es posible que desee realizar actualizaciones o inserciones en ambos vértices y los bordes que los conectan. Puede combinar los ejemplos de lotes que se presentan aquí. En el siguiente ejemplo, se realizan actualizaciones o inserciones de 3 vértices y 2 bordes:

Por lo general, las actualizaciones o inserciones se realizan con un elemento a la vez. Si sigue los patrones de actualización o inserción que se presentan aquí, cada operación de actualización o inserción emite un único recorrido, lo que hace que la siguiente operación se ejecute solo una vez.

Sin embargo, a veces es posible que desee mezclar actualizaciones o inserciones con inserciones. Podría ser así, por ejemplo, si utiliza bordes para representar instancias de acciones o eventos. Una solicitud puede usar actualizaciones o inserciones para garantizar que existan todos los vértices necesarios y, a continuación, usar inserciones para añadir bordes. En el caso de solicitudes de este tipo, preste atención a la cantidad potencial de recorridos que emite cada operación.

Examine el siguiente ejemplo, en el que se combinan actualizaciones o inserciones e inserciones para añadir bordes que representen eventos en el gráfico:

```
// Fully optimized, but inserts too many edges
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2')
                         .property('name', 'person-2@example.org'))
 .V('p-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-3')
                         .property('name', 'person-3@example.org'))
 .V('c-1')
 .fold()
 .coalesce(unfold(),
           addV('City').property(id, 'c-1')
                       .property('name', 'city-1'))
 .V('p-1', 'p-2')
 .addE('FOLLOWED')
 .to(V('p-1'))
 .V('p-1', 'p-2', 'p-3')
 .addE('VISITED')
 .to(V('c-1'))
 .id()
```

La consulta debe insertar 5 bordes: 2 bordes FOLLOWED y 3 bordes VISITED. Sin embargo, la consulta tal como está escrita inserta 8 bordes: 2 FOLLOWED y 6 VISITED. Esto se debe a que la operación que inserta los 2 bordes FOLLOWED emite 2 recorridos, lo que provoca que la siguiente operación de inserción, que inserta 3 bordes, se ejecute dos veces.

La solución consiste en añadir un paso `fold()` después de cada operación que pueda emitir más de un recorrido:

```
g.V('p-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-1')
                         .property('email', 'person-1@example.org'))
 .V('p-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-2').
                         .property('name', 'person-2@example.org'))
 .V('p-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'p-3').
                         .property('name', 'person-3@example.org'))
 .V('c-1')
 .fold().
 .coalesce(unfold(),
            addV('City').property(id, 'c-1').
                        .property('name', 'city-1'))
 .V('p-1', 'p-2')
 .addE('FOLLOWED')
 .to(V('p-1'))
 .fold()
 .V('p-1', 'p-2', 'p-3')
 .addE('VISITED')
 .to(V('c-1')).
 .id()
```

Aquí hemos insertado un paso `fold()` después de la operación que inserta los bordes FOLLOWED. Esto da como resultado un único recorrido, lo que hace que la siguiente operación se ejecute solo una vez.

La desventaja de este enfoque es que la consulta ahora no se optimiza por completo, porque `fold()` no se optimiza. La operación de inserción que sigue a `fold()` no estará optimizada.

Si necesita usar `fold()` para reducir el número de recorridos para los pasos posteriores, intente ordenar las operaciones de manera que las menos costosas ocupen la parte no optimizada de la consulta.

Actualizaciones o inserciones que modifican vértices y bordes existentes

A veces, desea crear un vértice o un borde si no existe y, a continuación, añadir o actualizar una propiedad, independientemente de si se trata de un vértice o un borde nuevo o existente.

Para añadir o modificar una propiedad, utilice el paso `property()`. Utilice este paso fuera del paso `coalesce()`. Si intenta modificar la propiedad de un vértice o borde existente dentro del paso `coalesce()`, es posible que el motor de consultas de Neptune no optimice la consulta.

La siguiente consulta añade o actualiza una propiedad de contador en cada vértice en el que se ha realizado una actualización o inserción. Cada paso `property()` tiene una cardinalidad única para garantizar que los nuevos valores sustituyan a los valores existentes, en lugar de añadirlos a un conjunto de valores existentes.

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .property(single, 'counter', 1)
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .property(single, 'counter', 2)
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .property(single, 'counter', 3)
 .id()
```

Si tiene un valor de propiedad, como un valor de marca temporal `lastUpdated`, que se aplica a todos los elementos en los que se ha realizado una actualización o inserción, puede añadirlo o actualizarlo al final de la consulta:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org'))
 .V('v-2').
 .fold().
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org'))
 .V('v-3')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-3')
                         .property('email', 'person-3@example.org'))
 .V('v-1', 'v-2', 'v-3')
 .property(single, 'lastUpdated', datetime('2020-02-08'))
 .id()
```

Si hay condiciones adicionales que determinan si se debe seguir modificando un vértice o un borde, puede utilizar un paso `has()` para filtrar los elementos a los que se aplicará la modificación. En el siguiente ejemplo, se utiliza un paso `has()` para filtrar los vértices en los que se han realizado actualizaciones o inserciones en función del valor de su propiedad `version`. A continuación, la consulta actualiza a 3 la `version` de cualquier vértice cuyo `version` es menor que 3:

```
g.V('v-1')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-1')
                         .property('email', 'person-1@example.org')
                         .property('version', 3))
 .V('v-2')
 .fold()
 .coalesce(unfold(),
           addV('Person').property(id, 'v-2')
                         .property('email', 'person-2@example.org')
```

```
                        .property('version', 3))
.V('v-3')
.fold()
.coalesce(unfold(),
          addV('Person').property(id, 'v-3')
                        .property('email', 'person-3@example.org')
                        .property('version', 3))
.V('v-1', 'v-2', 'v-3')
.has('version', lt(3))
.property(single, 'version', 3)
.id()
```

Análisis de la ejecución de las consultas de Neptune con **`explain`** de Gremlin

Amazon Neptune ha añadido una característica de Gremlin denominada explain. Esta característica es una herramienta de autoservicio para comprender el enfoque de ejecución adoptado por el motor de Neptune. Puede invocarla añadiendo un parámetro `explain` a una llamada HTTP que envíe una consulta de Gremlin.

La característica `explain` proporciona información sobre la estructura lógica de los planes de ejecución de consultas. Puede utilizar esta información para identificar posibles cuellos de botella de evaluación y ejecución y ajustar la consulta, tal y como se explica en Ajuste de las consultas de Gremlin. También puede usar sugerencias de consulta para mejorar los planes de ejecución de consultas.

> **Note**
>
> Esta característica solo está disponible a partir de Versión 1.0.1.0.200463.0 (15/10/2019).

Temas

- Descripción de cómo funcionan las consultas de Gremlin en Neptune
- Uso de la API explain de Gremlin en Neptune
- API profile de Gremlin en Neptune
- Ajuste de las consultas de Gremlin mediante explain y profile
- Compatibilidad nativa con pasos de Gremlin en Amazon Neptune

Descripción de cómo funcionan las consultas de Gremlin en Neptune

Para sacar el máximo partido de los informes `explain` y `profile` de Gremlin en Amazon Neptune, es útil tener información general sobre las consultas de Gremlin.

Temas

Instrucciones de Gremlin en Neptune

Los datos de gráficos de propiedades en Amazon Neptune se componen de instrucciones de cuatro posiciones (cuádruples). Cada una de estas instrucciones representa una unidad atómica individual de datos de gráficos de propiedades. Para obtener más información, consulte Modelo de datos de gráficos de Neptune.. Al igual que el modelo de datos del marco de descripción de recursos (RDF), estas cuatro posiciones son las siguientes:

- `subject (S)`
- `predicate (P)`
- `object (O)`
- `graph (G)`

Cada instrucción es una afirmación sobre uno o varios recursos. Por ejemplo, una instrucción puede afirmar la existencia de una relación entre dos recursos o puede asociar una propiedad (par clave/valor) a algún recurso.

Puede considerar el predicado como el verbo de la instrucción, que describe el tipo de relación o propiedad. El objeto es el objetivo de la relación o el valor de la propiedad. La posición del gráfico es opcional y se puede utilizar de muchas maneras diferentes. En el caso de los datos de gráficos-propiedades (PG) de Neptune, o bien no se usan (gráfico nulo) o se utilizan para representar el identificador de un borde. Un conjunto de instrucciones con identificadores de recursos compartidos crea un gráfico.

Existen tres clases de instrucciones en el modelo de datos de gráficos de propiedades de Neptune:

Temas

- Instrucciones de etiqueta de vértice de Gremlin
- Instrucciones de borde de Gremlin
- Instrucciones de propiedades de Gremlin

Instrucciones de etiqueta de vértice de Gremlin

Las instrucciones de etiqueta de vértice en Neptune sirven para dos fines:

- Realizan un seguimiento de las etiquetas de un vértice.
- La presencia de al menos una de estas instrucciones es lo que implica la existencia de un vértice determinado en el gráfico.

El asunto de estas instrucciones es un identificador de vértice y el objeto es una etiqueta que especifica el usuario. Puede utilizar un predicado fijo especial para estas instrucciones, que se muestra como `<~label>`, y un identificador de gráfico predeterminado (el gráfico nulo), que se muestra como `<~>`.

Por ejemplo, fíjese en el siguiente recorrido de `addV`.

```
g.addV("Person").property(id, "v1")
```

Este recorrido hace que se añada la siguiente instrucción al gráfico.

```
StatementEvent[Added(<v1> <~label> <Person> <~>) .]
```

Instrucciones de borde de Gremlin

Una instrucción de borde de Gremlin es lo que implica la existencia de un límite entre dos vértices en un gráfico en Neptune. El asunto (S) de una instrucción de borde es el vértice `from` de origen. El predicado (P) es una etiqueta de borde proporcionada por el usuario. El objeto (O) es el vértice `to` de destino. El gráfico (G) es un identificador de borde proporcionado por el usuario.

Por ejemplo, fíjese en el siguiente recorrido de `addE`.

```
g.addE("knows").from(V("v1")).to(V("v2")).property(id, "e1")
```

El recorrido hace que se añada la siguiente instrucción al gráfico.

```
StatementEvent[Added(<v1> <knows> <v2> <e1>) .]
```

Instrucciones de propiedades de Gremlin

Una instrucción de propiedad de Gremlin en Neptune confirma un valor de propiedad individual para un vértice o borde. El asunto es un vértice o un identificador de borde proporcionado por el usuario. El predicado es el nombre de la propiedad (clave) y el objeto es el valor de la propiedad individual. El gráfico (G) es de nuevo el identificador de gráfico predeterminado, el gráfico nulo, que se muestra como <~>.

Considere el siguiente ejemplo.

```
g.V("v1").property("name", "John")
```

Esta instrucción da como resultado lo siguiente.

```
StatementEvent[Added(<v1> <name> "John" <~>) .]
```

Las instrucciones de propiedad difieren de otras en que su objeto es un valor primitivo (`string`, `date`, `byte`, `short`, `int`, `long`, `float` o `double`). Su objeto no es un identificador de recursos que pueda utilizarse como sujeto de otra afirmación.

Para varias propiedades, cada valor de propiedad individual del conjunto recibe su propia instrucción.

```
g.V("v1").property(set, "phone", "956-424-2563").property(set, "phone", "956-354-3692
 (tel:9563543692)")
```

Se obtiene el siguiente resultado:

```
StatementEvent[Added(<v1> <phone> "956-424-2563" <~>) .]
StatementEvent[Added(<v1> <phone> "956-354-3692" <~>) .]
```

Cómo procesa Neptune las consultas de Gremlin mediante índices de instrucción

Para acceder a las instrucciones en Amazon Neptune, se utilizan tres índices de instrucción, tal y como se detalla en Cómo se indexan las instrucciones en Neptune. Neptune extrae un patrón de instrucciones de una consulta de Gremlin en el que se conocen algunas posiciones y el resto se deja para que se descubran mediante una búsqueda de índices.

Neptune asume que el tamaño del esquema de gráfico de propiedades no es grande. Esto significa que el número de etiquetas de borde y nombres de propiedades distintos es bastante bajo, lo que genera un número total bajo de predicados distintos. Neptune realiza un seguimiento de los predicados distintos en un índice independiente. Utiliza esta caché de predicados para realizar un análisis de unión de `{ all P x POGS }` en lugar de utilizar un índice de OSGP. Evitar la necesidad de un índice de OSGP de recorrido inverso ahorra espacio de almacenamiento y rendimiento de carga.

La API de Explain/Profile de Gremlin en Neptune le permite obtener el recuento de predicados en el gráfico. A continuación, puede determinar si la aplicación invalida la suposición de Neptune de que el esquema de gráficos de propiedades es pequeño.

Los siguientes ejemplos ayudan a demostrar cómo utiliza Neptune los índices para procesar consultas de Gremlin.

Pregunta: ¿Cuáles son las etiquetas del vértice **v1**?

```
Gremlin code:      g.V('v1').label()
Pattern:           (<v1>, <~label>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<~label>:*
```

Pregunta: ¿Cuáles son los bordes de tipo "knows" (conoce) del vértice **v1**?

```
Gremlin code:      g.V('v1').out('knows')
Pattern:           (<v1>, <knows>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<knows>:*
```

Pregunta: ¿Qué vértices tienen una etiqueta de vértice de **Person**?

```
Gremlin code:      g.V().hasLabel('Person')
Pattern:           (?, <~label>, <Person>, <~>)
Known positions:   POG
Lookup positions:  S
Index:             POGS
Key range:         <~label>:<Person>:<~>:*
```

Pregunta: ¿Cuáles son los vértices de origen/destino de un borde determinado **e1**?

```
Gremlin code:      g.E('e1').bothV()
Pattern:           (?, ?, ?, <e1>)
Known positions:   G
Lookup positions:  SPO
Index:             GPSO
Key range:         <e1>:*
```

Un índice de instrucción que no tiene Neptune es un índice OSGP de recorrido inverso. Este índice podría utilizarse para reunir todos los bordes entrantes de todas las etiquetas de borde, como en el siguiente ejemplo.

Pregunta: ¿Cuáles son los vértices **v1** adyacentes entrantes?

```
Gremlin code:      g.V('v1').in()
Pattern:           (?, ?, <v1>, ?)
Known positions:   O
Lookup positions:  SPG
Index:             OSGP  // <-- Index does not exist
```

Cómo se procesan las consultas de Gremlin en Neptune

En Amazon Neptune, los recorridos más complejos se pueden representar mediante una serie de patrones que crean una relación basada en la definición de variables con nombre que se pueden compartir entre patrones para crear uniones. Esto se muestra en el siguiente ejemplo.

Pregunta: ¿Cuál es el vecindario de dos saltos del vértice **v1**?

```
Gremlin code:      g.V('v1').out('knows').out('knows').path()
Pattern:           (?1=<v1>, <knows>, ?2, ?) X Pattern(?2, <knows>, ?3, ?)

The pattern produces a three-column relation (?1, ?2, ?3) like this:
                   ?1     ?2     ?3
                   ================
                   v1     v2     v3
                   v1     v2     v4
                   v1     v5     v6
```

Cuando se comparte la variable ?2 entre los dos patrones (en la posición O en el primer patrón y en la posición S del segundo patrón), se crea una combinación entre los vecinos del primer salto

y los vecinos del segundo salto. Cada solución de Neptune tiene enlaces para las tres variables nombradas, que se pueden usar para recrear un TinkerPopTraverser (incluida la información de la ruta).

El primer paso en el procesamiento de consultas de Gremlin consiste en analizar la consulta para convertirla en un objeto TinkerPop transversal, compuesto por una serie de pasos. TinkerPop Estos pasos, que forman parte del TinkerPop proyecto Apache de código abierto, son los operadores lógicos y físicos que componen un recorrido de Gremlin en la implementación de referencia. Ambos se utilizan para representar el modelo de la consulta. Son operadores ejecutables que pueden producir soluciones de acuerdo con la semántica del operador que representan. Por ejemplo, `.V()` está representado y ejecutado a la vez por. TinkerPop GraphStep

Como estos off-the-shelf TinkerPop pasos son ejecutables, un TinkerPop Traversal de este tipo puede ejecutar cualquier consulta de Gremlin y producir la respuesta correcta. Sin embargo, cuando se ejecuta con un gráfico grande, TinkerPop los pasos a veces pueden ser muy ineficientes y lentos. En lugar de utilizarlos, Neptune intenta convertir el recorrido en una forma declarativa compuesta por grupos de patrones, tal y como se ha descrito anteriormente.

Neptune no admite actualmente todos los operadores de Gremlin (pasos) en su motor de consultas nativo. Por lo tanto, intenta contraer tantos pasos como sea posible en un único `NeptuneGraphQueryStep`, que contiene el plan de consulta lógica declarativa para todos los pasos que se han convertido. Idealmente, se convierten todos los pasos. Pero cuando se encuentra un paso que no se puede convertir, Neptune interrumpe la ejecución nativa y aplaza toda la ejecución de consultas desde ese punto hasta los pasos. TinkerPop No intenta entrelazar y salir de la ejecución nativa.

Después de convertir los pasos en un plan de consulta lógica, Neptune ejecuta una serie de optimizadores de consultas que reescriben el plan de consulta en función del análisis estático y las cardinalidades estimadas. Esos optimizadores realizan tareas como reordenar operadores en función del recuento de rangos, eliminar operadores innecesarios o redundantes, reorganizar filtros, insertar operadores en diferentes grupos, etc.

Después de generar un plan de consulta optimizado, Neptune crea una canalización de operadores físicos que realizan el trabajo de ejecución de la consulta. Esto incluye la lectura de datos de los índices de instrucción, la realización de combinaciones de varios tipos, el filtrado, la ordenación, etc. La canalización produce un flujo de soluciones que luego se convierte de nuevo en un flujo de objetos de TinkerPop Traverser.

Serialización de los resultados de las consultas

Actualmente, Amazon Neptune utiliza los serializadores de mensajes de TinkerPop respuesta para convertir los resultados de las consultas (TinkerPop Traversers) en datos serializados que se enviarán por cable al cliente. Estos formatos de serialización suelen ser bastante específicos.

Por ejemplo, para serializar el resultado de una consulta de vértice como `g.V().limit(1)`, el motor de consultas de Neptune debe realizar una única búsqueda para producir el resultado de la consulta. Sin embargo, el serializador de `GraphSON` realizaría un gran número de búsquedas adicionales para empaquetar el vértice en el formato de serialización. Tendría que realizar una búsqueda para obtener la etiqueta, una para obtener las claves de propiedad y una búsqueda por clave de propiedad para el vértice para obtener todos los valores de cada clave.

Algunos de los formatos de serialización son más eficientes, pero todos requieren búsquedas adicionales. Además, los TinkerPop serializadores no intentan evitar la duplicación de búsquedas, lo que suele provocar que muchas búsquedas se repitan innecesariamente.

Esto hace que sea muy importante escribir sus consultas para que soliciten específicamente solo la información que necesitan. Por ejemplo, `g.V().limit(1).id()` devolvería solo el ID de vértice y eliminaría todas las búsquedas de serializador adicionales. API `profile` de Gremlin en Neptune le permite ver cuántas llamadas de búsqueda se realizan durante la ejecución de consultas y durante la serialización.

Uso de la API **`explain`** de Gremlin en Neptune

La API de `explain` de Gremlin de Amazon Neptune devuelve el plan de consulta que se ejecutaría si se realizara una consulta especificada. Puesto que la API no ejecuta realmente la consulta, el plan se devuelve casi instantáneamente.

Se diferencia del paso TinkerPop .explain () para poder reportar información específica del motor de Neptune.

Información contenida en un informe **`explain`** de Gremlin

Un informe `explain` contiene la siguiente información:

- La cadena de consulta según se solicita.
- El recorrido original. Este es el objeto TinkerPop Traversal que se produce al analizar la cadena de consulta en pasos. TinkerPop Es equivalente a la consulta original producida al ejecutar `.explain()` la consulta contra el. TinkerPop TinkerGraph

- El recorrido convertido. Este es el recorrido de Neptuno producido al convertir el TinkerPop recorrido en la representación del plan de consulta lógico de Neptuno. En muchos casos, todo el TinkerPop recorrido se convierte en dos pasos de Neptuno: uno que ejecuta toda la consulta `NeptuneGraphQueryStep` () y otro que convierte la salida del motor de consultas de Neptuno en Traversers (). TinkerPop `NeptuneTraverserConverterStep`
- El recorrido optimizado. Esta es la versión optimizada del plan de consultas de Neptune después de que se haya ejecutado a través de una serie de optimizadores estáticos de reducción de trabajo que reescriben la consulta en función del análisis estático y las cardinalidades estimadas. Esos optimizadores realizan tareas como reordenar operadores en función del recuento de rangos, eliminar operadores innecesarios o redundantes, reorganizar filtros, insertar operadores en diferentes grupos, etc.
- El recuento de predicados. Debido a la estrategia de indexación de Neptune descrita anteriormente, disponer de un gran número de predicados diferentes puede provocar problemas de rendimiento. Esto es especialmente cierto para las consultas que utilizan operadores de recorrido inverso sin etiqueta de borde (`.in` o `.both`). Si se utilizan estos operadores y el recuento de predicados es lo suficientemente alto, el informe `explain` muestra un mensaje de advertencia.
- Información de DFE Cuando el motor alternativo de DFE está habilitado, es posible que los siguientes componentes de recorrido aparezcan en el recorrido optimizado:
  - **`DFEStep`**: un paso de DFE optimizado para Neptune en el recorrido que contiene un `DFENode`secundario. `DFEStep` representa la parte del plan de consultas que se ejecuta en el motor DFE.
  - **`DFENode`**: contiene la representación intermedia como uno o más `DFEJoinGroupNodes` secundarios.
  - **`DFEJoinGroupNode`**: representa una unión de uno o más elementos `DFENode` o `DFEJoinGroupNode`.
  - **`NeptuneInterleavingStep`**: un paso de DFE optimizado para Neptune en el recorrido que contiene un `DFEStep`secundario.

    También contiene un elemento `stepInfo` que contiene información sobre el recorrido, como el elemento fronterizo, los elementos de la ruta utilizados, etc. Esta información se usa para procesar el `DFEStep` secundario.

  Una forma sencilla de averiguar si el DFE está evaluando su consulta consiste en comprobar si el resultado de `explain` contiene un `DFEStep`. Cualquier parte del recorrido que no forme parte del recorrido no será ejecutada por el DFE y `DFEStep` será ejecutada por el motor. TinkerPop

Consulte Ejemplo con DFE habilitado para ver un informe de ejemplo.

Sintaxis de **`explain`** de Gremlin

La sintaxis de la API `explain` es la misma que la de la API HTTP para consultas, salvo que utiliza `/gremlin/explain` como punto de enlace en lugar de `/gremlin`, como en el siguiente ejemplo.

```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().limit(1)"}'
```

La consulta anterior produciría el siguiente resultado.

```
=======================================================
               Neptune Gremlin Explain
=======================================================

Query String
============
g.V().limit(1)

Original Traversal
==================
[GraphStep(vertex,[]), RangeGlobalStep(0,1)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
```

```
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, finishers=[limit(1)], annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Pasos no convertidos TinkerPop

Lo ideal es que todos los TinkerPop pasos de un recorrido cuenten con la cobertura nativa del operador de Neptune. Cuando este no es el caso, Neptune recurre a la ejecución TinkerPop escalonada para cubrir las brechas en la cobertura de sus operadores. Si un recorrido utiliza un paso para el que Neptune todavía no tiene cobertura nativa, el informe `explain` muestra una advertencia que indica dónde se produjo la brecha.

Cuando se encuentra un paso sin un operador nativo de Neptuno correspondiente, todo el recorrido desde ese punto en adelante se ejecuta mediante TinkerPop pasos, incluso si los pasos posteriores tienen operadores nativos de Neptuno.

La excepción a esto se produce cuando se invoca la búsqueda de texto completo de Neptune. NeptuneSearchStep Implementa los pasos sin equivalentes nativos como pasos de búsqueda de texto completo.

Ejemplo de resultado de **`explain`** donde todos los pasos de una consulta tienen equivalentes nativos

A continuación se muestra un ejemplo de informe `explain` para una consulta en la que todos los pasos tienen equivalentes nativos:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============
g.V().out()
```

```
Original Traversal
==================
[GraphStep(vertex,[]), VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 18
```

Ejemplo en el que algunos pasos de una consulta no tienen equivalentes nativos

Neptune gestiona `GraphStep` y `VertexStep` de forma nativa, pero si introduce un `FoldStep` y `UnfoldStep`, el `explain` resultante es diferente.

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************
```

```
Query String
============
g.V().fold().unfold().out()

Original Traversal
==================
[GraphStep(vertex,[]), FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep,
    NeptuneMemoryTrackerStep
]
+ not converted into Neptune steps: [FoldStep, UnfoldStep, VertexStep(OUT,vertex)]

WARNING: >> FoldStep << is not supported natively yet
```

En este caso, `FoldStep` elimina la ejecución nativa. Sin embargo, incluso el `VertexStep` siguiente ya no se gestiona de forma nativa, ya que aparece después de los pasos `Fold/Unfold`.

Para ahorrar rendimiento y costes, es importante que intente formular los recorridos de forma que la mayor cantidad de trabajo posible se realice de forma nativa dentro del motor de consultas de Neptune, en lugar de hacerlo paso a paso. TinkerPop

Ejemplo de consulta que usa Neptune full-text-search

En la siguiente consulta se utiliza la búsqueda de texto completo de Neptune:

```
g.withSideEffect("Neptune#fts.endpoint", "some_endpoint")
  .V()
  .tail(100)
  .has("Neptune#fts mark*")
  -------
  .has("name", "Neptune#fts mark*")
  .has("Person", "name", "Neptune#fts mark*")
```

La parte `.has("name", "Neptune#fts mark*")` limita la búsqueda a los vértices con `name`, mientras que `.has("Person", "name", "Neptune#fts mark*")` limita la búsqueda a los vértices con `name` y la etiqueta `Person`. Esto da como resultado el siguiente recorrido en el informe `explain`:

```
Final Traversal
[NeptuneGraphQueryStep(Vertex) {
    JoinGroupNode {
        PatternNode[(?1, termid(1,URI), ?2, termid(0,URI)) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
    }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
}, NeptuneTraverserConverterStep, NeptuneTailGlobalStep(10),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=some_endpoint}
    }
}]
```

Ejemplo de uso de **`explain`** cuando DFE está habilitado

El siguiente es un ejemplo de un informe `explain` cuando el motor de consultas alternativo DFE está habilitado:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().as("a").out().has("name", "josh").out().in().where(eq("a"))


Original Traversal
==================
[GraphStep(vertex,[])@[a], VertexStep(OUT,vertex), HasStep([name.eq(josh)]),
 VertexStep(OUT,vertex), VertexStep(IN,vertex), WherePredicateStep(eq(a))]

Converted Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, ?2,
 <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>) . project DISTINCT[?1]
 {rangeCountEstimate=unknown}],
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!
= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: HasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneInterleavingStep {
```

```
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12]
 graphFilters=(!= <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph> . ),
 {rangeCountEstimate=unknown}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


Optimized Traversal
===================
Neptune steps:
[
    DFEStep(Vertex) {
      DFENode {
        DFEJoinGroupNode[ children={
          DFEPatternNode[(?1, ?3, ?4, ?5) . project ALL[?1, ?4] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
        }, {rangeCountEstimate=unknown}
        ]
      } [Vertex(?1):GraphStep@[a], Vertex(?4):VertexStep]
    } ,
    NeptuneTraverserConverterDFEStep
]
+ not converted into Neptune steps: NeptuneHasStep([name.eq(josh)]),
Neptune steps:
[
    NeptuneMemoryTrackerStep,
    NeptuneInterleavingStep {
```

```
      StepInfo[joinVars=[?7, ?1], frontierElement=Vertex(?7):HasStep,
 pathElements={a=(last,Vertex(?1):GraphStep@[a])}, listPathElement={}, indexTime=0ms],
      DFEStep(Vertex) {
        DFENode {
          DFEJoinGroupNode[ children={
            DFEPatternNode[(?7, ?8, ?9, ?10) . project ALL[?7, ?9] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807}]
          }, {rangeCountEstimate=unknown}
          ]
        } [Vertex(?9):VertexStep, Vertex(?12):VertexStep]
      }
    }
]
+ not converted into Neptune steps: WherePredicateStep(eq(a)),
Neptune steps:
[
    DFECleanupStep
]


WARNING: >> [NeptuneHasStep([name.eq(josh)]), WherePredicateStep(eq(a))] << (or one of
 the children for each step) is not supported natively yet

Predicates
==========
# of predicates: 8
```

Consulte Información en `explain` para obtener una descripción de las secciones específicas de DFE en el informe.

API **`profile`** de Gremlin en Neptune

La API `profile` de Gremlin de Neptune ejecuta un recorrido de Gremlin especificado, recopila varias métricas sobre la ejecución y produce un informe de perfil como salida.

> **Note**
>
> Esta característica solo está disponible a partir de Versión 1.0.1.0.200463.0 (15/10/2019).

Se diferencia del paso TinkerPop .profile () para poder reportar información específica del motor de Neptune.

El informe de perfil incluye la siguiente información sobre el plan de consulta:

- La canalización del operador físico.
- Las operaciones de índice para la ejecución y serialización de consultas.
- El tamaño del resultado.

La API de `profile` utiliza una versión ampliada de la sintaxis de la API HTTP para las consultas, con `/gremlin/profile` como punto de enlace en lugar de `/gremlin`.

Parámetros específicos de **`profile`** de Gremlin en Neptune

- profile.results: `boolean`, valores permitidos: `TRUE` y `FALSE`, valor predeterminado: `TRUE`.

  Si es true, los resultados de la consulta se recopilan y se muestran como parte del informe de `profile`. Si es false, solo se muestra el recuento de resultados.
- profile.chop: `int`, valor predeterminado: 250.

  Si es distinto de cero, hace que la cadena de resultados se trunque con ese número de caracteres. Esto no impide que se capturen todos los resultados. Simplemente limita el tamaño de la cadena en el informe de perfil. Si se establece en cero, la cadena contiene todos los resultados.
- profile.serializer: `string`, valor predeterminado: `<null>`.

  Si no es nulo, los resultados recopilados se devuelven en un mensaje de respuesta serializado en el formato especificado por este parámetro. El número de operaciones de índice necesarias para producir ese mensaje de respuesta se notifica junto con el tamaño en bytes que se va a enviar al cliente.

  Los valores permitidos son `<null>` o cualquiera de los valores de enumeración de los «serializadores» válidos del tipo MIME o TinkerPop del controlador.

```
"application/json" or "GRAPHSON"
"application/vnd.gremlin-v1.0+json" or "GRAPHSON_V1"
"application/vnd.gremlin-v1.0+json;types=false" or "GRAPHSON_V1_UNTYPED"
"application/vnd.gremlin-v2.0+json" or "GRAPHSON_V2"
"application/vnd.gremlin-v2.0+json;types=false" or "GRAPHSON_V2_UNTYPED"
```

```
"application/vnd.gremlin-v3.0+json" or "GRAPHSON_V3"
"application/vnd.gremlin-v3.0+json;types=false" or "GRAPHSON_V3_UNTYPED"
"application/vnd.graphbinary-v1.0" or "GRAPHBINARY_V1"
```

- profile.indexOps: `boolean`, valores permitidos: `TRUE` y `FALSE`, valor predeterminado: `FALSE`.

  Si es verdadero, muestra un informe detallado de todas las operaciones de índice que se realizaron durante la ejecución y serialización de consultas. Advertencia: este informe puede ser detallado.

Salida de ejemplo de **`profile`** de Gremlin en Neptune

A continuación, se muestra una consulta `profile` de ejemplo.

```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile \
     -d '{"gremlin":"g.V().hasLabel(\"airport\")
                         .has(\"code\", \"AUS\")
                         .emit()
                         .repeat(in().simplePath())
                         .times(2)
                         .limit(100)",
          "profile.serializer":"application/vnd.gremlin-v3.0+gryo"}'
```

Esta consulta genera el siguiente informe `profile` cuando se ejecuta en el gráfico de muestra de rutas aéreas de la entrada de blog, Let Me Graph That For You - Part 1 - Air Routes.

```
*******************************************************
                Neptune Gremlin Profile
*******************************************************

Query String
==================
g.V().hasLabel("airport").has("code",
 "AUS").emit().repeat(in().simplePath()).times(2).limit(100)

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([~label.eq(airport), code.eq(AUS)]),
 RepeatStep(emit(true),[VertexStep(IN,vertex), PathFilterStep(simple),
 RepeatEndStep],until(loops(2))), RangeGlobalStep(0,100)]
```

```
Optimized Traversal
==================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true, joinTime=3, actualTotalOutput=1}
            PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project ask .],
 {estimatedCardinality=3374, indexTime=29, hashJoin=true, joinTime=0,
 actualTotalOutput=61}
            RepeatNode {
                Repeat {
                    PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 SimplePathFilter(?1, ?3)) .], {hashJoin=true, estimatedCardinality=50148, indexTime=0,
 joinTime=3}
                }
                Emit {
                    Filter(true)
                }
                LoopsCondition {
                    LoopsFilter([?1, ?3],eq(2))
                }
            }, annotations={repeatMode=BFS, emitFirst=true, untilFirst=false, leftVar=?
1, rightVar=?3}
        }, finishers=[limit(100)], annotations={path=[Vertex(?1):GraphStep,
 Repeat[Vertex(?3):VertexStep]], joinStats=true, optimizationTime=495, maxVarId=7,
 executionTime=323}
    },
    NeptuneTraverserConverterStep
]

Physical Pipeline
=================
NeptuneGraphQueryStep
    |-- StartOp
    |-- JoinGroupOp
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <code>, "AUS", ?) . project ?1 .],
 {estimatedCardinality=1, indexTime=84, hashJoin=true})
        |-- SpoolerOp(100)
        |-- DynamicJoinOp(PatternNode[(?1, <~label>, ?2=<airport>, <~>) . project
 ask .], {estimatedCardinality=3374, indexTime=29, hashJoin=true})
        |-- RepeatOp
```

```
            |-- <upstream input> (Iteration 0) [visited=1, output=1 (until=0, emit=1),
 next=1]
            |-- BindingSetQueue (Iteration 1) [visited=61, output=61 (until=0,
 emit=61), next=61]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
            |-- BindingSetQueue (Iteration 2) [visited=38, output=38 (until=38,
 emit=0), next=0]
                |-- SpoolerOp(100)
                |-- DynamicJoinOp(PatternNode[(?3, ?5, ?1, ?6) . project ?
1,?3 . IsEdgeIdFilter(?6) . SimplePathFilter(?1, ?3)) .], {hashJoin=true,
 estimatedCardinality=50148, indexTime=0})
        |-- LimitOp(100)

Runtime (ms)
============
Query Execution:  392.686
Serialization:   2636.380

Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                        100         100
      314.162    82.78
NeptuneTraverserConverterStep                                        100         100
       65.333    17.22
                                           >TOTAL                     -           -
      379.495        -

Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
------------------------------------------------------
        0        1        1        0        1        1
        1       61       61        0       61       61
        2       38       38       38        0        0
------------------------------------------------------
              100      100       38       62       62

Predicates
```

```
==========
# of predicates: 16

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance

Results
=======
Count: 100
Output: [v[3], v[3600], v[3614], v[4], v[5], v[6], v[7], v[8], v[9], v[10], v[11],
 v[12], v[47], v[49], v[136], v[13], v[15], v[16], v[17], v[18], v[389], v[20], v[21],
 v[22], v[23], v[24], v[25], v[26], v[27], v[28], v[416], v[29], v[30], v[430], v[31],
 v[9...
Response serializer: GRYO_V3D0
Response size (bytes): 23566

Index Operations
================
Query execution:
    # of statement index ops: 3
    # of unique statement index ops: 3
    Duplication ratio: 1.0
    # of terms materialized: 0
Serialization:
    # of statement index ops: 200
    # of unique statement index ops: 140
    Duplication ratio: 1.43
    # of terms materialized: 393
```

Además de los planes de consulta devueltos por una llamada a `explain` de Neptune, los resultados de `profile` incluyen estadísticas de tiempo de ejecución sobre la ejecución de consultas. Cada operación Join se etiqueta con el tiempo que se tarda en realizar su combinación, así como el número real de soluciones que ha pasado a través de ella.

La salida `profile` incluye el tiempo necesario durante la fase de ejecución de consultas principales, así como la fase de serialización si se especificó la opción `profile.serializer`.

El desglose de las operaciones de índice realizadas durante cada fase también se incluye en la parte inferior de la salida `profile`.

Tenga en cuenta que las ejecuciones consecutivas de la misma consulta pueden mostrar resultados diferentes en términos de tiempo de ejecución e índice debido al almacenamiento en caché.

Para las consultas que utilizan el paso `repeat()`, hay disponible un desglose de la frontera en cada iteración si el paso `repeat()` se bajó como parte de `NeptuneGraphQueryStep`.

Diferencias en los informes de **`profile`** cuando DFE está habilitado

Cuando el motor de consultas alternativo DFE de Neptune está habilitado, la salida de `profile` es ligeramente diferente:

Recorrido optimizado: esta sección es similar a la de la salida de `explain`, pero contiene información adicional. Esto incluye el tipo de operadores de DFE que se han tenido en cuenta en la planificación y las estimaciones de los costos correspondientes en el peor y el mejor de los casos.

Canalización física: en esta sección se muestran los operadores que se utilizan para ejecutar la consulta. Los elementos `DFESubQuery` resumen el plan físico que utiliza DFE para ejecutar la parte del plan de la que es responsable. Los elementos `DFESubQuery` se detallan en la siguiente sección, donde se enumeran las estadísticas de DFE.

QueryEngine Estadísticas del DFE: esta sección solo aparece cuando el DFE ejecuta al menos una parte de la consulta. En ella se describen varias estadísticas de tiempo de ejecución que son específicas de DFE y contiene un desglose detallado del tiempo empleado en las distintas partes de la ejecución de la consulta, por `DFESubQuery`.

En esta sección, las subconsultas anidadas en distintos elementos `DFESubQuery` se aplanan y los identificadores únicos se marcan con un encabezado que comienza con `subQuery=`.

Métricas transversales: en esta sección se muestran las métricas transversales a nivel de pasos y, cuando el motor DFE ejecuta toda la consulta o parte de ella, se muestran métricas para `DFEStep` y/o `NeptuneInterleavingStep`. Consulte Ajuste de las consultas de Gremlin mediante `explain` y `profile`.

> **Note**
>
> El DFE es una característica experimental publicada en el modo de laboratorio, por lo que el formato exacto de la salida de `profile` aún está sujeto a cambios.

Salida de **`profile`** de ejemplo cuando el motor de flujo de datos de Neptune (DFE) está habilitado

Cuando se utiliza el motor DFE para ejecutar consultas de Gremlin, la salida de API `profile` de Gremlin tiene el formato que se muestra en el siguiente ejemplo.

Consulta:

```
curl https://localhost:8182/gremlin/profile \
  -d "{\"gremlin\": \"g.withSideEffect('Neptune#useDFE', true).V().has('code',
 'ATL').out()\"}"
```

```
********************************************************
                    Neptune Gremlin Profile
    ********************************************************

    Query String
    ==================
    g.withSideEffect('Neptune#useDFE', true).V().has('code', 'ATL').out()

    Original Traversal
    ==================
    [GraphStep(vertex,[]), HasStep([code.eq(ATL)]), VertexStep(OUT,vertex)]

    Optimized Traversal
    ===================
    Neptune steps:
    [
        DFEStep(Vertex) {
          DFENode {
            DFEJoinGroupNode[null](
              children=[
                DFEPatternNode((?1, vp://code[419430926], ?4, defaultGraph[526]) .
 project DISTINCT[?1] objectFilters=(in(ATL[452987149]) . ), {rangeCountEstimate=1},
                  opInfo=(type=PipelineJoin,
 cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.00
                    disc=(type=PipelineScan,
 cost=(exp=(in=1.00,out=1.00,io=0.00,comp=0.00,mem=34.00),wc=(in=1.00,out=1.00,io=0.00,comp=0.0
                DFEPatternNode((?1, ?5, ?6, ?7) . project ALL[?1, ?6] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=9223372036854775807})],
              opInfo=[
                OperatorInfoWithAlternative[
                  rec=(type=PipelineJoin,
 cost=(exp=(in=1.00,out=27.76,io=0.00,comp=0.00,mem=0.00),wc=(in=1.00,out=27.76,io=0.00,comp=0.
                    disc=(type=PipelineScan,
 cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
                  alt=(type=PipelineScan,
 cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
          } [Vertex(?1):GraphStep, Vertex(?6):VertexStep]
```

```
        } ,
        NeptuneTraverserConverterDFEStep,
        DFECleanupStep
    ]


    Physical Pipeline
    =================
    DFEStep
        |-- DFESubQuery1

    DFEQueryEngine Statistics
    =================
    DFESubQuery1

 ###########################################################################################
    # ID # Out #1 # Out #2 # Name                  # Arguments
                                                                          # Mode #
 Units In # Units Out # Ratio  # Time (ms) #

 ###########################################################################################
    # 0  # 1      # -      # DFESolutionInjection  # solutions=[]
                                                                       # -    # 0
      # 1         # 0.00   # 0.01      #
    #    #        #        #                      # outSchema=[]
                                                                       #      #
      #           #        #           #

 ###########################################################################################
    # 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1 # -    #
 1        # 1         # 1.00   # 0.02      #

 ###########################################################################################
    # 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2 # -    #
 1        # 242       # 242.00 # 0.02      #

 ###########################################################################################
    # 3  # 4      # -      # DFEMergeChunks        # -
                                                                       # -    # 242
      # 242       # 1.00   # 0.01      #

 ###########################################################################################
```

```
    # 4  # -      # -      # DFEDrain              # -
                                                                  # -    # 242
      # 0         # 0.00   # 0.01      #

 ##########################################################################################


    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_1

 ##########################################################################################
    # ID # Out #1 # Out #2 # Name                 # Arguments
                          # Mode # Units In # Units Out # Ratio # Time (ms) #

 ##########################################################################################
    # 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?1) with property
 'code' as ?4 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.22      #
    #    #        #        #                     # inlineFilters=[(?4 IN ["ATL"])]
                          #      #          #           #       #           #
    #    #        #        #                     # patternEstimate=1
                          #      #          #           #       #           #

 ##########################################################################################
    # 1  # 2      # -      # DFEMergeChunks       # -
                          # -    # 1        # 1         # 1.00  # 0.02      #

 ##########################################################################################
    # 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                          # -    # 2        # 1         # 0.50  # 0.09      #

 ##########################################################################################
    # 3  # 2      # -      # DFESolutionInjection # solutions=[]
                          # -    # 0        # 1         # 0.00  # 0.01      #
    #    #        #        #                     # outSchema=[]
                          #      #          #           #       #           #

 ##########################################################################################
    # 4  # -      # -      # DFEDrain             # -
                          # -    # 1        # 0         # 0.00  # 0.01      #

 ##########################################################################################
```

```
    subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/
graph#089f43e3-4d71-4259-8d19-254ff63cee04/graph_2

 ########################################################################################################
    # ID # Out #1 # Out #2 # Name                 # Arguments
 # Mode # Units In # Units Out # Ratio  # Time (ms) #

 ########################################################################################################
    # 0  # 1      # -      # DFESolutionInjection # solutions=[]
 # -    # 0        # 1         # 0.00   # 0.01      #
    #    #        #        #                      # outSchema=[?1]
 #      #          #           #        #           #

 ########################################################################################################
    # 1  # 2      # 3      # DFETee               # -
 # -    # 1        # 2         # 2.00   # 0.01      #

 ########################################################################################################
    # 2  # 4      # -      # DFEDistinctColumn    # column=?1
 # -    # 1        # 1         # 1.00   # 0.21      #
    #    #        #        #                      # ordered=false
 #      #          #           #        #           #

 ########################################################################################################
    # 3  # 5      # -      # DFEHashIndexBuild    # vars=[?1]
 # -    # 1        # 1         # 1.00   # 0.03      #

 ########################################################################################################
    # 4  # 5      # -      # DFEPipelineJoin      # pattern=Edge((?1)-[?7:?5]->(?6))
 # -    # 1        # 242       # 242.00 # 0.51      #
    #    #        #        #                      # constraints=[]
 #      #          #           #        #           #
    #    #        #        #                      # patternEstimate=9223372036854775807
 #      #          #           #        #           #

 ########################################################################################################
    # 5  # 6      # 7      # DFESync              # -
 # -    # 243      # 243       # 1.00   # 0.02      #

 ########################################################################################################
    # 6  # 8      # -      # DFEForwardValue      # -
 # -    # 1        # 1         # 1.00   # 0.01      #

 ########################################################################################################
```

```
   # 7  # 8      # -      # DFEForwardValue      # -
# -    # 242      # 242       # 1.00   # 0.02      #

##################################################################################################
   # 8  # 9      # -      # DFEHashIndexJoin     # -
# -    # 243      # 242       # 1.00   # 0.31      #

##################################################################################################
   # 9  # -      # -      # DFEDrain             # -
# -    # 242      # 0         # 0.00   # 0.01      #

##################################################################################################


   Runtime (ms)
   ============
   Query Execution: 11.744

   Traversal Metrics
   =================
   Step                                                             Count
Traversers       Time (ms)    % Dur

--------------------------------------------------------------------------------------------------
   DFEStep(Vertex)                                                    242
242          10.849    95.48
   NeptuneTraverserConverterDFEStep                                   242
242           0.514     4.52
                                              >TOTAL                    -
-          11.363        -

   Predicates
   ==========
   # of predicates: 18

   Results
   =======
   Count: 242


   Index Operations
   ================
   Query execution:
       # of statement index ops: 0
```

```
        # of terms materialized: 0
```

> **Note**
>
> Dado que el DFE es una característica experimental publicada en el modo de laboratorio, el formato exacto de la salida de `profile` está sujeto a cambios.

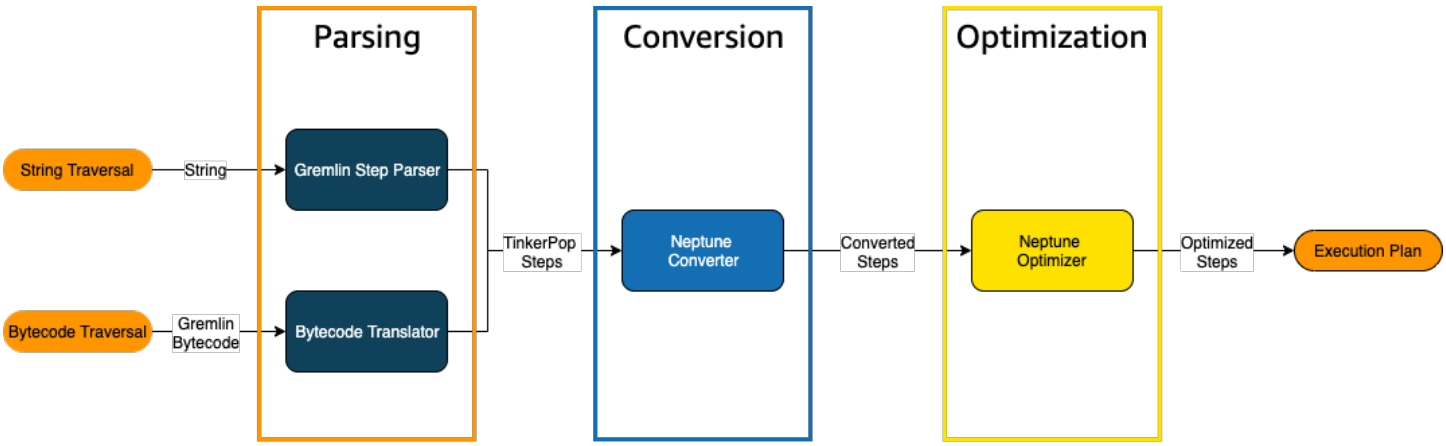
Ajuste de las consultas de Gremlin mediante **`explain`** y **`profile`**

Con frecuencia, puede ajustar sus consultas de Gremlin en Amazon Neptune para obtener un mejor rendimiento utilizando la información disponible en los informes que obtiene de las API explain y profile de Neptune. Para ello, es útil entender cómo Neptune procesa los recorridos de Gremlin.

> **Important**
>
> En la TinkerPop versión 3.4.11 se realizó un cambio que mejora la corrección del procesamiento de las consultas, pero por el momento a veces puede afectar gravemente al rendimiento de las consultas.
> Por ejemplo, una consulta de este tipo podría ejecutarse de una forma mucho más lenta:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```
>
> Los vértices situados tras el paso límite ahora se obtienen de forma no óptima debido al cambio de la versión 3.4.11. TinkerPop Para evitarlo, puede modificar la consulta añadiendo el paso barrier() en cualquier punto después de `order().by()`. Por ejemplo:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   barrier().
>   out()
> ```
>
> TinkerPop La versión 3.4.11 estaba habilitada en la versión 1.0.5.0 del motor de Neptune.

Descripción del procesamiento de recorridos de Gremlin en Neptune

Cuando se envía un recorrido de Gremlin a Neptune, hay tres procesos principales que transforman el recorrido en un plan de ejecución subyacente para que el motor lo ejecute. Estos procesos son el análisis, la conversión y la optimización,



El proceso de análisis de recorridos

El primer paso para procesar un recorrido es analizarlo en un lenguaje común. En Neptune, ese lenguaje común es el conjunto de TinkerPop pasos que forman parte de la TinkerPop API. Cada uno de estos pasos representa una unidad de cálculo dentro del recorrido.

Puede enviar un recorrido de Gremlin a Neptune como cadena o como código de bytes. El punto de conexión REST y el método `submit()` del controlador del cliente Java envían los recorridos como cadenas, como en este ejemplo:

```
client.submit("g.V()")
```

Las aplicaciones y los controladores de lenguaje que utilizan variantes del lenguaje Gremlin (GLV) envían los recorridos en código de bits.

El proceso de conversión del recorrido

El segundo paso para procesar un recorrido es convertir sus TinkerPop pasos en un conjunto de pasos de Neptune convertidos y no convertidos. La mayoría de los pasos del lenguaje de consulta Apache TinkerPop Gremlin se convierten en pasos específicos de Neptuno que están optimizados para ejecutarse en el motor Neptune subyacente. Cuando se encuentra un TinkerPop paso sin un equivalente a Neptuno en un recorrido, el motor de consulta procesa ese paso y todos los pasos subsiguientes del recorrido. TinkerPop

Para obtener más información sobre qué pasos se pueden convertir y en qué circunstancias, consulte Compatibilidad con pasos de Gremlin.

El proceso de optimización de recorridos

El último paso del procesamiento de recorridos consiste en ejecutar la serie de pasos convertidos y no convertidos a través del optimizador para tratar de determinar el mejor plan de ejecución. El resultado de esta optimización es el plan de ejecución que procesa el motor de Neptune.

Uso de la API **`explain`** en un recorrido de Gremlin de Neptune para ajustar las consultas

La API de explain de Neptune no es lo mismo que el paso `explain()` de Gremlin. Devuelve el plan de ejecución final que el motor de Neptune procesaría al ejecutar la consulta. Dado que no realiza ningún procesamiento, devuelve el mismo plan independientemente de los parámetros utilizados y su resultado no contiene estadísticas sobre la ejecución real.

Considere el siguiente recorrido simple que encuentra todos los vértices de los aeropuertos de Anchorage:

```
g.V().has('code','ANC')
```

Hay dos formas de ejecutar este recorrido a través de la API `explain` de Neptune. La primera es hacer una llamada REST al punto de conexión de explain de la siguiente manera:

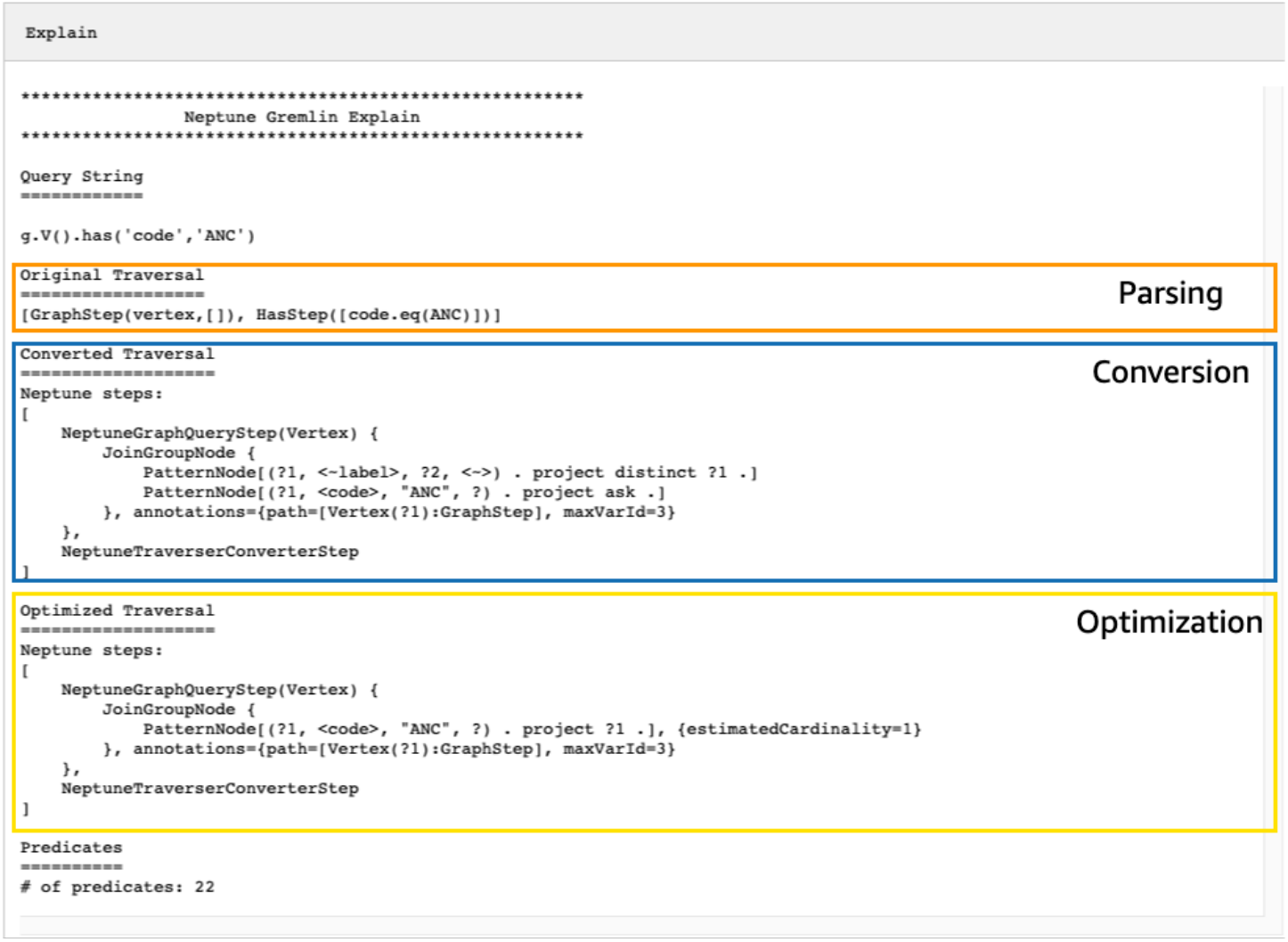
```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

La segunda forma es usar la magia de celda %%gremlin del entorno de trabajo de Neptune con el parámetro `explain`. Esto pasa el recorrido contenido en el cuerpo de la celda a la API `explain` de Neptune y, a continuación, muestra el resultado al ejecutar la celda:

```
%%gremlin explain

g.V().has('code','ANC')
```

El resultado de la API `explain` resultante describe el plan de ejecución de Neptune para el recorrido. Como puede ver en la imagen siguiente, el plan incluye cada uno de los tres pasos de la canalización de procesamiento:



Ajuste de un recorrido observando los pasos que no se convierten

Una de las primeras cosas que hay que buscar en el resultado de la API `explain` de Neptune son los pasos de Gremlin que no se convierten en pasos nativos de Neptune. En un plan de consulta, cuando se encuentra un paso que no se puede convertir en un paso nativo de Neptune, el servidor de Gremlin procesa este paso y todos los pasos subsiguientes del plan.

En el ejemplo anterior, se convirtieron todos los pasos del recorrido. Examinemos el resultado de la API de `explain` para este recorrido:

```
g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not
 an airport'))
```

Como puede ver en la imagen de abajo, Neptune no pudo convertir el paso `choose()`:

```
Explain
```

```
****************************************************
                Neptune Gremlin Explain
****************************************************

Query String
============

g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not an airport'))

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(OUT,vertex), ChooseStep([HasStep([~label.eq(airport)]), HasNex

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([HasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesSte

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1}
            PatternNode[(?1, ?5, ?3, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .], {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep], maxVarId=7}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [Proper

WARNING: >> ChooseStep([NeptuneHasStep([~label.eq(airport)]), HasNextStep],[[(eq(true)), [PropertiesStep([code],value), E

Predicates
==========
# of predicates: 26
```

Hay varias cosas que puede hacer para ajustar el rendimiento del recorrido. La primera sería reescribirlo de tal manera que se elimine el paso que no se pudo convertir. Otra sería mover el paso al final del recorrido para que todos los demás pasos se puedan convertir en pasos nativos.

No siempre es necesario ajustar un plan de consultas con pasos que no estén convertidos. Si los pasos que no se pueden convertir se encuentran al final del recorrido y están relacionados con el formato de la salida y no con la forma en que se recorre el gráfico, es posible que no afecten mucho al rendimiento.

Otra cosa que hay que tener en cuenta al examinar los resultados de la API de `explain` de Neptune son los pasos que no utilizan índices. En el siguiente recorrido, se muestran todos los aeropuertos con vuelos que aterrizan en Anchorage:

```
g.V().has('code','ANC').in().values('code')
```

La salida de la API de explain para este recorrido es:

```
*******************************************************
                Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC').in().values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
```

```
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .],
 {estimatedCardinality=INFINITY}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
       }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26

WARNING: reverse traversal with no edge label(s) - .in() / .both() may impact query
 performance
```

El mensaje `WARNING` en la parte inferior de la salida se produce porque el paso `in()` del recorrido no se puede gestionar con uno de los 3 índices que mantiene Neptune (consulte Cómo se indexan las instrucciones en Neptune y Instrucciones de Gremlin en Neptune). Como el paso `in()` no contiene ningún filtro de bordes, no se puede resolver mediante el índice `SPOG`, `POGS` o `GPSO`. En cambio, Neptune debe realizar un escaneo de unión para encontrar los vértices solicitados, lo cual es mucho menos eficiente.

Hay dos formas de ajustar el recorrido en esta situación. La primera consiste en añadir uno o más criterios de filtrado al paso `in()` para poder utilizar una búsqueda indexada para resolver la consulta. Para el ejemplo anterior, esto podría ser:

```
g.V().has('code','ANC').in('route').values('code')
```

La salida de la API `explain` de Neptune para el recorrido revisado ya no contiene el mensaje `WARNING`:

```
=======================================================
                Neptune Gremlin Explain
```

```
**********************************************************

Query String
============

g.V().has('code','ANC').in('route').values('code')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)]), VertexStep(IN,[route],vertex),
 PropertiesStep([code],value)]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
            PatternNode[(?3, ?5, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?6) .
 ContainsFilter(?5 in (<route>)) .]
            PatternNode[(?3, <~label>, ?4, <~>) . project ask .]
            PatternNode[(?3, ?7, ?8, <~>) . project ?3,?8 . ContainsFilter(?7 in
 (<code>)) .]
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(PropertyValue) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .],
 {estimatedCardinality=1}
            PatternNode[(?3, ?5=<route>, ?1, ?6) . project ?1,?3 . IsEdgeIdFilter(?
6) .], {estimatedCardinality=32042}
            PatternNode[(?3, ?7=<code>, ?8, <~>) . project ?3,?8 .],
 {estimatedCardinality=7564}
```

```
        }, annotations={path=[Vertex(?1):GraphStep, Vertex(?3):VertexStep,
 PropertyValue(?8):PropertiesStep], maxVarId=9}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 26
```

Otra opción si ejecuta muchos recorridos de este tipo es ejecutarlos en un clúster de base de datos de Neptune que tenga habilitado el índice OSGP opcional (consulte Habilitación de un índice OSGP). Habilitar un índice OSGP tiene sus inconvenientes:

- Debe habilitarse en un clúster de base de datos antes de cargar cualquier dato.
- Las tasas de inserción de vértices y bordes pueden disminuir hasta un 23 %.
- El uso del almacenamiento aumentará en torno a un 20 %.
- Es posible que en las consultas de lectura que dispersan las solicitudes en todos los índices aumente la latencia.

Tener un índice OSGP tiene mucho sentido en un conjunto restringido de patrones de consulta, pero a menos que los utilice con frecuencia, suele ser preferible asegurarse de que los recorridos que escribe se puedan resolver utilizando los tres índices principales.

Uso de una gran cantidad de predicados

Neptune trata cada etiqueta de borde y cada nombre de propiedad de vértice o borde distinto del gráfico como un predicado y está diseñado de forma predeterminada para funcionar con un número relativamente bajo de predicados distintos. Cuando hay más de unos pocos miles de predicados en los datos de gráficos, el rendimiento podría degradarse.

La salida de `explain` de Neptune le avisará si es así:

```
Predicates
==========
# of predicates: 9549
WARNING: high predicate count (# of distinct property names and edge labels)
```

Si no es conveniente reelaborar el modelo de datos para reducir el número de etiquetas y propiedades y, por lo tanto, el número de predicados, la mejor manera de ajustar los recorridos es

ejecutarlos en un clúster de base de datos que tenga el índice OSGP habilitado, como se ha explicado anteriormente.

Uso de la API de **`profile`** de Gremlin de Neptune para ajustar los recorridos

La API `profile` de Neptune es muy diferente del paso `profile()` de Gremlin. Al igual que la API de `explain`, su salida incluye el plan de consultas que utiliza el motor de Neptune al ejecutar el recorrido. Además, la salida de `profile` incluye estadísticas de ejecución reales del recorrido, teniendo en cuenta cómo están configurados sus parámetros.

De nuevo, tomemos como ejemplo el recorrido simple que encuentra todos los vértices del aeropuerto de Anchorage:

```
g.V().has('code','ANC')
```

Al igual que con la API de `explain`, puede invocar a la API de `profile` mediante una llamada REST:

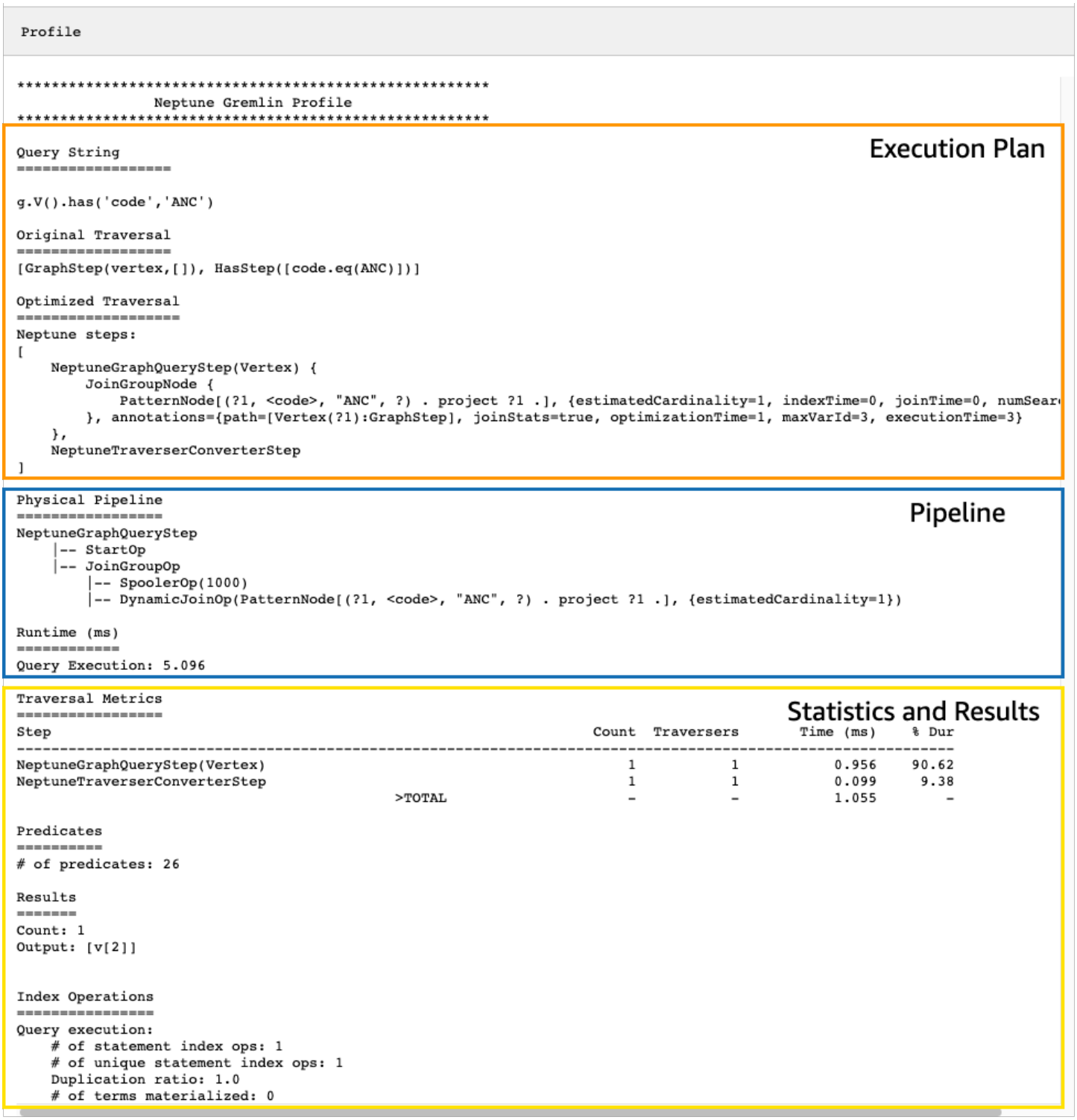
```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

También se puede usar la magia de celda %%gremlin del entorno de trabajo de Neptune con el parámetro `profile`. Esto pasa el recorrido contenido en el cuerpo de la celda a la API de `profile` de Neptune y, a continuación, muestra el resultado al ejecutar la celda:

```
%%gremlin profile

g.V().has('code','ANC')
```

La salida de la API de `profile` resultante contiene tanto el plan de ejecución de Neptune para el recorrido como las estadísticas sobre la ejecución del plan, como puede ver en esta imagen:



En la salida de `profile`, la sección del plan de ejecución solo contiene el plan de ejecución final para el recorrido, no los pasos intermedios. La sección de canalización contiene las operaciones físicas de la canalización que se realizaron, así como el tiempo real (en milisegundos) que ha tardado la ejecución del recorrido. La métrica del tiempo de ejecución es extremadamente útil

para comparar los tiempos que tardan dos versiones diferentes de un recorrido a medida que se optimizan.

> **Note**
>
> El tiempo de ejecución inicial de un recorrido suele ser mayor que el de los tiempos de ejecución posteriores, ya que el primero hace que los datos relevantes se almacenen en caché.

La tercera sección de la salida de `profile` contiene las estadísticas de ejecución y los resultados del recorrido. Para ver cómo esta información puede resultar útil para ajustar un recorrido, considere el siguiente recorrido, en el que se encuentran todos los aeropuertos cuyo nombre comienza por "Anchora" y todos los aeropuertos a los que se puede llegar en dos saltos desde esos aeropuertos, y se devuelven los códigos del aeropuerto de vuelta, las rutas de los vuelos y las distancias:

```
%%gremlin profile

g.withSideEffect("Neptune#fts.endpoint", "{your-OpenSearch-endpoint-URL").
    V().has("city", "Neptune#fts Anchora~").
    repeat(outE('route').inV().simplePath()).times(2).
    project('Destination', 'Route').
        by('code').
        by(path().by('code').by('dist'))
```

Métricas de recorrido en la salida de la API de **`profile`** de Neptune

El primer conjunto de métricas que está disponible en todas las salidas de `profile` es el de las métricas de recorrido. Son similares a las métricas de pasos de `profile()` de Gremlin, con algunas diferencias:

```
Traversal Metrics
=================
Step                                                               Count  Traversers
    Time (ms)    % Dur
-------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                       3856        3856
       91.701     9.09
NeptuneTraverserConverterStep                                       3856        3856
       38.787     3.84
```

```
ProjectStep([Destination, Route],[value(code), ...                    3856        3856
      878.786    87.07
  PathStep([value(code), value(dist)])                                3856        3856
      601.359
                                                >TOTAL                   -           -
     1009.274        -
```

La primera columna de la tabla de métricas de recorrido muestra los pasos que ha ejecutado el recorrido. Los dos primeros pasos son generalmente los pasos específicos de Neptune, `NeptuneGraphQueryStep` y `NeptuneTraverserConverterStep`.

`NeptuneGraphQueryStep` representa el tiempo de ejecución de toda la parte del recorrido que el motor de Neptune podría convertir y ejecutar de forma nativa.

`NeptuneTraverserConverterStep`representa el proceso de convertir la salida de esos pasos convertidos en TinkerPop travesaños que permiten procesar los pasos que no se pudieron convertir, si los hubiera, o devolver los resultados en un formato compatible. TinkerPop

En el ejemplo anterior, tenemos varios pasos no convertidos, por lo que vemos que cada uno de estos TinkerPop pasos (`ProjectStep`,`PathStep`) aparece entonces como una fila en la tabla.

La segunda columna de la tabla,`Count`, indica el número de travesaños representados que han pasado por el paso, mientras que la tercera columna,`Traversers`, indica el número de travesaños que han pasado por ese paso, tal y como se explica en la documentación del paso del perfil. TinkerPop

En nuestro ejemplo, hay 3856 vértices y 3856 recorridos que devuelve el `NeptuneGraphQueryStep`, y estos números permanecen iguales durante el resto del procesamiento, porque `ProjectStep` y `PathStep` están formateando los resultados, no filtrándolos.

> **Note**
>
> A diferencia TinkerPop, el motor Neptune no optimiza el rendimiento aumentando el volumen en sus *`NeptuneGraphQueryStep`* escalones. `NeptuneTraverserConverterStep` El aumento de volumen es la TinkerPop operación que combina los travesaños en el mismo vértice para reducir la sobrecarga operativa, y eso es lo que hace que los números y los números difieran. `Count` `Traversers` Como el aumento de volumen solo ocurre en los pasos en los que Neptuno delega TinkerPop y no en los pasos que Neptuno maneja de forma nativa, `Count` las columnas y rara vez difieren. `Traverser`

La columna de Tiempo indica el número de milisegundos que tardó el paso y la columna `% Dur` indica el porcentaje del tiempo total de procesamiento que tardó el paso. Estas son las métricas que indican dónde centrar el trabajo de ajuste, ya que muestran los pasos que han tardado más tiempo.

Métricas de operaciones de índice en la salida de la API de **`profile`** de Neptune

Otro conjunto de métricas en la salida de la API de profile de Neptune son las operaciones de índice:

```
Index Operations
================
Query execution:
    # of statement index ops: 23191
    # of unique statement index ops: 5960
    Duplication ratio: 3.89
    # of terms materialized: 0
```

Estas métricas informan de lo siguiente:

- El número total de búsquedas de índices.
- El número de búsquedas de índices únicas realizadas.
- La relación entre el total de búsquedas en el índice y las búsquedas únicas. Una relación más baja indica menos redundancia.
- El número de términos materializados a partir del diccionario de términos.

Métricas de repetición en la salida de la API de **`profile`** de Neptune

Si el recorrido utiliza un paso `repeat()` como en el ejemplo anterior, en la salida de `profile` aparecerá una sección que contiene las métricas de repetición:

```
Repeat Metrics
==============
Iteration  Visited   Output    Until     Emit     Next
-------------------------------------------------------
        0        2        0        0        0        2
        1       53        0        0        0       53
        2     3856     3856     3856        0        0
-------------------------------------------------------
              3911     3856     3856        0       55
```

Estas métricas informan de lo siguiente:

- El recuento de bucles de una fila (la columna `Iteration`).
- El número de elementos visitados por el bucle (la columna `Visited`).
- El número de elementos generados por el bucle (la columna `Output`).
- El último elemento generado por el bucle (la columna `Until`).
- El número de elementos emitidos por el bucle (la columna `Emit`).
- El número de elementos que se pasan del bucle al bucle siguiente (la columna `Next`).

Estas métricas de repetición son muy útiles para comprender el factor de ramificación del recorrido y así tener una idea del trabajo que está realizando la base de datos. Puede utilizar estos números para diagnosticar problemas de rendimiento, especialmente cuando el mismo recorrido funciona de forma muy diferente con parámetros diferentes.

Métricas de búsqueda de texto completo en la salida de la API **`profile`** de Neptune

Cuando un recorrido utiliza una búsqueda de texto completo, como en el ejemplo anterior, en la salida de `profile` aparece una sección que contiene las métricas de búsqueda de texto completo (FTS):

```
FTS Metrics
==============
SearchNode[(idVar=?1, query=Anchora~, field=city) . project ?1 .],
    {endpoint=your-OpenSearch-endpoint-URL, incomingSolutionsThreshold=1000,
 estimatedCardinality=INFINITY,
    remoteCallTimeSummary=[total=65, avg=32.500000, max=37, min=28],
    remoteCallTime=65, remoteCalls=2, joinTime=0, indexTime=0, remoteResults=2}

    2 result(s) produced from SearchNode above
```

Aquí se muestra la consulta enviada al clúster ElasticSearch (ES) y se muestran varias métricas sobre la interacción ElasticSearch que pueden ayudarle a identificar los problemas de rendimiento relacionados con la búsqueda de texto completo:

- Información resumida sobre las llamadas al ElasticSearch índice:
  - El número total de milisegundos que necesitan todas las remoteCalls para satisfacer la consulta (`total`).
  - El número medio de milisegundos empleados en una remoteCall (`avg`).
  - El número mínimo de milisegundos empleados en una remoteCall (`min`).

  - El número máximo de milisegundos empleados en una remoteCall (`max`).
- Tiempo total consumido por RemoteCalls to ElasticSearch ()`remoteCallTime`.
- El número de llamadas remotas realizadas a (). ElasticSearch `remoteCalls`
- El número de milisegundos invertidos en las uniones de ElasticSearch los resultados ()`joinTime`.
- El número de milisegundos empleados en las búsquedas de índices (`indexTime`).
- El número total de resultados devueltos por ElasticSearch (`remoteResults`).

Compatibilidad nativa con pasos de Gremlin en Amazon Neptune

El motor de Amazon Neptune actualmente no es totalmente compatible de forma nativa con todos los pasos de Gremlin, como se explica en Ajuste de las consultas de Gremlin. La compatibilidad actual se divide en cuatro categorías:

- Los pasos de Gremlin que siempre se pueden convertir en operaciones nativas del motor de Neptune
- Pasos de Gremlin que se pueden convertir en operaciones nativas del motor de Neptune en algunos casos
- Pasos de Gremlin que nunca se convierten en operaciones nativas del motor de Neptune
- Pasos de Gremlin que Neptune no admite en absoluto

Los pasos de Gremlin que siempre se pueden convertir en operaciones nativas del motor de Neptune

Muchos pasos de Gremlin se pueden convertir en operaciones nativas del motor de Neptune siempre que cumplan las siguientes condiciones:

- En la consulta no van precedidos de ningún paso que no se pueda convertir.
- Su paso principal, si lo hay, se puede convertir.
- Todos sus recorridos secundarios, si los hay, se pueden convertir.

Los siguientes pasos de Gremlin siempre se convierten en operaciones nativas del motor de Neptune si cumplen esas condiciones:

- and( )
- as( )

- count( )
- E( )
- emit( )
- explain( )
- group( )
- groupCount( )
- has( )
- identity( )
- is( )
- key( )
- label( )
- limit( )
- local( )
- loops( )
- not( )
- or( )
- profile( )
- properties( )
- subgraph( )
- until( )
- V( )
- value( )
- valueMap( )
- values( )

Pasos de Gremlin que se pueden convertir en operaciones nativas del motor de Neptune en algunos casos

Algunos pasos de Gremlin se pueden convertir en operaciones nativas del motor de Neptune en algunas situaciones, pero no en otras:

- addE( ): el paso `addE()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que vaya seguido inmediatamente de un paso `property()` que contenga un recorrido como clave.
- addV( ): el paso `addV()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que vaya seguido inmediatamente de un paso `property()` que contenga un recorrido como clave, o a menos que se asignen varias etiquetas.
- aggregate( ): el paso `aggregate()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que el paso se utilice en un recorrido secundario o en un subrecorrido, o a menos que el valor que se esté almacenando no sea un vértice, un borde, un identificador, una etiqueta o un valor de propiedad.

  En el ejemplo siguiente, `aggregate()` no se convierte porque se utiliza en un recorrido secundario:

  ```
  g.V().has('code','ANC').as('a')
       .project('flights').by(select('a')
       .outE().aggregate('x'))
  ```

  En este ejemplo, aggregate( ) no se convierte porque lo que se almacena es el `min()` de un valor:

  ```
  g.V().has('code','ANC').outE().aggregate('x').by(values('dist').min())
  ```

- barrier( ): el paso `barrier()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que el paso siguiente no se convierta.
- cap( ): el único caso en el que el paso `cap()` se convierte es cuando se combina con el paso `unfold()` para devolver una versión desplegada de un agregado de valores de vértice, borde, identificador o propiedad. En este ejemplo, `cap()` se convertirá porque va seguido de `.unfold()`.

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport').unfold()
  ```

  Sin embargo, si elimina `.unfold()`, `cap()` no se convertirá:

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport')
  ```

- coalesce (): el único caso en el que el `coalesce()` paso se convierte es cuando sigue el patrón Upsert recomendado en la página de recetas. TinkerPop No se permiten otros patrones de coalesce(). La conversión se limita al caso en el que todos los recorridos secundarios se pueden convertir, todos producen el mismo tipo que la salida (vértice, borde, identificador, valor, clave o etiqueta), todos se desplazan hacia un elemento nuevo y no contienen el paso `repeat()`.
- constant( ): actualmente, el paso constant() solo se convierte si se usa dentro de una parte de `sack().by()` de un recorrido para asignar un valor constante, como este:

```
g.V().has('code','ANC').sack(assign).by(constant(10)).out().limit(2)
```

- CyclicPath( ): el paso `cyclicPath()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que el paso se utilice con moduladores `by()`, `from()` o `to()`. En las siguientes consultas, por ejemplo, `cyclicPath()` no se convierte:

```
g.V().has('code','ANC').as('a').out().out().cyclicPath().by('code')
g.V().has('code','ANC').as('a').out().out().cyclicPath().from('a')
g.V().has('code','ANC').as('a').out().out().cyclicPath().to('a')
```

- drop( ): el paso `drop()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que el paso se utilice dentro de un paso `sideEffect(` o `optional()`.
- fold (): solo hay dos situaciones en las que se puede convertir el paso fold (), a saber, cuando se usa con el patrón Upsert recomendado en la página de TinkerPop recetas y cuando se usa en un contexto como este: `group().by()`

```
g.V().has('code','ANC').out().group().by().by(values('code', 'city').fold())
```

- id( ): el paso `id()` se convierte a menos que se utilice en una propiedad, como esta:

```
g.V().has('code','ANC').properties('code').id()
```

- order( ): el paso `order()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que se produzca algo de lo siguiente:
  - El paso `order()` se encuentra dentro de un recorrido secundario anidado, como este:

    ```
    g.V().has('code','ANC').where(V().out().order().by(id))
    ```

  - Se utilizan los pedidos locales, como por ejemplo con `order(local)`.

  - En la modulación `by()` se utiliza un comparador personalizado por el que ordenar. Un ejemplo es el uso de `sack()`:

```
g.withSack(0).
  V().has('code','ANC').
      repeat(outE().sack(sum).by('dist').inV()).times(2).limit(10).
      order().by(sack())
```

  - Hay varias ordenaciones en el mismo elemento.

- project( ): el paso `project()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que el número de instrucciones de `by()` que siguen a `project()` no coincida con el número de etiquetas especificado, como aquí:

```
g.V().has('code','ANC').project('x', 'y').by(id)
```

- range( ): el paso `range()` solo se convierte cuando el extremo inferior del rango en cuestión es cero (por ejemplo, `range(0,3)`).
- repeat( ): el paso `repeat()` generalmente se puede convertir en una operación nativa del motor de Neptune, a menos que esté anidado dentro de otro paso `repeat()`, como este:

```
g.V().has('code','ANC').repeat(out().repeat(out()).times(2)).times(2)
```

- sack(): el paso `sack()` generalmente se puede convertir en una operación nativa del motor de Neptune, excepto en los siguientes casos:
  - Si se utiliza un operador sack no numérico.
  - Si se utiliza un operador sack numérico que no sea `+`, `-`, `mult`, `div`, `min` ni `max`.
  - Si `sack()` se usa dentro de un paso `where()` para filtrar en función del valor de sack, como se muestra a continuación:

```
g.V().has('code','ANC').sack(assign).by(values('code')).where(sack().is('ANC'))
```

- sum( ): el paso `sum()` generalmente se puede convertir en una operación nativa del motor de Neptune, pero no cuando se usa para calcular una suma global, como esta:

```
g.V().has('code','ANC').outE('routes').values('dist').sum()
```

- union( ): el paso `union()` se puede convertir en una operación nativa del motor de Neptune siempre que sea el último paso de la consulta, aparte del paso terminal.

- unfold (): el `unfold()` paso solo se puede convertir en una operación nativa del motor de Neptuno cuando se usa con el patrón Upsert recomendado en TinkerPopla página de recetas y cuando se usa junto con esto: `cap()`

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport').unfold()
```

- where( ): el paso `where()` generalmente se puede convertir en una operación nativa del motor de Neptune, excepto en los siguientes casos:
  - Cuando se utilizan modulaciones by(), como esta:

```
g.V().hasLabel('airport').as('a')
     .where(gt('a')).by('runways')
```

  - Cuando se utilizan operadores de comparación que no sean `eq`, `neq`, `within` y `without`.
  - Cuando se utilizan agregaciones proporcionadas por el usuario.

Pasos de Gremlin que nunca se convierten en operaciones nativas del motor de Neptune

Los siguientes pasos de Gremlin son compatibles con Neptune, pero nunca se convierten en operaciones nativas del motor de Neptune. En su lugar, los ejecuta el servidor de Gremlin.

- choose( )
- coin( )
- inject( )
- match( )
- math( )
- max( )
- mean( )
- min( )
- option( )
- optional( )
- path( )
- propertyMap( )
- sample( )

- skip( )
- tail( )
- timeLimit( )
- tree( )

Pasos de Gremlin que Neptune no admite en absoluto

Los siguientes pasos de Gremlin no se admiten en absoluto en Neptune. En la mayoría de los casos, esto se debe a que requieren un `GraphComputer`, que Neptune no admite actualmente.

- connectedComponent( )
- io( )
- shortestPath( )
- withComputer( )
- pageRank( )
- peerPressure( )
- program( )

En realidad, el paso `io()` se admite parcialmente, ya que se puede usar `read()` desde una URL, pero no para `write()`.

Uso de Gremlin con el motor de consultas DFE de Neptune

Si habilita completamente el motor de consultas alternativo de Neptune, conocido como DFE, en el modo de laboratorio (configurando el parámetro del clúster de base de datos `neptune_lab_mode` en `DFEQueryEngine=enabled`), Neptune convierte las consultas/recorridos de Gremlin de solo lectura en una representación lógica intermedia y los ejecuta en el motor DFE siempre que sea posible.

Sin embargo, el DFE aún no admite todos los pasos de Gremlin. Cuando un paso no se puede ejecutar de forma nativa en el DFE, Neptune recurre a él TinkerPop para ejecutar el paso. Los informes `explain` y `profile` incluyen advertencias cuando esto ocurre.

> **Note**
>
> A partir de la versión 1.0.5.0 del motor, el comportamiento predeterminado del DFE para gestionar los pasos de Gremlin sin soporte nativo ha cambiado. Mientras que antes el motor DFE recaía sobre el motor Neptune Gremlin, ahora lo hace sobre el motor básico. TinkerPop

Pasos de Gremlin compatibles de forma nativa con el motor DFE

- **`GraphStep`**
- **`VertexStep`**
- **`EdgeVertexStep`**
- **`IdStep`**
- **`TraversalFilterStep`**
- **`PropertiesStep`**
- Compatibilidad de **`HasStep`** para filtrar vértices y bordes en propiedades, identificadores y etiquetas, con la excepción del texto y los predicados `Without`.
- **`WherePredicateStep`** con filtros `Path` de ámbito limitado, pero sin compatibilidad de búsqueda de `ByModulation`, `SideEffect` o `Map`.
- **`DedupGlobalStep`**, excepto la compatibilidad de búsqueda de `ByModulation`, `SideEffect` y `Map`.

Intercalación de planificación de consultas

Cuando el proceso de conversión encuentra un paso de Gremlin que no tiene un operador de DFE nativo correspondiente, antes de volver a utilizar Tinkerpop, intenta buscar otras partes de la consulta intermedias que se puedan ejecutar de forma nativa en el motor DFE. Para ello, aplica la lógica de intercalado al recorrido de nivel superior. El resultado es que, siempre que es posible, se utilizan pasos compatibles.

Cualquier conversión de consultas intermedias sin prefijo se representa mediante `NeptuneInterleavingStep` en las salidas `explain` y `profile`.

Para comparar el rendimiento, es posible que desee desactivar el intercalado en una consulta y, al mismo tiempo, seguir utilizando el motor DFE para ejecutar la parte del prefijo. O bien, puede

que desee utilizar solo el TinkerPop motor para la ejecución de consultas sin prefijo. Puede hacerlo mediante la sugerencia de consulta `disableInterleaving`.

Del mismo modo que la sugerencia de consulta useDFE con un valor de `false` impide que una consulta se ejecute en el DFE, la sugerencia de consulta `disableInterleaving` con un valor de `true` desactiva el intercalado del DFE para la conversión de una consulta. Por ejemplo:

```
g.with('Neptune#disableInterleaving', true)
 .V().has('genre','drama').in('likes')
```

Se ha actualizado la salida **`explain`** y **`profile`** de Gremlin

explain de Gremlin proporciona detalles sobre el recorrido optimizado que Neptune utiliza para ejecutar una consulta. Consulte la salida de `explain` del DFE de ejemplo para ver un ejemplo del aspecto que tiene la salida de `explain` cuando el motor DFE está habilitado.

API `profile` de Gremlin ejecuta un recorrido de Gremlin específico, recopila varias métricas sobre la ejecución y produce un informe de profile que contiene detalles sobre el plan de consultas optimizado y las estadísticas de tiempo de ejecución de varios operadores. Consulte la salida de `profile` del DFE de ejemplo para ver un ejemplo del aspecto que tiene la salida de `profile` cuando el motor DFE está habilitado.

> **Note**
>
> Dado que el DFE es una característica experimental publicada en el modo de laboratorio, el formato exacto de la salida de `explain` y `profile` está sujeto a cambios.

Acceso al gráfico de Neptune con openCypher

Neptune permite crear aplicaciones de gráficos mediante openCypher, que actualmente es uno de los lenguajes de consulta más populares para los desarrolladores que trabajan con bases de datos de gráficos. A los desarrolladores, analistas de negocio y científicos de datos les gusta la sintaxis de openCypher, que está inspirada en SQL, porque proporciona una estructura familiar para redactar consultas para aplicaciones gráficas.

openCypher es un lenguaje de consulta declarativo para gráficos de propiedades que desarrolló originalmente Neo4j, luego de código abierto en 2015, y que contribuyó al proyecto openCypher en

virtud de una licencia de código abierto Apache 2. Su sintaxis está documentada en Cypher Query Language Reference, versión 9.

Para conocer las limitaciones y diferencias en la compatibilidad con Neptune de la especificación de openCypher, consulte Conformidad con las especificaciones de OpenCypher en Amazon Neptune.

> **Note**
>
> La implementación actual de Neo4j del lenguaje de consulta Cypher se diferencia en algunos aspectos de la especificación de openCypher. Si está migrando el código de Cypher actual de Neo4j a Neptune, consulte Compatibilidad de Neptune con Neo4j y Reescritura de consultas de Cypher para ejecutarlas en openCypher en Neptune para obtener ayuda.

A partir de la versión 1.1.1.0 del motor, openCypher está disponible para su uso en producción en Neptune.

Gremlin frente a openCypher: similitudes y diferencias

Gremlin y openCypher son lenguajes de consulta basados en gráficos de propiedades y se complementan en muchos aspectos.

Gremlin se diseñó para atraer a los programadores y adaptarse perfectamente al código. Como resultado, Gremlin es imperativo por diseño, mientras que la sintaxis declarativa de openCypher puede resultarle más familiar a las personas con experiencia en SQL o SPARQL. Gremlin puede parecer más natural para un científico de datos que usa Python en un cuaderno de Jupyter, mientras que openCypher puede parecer más intuitivo para un usuario empresarial con experiencia en SQL.

Lo bueno es que no tiene que elegir entre Gremlin y openCypher en Neptune. Las consultas en cualquiera de los dos lenguajes pueden funcionar en el mismo gráfico, independientemente de cuál de los dos se haya utilizado para introducir esos datos. Puede que le resulte más cómodo usar Gremlin para algunas cosas y openCypher para otras, dependiendo de lo que esté haciendo.

Gremlin utiliza una sintaxis imperativa que le permite controlar cómo se mueve por el gráfico en una serie de pasos, cada uno de los cuales incluye un flujo de datos, realiza alguna acción sobre él (mediante un filtro, un mapa, etc.) y, a continuación, envía los resultados al siguiente paso. Por lo general, una consulta de Gremlin adopta la forma `g.V()` y va seguida de pasos adicionales.

En openCypher, se utiliza una sintaxis declarativa, inspirada en SQL, que especifica un patrón de nodos y relaciones que se pueden encontrar en el gráfico mediante una sintaxis de motivos (como

`()-[]->()`). Una consulta de openCypher suele empezar con una cláusula `MATCH`, seguida de otras cláusulas como `WHERE`, `WITH` y `RETURN`.

Introducción al uso de openCypher

Puede consultar datos de gráficos de propiedades en Neptune con openCypher independientemente de cómo se hayan cargado, pero no puede usar openCypher para consultar datos cargados como RDF.

El programa de carga masiva de Neptune acepta datos de gráficos de propiedades en un formato CSV para Gremlin y en un formato CSV para openCypher. Además, evidentemente, puede añadir datos de propiedades a su gráfico mediante consultas de Gremlin u openCypher.

Hay disponibles muchos tutoriales en línea para aprender el lenguaje de consultas Cypher. Aquí se ofrecen algunos ejemplos rápidos de consultas de openCypher que pueden ayudarle a hacerse una idea del lenguaje, pero la mejor y más sencilla forma de empezar a utilizar openCypher para consultar el gráfico de Neptune es utilizar los cuadernos de openCypher del entorno de trabajo de Neptune. El entorno de trabajo es de código abierto y está alojado en https://github.com/aws-samples/amazon-neptune-samples. GitHub

Encontrará los cuadernos OpenCypher en el repositorio de cuadernos gráficos de Neptune GitHub . En concreto, consulte la visualización de las rutas aéreas y los cuadernos de openCypher de los equipos de la liga inglesa.

Los datos que procesa openCypher adoptan la forma de una serie desordenada de mapas clave/valor. La principal forma de refinar, manipular y aumentar estos mapas es usar cláusulas que realicen tareas como la coincidencia de patrones, la inserción, la actualización y la eliminación de los pares clave/valor.

Existen varias cláusulas en openCypher para buscar patrones de datos en el gráfico, de las cuales `MATCH` es la más común. `MATCH` le permite especificar el patrón de nodos, relaciones y filtros que desea buscar en el gráfico. Por ejemplo:

- Obtener todos los nodos

```
MATCH (n) RETURN n
```

- Encontrar los nodos conectados

```
MATCH (n)-[r]->(d) RETURN n, r, d
```

- Encontrar una ruta

```
MATCH p=(n)-[r]->(d) RETURN p
```

- Obtener todos los nodos con una etiqueta

```
MATCH (n:airport) RETURN n
```

Tenga en cuenta que la primera consulta anterior devuelve todos los nodos del gráfico y las dos siguientes muestran todos los nodos que tienen una relación. Esto no suele ser recomendable. En casi todos los casos, es necesario restringir los datos que se devuelven, lo que puede hacer especificando las etiquetas y propiedades de los nodos o relaciones, como en el cuarto ejemplo.

Encontrará una hoja de referencia muy útil sobre la sintaxis de openCypher en el repositorio de ejemplos de GitHub de Neptune.

Servlet de estado de openCypher de Neptune y punto de conexión de estado

El punto de conexión de estado de openCypher proporciona acceso a la información sobre las consultas que se están ejecutando actualmente en el servidor o están en espera de ejecutarse. También le permite cancelar esas consultas. El punto de conexión es:

```
https://(the server):(the port number)/openCypher/status
```

Puede utilizar los métodos HTTP `GET` y `POST` para obtener el estado actual del servidor o para cancelar una consulta. También puede usar el método `DELETE` para cancelar una consulta en curso o en espera.

Parámetros de las solicitudes de estado

Parámetros de la consulta de estado

- **`includeWaiting`** (`true` o `false`): cuando se establece en `true` y otros parámetros no están presentes, se devuelve la información de estado tanto para las consultas en espera como para las consultas en ejecución.
- **`cancelQuery`**: se utiliza únicamente con los métodos `GET` y `POST` para indicar que se trata de una solicitud de cancelación. El método `DELETE` no necesita este parámetro.

El valor del parámetro `cancelQuery` no se utiliza, pero cuando `cancelQuery` está presente, el parámetro `queryId` es obligatorio para identificar qué consulta se va a cancelar.

- **`queryId`**: contiene el identificador de una consulta específica.

  Cuando se usa con el método `GET` o `POST` y el parámetro `cancelQuery` no está presente, `queryId` hace que se devuelva la información de estado de la consulta específica que identifica. Si el parámetro `cancelQuery` está presente, se cancela la consulta específica que `queryId` identifica.

  Cuando se usa con el método `DELETE`, `queryId` siempre indica que se debe cancelar una consulta específica.

- **`silent`**: solo se usa al cancelar una consulta. Si se establece en `true`, hace que la cancelación se produzca de forma silenciosa.

Campos de respuesta a la solicitud de estado

Campos de respuesta de estado si no se proporciona el identificador de una consulta específica

- aceptadas QueryCount: el número de consultas que se han aceptado pero que aún no se han completado, incluidas las consultas de la cola.
- en ejecución QueryCount: el número de consultas de OpenCypher que se están ejecutando actualmente.
- queries: una lista de las consultas de openCypher actuales.

Campos de respuesta de estado para una consulta específica

- queryId: un identificador GUID de la consulta. Neptune asigna automáticamente este valor de identificador a cada consulta o también puede asignar su propio identificador (consulte Inserte un identificador personalizado en una consulta de Neptune Gremlin o SPARQL).
- queryString: la consulta enviada. Esta se trunca en 1024 caracteres, si supera este número.
- consulta EvalStats: estadísticas de esta consulta:
  - waited: indica cuánto tiempo esperó la consulta, en milisegundos.
  - elapsed: el número de milisegundos que la consulta lleva en ejecución.
  - cancelled: `True` indica que la consulta se ha cancelado o `False` que no se ha cancelado.

Ejemplos de solicitudes de estado y respuestas

- Solicitud del estado de todas las consultas, incluidas las que están en espera:

```
curl https://server:port/openCypher/status \
  --data-urlencode "includeWaiting=true"
```

  Respuesta:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Solicitud del estado de todas las consultas, sin incluir las que están en espera:

```
curl https://server:port/openCypher/status
```

  Respuesta:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Solicitud del estado de una sola consulta:

```
curl https://server:port/openCypher/status \
 --data-urlencode "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

  Respuesta:

```
{
  "queryId" : "eadc6eea-698b-4a2f-8554-5270ab17ebee",
  "queryString" : "MATCH (n1)-[:knows]->(n2), (n2)-[:knows]->(n3), (n3)-[:knows]-
>(n4), (n4)-[:knows]->(n5), (n5)-[:knows]->(n6), (n6)-[:knows]->(n7), (n7)-[:knows]-
>(n8), (n8)-[:knows]->(n9), (n9)-[:knows]->(n10) RETURN COUNT(n1);",
  "queryEvalStats" : {
    "waited" : 0,
```

```
    "elapsed" : 23463,
    "cancelled" : false
  }
}
```

- Solicitudes para cancelar una consulta

  1. Uso de POST:

```
curl -X POST https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

  Respuesta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

  2. Uso de GET:

```
curl -X GET https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=588af350-cfde-4222-bee6-b9cedc87180d"
```

  Respuesta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

  3. Uso de DELETE:

```
curl -X DELETE \
  -s "https://server:port/openCypher/status?queryId=b9a516d1-d25c-4301-
bb80-10b2743ecf0e"
```

  Respuesta:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

El punto de conexión HTTPS de openCypher de Amazon Neptune

Temas

- Consultas de lectura y escritura de openCypher en el punto de conexión HTTPS
- El formato de resultados JSON de openCypher predeterminado

Consultas de lectura y escritura de openCypher en el punto de conexión HTTPS

El punto de conexión HTTPS de openCypher admite consultas de lectura y actualización mediante el método `GET` y `POST`. No se admiten los métodos `DELETE` y `PUT`.

Las siguientes instrucciones le guiarán a través de la conexión al punto de conexión de openCypher utilizando el comando `curl` y HTTPS. Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

La sintaxis es la siguiente:

```
HTTPS://(the server):(the port number)/openCypher
```

Estos son ejemplos de consultas de lectura, una que usa `POST` y otra que usa `GET`.

1. Uso de `POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n1) RETURN n1;"
```

2. Uso de `GET` (la cadena de consulta está codificada en una URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n1)%20RETURN%20n1"
```

Estos son ejemplos de consultas de escritura/actualización, una que usa `POST` y otra que usa `GET`:

1. Uso de POST:

```
curl HTTPS://server:port/openCypher \
  -d "query=CREATE (n:Person { age: 25 })"
```

2. Uso de GET (la cadena de consulta está codificada en una URL):

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=CREATE%20(n%3APerson%20%7B%20age%3A
%2025%20%7D)"
```

El formato de resultados JSON de openCypher predeterminado

El siguiente formato JSON se devuelve de forma predeterminada o si se establece explícitamente el encabezado de la solicitud en `Accept: application/json`. Este formato está diseñado para poder analizarse fácilmente en objetos utilizando las características del lenguaje nativo de la mayoría de las bibliotecas.

El documento JSON que se devuelve contiene un campo, `results`, que contiene los valores devueltos por la consulta. Los ejemplos siguientes muestran el formato JSON para valores comunes.

Ejemplo de respuesta de valor:

```
{
  "results": [
    {
      "count(a)": 121
    }
  ]
}
```

Ejemplo de respuesta de nodo:

```
{
  "results": [
    {
      "a": {
        "~id": "22",
        "~entityType": "node",
        "~labels": [
          "airport"
        ],
```

```
        "~properties": {
          "desc": "Seattle-Tacoma",
          "lon": -122.30899810791,
          "runways": 3,
          "type": "airport",
          "country": "US",
          "region": "US-WA",
          "lat": 47.4490013122559,
          "elev": 432,
          "city": "Seattle",
          "icao": "KSEA",
          "code": "SEA",
          "longest": 11901
        }
      }
    }
  ]
}
```

Ejemplo de respuesta de relación:

```
{
  "results": [
    {
      "r": {
        "~id": "7389",
        "~entityType": "relationship",
        "~start": "22",
        "~end": "151",
        "~type": "route",
        "~properties": {
          "dist": 956
        }
      }
    }
  ]
}
```

Ejemplo de respuesta de ruta:

```
{
  "results": [
    {
```

```
      "p": [
        {
          "~id": "22",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Seattle-Tacoma",
            "lon": -122.30899810791,
            "runways": 3,
            "type": "airport",
            "country": "US",
            "region": "US-WA",
            "lat": 47.4490013122559,
            "elev": 432,
            "city": "Seattle",
            "icao": "KSEA",
            "code": "SEA",
            "longest": 11901
          }
        },
        {
          "~id": "7389",
          "~entityType": "relationship",
          "~start": "22",
          "~end": "151",
          "~type": "route",
          "~properties": {
            "dist": 956
          }
        },
        {
          "~id": "151",
          "~entityType": "node",
          "~labels": [
            "airport"
          ],
          "~properties": {
            "desc": "Ontario International Airport",
            "lon": -117.600997924805,
            "runways": 2,
            "type": "airport",
            "country": "US",
```

```
            "region": "US-CA",
            "lat": 34.0559997558594,
            "elev": 944,
            "city": "Ontario",
            "icao": "KONT",
            "code": "ONT",
            "longest": 12198
          }
        }
      ]
    }
  ]
}
```

Uso del protocolo Bolt para realizar consultas de openCypher a Neptune

Bolt es un protocolo cliente/servidor orientado a declaraciones, desarrollado inicialmente por Neo4j y licenciado bajo la licencia Creative Commons 3.0 Attribution-. ShareAlike Está orientado al cliente, lo que significa que el cliente siempre inicia el intercambio de mensajes.

Para conectarse a Neptune mediante los controladores Bolt de Neo4j, simplemente sustituya la URL y el número de puerto por los puntos de conexión de su clúster mediante el esquema URI `bolt`. Si tiene una sola instancia de Neptune en ejecución, utilice el punto de conexión read_write. Si se están ejecutando varias instancias, se recomiendan dos controladores, uno para el escritor y otro para todas las réplicas de lectura. Si solo tiene los dos puntos de conexión predeterminados, basta con un controlador read_write y un controlador de solo lectura, pero si también tiene puntos de conexión personalizados, considere la posibilidad de crear una instancia de controlador para cada uno de ellos.

> **Note**
>
> Aunque la especificación de Bolt indica que Bolt puede conectarse mediante TCP o, WebSockets Neptune solo admite conexiones TCP para Bolt.

Neptune permite hasta 1000 conexiones Bolt simultáneas.

Para ver ejemplos de consultas de openCypher en varios lenguajes que utilizan los controladores de Bolt, consulte la documentación de las guías de lenguajes y controladores de Neo4j.

> **Important**
>
> Los controladores Neo4j Bolt para Python JavaScript, .NET y Golang inicialmente no admitían la renovación automática de los tokens de autenticación AWS Signature v4. Esto significa que, una vez caducada la firma (normalmente en 5 minutos), el controlador no pudo autenticarse y las solicitudes posteriores fallaron. Todos los ejemplos de Python JavaScript, .NET y Go que aparecen a continuación se vieron afectados por este problema. Consulte el problema #834 del controlador Python de Neo4j, el problema #664 de Neo4j.NET, el problema #993 del controlador Neo4j y el problema #429 JavaScript del controlador Neo4j GoLang, para obtener más información.
> A partir de la versión 5.8.0 del controlador, se lanzó una nueva versión preliminar de la API de reautenticación del controlador de Go (consulte v5.8.0 - Feedback wanted on re-authentication).

Uso de Bolt con Java para conectarse a Neptune

Puede descargar un controlador para cualquier versión que desee usar desde el repositorio MVN de Maven o puede añadir esta dependencia a su proyecto:

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
  <version>4.3.3</version>
</dependency>
```

A continuación, para conectarse a Neptune en Java mediante uno de estos controladores de Bolt, cree una instancia de controlador para la instancia principal/de escritura del clúster mediante un código como el siguiente:

```
import org.neo4j.driver.Driver;
import org.neo4j.driver.GraphDatabase;

final Driver driver =
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    AuthTokens.none(),
    Config.builder().withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Si tiene una o más réplicas de lector, también puede crear una instancia de controlador para ellas utilizando un código como el siguiente:

```
final Driver read_only_driver =              // (without connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
      Config.builder().withEncryption()
                      .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                      .build());
```

O bien, con un tiempo de espera:

```
final Driver read_only_timeout_driver =      // (with connection timeout)
  GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
    Config.builder().withConnectionTimeout(30, TimeUnit.SECONDS)
                    .withEncryption()
                    .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                    .build());
```

Si tiene puntos de conexión personalizados, puede que también valga la pena crear una instancia de controlador para cada uno de ellos.

Ejemplo de consulta de openCypher en Python con Bolt

Aquí se explica cómo hacer una consulta de openCypher en Python usando Bolt:

```
python -m pip install neo4j
```

```
from neo4j import GraphDatabase
uri = "bolt://(your cluster endpoint URL):(your cluster port)"
driver = GraphDatabase.driver(uri, auth=("username", "password"), encrypted=True)
```

Tenga en cuenta que los parámetros `auth` se omiten.

Ejemplo de consulta de openCypher en .NET con Bolt

Para realizar una consulta de OpenCypher en .NET usando Bolt, el primer paso es instalar el controlador Neo4j usando. NuHet Para realizar llamadas síncronas, use la versión `.Simple`, de esta manera:

```
Install-Package Neo4j.Driver.Simple-4.3.0
```

```
using Neo4j.Driver;

namespace hello
{
  // This example creates a node and reads a node in a Neptune
  // Cluster where IAM Authentication is not enabled.
  public class HelloWorldExample : IDisposable
  {
    private bool _disposed = false;
    private readonly IDriver _driver;
    private static string url = "bolt://(your cluster endpoint URL):(your cluster
 port)";
    private static string createNodeQuery = "CREATE (a:Greeting) SET a.message =
 'HelloWorldExample'";
    private static string readNodeQuery = "MATCH(n:Greeting) RETURN n.message";

    ~HelloWorldExample() => Dispose(false);

    public HelloWorldExample(string uri)
    {
      _driver = GraphDatabase.Driver(uri, AuthTokens.None, o =>
 o.WithEncryptionLevel(EncryptionLevel.Encrypted));
    }

    public void createNode()
    {
      // Open a session
      using (var session = _driver.Session())
      {
         // Run the query in a write transaction
        var greeting = session.WriteTransaction(tx =>
        {
          var result = tx.Run(createNodeQuery);
          // Consume the result
          return result.Consume();
        });

        // The output will look like this:
        //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
 'HelloWorldExample".....
        Console.WriteLine(greeting);
      }
    }
```

```
    public void retrieveNode()
    {
      // Open a session
      using (var session = _driver.Session())
      {
        // Run the query in a read transaction
        var greeting = session.ReadTransaction(tx =>
        {
          var result = tx.Run(readNodeQuery);
          // Consume the result. Read the single node
          // created in a previous step.
          return result.Single()[0].As<string>();
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }

    public void Dispose()
    {
      Dispose(true);
      GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
      if (_disposed)
        return;
      if (disposing)
      {
        _driver?.Dispose();
      }
      _disposed = true;
    }

    public static void Main()
    {
      using (var apiCaller = new HelloWorldExample(url))
      {
        apiCaller.createNode();
        apiCaller.retrieveNode();
      }
```

```
    }
  }
}
```

Ejemplo de consulta de openCypher en Java que utiliza Bolt con autenticación de IAM

El siguiente código de Java muestra cómo realizar consultas de openCypher en Java mediante Bolt con autenticación de IAM. El JavaDoc comentario describe su uso. Una vez que haya una instancia de controlador disponible, podrá utilizarla para realizar varias solicitudes autenticadas.

```
package software.amazon.neptune.bolt;

import com.amazonaws.DefaultRequest;
import com.amazonaws.Request;
import com.amazonaws.auth.AWS4Signer;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.http.HttpMethodName;
import com.google.gson.Gson;
import lombok.Builder;
import lombok.Getter;
import lombok.NonNull;
import org.neo4j.driver.Value;
import org.neo4j.driver.Values;
import org.neo4j.driver.internal.security.InternalAuthToken;
import org.neo4j.driver.internal.value.StringValue;

import java.net.URI;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

import static com.amazonaws.auth.internal.SignerConstants.AUTHORIZATION;
import static com.amazonaws.auth.internal.SignerConstants.HOST;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_DATE;
import static com.amazonaws.auth.internal.SignerConstants.X_AMZ_SECURITY_TOKEN;

/**
 * Use this class instead of `AuthTokens.basic` when working with an IAM
 * auth-enabled server. It works the same as `AuthTokens.basic` when using
 * static credentials, and avoids making requests with an expired signature
 * when using temporary credentials. Internally, it generates a new signature
 * on every invocation (this may change in a future implementation).
 *
```

```
 * Note that authentication happens only the first time for a pooled connection.
 *
 * Typical usage:
 *
 * NeptuneAuthToken authToken = NeptuneAuthToken.builder()
 *     .credentialsProvider(credentialsProvider)
 *     .region("aws region")
 *     .url("cluster endpoint url")
 *     .build();
 *
 * Driver driver = GraphDatabase.driver(
 *     authToken.getUrl(),
 *     authToken,
 *     config
 * );
 */

public class NeptuneAuthToken extends InternalAuthToken {
  private static final String SCHEME = "basic";
  private static final String REALM = "realm";
  private static final String SERVICE_NAME = "neptune-db";
  private static final String HTTP_METHOD_HDR = "HttpMethod";
  private static final String DUMMY_USERNAME = "username";
  @NonNull
  private final String region;
  @NonNull
  @Getter
  private final String url;
  @NonNull
  private final AWSCredentialsProvider credentialsProvider;
  private final Gson gson = new Gson();

  @Builder
  private NeptuneAuthToken(
      @NonNull final String region,
      @NonNull final String url,
      @NonNull final AWSCredentialsProvider credentialsProvider
  ) {
      // The superclass caches the result of toMap(), which we don't want
      super(Collections.emptyMap());
      this.region = region;
      this.url = url;
      this.credentialsProvider = credentialsProvider;
  }
```

```
  @Override
  public Map<String, Value> toMap() {
    final Map<String, Value> map = new HashMap<>();
    map.put(SCHEME_KEY, Values.value(SCHEME));
    map.put(PRINCIPAL_KEY, Values.value(DUMMY_USERNAME));
    map.put(CREDENTIALS_KEY, new StringValue(getSignedHeader()));
    map.put(REALM_KEY, Values.value(REALM));

    return map;
  }

  private String getSignedHeader() {
    final Request<Void> request = new DefaultRequest<>(SERVICE_NAME);
    request.setHttpMethod(HttpMethodName.GET);
    request.setEndpoint(URI.create(url));
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    request.setResourcePath("/opencypher");

    final AWS4Signer signer = new AWS4Signer();
    signer.setRegionName(region);
    signer.setServiceName(request.getServiceName());
    signer.sign(request, credentialsProvider.getCredentials());

    return getAuthInfoJson(request);
  }

  private String getAuthInfoJson(final Request<Void> request) {
    final Map<String, Object> obj = new HashMap<>();
    obj.put(AUTHORIZATION, request.getHeaders().get(AUTHORIZATION));
    obj.put(HTTP_METHOD_HDR, request.getHttpMethod());
    obj.put(X_AMZ_DATE, request.getHeaders().get(X_AMZ_DATE));
    obj.put(HOST, request.getHeaders().get(HOST));
    obj.put(X_AMZ_SECURITY_TOKEN, request.getHeaders().get(X_AMZ_SECURITY_TOKEN));

    return gson.toJson(obj);
  }
}
```

Ejemplo de consulta de openCypher en Python que utiliza Bolt con autenticación de IAM

La siguiente clase de Python le permite realizar consultas de openCypher en Python mediante Bolt con autenticación de IAM:

```
import json

from neo4j import Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
from botocore.auth import (
  SigV4Auth,
  _host_from_url,
)

SCHEME = "basic"
REALM = "realm"
SERVICE_NAME = "neptune-db"
DUMMY_USERNAME = "username"
HTTP_METHOD_HDR = "HttpMethod"
HTTP_METHOD = "GET"
AUTHORIZATION = "Authorization"
X_AMZ_DATE = "X-Amz-Date"
X_AMZ_SECURITY_TOKEN = "X-Amz-Security-Token"
HOST = "Host"


class NeptuneAuthToken(Auth):
  def __init__(
    self,
    credentials: Credentials,
    region: str,
    url: str,
    **parameters
  ):
    # Do NOT add "/opencypher" in the line below if you're using an engine version
 older than 1.2.0.0
    request = AWSRequest(method=HTTP_METHOD, url=url + "/opencypher")
    request.headers.add_header("Host", _host_from_url(request.url))
    sigv4 = SigV4Auth(credentials, SERVICE_NAME, region)
    sigv4.add_auth(request)
```

```
    auth_obj = {
      hdr: request.headers[hdr]
      for hdr in [AUTHORIZATION, X_AMZ_DATE, X_AMZ_SECURITY_TOKEN, HOST]
    }
    auth_obj[HTTP_METHOD_HDR] = request.method
    creds: str = json.dumps(auth_obj)
    super().__init__(SCHEME, DUMMY_USERNAME, creds, REALM, **parameters)
```

Esta clase se utiliza para crear un controlador de la siguiente manera:

```
  authToken = NeptuneAuthToken(creds, REGION, URL)
  driver = GraphDatabase.driver(URL, auth=authToken, encrypted=True)
```

Ejemplo de Node.js con la autenticación de IAM y Bolt

El siguiente código de Node.js usa el AWS SDK para la JavaScript versión 3 y la sintaxis de ES6 para crear un controlador que autentique las solicitudes:

```
import neo4j from "neo4j-driver";
import { HttpRequest }  from "@aws-sdk/protocol-http";
import { defaultProvider } from "@aws-sdk/credential-provider-node";
import { SignatureV4 } from "@aws-sdk/signature-v4";
import crypto from "@aws-crypto/sha256-js";
const { Sha256 } = crypto;
import assert from "node:assert";

const region = "us-west-2";
const serviceName = "neptune-db";
const host = "(your cluster endpoint URL)";
const port = 8182;
const protocol = "bolt";
const hostPort = host + ":" + port;
const url = protocol + "://" + hostPort;
const createQuery = "CREATE (n:Greeting {message: 'Hello'}) RETURN ID(n)";
const readQuery = "MATCH(n:Greeting) WHERE ID(n) = $id RETURN n.message";

async function signedHeader() {
  const req = new HttpRequest({
    method: "GET",
    protocol: protocol,
    hostname: host,
    port: port,
```

```
    // Comment out the following line if you're using an engine version older than
 1.2.0.0
    path: "/opencypher",
    headers: {
      host: hostPort
    }
  });

  const signer = new SignatureV4({
    credentials: defaultProvider(),
    region: region,
    service: serviceName,
    sha256: Sha256
  });

  return signer.sign(req, { unsignableHeaders: new Set(["x-amz-content-sha256"]) })
    .then((signedRequest) => {
      const authInfo = {
        "Authorization": signedRequest.headers["authorization"],
        "HttpMethod": signedRequest.method,
        "X-Amz-Date": signedRequest.headers["x-amz-date"],
        "Host": signedRequest.headers["host"],
        "X-Amz-Security-Token": signedRequest.headers["x-amz-security-token"]
      };
      return JSON.stringify(authInfo);
    });
}

async function createDriver() {
  let authToken = { scheme: "basic", realm: "realm", principal: "username",
 credentials: await signedHeader() };

  return neo4j.driver(url, authToken, {
      encrypted: "ENCRYPTION_ON",
      trust: "TRUST_SYSTEM_CA_SIGNED_CERTIFICATES",
      maxConnectionPoolSize: 1,
      // logging: neo4j.logging.console("debug")
    }
  );
}

function unmanagedTxn(driver) {
  const session = driver.session();
  const tx = session.beginTransaction();
```

```
  tx.run(createQuery)
  .then((res) => {
    const id = res.records[0].get(0);
    return tx.run(readQuery, { id: id });
  })
  .then((res) => {
    // All good, the transaction will be committed
    const msg = res.records[0].get("n.message");
    assert.equal(msg, "Hello");
  })
  .catch(err => {
    // The transaction will be rolled back, now handle the error.
    console.log(err);
  })
  .then(() => session.close());
}

createDriver()
.then((driver) => {
  unmanagedTxn(driver);
  driver.close();
})
.catch((err) => {
  console.log(err);
});
```

Ejemplo de consulta de openCypher en .NET que utiliza Bolt con autenticación de IAM

Para habilitar la autenticación de IAM en .NET, debe firmar una solicitud al establecer la conexión. En el siguiente ejemplo, se muestra cómo crear un ayudante de `NeptuneAuthToken` para generar un token de autenticación:

```
using Amazon.Runtime;
using Amazon.Util;
using Neo4j.Driver;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;
using System.Web;

namespace Hello
{
  /*
```

```
  * Use this class instead of `AuthTokens.None` when working with an IAM-auth-enabled
server.
  *
  * Note that authentication happens only the first time for a pooled connection.
  *
  * Typical usage:
  *
  * var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
Region).GetAuthToken(Host);
  * _driver = GraphDatabase.Driver(Url, authToken, o =>
o.WithEncryptionLevel(EncryptionLevel.Encrypted));
  */

 public class NeptuneAuthToken
 {
   private const string ServiceName = "neptune-db";
   private const string Scheme = "basic";
   private const string Realm = "realm";
   private const string DummyUserName = "username";
   private const string Algorithm = "AWS4-HMAC-SHA256";
   private const string AWSRequest = "aws4_request";

   private readonly string _accessKey;
   private readonly string _secretKey;
   private readonly string _region;

   private readonly string _emptyPayloadHash;

   private readonly SHA256 _sha256;


   public NeptuneAuthToken(string awsKey = null, string secretKey = null, string
region = null)
   {
     var awsCredentials = awsKey == null || secretKey == null
       ? FallbackCredentialsFactory.GetCredentials().GetCredentials()
       : null;

     _accessKey = awsKey ?? awsCredentials.AccessKey;
     _secretKey = secretKey ?? awsCredentials.SecretKey;
     _region = region ?? FallbackRegionFactory.GetRegionEndpoint().SystemName; //ex:
us-east-1

     _sha256 = SHA256.Create();
```

```
      _emptyPayloadHash = Hash(Array.Empty<byte>());
    }

    public IAuthToken GetAuthToken(string url)
    {
      return AuthTokens.Custom(DummyUserName, GetCredentials(url), Realm, Scheme);
    }

    /******************* AWS SIGNING FUNCTIONS ********************/
    private string Hash(byte[] bytesToHash)
    {
      return ToHexString(_sha256.ComputeHash(bytesToHash));
    }

    private static byte[] HmacSHA256(byte[] key, string data)
    {
      return new HMACSHA256(key).ComputeHash(Encoding.UTF8.GetBytes(data));
    }

    private byte[] GetSignatureKey(string dateStamp)
    {
      var kSecret = Encoding.UTF8.GetBytes($"AWS4{_secretKey}");
      var kDate = HmacSHA256(kSecret, dateStamp);
      var kRegion = HmacSHA256(kDate, _region);
      var kService = HmacSHA256(kRegion, ServiceName);
      return HmacSHA256(kService, AWSRequest);
    }

    private static string ToHexString(byte[] array)
    {
      return Convert.ToHexString(array).ToLowerInvariant();
    }

    private string GetCredentials(string url)
    {
      var request = new HttpRequestMessage
      {
        Method = HttpMethod.Get,
        RequestUri = new Uri($"https://{url}/opencypher")
      };

      var signedrequest = Sign(request);

      var headers = new Dictionary<string, object>
```

```
      {
        [HeaderKeys.AuthorizationHeader] =
signedrequest.Headers.GetValues(HeaderKeys.AuthorizationHeader).FirstOrDefault(),
        ["HttpMethod"] = HttpMethod.Get.ToString(),
        [HeaderKeys.XAmzDateHeader] =
signedrequest.Headers.GetValues(HeaderKeys.XAmzDateHeader).FirstOrDefault(),
        // Host should be capitalized, not like in Amazon.Util.HeaderKeys.HostHeader
        ["Host"] =
signedrequest.Headers.GetValues(HeaderKeys.HostHeader).FirstOrDefault(),
      };

      return JsonSerializer.Serialize(headers);
    }

    private HttpRequestMessage Sign(HttpRequestMessage request)
    {
      var now = DateTimeOffset.UtcNow;
      var amzdate = now.ToString("yyyyMMddTHHmmssZ");
      var datestamp = now.ToString("yyyyMMdd");

      if (request.Headers.Host == null)
      {
        request.Headers.Host = $"{request.RequestUri.Host}:{request.RequestUri.Port}";
      }

      request.Headers.Add(HeaderKeys.XAmzDateHeader, amzdate);

      var canonicalQueryParams = GetCanonicalQueryParams(request);

      var canonicalRequest = new StringBuilder();
      canonicalRequest.Append(request.Method + "\n");
      canonicalRequest.Append(request.RequestUri.AbsolutePath + "\n");
      canonicalRequest.Append(canonicalQueryParams + "\n");

      var signedHeadersList = new List<string>();
      foreach (var header in request.Headers.OrderBy(a => a.Key.ToLowerInvariant()))
      {
        canonicalRequest.Append(header.Key.ToLowerInvariant());
        canonicalRequest.Append(':');
        canonicalRequest.Append(string.Join(",", header.Value.Select(s => s.Trim())));
        canonicalRequest.Append('\n');
        signedHeadersList.Add(header.Key.ToLowerInvariant());
      }
      canonicalRequest.Append('\n');
```

```
      var signedHeaders = string.Join(";", signedHeadersList);
      canonicalRequest.Append(signedHeaders + "\n");
      canonicalRequest.Append(_emptyPayloadHash);

      var credentialScope = $"{datestamp}/{_region}/{ServiceName}/{AWSRequest}";
      var stringToSign = $"{Algorithm}\n{amzdate}\n{credentialScope}\n"
        + Hash(Encoding.UTF8.GetBytes(canonicalRequest.ToString()));

      var signing_key = GetSignatureKey(datestamp);
      var signature = ToHexString(HmacSHA256(signing_key, stringToSign));

      request.Headers.TryAddWithoutValidation(HeaderKeys.AuthorizationHeader,
        $"{Algorithm} Credential={_accessKey}/{credentialScope},
 SignedHeaders={signedHeaders}, Signature={signature}");

      return request;
    }

    private static string GetCanonicalQueryParams(HttpRequestMessage request)
    {
      var querystring = HttpUtility.ParseQueryString(request.RequestUri.Query);

      // Query params must be escaped in upper case (i.e. "%2C", not "%2c").
      var queryParams = querystring.AllKeys.OrderBy(a => a)
        .Select(key => $"{key}={Uri.EscapeDataString(querystring[key])}");
      return string.Join("&", queryParams);
    }
  }
}
```

A continuación, se explica cómo realizar una consulta de openCypher en .NET utilizando Bolt con autenticación de IAM. En el siguiente ejemplo se usa el ayudante `NeptuneAuthToken`.

```
using Neo4j.Driver;

namespace Hello
{
  public class HelloWorldExample
  {
    private const string Host = "(your hostname):8182";
    private const string Url = $"bolt://{Host}";
```

```
   private const string CreateNodeQuery = "CREATE (a:Greeting) SET a.message =
'HelloWorldExample'";
   private const string ReadNodeQuery = "MATCH(n:Greeting) RETURN n.message";

   private const string AccessKey = "(your access key)";
   private const string SecretKey = "(your secret key)";
   private const string Region = "(your AWS region)"; // e.g. "us-west-2"

   private readonly IDriver _driver;

   public HelloWorldExample()
   {
     var authToken = new NeptuneAuthToken(AccessKey, SecretKey,
Region).GetAuthToken(Host);

     // Note that when the connection is reinitialized after max connection lifetime
     // has been reached, the signature token could have already been expired (usually
5 min)
     // You can face exceptions like:
     //   `Unexpected server exception 'Signature expired: XXXX is now earlier than
YYYY (ZZZZ - 5 min.)`
     _driver = GraphDatabase.Driver(Url, authToken, o =>

o.WithMaxConnectionLifetime(TimeSpan.FromMinutes(60)).WithEncryptionLevel(EncryptionLevel.Encr
   }

   public async Task CreateNode()
   {
     // Open a session
     using (var session = _driver.AsyncSession())
     {
       // Run the query in a write transaction
       var greeting = await session.WriteTransactionAsync(async tx =>
       {
         var result = await tx.RunAsync(CreateNodeQuery);
         // Consume the result
         return await result.ConsumeAsync();
       });

       // The output will look like this:
       //   ResultSummary{Query=`CREATE (a:Greeting) SET a.message =
'HelloWorldExample".....
       Console.WriteLine(greeting.Query);
     }
```

```
    }

    public async Task RetrieveNode()
    {
      // Open a session
      using (var session = _driver.AsyncSession())
      {
        // Run the query in a read transaction
        var greeting = await session.ReadTransactionAsync(async tx =>
        {
          var result = await tx.RunAsync(ReadNodeQuery);
          var records = await result.ToListAsync();

          // Consume the result. Read the single node
          // created in a previous step.
          return records[0].Values.First().Value;
        });
        // The output will look like this:
        //   HelloWorldExample
        Console.WriteLine(greeting);
      }
    }
  }
}
```

Este ejemplo se puede iniciar ejecutando el siguiente código en `.NET 6` o `.NET 7` con los siguientes paquetes:

- **Neo4j**.Driver=4.3.0
- **AWSSDK**.Core=3.7.102.1

```
namespace Hello
{
  class Program
  {
    static async Task Main()
    {
      var apiCaller = new HelloWorldExample();

      await apiCaller.CreateNode();
      await apiCaller.RetrieveNode();
    }
```

```
  }
}
```

Ejemplo de consulta de openCypher en Golang que utiliza Bolt con autenticación de IAM

En el siguiente paquete de Golang, se muestra cómo realizar consultas de openCypher en el lenguaje Go utilizando Bolt con autenticación de IAM:

```
package main

import (
  "context"
  "encoding/json"
  "fmt"
  "github.com/aws/aws-sdk-go/aws/credentials"
  "github.com/aws/aws-sdk-go/aws/signer/v4"
  "github.com/neo4j/neo4j-go-driver/v5/neo4j"
  "log"
  "net/http"
  "os"
  "time"
)

const (
  ServiceName   = "neptune-db"
  DummyUsername = "username"
)

// Find node by id using Go driver
func findNode(ctx context.Context, region string, hostAndPort string, nodeId string)
 (string, error) {
  req, err := http.NewRequest(http.MethodGet, "https://"+hostAndPort+"/opencypher",
 nil)

  if err != nil {
    return "", fmt.Errorf("error creating request, %v", err)
  }

  // credentials must have been exported as environment variables
  signer := v4.NewSigner(credentials.NewEnvCredentials())
  _, err = signer.Sign(req, nil, ServiceName, region, time.Now())
```

```
  if err != nil {
    return "", fmt.Errorf("error signing request: %v", err)
  }

  hdrs := []string{"Authorization", "X-Amz-Date", "X-Amz-Security-Token"}
  hdrMap := make(map[string]string)
  for _, h := range hdrs {
    hdrMap[h] = req.Header.Get(h)
  }

  hdrMap["Host"] = req.Host
  hdrMap["HttpMethod"] = req.Method

  password, err := json.Marshal(hdrMap)
  if err != nil {
    return "", fmt.Errorf("error creating JSON, %v", err)
  }
  authToken := neo4j.BasicAuth(DummyUsername, string(password), "")
  // +s enables encryption with a full certificate check
  // Use +ssc to disable client side TLS verification
  driver, err := neo4j.NewDriverWithContext("bolt+s://"+hostAndPort+"/opencypher",
authToken)
  if err != nil {
    return "", fmt.Errorf("error creating driver, %v", err)
  }

  defer driver.Close(ctx)

  if err := driver.VerifyConnectivity(ctx); err != nil {
    log.Fatalf("failed to verify connection, %v", err)
  }

  config := neo4j.SessionConfig{}

  session := driver.NewSession(ctx, config)
  defer session.Close(ctx)

  result, err := session.Run(
    ctx,
    fmt.Sprintf("MATCH (n) WHERE ID(n) = '%s' RETURN n", nodeId),
    map[string]any{},
  )
  if err != nil {
    return "", fmt.Errorf("error running query, %v", err)
```

```
  }

  if !result.Next(ctx) {
    return "", fmt.Errorf("node not found")
  }

  n, found := result.Record().Get("n")
  if !found {
    return "", fmt.Errorf("node not found")
  }

  return fmt.Sprintf("+%v\n", n), nil
}

func main() {
  if len(os.Args) < 3 {
    log.Fatal("Usage: go main.go (region) (host and port)")
  }
  region := os.Args[1]
  hostAndPort := os.Args[2]
  ctx := context.Background()

  res, err := findNode(ctx, region, hostAndPort,
 "72c2e8c1-7d5f-5f30-10ca-9d2bb8c4afbc")
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(res)
}
```

Comportamiento de conexión de Bolt en Neptune

Aquí se incluyen algunos aspectos que debe tener en cuenta sobre las conexiones de Bolt de Neptune:

- Como las conexiones Bolt se crean en la capa TCP, no se puede usar un equilibrador de carga de aplicación delante de ellas, como sí puede hacer con un punto de conexión HTTP.
- El puerto que Neptune utiliza para las conexiones de Bolt es el puerto de su clúster de base de datos.
- Según el preámbulo de Bolt que se le haya pasado, el servidor de Neptune selecciona la versión de Bolt más alta adecuada (1, 2, 3 o 4.0).

- El número máximo de conexiones al servidor de Neptune que un cliente puede tener abiertas en cualquier momento es de 1000.
- Si el cliente no cierra una conexión después de una consulta, esa conexión se puede usar para ejecutar la siguiente consulta.
- Sin embargo, si una conexión está inactiva durante 20 minutos, el servidor la cierra automáticamente.
- Si la autenticación de IAM no está habilitada, puede utilizar `AuthTokens.none()` en lugar de proporcionar un nombre de usuario y una contraseña ficticios. Por ejemplo, en Java:

```
GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
 AuthTokens.none(),

 Config.builder().withEncryption().withTrustStrategy(TrustStrategy.trustSystemCertificates())
```

- Cuando la autenticación de IAM está habilitada, una conexión de Bolt siempre se desconecta unos minutos más de 10 días después de su establecimiento si no se ha cerrado aún por algún otro motivo.
- Si el cliente envía una consulta para su ejecución a través de una conexión sin haber consumido los resultados de una consulta anterior, la nueva consulta se descarta. Para descartar los resultados anteriores, el cliente debe enviar un mensaje de restablecimiento a través de la conexión.
- Solo se puede crear una transacción a la vez en una conexión determinada.
- Si se produce una excepción durante una transacción, el servidor de Neptune revierte la transacción y cierra la conexión. En este caso, el controlador crea una nueva conexión para la siguiente consulta.
- Tenga en cuenta que las sesiones no son seguras para los subprocesos. Las operaciones paralelas múltiples deben utilizar varias sesiones independientes.

Ejemplos de consultas parametrizadas de openCypher

Neptune admite consultas de openCypher parametrizadas. Esto le permite usar la misma estructura de consulta varias veces con argumentos diferentes. Como la estructura de la consulta no cambia, Neptune puede almacenar en caché su árbol de sintaxis abstracto (AST) en lugar de tener que analizarlo varias veces.

Ejemplo de una consulta parametrizada de openCypher con el punto de conexión HTTPS

A continuación se muestra un ejemplo de uso de una consulta parametrizada con el punto de conexión HTTPS de openCypher en Neptune. La consulta es:

```
MATCH (n {name: $name, age: $age})
RETURN n
```

Los parámetros se definen como sigue:

```
parameters={"name": "john", "age": 20}
```

Con `GET`, puede enviar la consulta parametrizada de la siguiente manera:

```
curl -k \
  "https://localhost:8182/openCypher?query=MATCH%20%28n%20%7Bname:\$name,age:\$age%7D
%29%20RETURN%20n&parameters=%7B%22name%22:%22john%22,%22age%22:20%7D"
```

Como alternativa también puede utilizar `POST`.

```
curl -k \
  https://localhost:8182/openCypher \
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d "parameters={\"name\": \"john\", \"age\": 20}"
```

O con `DIRECT POST`:

```
curl -k \
   -H "Content-Type: application/opencypher" \
  "https://localhost:8182/openCypher?parameters=%7B%22name%22:%22john%22,%22age
%22:20%7D" \
  -d "MATCH (n {name: \$name, age: \$age}) RETURN n"
```

Ejemplos de consultas parametrizadas de openCypher con Bolt

A continuación, se muestra un ejemplo de Python de una consulta parametrizada de openCypher que utiliza el protocolo Bolt:

```
from neo4j import GraphDatabase
```

```
uri = "bolt://[neptune-endpoint-url]:8182"
driver = GraphDatabase.driver(uri, auth=("", ""))

def match_name_and_age(tx, name, age):
  # Parameterized Query
  tx.run("MATCH (n {name: $name, age: $age}) RETURN n", name=name, age=age)

with driver.session() as session:
  # Parameters
  session.read_transaction(match_name_and_age, "john", 20)

driver.close()
```

A continuación, se muestra un ejemplo de Java de una consulta parametrizada de openCypher que utiliza el protocolo Bolt:

```
Driver driver = GraphDatabase.driver("bolt+s://(your cluster endpoint URL):8182");
HashMap<String, Object> parameters = new HashMap<>();
parameters.put("name", "john");
parameters.put("age", 20);
String queryString = "MATCH (n {name: $name, age: $age}) RETURN n";
Result result = driver.session().run(queryString, parameters);
```

Modelo de datos de openCypher

El motor de openCypher de Neptune se basa en el mismo modelo de gráficos de propiedades que Gremlin. En particular:

- Cada nodo tiene una o varias etiquetas. Si inserta un nodo sin etiquetas, se adjunta una etiqueta predeterminada denominada `vertex`. Si intenta eliminar todas las etiquetas de un nodo, se produce un error.
- Una relación es una entidad que tiene exactamente un tipo de relación y que forma una conexión unidireccional entre dos nodos (es decir, desde uno de los nodos al otro).
- Tanto los nodos como las relaciones pueden tener propiedades, pero no es necesario que las tengan. Neptune admite nodos y relaciones con propiedades cero.
- Neptune no admite metapropiedades, que tampoco están incluidas en la especificación de openCypher.
- Las propiedades del gráfico pueden tener varios valores si se crearon con Gremlin. Es decir, una propiedad de nodo o relación puede tener un conjunto de valores diferentes en lugar de solo uno.

Neptune ha ampliado la semántica de openCypher para gestionar correctamente las propiedades con varios valores.

Los tipos de datos compatibles se documentan en Formato de datos de openCypher. Sin embargo, por el momento no recomendamos insertar valores de propiedades `Array` en un gráfico de openCypher. Aunque es posible insertar un valor de propiedad de matriz mediante el programa de carga masiva, la versión actual de openCypher de Neptune lo trata como un conjunto de propiedades con varios valores en lugar de como un valor de lista único.

A continuación, se muestra la lista de tipos de datos compatibles con esta versión:

- `Bool`
- `Byte`
- `Short`
- `Int`
- `Long`
- `Float` (Incluye más y menos Infinity y NaN, pero no INF)
- `Double` (Incluye más y menos Infinity y NaN, pero no INF)
- `DateTime`
- `String`

La característica **`explain`** de openCypher

La característica `explain` de openCypher es una herramienta de autoservicio de Amazon Neptune que le ayuda a entender el enfoque de ejecución adoptado por el motor de Neptune. Para invocar a explain, se pasa un parámetro a una solicitud HTTPS de openCypher con`explain=`*`mode`*, donde el valor de *mode* puede ser uno de los siguientes:

- **`static`**: en el modo `static`, `explain` solo imprime la estructura estática del plan de consulta. En realidad, no ejecuta la consulta.
- **`dynamic`**: en el modo `dynamic`, `explain` también ejecuta la consulta e incluye los aspectos dinámicos del plan de consulta. Estos ellos, se puede incluir el número de enlaces intermedios que fluyen a través de los operadores, la proporción entre los enlaces entrantes y los enlaces salientes y el tiempo total que necesita cada operador.

- **`details`**: en el modo de `details`, `explain` imprime la información mostrada en el modo dinámico más detalles adicionales como la cadena de consulta de openCypher real y el recuento de intervalo estimado para el patrón subyacente de un operador de unión.

Por ejemplo, con `POST`:

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n) RETURN n LIMIT 1;" \
  -d "explain=dynamic"
```

O con `GET`:

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n)%20RETURN%20n%20LIMIT
%201&explain=dynamic"
```

Limitaciones de **`explain`** de openCypher en Neptune

La versión actual de explain de openCypher tiene las siguientes limitaciones:

- Actualmente, los planes de explain solo están disponibles para consultas que realizan operaciones de solo lectura. No se admiten consultas que realicen algún tipo de mutación, como `CREATE`, `DELETE`, `MERGE`, `SET`, etc.
- Los operadores y la salida de un plan específico pueden cambiar en futuras versiones.

Operadores de DFE en la salida de **`explain`** de openCypher

Para utilizar la información que proporciona la característica `explain` de openCypher, debe comprender algunos detalles sobre el funcionamiento del motor de consultas DFE (DFE es el motor que utiliza Neptune para procesar las consultas de openCypher).

El motor DFE traduce cada consulta en una canalización de operadores. Partiendo del primer operador, las soluciones intermedias fluyen de un operador al siguiente a través de esta canalización de operadores. Cada fila de la tabla explain representa un resultado, hasta el punto de evaluación.

Los operadores que pueden aparecer en un plan de consulta de DFE son los siguientes:

DFEApply: ejecuta la función especificada en la sección de argumentos, en el valor almacenado en la variable especificada

DFE BindRelation: une variables con los nombres especificados

DFE ChunkLocal SubQuery: se trata de una operación sin bloqueo que actúa como un contenedor alrededor de las subconsultas que se están realizando.

DFE DistinctColumn: devuelve el subconjunto distinto de los valores de entrada en función de la variable especificada.

DFE DistinctRelation: devuelve el subconjunto distinto de las soluciones de entrada en función de la variable especificada.

DFEDrain: aparece al final de una subconsulta para actuar como paso de finalización de esa subconsulta. El número de soluciones se registra como `Units In`. `Units Out` siempre es cero.

DFE ForwardValue: copia todos los fragmentos de entrada directamente como fragmentos de salida para pasarlos a su operador descendente.

DFE GroupBy HashIndex: realiza una operación de agrupamiento sobre las soluciones de entrada en función de un índice hash previamente calculado (mediante la operación). `DFEHashIndexBuild` Como salida, la entrada especificada se amplía con una columna que contiene una clave de grupo para cada solución de entrada.

Compilación DFE: HashIndex crea un índice hash sobre un conjunto de variables como efecto secundario. Este índice hash se suele reutilizar en operaciones posteriores. Consulte `DFEHashIndexJoin` o `DFEGroupByHashIndex` para ver dónde podría usarse este índice hash.

Combinación DFE HashIndex: realiza una combinación de las soluciones entrantes con respecto a un índice hash creado anteriormente. Consulte `DFEHashIndexBuild` para ver dónde podría crear este índice hash.

DFE JoinExists: toma una relación de entrada izquierda y derecha y retiene los valores de la relación izquierda que tienen un valor correspondiente en la relación derecha, tal como se define en las variables de unión dadas.

: se trata de una operación sin bloqueo que actúa como encapsulador de una subconsulta, lo que permite que se ejecute repetidamente para su uso en bucles.

DFE MergeChunks: se trata de una operación de bloqueo que combina fragmentos de su operador ascendente en un único bloque de soluciones para pasarlos a su operador descendente (a la inversa de). `DFESplitChunks`

DFEMinus: toma una relación de entrada de la izquierda y la derecha y conserva los valores de la relación de la izquierda que no tienen un valor correspondiente en la relación de la derecha, tal como se define en las variables de unión especificadas. Si no hay superposición de variables en ambas relaciones, este operador simplemente devuelve la relación de entrada de la izquierda.

DFE NotExists: toma una relación de entrada entre la izquierda y la derecha y conserva los valores de la relación izquierda que no tienen un valor correspondiente en la relación derecha, tal como se define en las variables de unión dadas. Si no hay superposición de variables en ambas relaciones, este operador devuelve una relación vacía.

DFE OptionalJoin: realiza una unión externa por la izquierda (también denominada unión OPCIONAL): las soluciones del lado izquierdo que tienen al menos un compañero de unión en el lado derecho se unen, y las soluciones del lado izquierdo sin un compañero de unión en el lado derecho se reenvían tal cual. Es una operación de bloqueo.

DFE PipelineJoin: une la entrada con el patrón de tuplas definido por el argumento. `pattern`

PipelineRangeRecuento de DFE: cuenta el número de soluciones que coinciden con un patrón determinado y devuelve una única solución uniaria que contiene el valor del recuento.

DFE PipelineScan: escanea la base de datos en busca del `pattern` argumento dado, con o sin un filtro determinado en las columnas.

DFEProject: toma varias columnas de entrada y proyecta solo las columnas deseadas.

DFEReduce: realiza la función de agregación especificada en variables especificadas.

DFE RelationalJoin: une la entrada del operador anterior en función de las claves de patrón especificadas mediante una combinación de combinación. Es una operación de bloqueo.

DFE RouteChunks: toma los fragmentos de entrada de su único borde de entrada y los enruta a lo largo de sus múltiples bordes de salida.

DFE SelectRows: este operador toma filas de forma selectiva de sus soluciones de relaciones de entrada de la izquierda y las reenvía a su operador descendente. Las filas se seleccionan en función de los identificadores de línea proporcionados en la relación de entrada derecha del operador.

DFESerialize: serializa los resultados finales de una consulta en una serialización de cadenas JSON, asignando cada solución de entrada al nombre de variable correspondiente. En el caso de resultados de nodos y bordes, estos resultados se serializan en un mapa de propiedades y metadatos de la entidad.

DFESort: toma una relación de entrada y produce una relación ordenada en función de la clave de clasificación proporcionada.

SplitByGrupo DFE: divide cada fragmento de entrada individual de un borde de entrada en fragmentos de salida más pequeños que corresponden a los grupos de filas identificados mediante los ID de fila del fragmento de entrada correspondiente del otro borde de entrada.

DFE SplitChunks: divide cada fragmento de entrada individual en fragmentos de salida más pequeños (a la inversa de). `DFEMergeChunks`

StreamingHashIndexBuildDFE: versión en streaming de. `DFEHashIndexBuild`

StreamingGroupByHashÍndice DFE: versión en streaming de. `DFEGroupByHashIndex`

DFESubquery: este operador aparece al principio de todos los planes y encapsula las partes del plan que se ejecutan en el motor DFE, que es el plan completo de openCypher.

Combinación DFE: SymmetricHash une la entrada del operador anterior en función de las claves de patrón especificadas mediante una combinación hash. Es una operación sin bloqueo.

DFESync: se trata de un operador de sincronización que admite planes sin bloqueo. Toma las soluciones de dos periferias entrantes y las envía a los periferias posteriores apropiadas. Para fines de sincronización, las entradas a lo largo de uno de estas periferias pueden almacenarse internamente en búfer.

DFETee: se trata de un operador de ramificación que envía el mismo conjunto de soluciones a varios operadores.

DFE TermResolution: realiza una operación de localización o globalización en sus entradas, lo que da como resultado columnas de identificadores localizados o globalizados, respectivamente.

: despliega listas de valores de una columna de entrada a la columna de salida como elementos individuales.

DFEUnion: toma dos o más relaciones de entrada y produce una unión de esas relaciones mediante el esquema de salida deseado.

SolutionInjection— Aparece antes que todo lo demás en el resultado explicativo, con un valor de 1 en la columna Unidades de salida. Sin embargo, no sirve para nada y en realidad no inyecta ninguna solución en el motor DFE.

TermResolution— Aparece al final de los planos y convierte los objetos del motor de Neptune en objetos de OpenCypher.

Columnas en la salida de **`explain`** de openCypher

La información del plan de consultas que Neptune genera como resultado de explain de openCypher contiene tablas con un operador por fila. Esta tabla tiene las siguientes columnas:

ID: el identificador numérico de este operador del plan.

Salida 1 (y Salida 2): los identificadores de los operadores que se encuentran más abajo de este operador. Puede haber como máximo dos operadores más abajo.

Nombre: el nombre de este operador.

Argumentos: cualquier detalle relevante para el operador. Incluye elementos como el esquema de entrada, el esquema de salida, el patrón (para `PipelineScan` y `PipelineJoin`), etc.

Modo: etiqueta que describe el comportamiento fundamental del operador. La mayor parte de esta columna está en blanco (`-`). Una excepción es `TermResolution`, donde el modo puede ser `id2value_opencypher`, que indica una resolución desde el identificador hasta el valor de openCypher.

Unidades de entrada: el número de soluciones que se pasan como entrada a este operador. Los operadores sin operadores por encima, como `DFEPipelineScan`, `SolutionInjections` y `DFESubquery`, sin un valor estático inyectado, tendrían un valor cero.

Unidades de salida: la cantidad de soluciones producidas como salida de este operador. `DFEDrain` es un caso especial en el que el número de soluciones que se están drenando se registra en `Units In` y `Units Out` siempre es cero.

Relación: la relación de `Units Out` a `Units In`.

Tiempo (ms): el tiempo de CPU que consume este operador, en milisegundos.

Salida de ejemplo básico de explain de openCypher

A continuación, se muestra un ejemplo básico de una salida de `explain` de openCypher: La consulta es una búsqueda de un solo nodo en el conjunto de datos de rutas aéreas para un nodo con

el código de aeropuerto ATL que invoca a `explain` utilizando el modo `details` en el formato de salida ASCII predeterminado:

```
curl -d "query=MATCH (n {code: 'ATL'}) RETURN n" -k https://localhost:8182/openCypher -
d "explain=details"
                                    ~
Query:
MATCH (n {code: 'ATL'}) RETURN n

############################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
############################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
############################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 1         # 0.00  # 4.00      #
############################################################################################
# 2  # -      # -      # TermResolution    # vars=[?n]          # id2value_opencypher #
 1        # 1         # 1.00  # 2.00      #
############################################################################################


subQuery1
############################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
############################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?n) with property 'code'
 as ?n_code2 and label 'ALL'                                       # -    # 0
   # 1         # 0.00  # 0.21      #
#    #        #        #                       # inlineFilters=[(?n_code2 IN
 ["ATL"^^xsd:string])]                                                  #
  #          #           #       #           #
#    #        #        #                       # patternEstimate=1
                                                                 #      #
  #           #       #           #
############################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#9d84f97c-c3b0-459a-98d5-955a8726b159/graph_1 # -    #
 1        # 1         # 1.00  # 0.04      #
```

```
##########################################################################################
# 2  # 3      # -      # DFEProject            # columns=[?n]
                                                                    # -    # 1
  # 1         # 1.00  # 0.04      #
##########################################################################################
# 3  # -      # -      # DFEDrain              # -
                                                                    # -    # 1
  # 0         # 0.00  # 0.03      #
##########################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#9d84f97c-
c3b0-459a-98d5-955a8726b159/graph_1
##########################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                     # Mode # Units In # Units Out # Ratio # Time (ms) #
##########################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?n, ?n_code2]
                     # -    # 0        # 1         # 0.00  # 0.02      #
##########################################################################################
# 1  # 2      # 3      # DFETee               # -
                     # -    # 1        # 2         # 2.00  # 0.02      #
##########################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?n
                     # -    # 1        # 1         # 1.00  # 0.20      #
#    #        #        #                      # ordered=false
                     #      #          #           #       #           #
##########################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?n]
                     # -    # 1        # 1         # 1.00  # 0.04      #
##########################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?n) with property 'ALL'
 and label '?n_label1' # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # patternEstimate=3506
                     #      #          #           #       #           #
##########################################################################################
# 5  # 6      # 7      # DFESync              # -
                     # -    # 2        # 2         # 1.00  # 0.02      #
##########################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                     # -    # 1        # 1         # 1.00  # 0.01      #
##########################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                     # -    # 1        # 1         # 1.00  # 0.01      #
```

```
#########################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                   # -    # 2        # 1         # 0.50  # 0.35      #
#########################################################################################
# 9  # -      # -      # DFEDrain              # -
                   # -    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################
```

En el nivel superior, `SolutionInjection` aparece antes que todo lo demás, con 1 unidad de salida. Tenga en cuenta que, en realidad, no inyecta ninguna solución. Puede ver que el siguiente operador, `DFESubquery`, tiene 0 unidades.

Después de `SolutionInjection` en el nivel superior, están los operadores `DFESubquery` y `TermResolution`. `DFESubquery` encapsula las partes del plan de ejecución de consultas que se envían al motor DFE (en el caso de las consultas de openCypher, el DFE ejecuta todo el plan de consultas). Todos los operadores del plan de consultas están anidados dentro de `subQuery1`, al que hace referencia `DFESubquery`. La única excepción es `TermResolution` que materializa los identificadores internos en objetos de openCypher completamente serializados.

Todos los operadores que se desplazan hacia abajo hasta el motor DFE tienen nombres que comienzan con un prefijo `DFE`. Como se ha mencionado anteriormente, el DFE ejecuta todo el plan de consultas de openCypher, por lo que todos los operadores, excepto el operador `TermResolution` final, comienzan con `DFE`.

En el interior de `subQuery1`, puede haber cero o más operadores `DFEChunkLocalSubQuery` o `DFELoopSubQuery` que encapsulen una parte del plan de ejecución desplazado que se ejecuta en un mecanismo limitado por la memoria. `DFEChunkLocalSubQuery`contiene aquí un `SolutionInjection` que se utiliza como entrada para la subconsulta. Para buscar la tabla de esa subconsulta en la salida, busque la `subQuery=`*`graph URI`* especificada en la columna `Arguments` correspondiente al operador `DFEChunkLocalSubQuery` o `DFELoopSubQuery`.

En `subQuery1`, `DFEPipelineScan` con el `ID` 0, escanea la base de datos en busca de un `pattern` especificado. El patrón busca una entidad con la propiedad `code` guardada como variable `?n_code2` en todas las etiquetas (puede filtrar una etiqueta específica agregando `airport` a `n:airport`). El argumento `inlineFilters` muestra el filtrado de la propiedad `code`, que equivale a `ATL`.

A continuación, el operador `DFEChunkLocalSubQuery` une los resultados intermedios de una subconsulta que contiene `DFEPipelineJoin`. Esto asegura que en realidad `?n` sea un nodo, ya que el `DFEPipelineScan` anterior busca cualquier entidad con la propiedad `code`.

Ejemplo de salida de **`explain`** para una búsqueda de relaciones con un límite

Esta consulta busca relaciones entre dos nodos anónimos con el tipo `route` y devuelve como máximo 10. De nuevo, el modo `explain` es `details` y el formato de salida es el formato ASCII predeterminado. Esta es la salida de `explain`.

Aquí, `DFEPipelineScan` busca bordes que comiencen en un nodo anónimo `?anon_node7` y terminen en otro nodo anónimo`?anon_node21`, con un tipo de relación guardado como `?p_type1`. Hay un filtro para `?p_type1` que es `el://route` (donde `el` significa la etiqueta de borde), que corresponde a `[p:route]` en la cadena de consulta.

`DFEDrain` recopila la solución de salida con un límite de 10, como se muestra en su columna `Arguments`. `DFEDrain`termina una vez que se alcanza el límite o se producen todas las soluciones, lo que ocurra primero.

```
curl -d "query=MATCH ()-[p:route]->() RETURN p LIMIT 10" -k https://localhost:8182/
openCypher -d "explain=details"
                                        ~
Query:
MATCH ()-[p:route]->() RETURN p LIMIT 10

#######################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
#######################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 10        # 0.00  # 5.00      #
#######################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 10       # 10        # 1.00  # 1.00      #
#######################################################################################


subQuery1
#######################################################################################
# ID # Out #1 # Out #2 # Name            # Arguments
              # Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
```

```
# 0  # 1      # -      # DFEPipelineScan # pattern=Edge((?anon_node7)-[?p:?p_type1]->(?
anon_node21)) # -    # 0        # 1000      # 0.00  # 0.66      #
#    #        #        #                 # inlineFilters=[(?p_type1 IN [<el://route>])]
             #      #          #           #       #           #
#    #        #        #                 # patternEstimate=26219
             #      #          #           #       #           #
################################################################################################
# 1  # 2      # -      # DFEProject      # columns=[?p]
             # -    # 1000     # 1000      # 1.00  # 0.14      #
################################################################################################
# 2  # -      # -      # DFEDrain        # limit=10
             # -    # 1000     # 0         # 0.00  # 0.11      #
################################################################################################
```

Ejemplo de salida de **`explain`** de una función de expresión de valores

La función es:

```
MATCH (a) RETURN DISTINCT labels(a)
```

En la siguiente salida de `explain`, `DFEPipelineScan` (ID 0) busca todas las etiquetas de los nodos. Esto se corresponde a `MATCH (a)`.

`DFEChunkLocalSubquery` (ID 1) agrega la etiqueta de `?a` por cada `?a`. Esto se corresponde a `labels(a)`. Puede ver esto en `DFEApply` y `DFEReduce`.

`BindRelation` (ID 2) se usa para cambiar el nombre de la columna genérica `?__gen_labelsOfa2` por `?labels(a)`.

`DFEDistinctRelation` (ID 4) recupera solo las etiquetas distintas (múltiple: los nodos de aeropuertos darían lugar a etiquetas duplicadas (a): ["airport"]). Esto se corresponde a `DISTINCT labels(a)`.

```
curl -d "query=MATCH (a) RETURN DISTINCT labels(a)" -k https://localhost:8182/
openCypher -d "explain=details"
                                          ~
Query:
MATCH (a) RETURN DISTINCT labels(a)

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
```

```
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                    #
 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                    #
 0        # 5         # 0.00  # 81.00     #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?labels(a)]  # id2value_opencypher #
 5        # 5         # 1.00  # 1.00      #
################################################################################################



subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                          # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'ALL'
 and label '?a_label1'                                                      # -    # 0
    # 3750      # 0.00  # 26.77     #
#    #        #        #                      # patternEstimate=3506
                                                                        #      #
  #          #       #          #
################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-a48a-c76a0465cfab/graph_1 # -    #
 3750     # 3750      # 1.00  # 0.04      #
################################################################################################
# 2  # 3      # -      # DFEBindRelation       # inputVars=[?a, ?__gen_labelsOfa2, ?
__gen_labelsOfa2]                                                           # -    # 3750
     # 3750      # 1.00  # 0.08      #
#    #        #        #                      # outputVars=[?a, ?__gen_labelsOfa2, ?
labels(a)]                                                                  #      #
    #          #       #          #
################################################################################################
# 3  # 4      # -      # DFEProject            # columns=[?labels(a)]
                                                                        # -    # 3750
  # 3750      # 1.00  # 0.05      #
################################################################################################
# 4  # 5      # -      # DFEDistinctRelation   # -
                                                                        # -    # 3750
  # 5         # 0.00  # 2.78      #
################################################################################################
```

```
# 5  # -      # -      # DFEDrain              # -
                                                                  # -    # 5
  # 0         # 0.00  # 0.03      #
#################################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#8b314f55-2cc7-456a-
a48a-c76a0465cfab/graph_1
#################################################################################################
# ID # Out #1 # Out #2 # Name                 # Arguments
                     # Mode     # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a]
                     # -        # 0         # 3750      # 0.00  # 0.02      #
#################################################################################################
# 1  # 2      # 3      # DFETee               # -
                     # -        # 3750     # 7500      # 2.00  # 0.02      #
#################################################################################################
# 2  # 4      # -      # DFEProject           # columns=[?a]
                     # -        # 3750     # 3750      # 1.00  # 0.04      #
#################################################################################################
# 3  # 17     # -      # DFEOptionalJoin      # -
                     # -        # 7500     # 3750      # 0.50  # 0.44      #
#################################################################################################
# 4  # 5      # -      # DFEDistinctRelation  # -
                     # -        # 3750     # 3750      # 1.00  # 2.23      #
#################################################################################################
# 5  # 6      # -      # DFEDistinctColumn    # column=?a
                     # -        # 3750     # 3750      # 1.00  # 1.50      #
#    #        #        #                      # ordered=false
                     #          #          #           #       #           #
#################################################################################################
# 6  # 7      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label3' # -        # 3750     # 3750      # 1.00  # 10.58     #
#    #        #        #                      # patternEstimate=3506
                     #          #          #           #       #           #
#################################################################################################
# 7  # 8      # 9      # DFETee               # -
                     # -        # 3750     # 7500      # 2.00  # 0.02      #
#################################################################################################
# 8  # 10     # -      # DFEBindRelation      # inputVars=[?a_label3]
                     # -        # 3750     # 3750      # 1.00  # 0.04      #
#    #        #        #                      # outputVars=[?100]
                     #          #          #           #       #           #
```

```
#################################################################################################
# 9  # 11     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 7500     # 3750      # 0.50  # 0.07      #
#    #        #        #                     # outputVars=[?a, ?a_label3, ?100]
                    #          #          #           #       #           #
#################################################################################################
# 10 # 9      # -      # DFETermResolution    # column=?100
                    # id2value # 3750     # 3750      # 1.00  # 7.60      #
#################################################################################################
# 11 # 12     # -      # DFEBindRelation      # inputVars=[?a, ?a_label3, ?100]
                    # -        # 3750     # 3750      # 1.00  # 0.06      #
#    #        #        #                     # outputVars=[?a, ?100, ?a_label3]
                    #          #          #           #       #           #
#################################################################################################
# 12 # 13     # -      # DFEApply             # functor=nodeLabel(?a_label3)
                    # -        # 3750     # 3750      # 1.00  # 0.55      #
#################################################################################################
# 13 # 14     # -      # DFEProject           # columns=[?a, ?a_label3_alias4]
                    # -        # 3750     # 3750      # 1.00  # 0.05      #
#################################################################################################
# 14 # 15     # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.02      #
#################################################################################################
# 15 # 16     # -      # DFEReduce            # functor=collect(?a_label3_alias4)
                    # -        # 3750     # 3750      # 1.00  # 6.37      #
#    #        #        #                     # segmentationKey=[?a]
                    #          #          #           #       #           #
#################################################################################################
# 16 # 3      # -      # DFEMergeChunks       # -
                    # -        # 3750     # 3750      # 1.00  # 0.03      #
#################################################################################################
# 17 # -      # -      # DFEDrain             # -
                    # -        # 3750     # 0         # 0.00  # 0.02      #
#################################################################################################
```

Ejemplo de salida de **explain** de una función de expresión de valores matemática

En este ejemplo, `RETURN abs(-10)` realiza una evaluación sencilla, tomando el valor absoluto de una constante, `-10`.

`DFEChunkLocalSubQuery` (ID 1) realiza una inyección de solución para el valor estático `-10`, que se almacena en la variable `?100`.

`DFEApply` (ID 2) es el operador que ejecuta la función de valor absoluto `abs()` en el valor estático almacenado en la variable `?100`.

Esta es la consulta y la salida de `explain` resultante:

```
curl -d "query=RETURN abs(-10)" -k https://localhost:8182/openCypher  -d
 "explain=details"
                                                    ~
Query:
RETURN abs(-10)

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments             # Mode
  # Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]        # -
  # 0        # 1        # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1    # -
  # 0        # 1        # 0.00  # 4.00      #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?_internalVar1] #
 id2value_opencypher # 1        # 1         # 1.00  # 1.00      #
################################################################################################


subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFESolutionInjection  # outSchema=[]
                                                                 # -    # 0
  # 1         # 0.00  # 0.01      #
################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#c4cc6148-cce3-4561-93c0-deb91f257356/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
################################################################################################
# 2  # 3      # -      # DFEApply             # functor=abs(?100)
                                                                 # -    # 1
  # 1         # 1.00  # 0.26      #
################################################################################################
```

```
# 3  # 4      # -      # DFEBindRelation       # inputVars=[?_internalVar2, ?
_internalVar2]                                                              # -
 # 1        # 1         # 1.00  # 0.04      #
#    #        #        #                       # outputVars=[?_internalVar2, ?
_internalVar1]                                                             #
 #          #          #       #           #
#################################################################################
# 4  # 5      # -      # DFEProject           # columns=[?_internalVar1]
                                                                  # -    # 1
  # 1         # 1.00  # 0.06      #
#################################################################################
# 5  # -      # -      # DFEDrain             # -
                                                                  # -    # 1
  # 0         # 0.00  # 0.05      #
#################################################################################

subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#c4cc6148-
cce3-4561-93c0-deb91f257356/graph_1
#################################################################################
# ID # Out #1 # Out #2 # Name                # Arguments                        #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################
# 0  # 1      # -      # DFESolutionInjection # solutions=[?100 -> [-10^^<LONG>]]   # -
    # 0        # 1         # 0.00  # 0.01      #
#    #        #        #                       # outSchema=[?100]                 #
    #          #          #       #           #
#################################################################################
# 1  # 3      # -      # DFERelationalJoin    # joinVars=[]                      # -
    # 2        # 1         # 0.50  # 0.18      #
#################################################################################
# 2  # 1      # -      # DFESolutionInjection # outSchema=[]                     # -
    # 0        # 1         # 0.00  # 0.01      #
#################################################################################
# 3  # -      # -      # DFEDrain             # -                                # -
    # 1        # 0         # 0.00  # 0.02      #
#################################################################################
```

Ejemplo de la salida de **`explain`** de una consulta de ruta de longitud variable (VLP)

Este es un ejemplo de un plan de consultas más complejo para gestionar una consulta de ruta de longitud variable. Para mayor claridad, en este ejemplo solo se muestra una parte de la salida de `explain`.

En `subQuery1`, `DFEPipelineScan` (ID 0) y `DFEChunkLocalSubQuery` (ID 1), que inyectan la subconsulta `...graph_1`, se encargan de buscar un nodo con el código YPO.

En `subQuery1`, `DFEChunkLocalSubQuery` (ID 2), que inyecta la subconsulta `...graph_2`, se encarga de buscar un nodo con el código LAX.

En `subQuery1`, `DFEChunkLocalSubQuery` (ID 3) inyecta la subconsulta `...graph3`, que contiene `DFELoopSubQuery` (ID 17), que a su vez inyecta la subconsulta `...graph5`. Esta operación es responsable de resolver el patrón de longitud variable `-[*2]->` en la cadena de consulta entre dos nodos.

```
curl -d "query=MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" -k https://
localhost:8182/openCypher -d "explain=details"

                                                  ~
Query:
MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p

################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments          # Mode                #
 Units In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]     # -                   #
 0        # 1         # 0.00  # 0         #
################################################################################################
# 1  # 2      # -      # DFESubquery       # subQuery=subQuery1 # -                   #
 0        # 0         # 0.00  # 84.00     #
################################################################################################
# 2  # -      # -      # TermResolution    # vars=[?p]          # id2value_opencypher #
 0        # 0         # 0.00  # 0         #
################################################################################################


subQuery1
################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode # Units
 In # Units Out # Ratio # Time (ms) #
################################################################################################
# 0  # 1      # -      # DFEPipelineScan       # pattern=Node(?a) with property 'code'
 as ?a_code7 and label 'ALL'                                       # -    # 0
   # 1         # 0.00  # 0.68      #
```

```
#    #        #        #                       # inlineFilters=[(?a_code7 IN
 ["YPO"^^xsd:string])]                                                        #
  #          #           #       #           #
#    #        #        #                       # patternEstimate=1
                                                                    #      #
  #          #       #           #
#################################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_1 # -    #
 1        # 1         # 1.00  # 0.03      #
#################################################################################################
# 2  # 3      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_2 # -    #
 1        # 1         # 1.00  # 0.02      #
#################################################################################################
# 3  # 4      # -      # DFEChunkLocalSubQuery # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_3 # -    #
 1        # 0         # 0.00  # 0.04      #
#################################################################################################
# 4  # 5      # -      # DFEBindRelation       # inputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?__gen_path6]                          # -
   # 0        # 0         # 0.00  # 0.10      #
#    #        #        #                       # outputVars=[?__gen_path6, ?
anon_rel26, ?b_code8, ?b, ?a_code7, ?a, ?p]                                    #
  #          #           #       #           #
#################################################################################################
# 5  # 6      # -      # DFEProject            # columns=[?p]
                                                                    # -    # 0
  # 0         # 0.00  # 0.05      #
#################################################################################################
# 6  # -      # -      # DFEDrain              # -
                                                                    # -    # 0
  # 0         # 0.00  # 0.02      #
#################################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_1
#################################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                    # Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                    # -    # 0        # 1         # 0.00  # 0.01      #
```

```
#################################################################################################
# 1  # 2      # 3      # DFETee               # -
                    # -    # 1        # 2         # 2.00  # 0.01      #
#################################################################################################
# 2  # 4      # -      # DFEDistinctColumn    # column=?a
                    # -    # 1        # 1         # 1.00  # 0.25      #
#    #        #        #                      # ordered=false
                    #      #          #           #       #           #
#################################################################################################
# 3  # 5      # -      # DFEHashIndexBuild    # vars=[?a]
                    # -    # 1        # 1         # 1.00  # 0.05      #
#################################################################################################
# 4  # 5      # -      # DFEPipelineJoin      # pattern=Node(?a) with property 'ALL'
 and label '?a_label1' # -    # 1        # 1         # 1.00  # 0.47      #
#    #        #        #                      # patternEstimate=3506
                    #      #          #           #       #           #
#################################################################################################
# 5  # 6      # 7      # DFESync              # -
                    # -    # 2        # 2         # 1.00  # 0.04      #
#################################################################################################
# 6  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#################################################################################################
# 7  # 8      # -      # DFEForwardValue      # -
                    # -    # 1        # 1         # 1.00  # 0.01      #
#################################################################################################
# 8  # 9      # -      # DFEHashIndexJoin     # -
                    # -    # 2        # 1         # 0.50  # 0.26      #
#################################################################################################
# 9  # -      # -      # DFEDrain             # -
                    # -    # 1        # 0         # 0.00  # 0.02      #
#################################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_2
#################################################################################################
# ID # Out #1 # Out #2 # Name                # Arguments
                           # Mode # Units In # Units Out # Ratio # Time (ms) #
#################################################################################################
# 0  # 1      # -      # DFEPipelineScan      # pattern=Node(?b) with property 'code'
 as ?b_code8 and label 'ALL' # -    # 0        # 1         # 0.00  # 0.38      #
```

```
#    #        #        #                    # inlineFilters=[(?b_code8 IN
 ["LAX"^^xsd:string])]                #      #          #           #       #
  #
#    #        #        #                    # patternEstimate=1
                           #      #          #           #       #           #
#####################################################################################
# 1  # 2      # -      # DFEMergeChunks       # -
                           # -    # 1        # 1         # 1.00  # 0.02      #
#####################################################################################
# 2  # 4      # -      # DFERelationalJoin    # joinVars=[]
                           # -    # 2        # 1         # 0.50  # 0.19      #
#####################################################################################
# 3  # 2      # -      # DFESolutionInjection # outSchema=[?a, ?a_code7]
                           # -    # 0        # 1         # 0.00  # 0         #
#####################################################################################
# 4  # -      # -      # DFEDrain             # -
                           # -    # 1        # 0         # 0.00  # 0.01      #
#####################################################################################


subQuery=http://aws.amazon.com/neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-
bbe3-9e99558eca46/graph_3
#####################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments
                                                                  # Mode     #
 Units In # Units Out # Ratio # Time (ms) #
#####################################################################################
...
# 17 # 18     # -      # DFELoopSubQuery       # subQuery=http://aws.amazon.com/
neptune/vocab/v01/dfe/past/graph#cc05129f-d07e-4622-bbe3-9e99558eca46/graph_5 # -
  # 1        # 2         # 2.00  # 0.31      #
...
```

Transacciones en openCypher de Neptune

La implementación de openCypher en Amazon Neptune utiliza la semántica de transacciones definida por Neptune. Sin embargo, los niveles de aislamiento proporcionados por el controlador Bolt tienen algunas implicaciones específicas para la semántica de las transacciones de Bolt, tal y como se describe en las secciones siguientes.

Consultas de transacciones de Bolt de solo lectura

Hay varias formas de procesar las consultas de solo lectura, con diferentes modelos de transacciones y niveles de aislamiento, como se indica a continuación:

Consultas de transacciones implícitas de solo lectura

A continuación se ofrece un ejemplo de una transacción implícita de solo lectura:

```
public void executeReadImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
          Config.builder().withEncryption()
                          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
                          .build());

  // create the session config
  SessionConfig sessionConfig = SessionConfig.builder()
                                             .withFetchSize(1000)
                                             .withDefaultAccessMode(AccessMode.READ)
                                             .build();

  // run the query as access mode read
  driver.session(sessionConfig).readTransaction(new TransactionWork<String>()
    {
      final StringBuilder resultCollector = new StringBuilder();

      @Override
      public String execute(final Transaction tx)
      {
        // execute the query
        Result queryResult = tx.run(READ_QUERY);

        // Read the result
        for (Record record : queryResult.list())
        {
          for (String key : record.keys())
```

```
          {
            resultCollector.append(key)
                           .append(":")
                           .append(record.get(key).asNode().toString());
          }
        }
        return resultCollector.toString();
      }

    }
  );

  // close the driver.
  driver.close();
}
```

Dado que las réplicas de lectura solo aceptan consultas de solo lectura, todas las consultas realizadas con respecto a réplicas de lectura se ejecutan como transacciones de lectura implícita, independientemente del modo de acceso establecido en la configuración de la sesión. Neptune evalúa las transacciones implícitas de lectura como consultas de solo lectura según la semántica de aislamiento de SNAPSHOT.

En caso de error, las transacciones de lectura implícita se vuelven a intentar de forma predeterminada.

Consultas de transacciones de solo lectura con confirmación automática

A continuación se ofrece un ejemplo de una transacción con confirmación automática de solo lectura:

```
public void executeAutoCommitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();
```

```
  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // run the query
  final Result queryResult = session.run(READ_QUERY);
  for (final Record record : queryResult.list())
  {
    for (String key : record.keys())
    {
      resultCollector.append(key)
                     .append(":")
                     .append(record.get(key).asNode().toString());
    }
  }

  // close the session
  session.close();

  // close the driver
  driver.close();
}
```

Si el modo de acceso está establecido en `READ` en la configuración de la sesión, Neptune evalúa las consultas de transacciones de confirmación automática como consultas de solo lectura en la semántica de aislamiento `SNAPSHOT`. Tenga en cuenta que las réplicas de lectura solo aceptan consultas de solo lectura.

Si no se pasa una configuración de sesión, las consultas con confirmación automática se procesan de forma predeterminada con el aislamiento de las consultas por mutación, por lo que es importante incluir una configuración de sesión que establezca explícitamente el modo de acceso en `READ`.

En caso de error, las consultas con confirmación automática de solo lectura no se vuelven a intentar.

Consultas de transacciones explícitas de solo lectura

A continuación, se muestra un ejemplo de una transacción de solo lectura explícita:

```
public void executeReadExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // read query
  final String READ_QUERY = "MATCH (n) RETURN n limit 10";

  // Create the session config.
  final SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.READ)
    .build();

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // result collector
  final StringBuilder resultCollector = new StringBuilder();

  // create a session
  final Session session = driver.session(sessionConfig);

  // begin transaction
  final Transaction tx = session.beginTransaction();

  // run the query on transaction
  final List<Record> list = tx.run(READ_QUERY).list();

  // read the result
  for (final Record record : list)
  {
    for (String key : record.keys())
    {
      resultCollector
```

```
        .append(key)
        .append(":")
        .append(record.get(key).asNode().toString());
    }
  }

  // commit the transaction and for rollback we can use beginTransaction.rollback();
  tx.commit();

  // close the driver
  driver.close();
}
```

Si el modo de acceso está establecido en READ en la configuración de la sesión, Neptune evalúa las transacciones de solo lectura explícitas como consultas de solo lectura en la semántica de aislamiento SNAPSHOT. Tenga en cuenta que las réplicas de lectura solo aceptan consultas de solo lectura.

Si no se pasa una configuración de sesión, las transacciones de solo lectura explícitas se procesan de forma predeterminada con el aislamiento de las consultas por mutación, por lo que es importante incluir una configuración de sesión que establezca explícitamente el modo de acceso en READ.

En caso de error, las consultas explícitas de solo lectura se vuelven a intentar de forma predeterminada.

Consultas de transacciones de Bolt de mutación

Al igual que con las consultas de solo lectura, hay varias formas de procesar las consultas de mutación, con diferentes modelos de transacciones y niveles de aislamiento, como se indica a continuación:

Consultas de transacciones de mutación implícita

A continuación, se muestra un ejemplo de una transacción de mutación implícita:

```
public void executeWriteImplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // create node with label as label and properties.
  final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";
```

```
// Read the vertex created with label as label.
final String READ_QUERY = "MATCH (n:label) RETURN n";

// create the driver
final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
  Config.builder()
        .withEncryption()
        .withTrustStrategy(TrustStrategy.trustSystemCertificates())
        .build());

// create the session config
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();

final StringBuilder resultCollector = new StringBuilder();

// run the query as access mode write
driver.session(sessionConfig).writeTransaction(new TransactionWork<String>()
{
  @Override
  public String execute(final Transaction tx)
  {
    // execute the write query and consume the result.
    tx.run(WRITE_QUERY).consume();

    // read the vertex written in the same transaction
    final List<Record> list = tx.run(READ_QUERY).list();

    // read the result
    for (final Record record : list)
    {
      for (String key : record.keys())
      {
        resultCollector
          .append(key)
          .append(":")
          .append(record.get(key).asNode().toString());
      }
    }
    return resultCollector.toString();
  }
```

```
  }); // at the end, the transaction is automatically committed.

  // close the driver.
  driver.close();
}
```

Las lecturas realizadas como parte de las consultas de mutación se ejecutan en el aislamiento `READ COMMITTED` con las garantías habituales para las transacciones de mutación de Neptune.

Independientemente de si se pasa específicamente una configuración de sesión o no, la transacción siempre se trata como una transacción de escritura.

Para conocer los conflictos, consulte Resolución de conflictos mediante tiempos de espera de bloqueo.

Consultas de transacciones de mutación con confirmación automática

Las consultas con confirmación automática de mutación heredan el mismo comportamiento que las transacciones implícitas de mutación.

Si no se pasa una configuración de sesión, la transacción se trata de forma predeterminada como una transacción de escritura.

En caso de error, las consultas con confirmación automática de mutación no se vuelven a intentar automáticamente.

Consultas de transacciones de mutación explícita

A continuación, se muestra un ejemplo de una transacción de mutación explícita:

```
public void executeWriteExplicitTransaction()
{
  // end point
  final String END_POINT = "(End Point URL)";

  // create node with label as label and properties.
  final String WRITE_QUERY = "CREATE (n:label {name : 'foo'})";

  // Read the vertex created with label as label.
  final String READ_QUERY = "MATCH (n:label) RETURN n";

  // create the driver
  final Driver driver = GraphDatabase.driver(END_POINT, AuthTokens.none(),
```

```
    Config.builder()
          .withEncryption()
          .withTrustStrategy(TrustStrategy.trustSystemCertificates())
          .build());

  // create the session config
  SessionConfig sessionConfig = SessionConfig
    .builder()
    .withFetchSize(1000)
    .withDefaultAccessMode(AccessMode.WRITE)
    .build();

  final StringBuilder resultCollector = new StringBuilder();

  final Session session = driver.session(sessionConfig);

  // run the query as access mode write
  final Transaction tx = driver.session(sessionConfig).beginTransaction();

  // execute the write query and consume the result.
  tx.run(WRITE_QUERY).consume();

  // read the result from the previous write query in a same transaction.
  final List<Record> list = tx.run(READ_QUERY).list();

  // read the result
  for (final Record record : list)
  {
    for (String key : record.keys())
    {
      resultCollector
        .append(key)
        .append(":")
        .append(record.get(key).asNode().toString());
    }
  }

  // commit the transaction and for rollback we can use tx.rollback();
  tx.commit();

  // close the session
  session.close();

  // close the driver.
```

```
  driver.close();
}
```

Las consultas de mutación explícita heredan el mismo comportamiento que las transacciones de mutación implícita.

Si no se pasa una configuración de sesión, la transacción se trata de forma predeterminada como una transacción de escritura.

Para conocer los conflictos, consulte Resolución de conflictos mediante tiempos de espera de bloqueo.

Restricciones de openCypher de Neptune

La versión de openCypher para Amazon Neptune aún no admite todo lo que se especifica en la referencia del lenguaje de consulta de Cypher, versión 9, como se detalla en Conformidad con las especificaciones de OpenCypher. Se espera que en las versiones futuras aborden muchas de esas limitaciones.

Excepciones de openCypher de Neptune

Al trabajar con openCypher en Amazon Neptune, pueden producirse diversas excepciones. A continuación, se detallan las excepciones más comunes que puede recibir, ya sea del punto de conexión de HTTPS o del controlador de Bolt (todas las excepciones del controlador de Bolt se registran como excepciones del estado del servidor):

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
400	(error de sintaxis, propagado directame nte desde el analizador de openCypher)	No	Corrija la sintaxis de la consulta y vuelva a intentarl o.	
500	`Operation terminate`	Sí	Vuelva a crear la consulta para	

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
	`d (out of memory)`		añadir criterios de filtrado adicionales con el fin de reducir la memoria requerida	
500	La operación ha finalizado (se ha superado el plazo)	Sí	Aumente el tiempo de espera de la consulta en el grupo de parámetros del clúster de base de datos o vuelva a intentar la solicitud.	
500	La operación ha finalizado (el usuario la ha cancelado)	Sí	Intente realizar de nuevo la solicitud .	
500	El restablec imiento de la base de datos está en curso. Vuelva a intentar la consulta cuando el clúster esté disponible.	Sí	Vuelva a intentarlo cuando se haya completado el restablecimiento.	

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
500	La operación ha fallado debido a operaciones simultáneas conflictivas (vuelva a intentarlo). En estos momentos, las transacciones se están revirtiendo.	Sí	Vuelva a intentarlo con una estrategia de retroceso exponencial y reintento.	
400	Excepción debida a que la operación/característica *(nombre de la operación)* no se admite	No	La operación especificada no es compatible.	
400	Se intentó una actualización de openCypher en una réplica de solo lectura	No	Cambie el punto final de destino por el punto final del escritor.	
400	MalformedQueryException (Neptune no muestra el estado interno del analizador)	No	Corrija la sintaxis de la consulta y vuelva a intentarlo.	

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
400	No se puede eliminar el nodo porque todavía tiene relaciones. Para eliminar este nodo, primero debe eliminar sus relaciones.	No	En lugar de usar, `MATCH (n) DELETE n` utilice `MATCH(n) DETACH DELETE(n)`	

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
400	Operación no válida: intento de eliminar la última etiqueta de un nodo. Un nodo debe tener al menos una etiqueta.	No	Neptune requiere que todos los nodos tengan al menos una etiqueta y, si los nodos se crean sin una etiqueta explícita, se asigna una etiqueta predeterminada `vertex`. Cambie la lógica de la consulta y/o aplicación para no eliminar la última etiqueta. La etiqueta singleton de un nodo se puede actualizar configurando una nueva etiqueta y, a continuación, quitando la antigua.	

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
500	Se ha superado el número máximo de solicitudes, Configure dQueueCapacity = {} para ConnID = {}	Sí	Actualmente, solo se pueden procesar 8192 solicitudes simultáneas, independientemente de la pila y el protocolo.	
500	Se ha superado el límite máximo de conexiones.	Sí	Solo se permiten 1000 conexiones de Bolt simultáneas por instancia (para HTTP no hay límite).	
400	Se esperaba [uno de los siguientes: nodo, relación o ruta] y se obtuvo un literal	No	Compruebe que está pasando los argumentos correctos y que la sintaxis de consulta es correcta, y vuelva a intentarlo.	

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
400	El valor de la propiedad debe ser un literal simple. O bien: se esperaba un mapa para las propiedades del conjunto, pero no se encontró ninguno.	No	Una cláusula SET solo acepta literales simples, no tipos compuestos.	
400	se encuentra la entidad pasada para su eliminación	No	Compruebe que la entidad que está intentando eliminar existe en la base de datos.	
400	El usuario no tiene acceso a la base de datos.	No	Compruebe la política en el rol de IAM que se está utilizando.	
400	No se ha pasado ningún token como parte de la solicitud	No	Se debe pasar un token debidamente firmado como parte de la solicitud de consulta en un clúster habilitado para IAM.	

Código de HTTP	Mensaje de error	¿Se puede volver a intentar?	Solución	
400	El mensaje de error se propaga.	No	Póngase en contacto con AWS Support con el identific ador de la solicitud.	
500	La operación ha finalizado (error interno)	Sí	Póngase en contacto con AWS Support con el identific ador de la solicitud.	

Acceso al gráfico de Neptune con SPARQL

SPARQL es un lenguaje de consulta para el marco de descripción de recursos (RDF), que es un formato de datos de gráficos diseñado para la web. Amazon Neptune es compatible con SPARQL 1.1. Esto significa que puede conectarse a una instancia de base de datos de Neptune y consultar el gráfico utilizando el lenguaje de consulta descrito en la especificación de SPARQL 1.1 Query Language.

Una consulta en SPARQL se compone de una cláusula `SELECT` para especificar las variables que se devolverán y una cláusula `WHERE` para especificar los datos del gráfico que deben corresponderse. Si no está familiarizado con las consultas SPARQL, consulte la sección sobre escritura de consultas sencillas en la documentación del lenguaje de consulta SPARQL 1.1.

> **Important**
>
> Para cargar datos, `SPARQL UPDATE INSERT` puede funcionar bien para un conjunto de datos pequeño, pero si necesita cargar una cantidad considerable de datos de un archivo, consulte Uso del programa de carga masiva de Amazon Neptune para adquirir datos.

Para obtener más información sobre la información específica de la implementación de SPARQL de Neptune, consulte Conformidad con los estándares de SPARQL.

Antes de comenzar, debe disponer de lo siguiente:

- Una instancia de base de datos de Neptune. Para obtener información acerca de la creación de una instancia de base de datos de Neptune, consulte Creación de un nuevo clúster de base de datos de Neptune.
- Una instancia de Amazon EC2 en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Temas

Uso de la consola de RDF4J para conectarse a una instancia de base de datos de Neptune

La consola RDF4J le permite experimentar con gráficos y consultas del Resource Description Framework (RDF) en un entorno REPL (bucle). read-eval-print

Puede añadir una base de datos de gráficos remota como repositorio y consultarla desde la consola de RDF4J. En esta sección, se explica la configuración de la consola de RDF4J para conectarse de forma remota a una instancia de base de datos de Neptune.

Para conectarse a Neptune mediante la consola de RDF4J

1. Descargue el SDK de RDF4J de la página de descargas del sitio web de RDF4J.
2. Descomprima el archivo .zip del SDK de RDF4J.
3. En un terminal, vaya al directorio del SDK de RDF4J y, a continuación, escriba el siguiente comando para ejecutar la consola de RDF4J:

   ```
   bin/console.sh
   ```

   Debería ver una salida similar a esta:

   ```
   14:11:51.126 [main] DEBUG o.e.r.c.platform.PlatformFactory - os.name = linux
   14:11:51.130 [main] DEBUG o.e.r.c.platform.PlatformFactory - Detected Posix
    platform
   Connected to default data directory
   RDF4J Console 3.6.1

   3.6.1
   Type 'help' for help.
   >
   ```

   Ahora se encuentra en >. Este es el símbolo general para la consola de RDF4J, que se utiliza para configurar los repositorios y otras operaciones. Un repositorio tiene su propio símbolo para ejecutar consultas.
4. En el símbolo del sistema de >, escriba lo siguiente si quiere crear un repositorio de SPARQL para la instancia de base de datos de Neptune:

   ```
   create sparql
   ```
5. La consola de RDF4J solicita los valores de las variables que se requieren para conectarse al punto de enlace de SPARQL.

   ```
   Please specify values for the following variables:
   ```

   Especifique los siguientes valores:

Nombre de variable	Valor

Punto de enlace de consulta de SPARQL	`https://your-neptune-endpoint :port/sparql`
Punto de enlace de actualización de SPARQL	`https://your-neptune-endpoint :port/sparql`
ID del repositorio local [endpoint@localhost]	`neptune`
Título del repositorio [repositorio de punto de enlace de SPARQL @localhost]	`Neptune DB instance`

Para obtener información acerca de cómo encontrar la dirección de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

Si la operación se realiza correctamente, aparecerá el mensaje siguiente:

```
Repository created
```

6. En el símbolo del sistema de >, escriba lo siguiente para conectarse a la instancia de base de datos de Neptune:

```
open neptune
```

Si la operación se realiza correctamente, aparecerá el mensaje siguiente:

```
Opened repository 'neptune'
```

Ahora se encuentra en `neptune>`. Aquí puede ejecutar consultas en el gráfico de Neptune.

> **Note**
>
> Una vez que ha añadido el repositorio, la próxima vez que ejecute `bin/console.sh`, podrá ejecutar inmediatamente el comando `open neptune` para conectarse a la instancia de base de datos de Neptune.

7. En el símbolo del sistema `neptune>`, escriba lo siguiente para ejecutar una consulta SPARQL que devuelva hasta un máximo de 10 valores triples (sujeto-predicado-objeto) en el gráfico utilizando la consulta `?s ?p ?o` con un límite de 10. Para otras consultas, sustituya el texto después del comando `sparql` por otra consulta SPARQL.

```
sparql select ?s ?p ?o where {?s ?p ?o} limit 10
```

Uso de RDF4J Workbench para conectarse a una instancia de base de datos de Neptune

En esta sección se explica cómo conectarse a una instancia de base de datos de Amazon Neptune utilizando RDF4J Workbench y RDF4J Server. RDF4J Server es obligatorio porque actúa como proxy entre el punto de conexión HTTP REST de SPARQL para Neptune y RDF4J Workbench.

RDF4J Workbench proporciona una interfaz sencilla para experimentar con un gráfico, incluida la carga de archivos locales. Para obtener más información, consulte la sección Add en la documentación de RDF4J.

Requisitos previos

Antes de comenzar, haga lo siguiente:

- Instale Java 1.8 o versiones posteriores.
- Instale RDF4J Server y RDF4J Workbench. Para obtener más información, consulte la sección sobre cómo instalar RDF4J Server y RDF4J Workbench.

Para utilizar RDF4J Workbench para conectarse a Neptune

1. En un explorador web, vaya a la dirección URL donde está implementada la aplicación web de RDF4J Workbench. Por ejemplo, si utiliza Apache Tomcat, la dirección URL es: https://*ec2_hostname*:8080/rdf4j-workbench/.
2. Si se le pide que realice la acción Connect to RDF4J Server (Conectar al servidor RDF4J), verifique que se ha instalado RDF4J Server (Servidor RDF4J), que está en ejecución y que la URL del servidor es correcta. Después, continúe con el siguiente paso.
3. En el panel izquierdo, elija New repository (Nuevo repositorio).

   En New repository (Nuevo repositorio):

- En la lista desplegable Type (Tipo), elija SPARQL endpoint proxy (Proxy de punto de enlace de SPARQL).
- En ID, escriba neptune.
- En Título, escriba Instancia de base de datos de Neptune.

Elija Siguiente.

4. En New repository (Nuevo repositorio):

   - En SPARQL query endpoint URL (URL de punto de enlace de consulta de SPARQL), escriba `https://your-neptune-endpoint:port/sparql`.
   - En SPARQL update endpoint URL (URL de punto de enlace de actualización de SPARQL), escriba `https://your-neptune-endpoint:port/sparql`.

   Para obtener información acerca de cómo encontrar la dirección de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

   Seleccione Crear.

5. El repositorio de neptune aparece ahora en la lista de repositorios. Puede que tenga que esperar unos minutos antes de poder usar el nuevo repositorio.
6. En la columna Id de la tabla, elija el enlace neptune.
7. En el panel izquierdo, elija Query (Consulta).

> **Note**
>
> Si los elementos del menú Explore (Explorar) están deshabilitados, puede que tenga que volver a conectar con RDF4J Server y elegir de nuevo el repositorio neptune.
> Para ello, puede usar los enlaces [change] ([cambiar]) en la esquina superior derecha.

8. En el campo de consulta, escriba la siguiente consulta SPARQL y, a continuación, elija Execute (Ejecutar).

```
select ?s ?p ?o where {?s ?p ?o} limit 10
```

El ejemplo anterior devuelve hasta 10 de los triples (sujeto-predicado-objeto) del gráfico utilizando la consulta `?s ?p ?o` con un límite de 10.

Uso de Java para conectarse a una instancia de base de datos de Neptune

En esta sección, se indica cómo ejecutar un ejemplo de Java completo que se conecta a una instancia de base de datos de Amazon Neptune y realiza una consulta SPARQL.

Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

Para conectarse a Neptune mediante Java

1. Instale Apache Maven en la instancia EC2. En primer lugar, escriba lo siguiente para añadir un repositorio con un paquete de Maven:

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   Escriba lo siguiente para establecer el número de versión de los paquetes:

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   A continuación, puede usar el comando yum para instalar Maven:

   ```
   sudo yum install -y apache-maven
   ```

2. Este ejemplo solo se ha probado con Java 8. Escriba lo siguiente para instalar Java 8 en la instancia EC2:

   ```
   sudo yum install java-1.8.0-devel
   ```

3. Escriba lo siguiente para establecer Java 8 como tiempo de ejecución predeterminado en la instancia EC2:

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   Cuando se le solicite, escriba el número para Java 8.

4. Escriba lo siguiente para establecer Java 8 como compilador predeterminado en la instancia EC2:

```
sudo /usr/sbin/alternatives --config javac
```

   Cuando se le solicite, escriba el número para Java 8.

5. En un directorio nuevo, cree un archivo `pom.xml` y, a continuación, ábralo en un editor de texto.

6. Copie lo siguiente en el archivo `pom.xml` y guárdelo (normalmente puede ajustar los números de versión a la última versión estable):

```
<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amazonaws</groupId>
  <artifactId>RDFExample</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>RDFExample</name>
  <url>https://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>org.eclipse.rdf4j</groupId>
      <artifactId>rdf4j-runtime</artifactId>
      <version>3.6</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.2.1</version>
          <configuration>
            <mainClass>com.amazonaws.App</mainClass>
          </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Si está modificando un proyecto de Maven ya existente, la dependencia requerida aparece resaltada en el código anterior.

7. Escriba lo siguiente en la línea de comandos a fin de crear subdirectorios para el código fuente de ejemplo (`src/main/java/com/amazonaws/`):

```
mkdir -p src/main/java/com/amazonaws/
```

8. En el directorio `src/main/java/com/amazonaws/`, cree un archivo llamado `App.java` y, a continuación, ábralo en un editor de texto.
9. Copie lo siguiente en el archivo `App.java`. Sustituya *`your-neptune-endpoint`* por la dirección de su instancia de base de datos de Neptune.

> **Note**
>
> Para obtener información acerca de cómo encontrar el nombre de host de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

```
package com.amazonaws;

import org.eclipse.rdf4j.repository.Repository;
import org.eclipse.rdf4j.repository.http.HTTPRepository;
import org.eclipse.rdf4j.repository.sparql.SPARQLRepository;

import java.util.List;
import org.eclipse.rdf4j.RDF4JException;
```

```
import org.eclipse.rdf4j.repository.RepositoryConnection;
import org.eclipse.rdf4j.query.TupleQuery;
import org.eclipse.rdf4j.query.TupleQueryResult;
import org.eclipse.rdf4j.query.BindingSet;
import org.eclipse.rdf4j.query.QueryLanguage;
import org.eclipse.rdf4j.model.Value;

public class App
{
    public static void main( String[] args )
    {
        String sparqlEndpoint = "https://your-neptune-endpoint:port/sparql";
        Repository repo = new SPARQLRepository(sparqlEndpoint);
        repo.initialize();

        try (RepositoryConnection conn = repo.getConnection()) {
           String queryString = "SELECT ?s ?p ?o WHERE { ?s ?p ?o } limit 10";

           TupleQuery tupleQuery = conn.prepareTupleQuery(QueryLanguage.SPARQL,
 queryString);

           try (TupleQueryResult result = tupleQuery.evaluate()) {
              while (result.hasNext()) {  // iterate over the result
                   BindingSet bindingSet = result.next();

                   Value s = bindingSet.getValue("s");
                   Value p = bindingSet.getValue("p");
                   Value o = bindingSet.getValue("o");

                   System.out.print(s);
                   System.out.print("\t");
                   System.out.print(p);
                   System.out.print("\t");
                   System.out.println(o);
              }
           }
        }
    }
}
```

10. Use el siguiente comando Maven para compilar y ejecutar el ejemplo:

```
mvn compile exec:java
```

El ejemplo anterior devuelve hasta 10 de los triples (sujeto-predicado-objeto) del gráfico utilizando la consulta `?s ?p ?o` con un límite de 10. Para otras consultas, sustitúyala por otra consulta SPARQL.

La iteración de los resultados en el ejemplo imprime el valor de cada variable devuelta. El objeto `Value` se convierte en `String` y después se imprime. Si cambia la parte `SELECT` de la consulta, debe modificar el código.

API HTTP de SPARQL

Las solicitudes HTTP de SPARQL se aceptan en el siguiente punto de enlace: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`

Para obtener más información acerca de la conexión a Amazon Neptune con SPARQL, consulte Acceso al gráfico de Neptune con SPARQL.

Para obtener más información acerca del lenguaje de consulta y el protocolo SPARQL, consulte la especificación sobre el protocolo SPARQL 1.1 y el lenguaje de consulta SPARQL 1.1.

En los siguientes temas, se proporciona información acerca de los formatos de serialización RDF y SPARQL y cómo utilizar la API HTTP de SPARQL con Neptune.

Contenido

Uso del punto de conexión HTTP REST para conectarse a una instancia de base de datos de Neptune

Amazon Neptune proporciona un punto de conexión HTTP para las consultas SPARQL. La interfaz de REST es compatible con SPARQL versión 1.1.

> **Important**
>
> En Versión: 1.0.4.0 (12/10/2020), se hizo que TLS 1.2 y HTTPS fueran obligatorios para todas las conexiones a Amazon Neptune. Ya no es posible conectarse a Neptune mediante HTTP no seguro o mediante HTTPS con una versión de TLS anterior a la 1.2.

Las siguientes instrucciones le ayudarán a conectar con el punto de enlace de SPARQL mediante el comando curl a través de HTTPS y mediante la sintaxis HTTP. Siga estas instrucciones desde una instancia de Amazon EC2 que esté en la misma nube privada virtual (VPC) que su instancia de base de datos de Neptune.

El punto de conexión HTTP para las consultas de SPARQL a una instancia de base de datos de Neptune es: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

> **Note**
>
> Para obtener información acerca de cómo encontrar el nombre de host de la instancia de base de datos de Neptune, consulte la sección Conexión a los puntos de conexión de Amazon Neptune.

QUERY con HTTP POST

En el siguiente ejemplo se utiliza curl para enviar una **`QUERY`** de SPARQL a través de HTTP POST.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

El ejemplo anterior devuelve hasta 10 de los triples (sujeto-predicado-objeto) del gráfico utilizando la consulta `?s ?p ?o` con un límite de 10. Para otras consultas, sustitúyalo por otra consulta SPARQL.

> **Note**
>
> El tipo de medio MIME predeterminado de una respuesta es `application/sparql-results+json` para las consultas `SELECT` y `ASK`.
> El tipo MIME predeterminado de una respuesta es `application/n-quads` para las consultas `CONSTRUCT` y `DESCRIBE`.
> Para obtener una lista de los tipos de medios que utiliza Neptune para la serialización, consulte Formatos de serialización de RDF utilizados por SPARQL para Neptune.

UPDATE con HTTP POST

En el siguiente ejemplo se utiliza curl para enviar una **`UPDATE`** de SPARQL a través de HTTP POST.

```
curl -X POST --data-binary 'update=INSERT DATA { <https://test.com/s> <https://
test.com/p> <https://test.com/o> . }' https://your-neptune-endpoint:port/sparql
```

El ejemplo anterior inserta el siguiente triple en el gráfico predeterminado de SPARQL: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

Encabezados finales HTTP opcionales para las respuestas de SPARQL de varias partes

> **Note**
>
> Esta característica está disponible a partir de la versión 1.0.3.0 del motor de Neptune.

La respuesta HTTP a las consultas y actualizaciones de SPARQL suele devolverse en más de una parte o en fragmentos. Puede resultar difícil diagnosticar un error que se produce después de que una consulta o actualización comience a enviar estos fragmentos, especialmente si se tiene en cuenta que el primero llega con un código de estado HTTP de `200`.

A menos que solicite explícitamente los encabezados finales, Neptune solo informa de este error añadiendo un mensaje de error al cuerpo del mensaje, que suele estar dañado.

Para facilitar la detección y el diagnóstico de este tipo de problemas, puede incluir un encabezado final de codificación por transferencia (TE) (`te: trailers`) en su solicitud (consulte, por ejemplo,

la página de MDN sobre los encabezados de solicitudes de TE). Si lo hace, Neptune incluirá dos nuevos campos de encabezados dentro de los encabezados finales de los fragmentos de respuesta:

- `X-Neptune-Status`: contiene el código de respuesta seguido de un nombre abreviado. Por ejemplo, en caso de que se realizara correctamente, el encabezado final sería: `X-Neptune-Status: 200 OK`. En caso de fallo, el código de respuesta sería uno de los códigos de error del motor de Neptune, como `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail`: está vacío si las solicitudes se han realizado correctamente. En caso de errores, contiene el mensaje de error JSON. Como solo se permiten caracteres ASCII en los valores de los encabezados HTTP, la cadena JSON está codificada en URL. El mensaje de error también se sigue adjuntando al cuerpo del mensaje de respuesta.

Tipos de medios de RDF utilizados por SPARQL en Neptune

Los datos del marco de descripción de recursos (RDF) se pueden serializar de varias maneras diferentes, pudiendo SPARQL producir o consumir la mayoría:

Formatos de serialización de RDF utilizados por SPARQL para Neptune

- RDF/XML: serialización de XML de RDF, definida en RDF 1.1 XML Syntax. Tipo de medios: `application/rdf+xml`. Extensión de archivo típica: `.rdf`.
- N-Triples: un formato basado en líneas no cifrado para codificar un gráfico de RDF, definido en RDF 1.1 N-Triples. Tipo de medios: `application/n-triples`, `text/turtle` o `text/plain`. Extensión de archivo típica: `.nt`.
- N-Quads: un formato basado en líneas no cifrado para codificar un gráfico de RDF, definido en RDF 1.1 N-Quads. Es una extensión de N-Triples. Tipo de medios: `application/n-quads` o `text/x-nquads` cuando se codifique con US-ASCII de 7 bits. Extensión de archivo típica: `.nq`.
- Turtle: una sintaxis textual para RDF definida en RDF 1.1 Turtle que permite que un gráfico RDF se escriba por completo en un formato de texto compacto y natural, con abreviaturas para los patrones de uso y tipos de datos comunes. Turtle proporciona niveles de compatibilidad con el formato N-Triples además de con la sintaxis del patrón triple de SPARQL. Tipo de medios: `text/turtle` Extensión de archivo típica: `.ttl`.
- TriG: una sintaxis textual para RDF definida en RDF 1.1 TriG que permite que un gráfico RDF se escriba por completo en un formato de texto compacto y natural, con abreviaturas para los patrones de uso y tipos de datos comunes. TriG es una extensión del formato Turtle. Tipo de medios: `application/trig`. Extensión de archivo típica: `.trig`.

- N3 (Notation3: un lenguaje de lógica y de aserción definido en Notation3 (N3): A readable RDF syntax. N3 amplía el modelo de datos de RDF al añadir fórmulas (literales que son gráficos en sí mismos), variables, implicación lógica, predicados funcionales y ofrece una sintaxis textual alternativa a RDF/XML. Tipo de medios: `text/n3`. Extensión de archivo típica: `.n3`.
- JSON-LD: una serialización de datos y un formato de mensajería definido en JSON-LD 1.0. Tipo de medios: `application/ld+json`. Extensión de archivo típica: `.jsonld`.
- TriX: una serialización de RDF en XML, definida en TriX: RDF Triples in XML. Tipo de medios: `application/trix`. Extensión de archivo típica: `.trix`.
- Resultados de JSON para SPARQL: una serialización de RDF utilizando SPARQL 1.1 Query Results JSON Format. Tipo de medios: `application/sparql-results+json`. Extensión de archivo típica: `.srj`.
- Formato binario para RDF4J: un formato binario para codificar datos de RDF, documentado en RDF4J Binary RDF Format. Tipo de medios: `application/x-binary-rdf`.

Formatos de serialización de resultados de SPARQL utilizados en Neptune SPARQL

- Resultados de XML para SPARQL: un formato XML para los formatos de resultados booleanos y vinculantes variables ofrecidos por el lenguaje de consulta SPARQL, definido en SPARQL Query Results XML Format (Second Edition). Tipo de medios: `application/sparql-results+xml`. Extensión de archivo típica: `.srx`.
- Resultados de CSV y TSV para SPARQL: el uso de los valores separados por comas y los valores separados por tabulaciones para expresar los resultados de la consulta de SPARQL de las consultas de `SELECT`, definidas en SPARQL 1.1 Query Results CSV and TSV Formats. Tipo de medios: `text/csv` para valores separados por comas y `text/tab-separated-values` para valores separados por tabulaciones. Extensiones de archivo típicas: `.csv` para los valores separados por comas y `.tsv` para los valores separados por tabulaciones.
- Tabla de resultados binarios: un formato binario para codificar la salida de las consultas SPARQL. Tipo de medios: `application/x-binary-rdf-results-table`.
- Resultados de JSON para SPARQL: una serialización de RDF utilizando SPARQL 1.1 Query Results JSON Format. Tipo de medios: `application/sparql-results+json`.

Tipos de medios que Neptune puede utilizar para importar datos de RDF

Tipos de medios que admite el programa de carga masivo de Neptune

- N-Triples
- N-Quads
- RDF/XML
- Turtle

Tipos de medios que puede importar SPARQL UPDATE LOAD

- N-Triples
- N-Quads
- RDF/XML
- Turtle
- TriG
- N3
- JSON-LD

Tipos de medios que Neptune puede utilizar para exportar los resultados de las consultas

Para especificar el formato de salida para la respuesta de una consulta SPARQL, envíe un encabezado"`Accept: media-type`" con la respuesta de la consulta. Por ejemplo:

```
curl -H "Accept: application/nquads" ...
```

Tipos de medios de RDF que SPARQL SELECT puede producir desde Neptune

- Resultados de JSON para SPARQL (Esta es la opción predeterminada)
- Resultados de XML para SPARQL
- Tabla de resultados binarios (tipo de medios: `application/x-binary-rdf-results-table`)
- Valores separados por comas (CSV)
- TSV (valores separados por tabulaciones)

Tipos de medios de RDF que SPARQL ASK puede producir desde Neptune

- Resultados de JSON para SPARQL (Esta es la opción predeterminada)
- Resultados de XML para SPARQL
- Booleano (tipo de medios: `text/boolean`, significa “verdadero” o “falso”)

Tipos de medios de RDF que SPARQL CONSTRUCT puede producir desde Neptune

- N-Quads (Esta es la opción predeterminada)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- Resultados de JSON para SPARQL
- Formato RDF binario RDF4J

Tipos de medios de RDF que SPARQL DESCRIBE puede producir desde Neptune

- N-Quads (Esta es la opción predeterminada)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- Resultados de JSON para SPARQL
- Formato RDF binario RDF4J

Uso de SPARQL UPDATE LOAD para importar datos a Neptune

La sintaxis del comando UPDATE LOAD de SPARQL se especifica en la recomendación de actualización de SPARQL 1.1:

```
LOAD SILENT (URL of data to be loaded) INTO GRAPH (named graph into which to load the
 data)
```

- **`SILENT`**: (opcional) hace que la operación se realice correctamente aunque se haya producido un error durante el procesamiento.

  Puede resultar útil cuando una sola transacción contiene varias instrucciones, como "`LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;`", y si desea que la transacción se complete aunque algunos de los datos remotos no se hayan podido procesar.
- *`URL de los datos que se van a cargar`*: (obligatorio) especifica un archivo de datos remoto que contiene los datos que se van a cargar en un gráfico.

  El archivo remoto debe tener una de las siguientes extensiones:

  - `.nt` para NTriples.
  - `.nq` para NQuads.
  - `.trig` para Trig.
  - `.rdf` para RDF/XML.
  - `.ttl` para Turtle.
  - `.n3` para N3.
  - `.jsonld` para JSON-LD.
- **`INTO GRAPH`***`(gráfico con nombre en el que cargar los datos)`*: (opcional) especifica el gráfico en el que se deben cargar los datos.

  Neptune asocia cada triple con un gráfico con nombre. Puede especificar el gráfico con nombre predeterminado utilizando el URI de gráfico con nombre de reserva, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, de la siguiente manera:

  ```
  INTO GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

> **Note**
>
> Cuando necesite cargar muchos datos, le recomendamos que utilice el programa de carga masiva de Neptune en lugar de UPDATE LOAD. Para obtener más información acerca del programa de carga masiva, consulte Uso del programa de carga masiva de Amazon Neptune para adquirir datos.

Puede utilizar `SPARQL UPDATE LOAD` para cargar datos directamente desde Amazon S3 o desde archivos obtenidos de un servidor web con alojamiento propio. Los recursos que se cargarán deben residir en la misma región que el servidor de Neptune y el punto de conexión de los recursos debe estar permitido en la VPC. Para obtener información sobre cómo crear un punto de conexión de Amazon S3, consulte Creación de un punto de conexión de VPC de Amazon S3.

Todos los URI de `SPARQL UPDATE LOAD` deben empezar por `https://`. Esto incluye URL de Amazon S3.

En contraste con el programa de carga masiva de Neptune, una llamada a `SPARQL UPDATE LOAD` es completamente transaccional.

Carga de archivos directamente desde Amazon S3 en Neptune con SPARQL UPDATE LOAD

Dado que Neptune no le permite pasar un rol de IAM a Amazon S3 al usar SPARQL UPDATE LOAD, el bucket de Amazon S3 en cuestión debe ser público o debe usar una URL prefirmada de Amazon S3 en la consulta LOAD.

Para generar una URL prefirmada para un archivo de Amazon S3, puede utilizar un AWS CLI comando como este:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to load)
```

A continuación, puede utilizar la URL prefirmada resultante en su comando de la `LOAD`:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=load (pre-signed URL of the remote Amazon S3 file of data to
 be loaded) \
                           into graph (named graph)'
```

Para obtener más información, consulte la sección sobre autenticación de solicitudes: uso de parámetros de consulta. La documentación de Boto3 muestra cómo usar un script de Python para generar una URL prefirmada.

además, el tipo de contenido de los archivos se cargarán se debe configurar correctamente.

1. Establezca el tipo de contenido de los archivos cuando los cargue en Amazon S3 con el parámetro `-metadata` de esta manera:

   ```
   aws s3 cp test.nt s3://bucket-name/my-plain-text-input/test.nt --metadata Content-
   Type=text/plain
   aws s3 cp test.rdf s3://bucket-name/my-rdf-input/test.rdf --metadata Content-
   Type=application/rdf+xml
   ```

2. Confirme que la información del tipo de medios está presente. Ejecute:

   ```
   curl -v bucket-name/folder-name
   ```

   El resultado de este comando debería mostrar la información del tipo de medios que configura cuando carga los archivos.

3. A continuación puede utilizar el comando de la `SPARQL UPDATE LOAD` para importar estos archivos a Neptune:

   ```
   curl https://your-neptune-endpoint:port/sparql \
     -d "update=LOAD <https://s3.amazonaws.com/bucket-name/my-rdf-input/test.rdf>"
   ```

Los pasos anteriores solo funcionan para un bucket de Amazon S3 público o para un bucket al que se accede mediante una URL de Amazon S3 prefirmada en la consulta LOAD.

También puede configurar un servidor proxy web para cargarlo desde un bucket de Amazon S3 privado, como se muestra a continuación:

Uso de un servidor web para cargar archivos en Neptune con SPARQL UPDATE LOAD

1. Instale un servidor web en una maquina que se ejecute en la VPC que hospeda Neptune y los archivos que se cargarán. Por ejemplo, si utiliza Amazon Linux, puede instalar Apache de la siguiente manera:

   ```
   sudo yum install httpd mod_ssl
   ```

```
sudo /usr/sbin/apachectl start
```

2. Defina los tipos MIME del contenido del archivo de RDF que va a cargar. SPARQL utiliza el encabezado `Content-type` que envía el servidor web para determinar el formato de entrada del contenido, por lo tanto debe definir los tipos MIME relevantes para el servidor web.

   Por ejemplo, suponga que utiliza las siguientes extensiones de archivo para identificar formatos de archivo:

   - `.nt` para NTriples.
   - `.nq` para NQuads.
   - `.trig` para Trig.
   - `.rdf` para RDF/XML.
   - `.ttl` para Turtle.
   - `.n3` para N3.
   - `.jsonld` para JSON-LD.

   Si utiliza Apache 2 como servidor web, editaría el archivo `/etc/mime.types` y agregaría los siguientes tipos:

   ```
   text/plain nt
   application/n-quads nq
   application/trig trig
   application/rdf+xml rdf
   application/x-turtle ttl
   text/rdf+n3 n3
   application/ld+json jsonld
   ```

3. Confirme que la asignación del tipo MIME funciona. Una vez que tenga su servidor web en funcionamiento y hospede archivos RDF con el formato de su elección, puede probar la configuración enviando una solicitud al servidor web desde su host local.

   Por ejemplo, podría enviar una solicitud como esta:

   ```
   curl -v http://localhost:80/test.rdf
   ```

   Después, en la salida detallada de `curl`, debería ver una línea como:

```
Content-Type: application/rdf+xml
```

Esto muestra que la asignación del tipo de contenido se definió con éxito.

4. Ahora está listo para cargar datos utilizando el comando SPARQL UDPATE:

```
curl https://your-neptune-endpoint:port/sparql \
    -d "update=LOAD <http://web_server_private_ip:80/test.rdf>"
```

> **Note**
>
> El uso de `SPARQL UPDATE LOAD` puede desencadenar un tiempo de espera en el servidor web cuando el archivo de origen que se está cargando es grande. Neptune procesa los datos del archivo a medida que se transmiten y para un archivo grande puede tardar un tiempo superior al tiempo de espera configurado en el servidor. Esto, a su vez, puede hacer que el servidor cierre la conexión, lo que puede dar lugar al siguiente mensaje de error cuando Neptune encuentra un EOF inesperado en la secuencia:
>
> ```
> {
>   "detailedMessage":"Invalid syntax in the specified file",
>   "code":"InvalidParameterException"
> }
> ```
>
> Si recibe este mensaje y no cree que el archivo de origen contenga sintaxis no válida, pruebe a aumentar la configuración de tiempo de espera en el servidor web. También puede diagnosticar el problema habilitando los registros de depuración en el servidor y buscando los tiempos de espera.

Uso de SPARQL UPDATE UNLOAD para eliminar datos de Neptune

Neptune también proporciona una operación SPARQL personalizada, `UNLOAD`, para eliminar los datos que se especifican en un origen remoto. `UNLOAD` puede considerarse como una contrapartida de la operación `LOAD`. Su sintaxis es la siguiente:

> **Note**
>
> Esta característica está disponible a partir de la versión 1.0.4.1 del motor de Neptune.

```
UNLOAD SILENT (URL of the remote data to be unloaded) FROM GRAPH (named graph from
 which to remove the data)
```

- **`SILENT`**: (opcional) hace que la operación se realice correctamente aunque se haya producido un error al procesar los datos.

  Puede resultar útil cuando una sola transacción contiene varias instrucciones, como "`LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;`", y si desea que la transacción se complete aunque algunos de los datos remotos no se hayan podido procesar.
- *`URL de los datos remotos que se van a descargar`*: (obligatorio) especifica un archivo de datos remoto que contiene los datos que se van a descargar de un gráfico.

  El archivo remoto debe tener una de las siguientes extensiones (son los mismos formatos que admite UPDATE-LOAD):

  - `.nt` para NTriples.
  - `.nq` para NQuads.
  - `.trig` para Trig.
  - `.rdf` para RDF/XML.
  - `.ttl` para Turtle.
  - `.n3` para N3.
  - `.jsonld` para JSON-LD.

  La operación `UNLOAD` eliminará todos los datos que contiene este archivo del clúster de base de datos.

  Cualquier autenticación de Amazon S3 debe incluirse en la URL para que se descarguen los datos. Puede prefirmar un archivo de Amazon S3 y, a continuación, utilizar la URL resultante para acceder a él de forma segura. Por ejemplo:

  ```
  aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
   data to unload)
  ```

Después:

```
curl https://(a Neptune endpoint URL):8182/sparql \
  --data-urlencode 'update=unload (pre-signed URL of the remote Amazon S3 data to be
 unloaded) \
                           from graph (named graph)'
```

Para obtener más información, consulte la sección sobre autenticación de solicitudes: uso de parámetros de consulta.

- **FROM GRAPH** *(gráfico con nombre del que se van a eliminar los datos)*: (opcional) especifica el gráfico con nombre del que se deben descargar los datos remotos.

  Neptune asocia cada triple con un gráfico con nombre. Puede especificar el gráfico con nombre predeterminado utilizando el URI de gráfico con nombre de reserva, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, de la siguiente manera:

  ```
  FROM GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

De la misma manera que `LOAD` se corresponde con `INSERT DATA { (inline data) }`, `UNLOAD` se corresponde con `DELETE DATA { (inline data) }`. Al igual que `DELETE DATA`, `UNLOAD` no funciona con datos que contienen nodos en blanco.

Por ejemplo, si un servidor web local publica un archivo denominado `data.nt` que contiene los dos triples siguientes:

```
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#b> .
<http://example.org/resource#a> <http://example.org/resource#p> <http://example.org/
resource#c> .
```

El siguiente comando `UNLOAD` eliminaría esos dos triples del gráfico con nombre, `<http://example.org/graph1>`.

```
UNLOAD <http://localhost:80/data.nt> FROM GRAPH <http://example.org/graph1>
```

Esto tendría el mismo efecto que usar el siguiente comando `DELETE DATA`:

```
DELETE DATA {
```

```
  GRAPH <http://example.org/graph1> {
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#b> .
    <http://example.org/resource#a> <http://example.org/resource#p> <http://
example.org/resource#c> .
  }
}
```

Excepciones generadas por el comando **`UNLOAD`**

- **`InvalidParameterException`**: había nodos en blanco en los datos. Estado HTTP: solicitud incorrecta 400.

  Mensaje: `Blank nodes are not allowed for UNLOAD`

- **`InvalidParameterException`**: había una sintaxis interrumpida en los datos. Estado HTTP: solicitud incorrecta 400.

  Mensaje: `Invalid syntax in the specified file.`

- **`UnloadUrlAccessDeniedException`** : el acceso se ha denegado. Estado HTTP: solicitud incorrecta 400.

  Mensaje: `Update failure: Endpoint` *`(Neptune endpoint)`* `reported access denied error. Please verify access.`

- **`BadRequestException`** : no se pueden recuperar los datos remotos. Estado HTTP: solicitud incorrecta 400.

  Mensaje: (depende de la respuesta HTTP).

Sugerencias de consulta SPARQL

Puede utilizar sugerencias de consulta para especificar estrategias de optimización y evaluación para una consulta SPARQL concreta en Amazon Neptune.

Las sugerencias de consulta se expresan utilizando patrones triples adicionales que están insertados en la consulta SPARQL con las siguientes partes:

```
scope hint value
```

- scope: determina la parte de la consulta a la que se aplica la sugerencia de consulta, como un grupo determinado de la consulta o la consulta completa.
- hint: identifica el tipo de sugerencia que se va a aplicar.
- value: determina el comportamiento del aspecto del sistema considerado.

Las sugerencias y los ámbitos de consulta se exponen como términos predefinidos en el espacio de nombres `http://aws.amazon.com/neptune/vocab/v01/QueryHints#` de Amazon Neptune. Los ejemplos de esta sección incluyen el espacio de nombres como un prefijo `hint` que se define y se incluye en la consulta:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Por ejemplo, a continuación se muestra cómo incluir una sugerencia `joinOrder` en una consulta `SELECT`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... {
 hint:Query hint:joinOrder "Ordered" .
 ...
}
```

La consulta anterior indica al motor de Neptune que evalúe las uniones en la consulta en el orden indicado y deshabilita cualquier reordenación automática.

Tenga en cuenta lo siguiente al usar sugerencias de consulta:

- Puede combinar diferentes sugerencias de consulta en una sola consulta. Por ejemplo, puede utilizar la sugerencia de consulta `bottomUp` para anotar una subconsulta para la evaluación ascendente y una sugerencia de consulta `joinOrder` para corregir el orden de unión dentro de la subconsulta.
- Puede utilizar la misma sugerencia de consulta varias veces, en diferentes ámbitos no solapados.

- Las sugerencias de consulta son sugerencias. Aunque el motor de consulta generalmente tiene como objetivo tener en cuenta determinadas sugerencias de consulta, también puede ignorarlas.
- Las sugerencias de consulta mantienen la semántica. La adición de una sugerencia de consulta no modifica la salida de la consulta (excepto el orden potencial de los resultados cuando no se dan garantías de ordenación, es decir, cuando el orden de los resultados no se aplica explícitamente mediante el uso de ORDER BY).

En las siguientes secciones se proporciona más información sobre las sugerencias de consulta disponibles y su uso en Neptune.

Temas

Ámbito de las sugerencias de consulta SPARQL en Neptune

En la siguiente tabla, se muestran los ámbitos disponibles, las sugerencias asociadas y las descripciones para las sugerencias de consulta SPARQL en Amazon Neptune. El prefijo `hint` en estas entradas representa el espacio de nombres de Neptune para las sugerencias:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Ámbito	Sugerencia admitida	Descripción
`hint:Query`	joinOrder	La sugerencia de consulta se aplica a toda la consulta.
`hint:Query`	queryTimeout	El valor de expiración se aplica a toda la consulta

Ámbito	Sugerencia admitida	Descripción
`hint:Query`	rangeSafe	La promoción de tipos está deshabilitada para toda la consulta.
`hint:Query`	queryId	El valor del identificador de consulta se aplica a toda la consulta
`hint:Query`	useDFE	El uso del DFE está habilitado (o deshabilitado) para toda la consulta.
`hint:Group`	joinOrder	La sugerencia de consulta se aplica a los elementos de nivel superior del grupo especificado, pero no a los elementos anidados (como las subconsultas) o a los elementos principales.
`hint:SubQuery`	evaluationStrategy	La sugerencia se especifica y se aplica a una subconsul ta SELECT anidada. La subconsulta se evalúa de forma independiente, sin tener en cuenta las solucione s calculadas antes de la subconsulta.

La sugerencia de consulta SPARQL **`joinOrder`**

Cuando envía una consulta SPARQL, el motor de consultas de Amazon Neptune investiga la estructura de la consulta. Reordena partes de la consulta y trata de minimizar la cantidad de trabajo necesario para la evaluación y el tiempo de respuesta de la consulta.

Por ejemplo, una secuencia de patrones triples conectados normalmente no se evalúa en el orden dado. Se reordena mediante heurística y estadísticas como la selectividad de los patrones individuales y cómo están conectados a través de variables compartidas. Además, si la consulta contiene patrones más complejos como subconsultas, cláusulas FILTER o bloques OPTIONAL o MINUS complejos, el motor de consulta de Neptune los reordena cuando es posible, con el objetivo de conseguir un orden de evaluación eficiente.

Para consultas más complejas, el orden en el que Neptune decide evaluar la consulta puede no ser siempre el óptimo. Por ejemplo, Neptune podría perder las características específicas de datos de instancias (como el alcance de nodos Power en el gráfico) que surgen durante la evaluación de la consulta.

Si conoce las características exactas de los datos y desea dictar manualmente el orden de ejecución de la consulta, utilice la sugerencia de consulta `joinOrder` de Neptune para especificar que la consulta se evalúe en el orden indicado.

Sintaxis de sugerencias SPARQL de **`joinOrder`**

La sugerencia de consulta `joinOrder` se especifica como un patrón triple incluido en una consulta SPARQL.

Para una mayor claridad, los siguientes usos de sintaxis utilizan un prefijo `hint` que se define e incluye en la consulta para especificar el espacio de nombres de la sugerencia de consulta de Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
scope hint:joinOrder "Ordered" .
```

Ámbitos disponibles

- `hint:Query`
- `hint:Group`

Para obtener más información acerca de los ámbitos de sugerencia de consulta, vea Ámbito de las sugerencias de consulta SPARQL en Neptune.

Ejemplo de sugerencia SPARQL **`joinOrder`**

En esta sección se muestra una consulta escrita con la sugerencia de consulta `joinOrder` y sin, así como las optimizaciones relacionadas.

En este ejemplo, suponga que el conjunto de datos contiene lo siguiente:

- Una sola persona llamada `John` que indica que le gustan (`:likes`) 1000 personas, incluyendo a `Jane`.
- Una sola persona llamada `Jane` que indica que le gustan (`:likes`) 10 personas, incluyendo a `John`.

Sin sugerencias de consulta

La siguiente consulta SPARQL extrae todos los pares de personas que se llaman `John` y `Jane`, y que se gustan entre sí de un conjunto de datos de redes sociales:

```
PREFIX : <https://example.com/>
SELECT ?john ?jane {
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

El motor de consultas de Neptune puede evaluar las instrucciones en un orden diferente al escrito. Por ejemplo, puede elegir evaluar en el siguiente orden:

1. Buscar todas las personas llamadas `John`.
2. Buscar todas las personas conectadas a `John` por un borde `:likes`.
3. Filtrar este conjunto por personas llamadas `Jane`.
4. Filtrar este conjunto por las conectadas a `John` por un borde `:likes`.

Según el conjunto de datos, la evaluación en este orden da como resultado la extracción de 1000 entidades en el segundo paso. El tercer paso lo limita hasta el nodo individual, `Jane`. El paso final determina que a `Jane` también le gusta (`:likes`) el nodo `John`.

Sugerencia de consulta

Sería favorable comenzar con el nodo `Jane` porque solo tiene 10 bordes `:likes` salientes. De este modo se reduce la cantidad de trabajo durante la evaluación de la consulta al evitar la extracción de las 1000 entidades durante el segundo paso.

En el ejemplo siguiente, se utiliza la sugerencia de consulta joinOrder para asegurar que el nodo `Jane` y sus bordes de salida se procesen primero deshabilitando de todo reordenamiento automático de uniones para la consulta:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Un escenario aplicable real podría ser una aplicación de red social en la que las personas de dicha red se clasifican como personas con influencia con muchas conexiones o como usuarios normales con pocas conexiones. En tal escenario, podría asegurarse de que el usuario normal (`Jane`) se procese antes que la persona influyente (`John`) en una consulta como la del ejemplo anterior.

Sugerencia de consulta y reordenación

Puede llevar este ejemplo un paso más allá. Si sabe que el atributo `:name` es único para un solo nodo, podría acelerar la consulta mediante la reordenación y el uso de la sugerencia de consulta `joinOrder`. Este paso garantiza que los nodos únicos se extraigan primero.

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?john ?jane {
  hint:Query hint:joinOrder "Ordered" .
  ?person1 :name "Jane" .
  ?person2 :name "John" .
  ?person1 :likes ?person2 .
  ?person2 :likes ?person1 .
}
```

En este caso, puede reducir la consulta a las siguientes acciones individuales en cada paso:

1. Buscar la persona con `:name Jane`.
2. Buscar la persona con `:name John`.
3. Comprobar que el primer nodo está conectado al segundo con un borde `:likes`.

4. Comprobar que el segundo nodo está conectado al primero con un borde `:likes`.

> **Important**
>
> Si elige un orden incorrecto, la sugerencia de consulta `joinOrder` puede provocar un descenso considerable del rendimiento. Por ejemplo, el ejemplo anterior no sería eficiente si los atributos `:name` no fueran únicos. Si los 100 nodos se llamaran `Jane` y los 1000 nodos se llamaran `John`, la consulta terminaría verificando 1000 * 100 (100.000) pares para los bordes `:likes`.

La sugerencia de consulta **`evaluationStrategy`** de SPARQL

La sugerencia de consulta `evaluationStrategy` indica al motor de consultas de Amazon Neptune que el fragmento de la consulta anotada debe evaluarse de abajo arriba como una unidad independiente. Esto significa que no se utiliza ninguna solución de los pasos de evaluación anteriores para calcular el fragmento de consulta. El fragmento de consulta se evalúa como una unidad independiente y las soluciones producidas se unen con el resto de la consulta una vez calculada.

El uso de la sugerencia de consulta `evaluationStrategy` implica un plan de consulta de bloqueo (sin canalización), lo que significa que las soluciones del fragmento anotado con la sugerencia de consulta se materializan y se almacenan en búfer en la memoria principal. El uso de esta sugerencia de consulta puede aumentar considerablemente la cantidad de memoria principal necesaria para evaluar la consulta, especialmente si el fragmento de consulta anotado calcula un gran número de resultados.

Sintaxis de sugerencias SPARQL de **`evaluationStrategy`**

La sugerencia de consulta `evaluationStrategy` se especifica como un patrón triple incluido en una consulta SPARQL.

Para una mayor claridad, los siguientes usos de sintaxis utilizan un prefijo `hint` que se define e incluye en la consulta para especificar el espacio de nombres de la sugerencia de consulta de Neptune:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

Ámbitos disponibles

- `hint:SubQuery`

> **Note**
>
> Esta sugerencia de consulta solo es compatible con las subconsultas anidadas.

Para obtener más información acerca de los ámbitos de sugerencia de consulta, vea Ámbito de las sugerencias de consulta SPARQL en Neptune.

Ejemplo de sugerencia SPARQL **`evaluationStrategy`**

En esta sección se muestra una consulta escrita con la sugerencia de consulta `evaluationStrategy` y sin, así como las optimizaciones relacionadas.

En este ejemplo, suponga que el conjunto de datos tiene las siguientes características:

- Contiene 1000 bordes etiquetados `:connectedTo`.
- Cada nodo `component` está conectado a una media de otros 100 nodos `component`.
- El número típico de conexiones cíclicas de cuatro saltos entre nodos es de unos 100.

Sin sugerencias de consulta

La siguiente consulta SPARQL extrae todos los nodos `component` que están cíclicamente conectados entre sí a través de cuatro saltos:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  ?component1 :connectedTo ?component2 .
  ?component2 :connectedTo ?component3 .
  ?component3 :connectedTo ?component4 .
  ?component4 :connectedTo ?component1 .
}
```

El enfoque del motor de consultas de Neptune es evaluar esta consulta utilizando los siguientes pasos:

- Extraer los 1000 bordes `connectedTo` del gráfico.
- Expandir 100 veces (el número de bordes `connectedTo` de salida desde component2).

  Resultados intermedios: 100.000 nodos.
- Expandir 100 veces (el número de bordes `connectedTo` de salida desde component3).

  Resultados intermedios: 10.000.000 nodos.
- Escanear los 10.000.000 de nodos para el cierre del ciclo.

Esto da como resultado un plan de consulta de streaming, que tiene una cantidad constante de memoria principal.

Sugerencia de consulta y subconsultas

Es posible que desee intercambiar el espacio de memoria principal para acelerar el cálculo. Al reescribir la consulta mediante una sugerencia de consulta `evaluationStrategy`, puede forzar al motor a calcular una unión entre dos subconjuntos más pequeños y materializados.

```
PREFIX : <https://example.com/>
          PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component1 :connectedTo ?component2 .
      ?component2 :connectedTo ?component3 .
    }
  }
  {
    SELECT * WHERE {
      hint:SubQuery hint:evaluationStrategy "BottomUp" .
      ?component3 :connectedTo ?component4 .
      ?component4 :connectedTo ?component1 .
    }
  }
}
```

En lugar de evaluar los patrones triples en secuencia mientras se usan iterativamente los resultados del patrón triple anterior como entrada para los patrones siguientes, la sugerencia `evaluationStrategy` provoca que las dos subconsultas se evalúen de forma independiente.

Ambas subconsultas producen 100.000 nodos para los resultados intermedios, que después se unen para formar la salida final.

En concreto, cuando ejecuta Neptune en los tipos de instancia más grandes, el almacenamiento temporal de estos dos subconjuntos de 100 000 en la memoria principal aumenta el uso de memoria a cambio de acelerar considerablemente la evaluación.

La sugerencia de consulta **`queryTimeout`** de SPARQL

La sugerencia de consulta `queryTimeout` especifica un tiempo de espera que es menor que el conjunto de valores de`neptune_query_timeout` en el grupo de parámetros de base de datos.

Si la consulta termina como resultado de esta sugerencia, se genera una `TimeLimitExceededException`, con un mensaje `Operation terminated (deadline exceeded)`.

Sintaxis de sugerencias SPARQL de **`queryTimeout`**

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... WHERE {
    hint:Query hint:queryTimeout 10 .
    # OR
    hint:Query hint:queryTimeout "10" .
    # OR
    hint:Query hint:queryTimeout "10"^^xsd:integer .
 ...
}
```

El valor del tiempo de espera se expresa en milisegundos.

El valor del tiempo de espera debe ser menor que el valor de `neptune_query_timeout` establecido en el grupo de parámetros de la base de datos. De lo contrario, se generará una excepción `MalformedQueryException` con un mensaje `Malformed query: Query hint 'queryTimeout' must be less than neptune_query_timeout DB Parameter Group`.

Se debe especificar la sugerencia de consulta `queryTimeout` en la cláusula `WHERE` de la consulta principal o en la cláusula `WHERE` de uno de las subconsultas tal como se en el ejemplo a continuación:

Debe configurarse una sola vez en todas las secciones de consultas/subconsultas y actualizaciones de SPARQL (como INSERT y DELETE). De lo contrario, se generará una

excepción `MalformedQueryException` con un mensaje `Malformed query: Query hint 'queryTimeout' must be set only once.`

Ámbitos disponibles

La sugerencia `queryTimeout` se puede aplicar tanto a las consultas SPARQL como a las actualizaciones.

- En una consulta SPARQL, puede aparecer en la cláusula WHERE de la consulta principal o en una subconsulta.
- En una actualización de SPARQL, se puede establecer en la cláusula INSERT, DELETE o WHERE. Si hay varias cláusulas de actualización, solo se puede establecer en una de ellas.

Para obtener más información acerca de los ámbitos de sugerencia de consulta, vea Ámbito de las sugerencias de consulta SPARQL en Neptune.

Ejemplo de sugerencia SPARQL **`queryTimeout`**

A continuación se muestra un ejemplo de uso de `hint:queryTimeout` en la cláusula principal `WHERE` de una consulta `UPDATE`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
INSERT {
    ?s ?p ?o
} WHERE {
    hint:Query hint:queryTimeout 100 .
    ?s ?p ?o .
}
```

Aquí, la `hint:queryTimeout` se encuentra en la cláusula `WHERE` de una subconsulta:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   {
      SELECT ?s WHERE {
         hint:Query hint:queryTimeout 100 .
         ?s ?p1 ?o1 .
      }
   }
```

```
}
```

La sugerencia de consulta **`rangeSafe`** de SPARQL

Utilice esta sugerencia de consulta para desactivar la promoción de tipos en una consulta de SPARQL.

Cuando envía una consulta SPARQL que incluye un `FILTER` en un valor o rango numérico, normalmente el motor de consultas de Neptune debe usar la promoción de tipos cuando ejecuta la consulta. Esto significa que tiene que examinar los valores de todos los tipos que puedan contener el valor por el que se está filtrando.

Por ejemplo, si está filtrando valores iguales a 55, el motor debe buscar números enteros iguales a 55, enteros largos iguales a 55L, flotantes iguales a 55,0, etc. Cada promoción de tipo requiere una búsqueda adicional en el almacenamiento, lo que puede provocar que una consulta aparentemente simple tarde un tiempo inesperadamente largo en completarse.

A menudo, la promoción de tipos no es necesaria porque ya sabe de antemano que solo necesita encontrar valores de un tipo específico. Cuando sea así, puede acelerar considerablemente sus consultas utilizando la sugerencia de consulta `rangeSafe` para desactivar la promoción de tipos.

Sintaxis de sugerencias SPARQL de **`rangeSafe`**

La sugerencia de consulta `rangeSafe` toma un valor de `true` para desactivar la promoción de tipos. También acepta un valor de `false` (el valor predeterminado).

Ejemplo. El siguiente ejemplo muestra cómo desactivar la promoción de tipos al filtrar por un valor entero de o superior a 1:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * {
   ?s ?p ?o .
   hint:Prior hint:rangeSafe 'true' .
   FILTER (?o > '1'^^<http://www.w3.org/2001/XMLSchema#int>)
```

La sugerencia de consulta SPARQL **`queryId`**

Utilice esta sugerencia de consulta para asignar su propio valor de queryId a una consulta SPARQL.

Ejemplo:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * WHERE {
  hint:Query hint:queryId "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
  {?s ?p ?o}}
```

El valor que asigne debe ser único en todas las consultas de la base de datos de Neptune.

La sugerencia de consulta **`useDFE`** de SPARQL

Utilice esta sugerencia de consulta para permitir el uso del DFE para ejecutar la consulta. De forma predeterminada, Neptune no usa el DFE sin que esta sugerencia de consulta esté establecida en `true`, ya que el parámetro de instancia neptune_dfe_query_engine tiene el valor predeterminado `viaQueryHint`. Si establece ese parámetro de instancia en `enabled`, el motor DFE se utiliza para todas las consultas excepto para las que la sugerencia de consulta `useDFE` está establecida en `false`.

Ejemplo de cómo habilitar el uso del DFE para una consulta:

```
PREFIX : <https://example.com/>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>

SELECT ?john ?jane
{
  hint:Query hint:useDFE true .
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Sugerencias de consulta de SPARQL utilizadas con DESCRIBE

Una consulta `DESCRIBE` de SPARQL proporciona un mecanismo flexible para solicitar descripciones de recursos. Sin embargo, las especificaciones de SPARQL no definen la semántica precisa de `DESCRIBE`.

A partir de la versión 1.2.0.2 del motor, Neptune admite varios modos y algoritmos `DESCRIBE`diferentes que se adaptan a diferentes situaciones.

Este conjunto de datos de muestra puede ayudar a ilustrar los diferentes modos:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <https://example.com/> .

:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JohnDoe :firstName "John" .
:JaneDoe :knows _:b1 .
_:b1 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .
:RichardRoe :firstName "Richard" .

_:s1 rdf:type rdf:Statement .
_:s1 rdf:subject :JaneDoe .
_:s1 rdf:predicate :knows .
_:s1 rdf:object :JohnDoe .
_:s1 :knowsFrom "Berlin" .

:ref_s2 rdf:type rdf:Statement .
:ref_s2 rdf:subject :JaneDoe .
:ref_s2 rdf:predicate :knows .
:ref_s2 rdf:object :JohnDoe .
:ref_s2 :knowsSince 1988 .
```

En los ejemplos siguientes, se supone que se solicita una descripción del recurso `:JaneDoe` mediante una consulta SPARQL como esta:

```
DESCRIBE <https://example.com/JaneDoe>
```

La sugerencia de consulta **`describeMode`** de SPARQL

La sugerencia de consulta `hint:describeMode` de SPARQL se utiliza para seleccionar uno de los siguientes modos `DESCRIBE` de SPARQL compatibles con Neptune:

El modo **`ForwardOneStep`** de DESCRIBE

El modo `ForwardOneStep` se invoca con la sugerencia de consulta `describeMode` de la siguiente manera:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
```

```
{
  hint:Query hint:describeMode "ForwardOneStep"
}
```

El modo `ForwardOneStep` solo devuelve los atributos y los enlaces de reenvío del recurso que se va a describir. En el caso de ejemplo, esto significa que devuelve los triples que tienen `:JaneDoe`, que es el recurso que se va a describir, como asunto:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b301990159 .
```

Tenga en cuenta que la consulta DESCRIBE puede devolver triples con nodos en blanco, por ejemplo `_:b301990159`, que tienen identificadores diferentes cada vez, en comparación con el conjunto de datos de entrada.

El modo **`SymmetricOneStep`** de DESCRIBE

`SymmetricOneStep` es el modo de DESCRIBE predeterminado si no proporciona una sugerencia de consulta. También puede invocarlo de forma explícita con la sugerencia de consulta `describeMode` de la siguiente manera:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SymmetricOneStep"
}
```

En la semántica `SymmetricOneStep`, `DESCRIBE` devuelve los atributos, los enlaces directos y los enlaces inversos del recurso que se va a describir:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b318767375 .

_:b318767631 rdf:subject :JaneDoe .

:RichardRoe :knows :JaneDoe .

:ref_s2 rdf:subject :JaneDoe .
```

El modo de DESCRIBE de descripción limitada concisa (**CBD**)

El modo de descripción limitada concisa (CBD) se invoca con la sugerencia de consulta `describeMode` de la siguiente manera:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "CBD"
}
```

En la semántica CBD, `DESCRIBE` devuelve la descripción limitada concisa (tal como la define el W3C) del recurso que se va a describir:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b285212943 .
_:b285212943 :knows :RichardRoe .

_:b285213199 rdf:subject :JaneDoe .
_:b285213199 rdf:type rdf:Statement .
_:b285213199 rdf:predicate :knows .
_:b285213199 rdf:object :JohnDoe .
_:b285213199 :knowsFrom "Berlin" .

:ref_s2 rdf:subject :JaneDoe .
```

La descripción limitada concisa de un recurso de RDF (es decir, un nodo de un gráfico RDF) es el subgráfico más pequeño centrado alrededor de ese nodo que puede ser independiente. En la práctica, esto significa que si piensa en este gráfico como un árbol, con el nodo designado como raíz, no hay nodos en blanco (nodos b) como hojas de ese árbol. Como los nodos b no se pueden direccionar externamente ni se pueden usar en consultas posteriores, no basta con navegar por el gráfico para encontrar los siguientes saltos individuales desde el nodo actual. También hay que ir lo suficientemente lejos como para encontrar algo que pueda usarse en consultas posteriores (es decir, algo que no sea un nodo b).

Cálculo del CBD

Dado un nodo en particular (el nodo de inicio o raíz) en el gráfico RDF de origen, el CBD de ese nodo se calcula de la siguiente manera:

1. Incluya en el subgráfico todas las instrucciones del gráfico de origen en las que el sujeto de la instrucción sea el nodo inicial.
2. De forma recursiva, para todas las instrucciones del subgráfico que hasta ahora tengan un objeto de nodo en blanco, incluya en el subgráfico todas las instrucciones del gráfico de origen cuyo sujeto sea ese nodo en blanco y que aún no estén incluidas en el subgráfico.
3. De forma recursiva, para todas las instrucciones incluidas en el subgráfico hasta ahora, para todas las reificaciones de estas instrucciones en el gráfico de origen, incluya el CBD comenzando por el nodo `rdf:Statement` de cada reificación.

Esto da como resultado un subgráfico en el que los nodos objeto son referencias o literales del IRI, o nodos en blanco que no sirven de sujeto a ninguna instrucción del gráfico. Tenga en cuenta que el CBD no se puede calcular con una sola consulta SELECT o CONSTRUCT de SPARQL.

El modo DESCRIBE de descripción limitada concisa simétrica (**`SCBD`**)

El modo de descripción limitada concisa simétrica (`SCBD`) se invoca con la sugerencia de consulta `describeMode` de la siguiente manera:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SCBD"
}
```

En la semántica de `SCBD`, `DESCRIBE` devuelve la descripción limitada concisa simétrica del recurso (tal como la define el W3C en Describing Linked Datasets with the VoID Vocabulary:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b335544591 .
_:b335544591 :knows :RichardRoe .

:RichardRoe :knows :JaneDoe .

_:b335544847 rdf:subject :JaneDoe .
_:b335544847 rdf:type rdf:Statement .
_:b335544847 rdf:predicate :knows .
_:b335544847 rdf:object :JohnDoe .
_:b335544847 :knowsFrom "Berlin" .
```

```
:ref_s2 rdf:subject :JaneDoe .
```

La ventaja de CBD y SCBD con respecto a los modos `ForwardOneStep` y `SymmetricOneStep` es que los nodos vacíos siempre se expanden para incluir su representación. Esto puede ser una ventaja importante, ya que no se puede consultar un nodo en blanco con SPARQL. Además, los modos CBD y SCBD también consideran las reificaciones.

Tenga en cuenta que la sugerencia de consulta `describeMode` también puede formar parte de una cláusula `WHERE`:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE ?s
WHERE {
  hint:Query hint:describeMode "CBD" .
  ?s rdf:type <https://example.com/Person>
}
```

La sugerencia de consulta **`describeIterationLimit`** de SPARQL

La sugerencia de consulta de SPARQL `hint:describeIterationLimit` proporciona una restricción opcional con respecto al número máximo de expansiones iterativas que se deben realizar para los algoritmos DESCRIBE iterativos, como CBD y SCBD.

Los límites de DESCRIBE se suman con AND. Por lo tanto, si se especifican tanto el límite de iteración como el límite de las instrucciones, ambos límites deben cumplirse antes de que se interrumpa la consulta DESCRIBE.

El valor predeterminado de este valor es 5. Puede establecerlo en CERO (0) para no especificar ningún límite en el número de expansiones iterativas.

La sugerencia de consulta **`describeStatementLimit`** de SPARQL

La sugerencia de consulta `hint:describeStatementLimit` de SPARQL proporciona una restricción opcional con respecto al número máximo de instrucciones que pueden estar presentes en una respuesta a una consulta DESCRIBE. Solo se aplica a los algoritmos DESCRIBE iterativos, como CBD y SCBD.

Los límites de DESCRIBE se suman con AND. Por lo tanto, si se especifican tanto el límite de iteración como el límite de las instrucciones, ambos límites deben cumplirse antes de que se interrumpa la consulta DESCRIBE.

El valor predeterminado de este valor es 5000. Puede establecerlo en CERO (0) para no especificar ningún límite en el número de instrucciones devueltas.

Comportamiento de DESCRIBE de SPARQL con respecto al gráfico predeterminado

El formulario de consulta `DESCRIBE` de SPARQL le permite recuperar información sobre los recursos sin conocer la estructura de los datos y sin tener que redactar una consulta. La forma en que se recopila esta información depende de la implementación de SPARQL. Neptune proporciona varias sugerencias de consulta que invocan diferentes modos y algoritmos para que los use `DESCRIBE`.

En la implementación de Neptune, independientemente del modo, `DESCRIBE` solo usa los datos presentes en el gráfico predeterminado de SPARQL. Esto es coherente con la forma en que SPARQL trata los conjuntos de datos (consulte Specifying RDF Datasets en la especificación de SPARQL).

En Neptune, el gráfico predeterminado contiene todos los triples únicos en la unión de todos los gráficos con nombre de la base de datos, a menos que los gráficos con nombre en particular se especifiquen mediante cláusulas `FROM` y/o `FROM NAMED`. Todos los datos RDF de Neptune se almacenan en un gráfico con nombre. Si se inserta un triple sin un contexto de gráfico con nombre, Neptune lo almacena en un gráfico con nombre designado `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Cuando se especifican uno o más gráficos con nombre mediante la cláusula `FROM`, el gráfico predeterminado es la unión de todos los triples únicos de esos gráficos con nombre. Si no hay ninguna cláusula `FROM` y hay una o más cláusulas `FROM NAMED`, el gráfico predeterminado está vacío.

Ejemplos de **`DESCRIBE`** de SPARQL

Analice los siguientes datos:

```
PREFIX ex: <https://example.com/>

GRAPH ex:g1 {
    ex:s ex:p1 "a" .
    ex:s ex:p2 "c" .
}

GRAPH ex:g2 {
```

```
    ex:s ex:p3 "b" .
    ex:s ex:p2 "c" .
}

ex:s ex:p3 "d" .
```

Para esta consulta:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM ex:g1
FROM NAMED ex:g2
WHERE {
  GRAPH ex:g2 { ?s ?p "b" . }
}
```

Neptune devolvería:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
```

Aquí, el patrón del gráfico `GRAPH ex:g2 { ?s ?p "b" }` se evalúa primero, lo que da como resultado enlaces para `?s`, y luego la parte `DESCRIBE` se evalúa con respecto al gráfico predeterminado, que ahora es solo `ex:g1`.

Sin embargo, para esta consulta:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM NAMED ex:g1
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Neptune no devolvería nada, porque cuando una cláusula `FROM NAMED` está presente sin ninguna cláusula `FROM`, el gráfico predeterminado está vacío.

En la siguiente consulta, `DESCRIBE` se usa sin presencia de ninguna cláusula `FROM` o `FROM NAMED`:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
```

```
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

En esta situación, el gráfico predeterminado se compone de todos los triples únicos en la unión de todos los gráficos con nombre de la base de datos (formalmente, la combinación RDF), por lo que Neptune devolvería:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
ex:s ex:p3 "b" .
ex:s ex:p3 "d" .
```

API de estado de la consulta SPARQL

Para obtener el estado de las consultas SPARQL, utilice HTTP `GET` o `POST` para realizar una solicitud al punto de enlace `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`.

Parámetros de solicitud de estado de consultas SPARQL

queryId (opcional)

El ID de una consulta SPARQL en ejecución. Solo muestra el estado de la consulta especificada.

Sintaxis de respuesta de estado de consultas SPARQL

```
{
    "acceptedQueryCount": integer,
    "runningQueryCount": integer,
    "queries": [
      {
        "queryId":"guid",
        "queryEvalStats":
          {
            "subqueries": integer,
            "elapsed": integer,
            "cancelled": boolean
          },
        "queryString": "string"
      }
    ]
```

```
}
```

Valores de respuesta de estado de consultas SPARQL

aceptado QueryCount

El número de consultas aceptadas desde el último reinicio del motor de Neptune.

corriendo QueryCount

El número de consultas SPARQL que se están ejecutando actualmente.

queries

Una lista de las consultas SPARQL actuales.

queryId

Un ID de GUID para la consulta. Neptune asigna automáticamente este valor de identificador a cada consulta o también puede asignar su propio identificador (consulte Inserte un identificador personalizado en una consulta de Neptune Gremlin o SPARQL).

consulta EvalStats

Estadísticas de esta consulta.

subqueries

El número de subconsultas de esta consulta.

elapsed

El número de milisegundos que la consulta lleva en ejecución.

cancelled

True indica que se canceló la consulta.

queryString

La consulta enviada.

Ejemplo de estado de consultas SPARQL

A continuación se muestra un ejemplo de comando de estado que utiliza `curl` y HTTP `GET`.

```
curl https://your-neptune-endpoint:port/sparql/status
```

Esta salida muestra una única consulta en ejecución.

```
{
    "acceptedQueryCount":9,
    "runningQueryCount":1,
    "queries": [
        {
            "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
            "queryEvalStats":
                {
                    "subqueries": 0,
                    "elapsed": 29256,
                    "cancelled": false
                },
            "queryString": "SELECT ?s ?p ?o WHERE {?s ?p ?o}"
        }
    ]
}
```

Cancelación de consultas SPARQL

Para obtener el estado de las consultas SPARQL, utilice HTTP `GET` o `POST` para realizar una solicitud al punto de enlace `https://your-neptune-endpoint:port/sparql/status`.

Parámetros de solicitud de cancelación de consultas SPARQL

cancelQuery

Indica al comando de estado que cancele una consulta (obligatorio). Este parámetro no selecciona un valor.

queryId

El ID de la consulta SPARQL en ejecución que se va a cancelar (obligatorio).

silent

Si es `silent=true`, se cancela la consulta en ejecución y el código de respuesta HTTP es 200 (opcional). Si `silent` no está presente o `silent=false`, la consulta se cancela con un código de estado HTTP 500.

Ejemplos de cancelación de consultas SPARQL

Ejemplo 1: Cancelación con **`silent=false`**

A continuación se muestra un ejemplo de comando de estado que utiliza `curl` para cancelar una consulta con el parámetro `silent` establecido en `false`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=false"
```

A menos que la consulta ya haya comenzado a transmitir resultados, la consulta cancelada devolvería un código HTTP 500 con una respuesta como esta:

```
{
  "code": "CancelledByUserException",
  "requestId": "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47",
  "detailedMessage": "Operation terminated (cancelled by user)"
}
```

Si la consulta ya ha devuelto un código HTTP 200 (OK) y ha iniciado la transmisión de resultados antes de su cancelación, la información de excepción de tiempo de espera se enviará a la secuencia de salida normal.

Ejemplo 2: Cancelación con **`silent=true`**

A continuación se muestra un ejemplo del mismo comando de estado que el anterior, excepto con el parámetro `silent` ahora establecido en `true`:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=true"
```

Este comando devolvería la misma respuesta que con `silent=false`, pero la consulta cancelada devolvería ahora un código HTTP 200 con una respuesta similar a la siguiente:

```
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
```

```
  },
  "results" : {
    "bindings" : [ ]
  }
}
```

Uso del protocolo HTTP de almacén de gráficos (GSP) de SPARQL 1.1 en Amazon Neptune

En la recomendación del protocolo HTTP de almacén de gráficos de SPARQL 1.1, el W3C definió un protocolo HTTP para administrar los gráficos RDF. Define las operaciones para eliminar, crear y reemplazar el contenido de los gráficos RDF, así como para añadir instrucciones RDF al contenido existente.

El protocolo de almacén de gráficos (GSP) proporciona una forma cómoda de manipular todo el gráfico sin tener que escribir consultas SPARQL complejas.

A partir de Versión: 1.0.5.0 (27/07/2021), Neptune es totalmente compatible con este protocolo.

El punto de conexión del protocolo de almacén de gráficos (GSP) es:

```
https://your-neptune-cluster:port/sparql/gsp/
```

Para acceder al gráfico predeterminado con GSP, utilice:

```
https://your-neptune-cluster:port/sparql/gsp/?default
```

Para acceder a un gráfico con nombre con GSP, utilice:

```
https://your-neptune-cluster:port/sparql/gsp/?graph=named-graph-URI
```

Detalles especiales de la implementación de GSP en Neptune

Neptune implementa plenamente la recomendación del W3C que define GSP. Sin embargo, hay algunas situaciones que la especificación no cubre.

Una de ellas es cuando una solicitud PUT o POST especifica uno o más gráficos con nombre en el cuerpo de la solicitud que difieren del gráfico especificado en la URL de la solicitud. Esto solo puede ocurrir cuando el formato RDF del cuerpo de la solicitud admite gráficos con nombre,

como, por ejemplo, si se utiliza `Content-Type: application/n-quads` o `Content-Type: application/trig`.

En esta situación, Neptune añade o actualiza todos los gráficos con nombre presentes en el cuerpo, así como el gráfico con nombre especificado en la URL.

Por ejemplo, supongamos que, partiendo de una base de datos vacía, se envía una solicitud PUT para realizar actualizaciones o inserciones en los votos en tres gráficos. Uno, denominado `urn:votes`, contiene todos los votos de todos los años electorales. Otros dos, denominados `urn:votes:2005` y `urn:votes:2019`, contienen votos de años electorales específicos. La solicitud y su carga tienen el siguiente aspecto:

```
PUT "http://your-Neptune-cluster:port/sparql/gsp/?graph=urn:votes"
  Host: example.com
  Content-Type: application/n-quads

  PAYLOAD:

  <urn:JohnDoe> <urn:votedFor> <urn:Labour> <urn:votes:2005>
  <urn:JohnDoe> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
  <urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
  <urn:JaneSmith> <urn:votedFor> <urn:Conservative> <urn:votes:2019>
```

Una vez ejecutada la solicitud, los datos de la base de datos tienen el siguiente aspecto:

```
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes:2005>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes:2005>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes:2019>
<urn:JohnDoe>   <urn:votedFor> <urn:Labour>           <urn:votes>
<urn:JohnDoe>   <urn:votedFor> <urn:Conservative>     <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:LiberalDemocrats> <urn:votes>
<urn:JaneSmith> <urn:votedFor> <urn:Conservative>     <urn:votes>
```

Otra situación ambigua es cuando se especifica más de un gráfico en la propia URL de la solicitud, utilizando `PUT`, `POST`, `GET` o `DELETE`. Por ejemplo:

```
POST "http://your-Neptune-cluster:port/sparql/gsp/?
graph=urn:votes:2005&graph=urn:votes:2019"
```

O bien:

```
GET "http://your-Neptune-cluster:port/sparql/gsp/?default&graph=urn:votes:2019"
```

En esta situación, Neptune devuelve un HTTP 400 con un mensaje que indica que solo se puede especificar un gráfico en la URL de la solicitud.

Análisis de la ejecución de las consultas de Neptune con **`explain`** de SPARQL

Amazon Neptune ha añadido una característica de SPARQL denominada explain. Esta característica es una herramienta de autoservicio para comprender el enfoque de ejecución adoptado por el motor de Neptune. Puede invocarla añadiendo un parámetro `explain` a una llamada HTTP que envíe una consulta SPARQL.

La característica `explain` proporciona información sobre la estructura lógica de los planes de ejecución de consultas. Puede utilizar esta información para identificar posibles cuellos de botella en la evaluación y la ejecución. A continuación, puede usar sugerencias de consulta para mejorar los planes de ejecución de consultas.

Temas

- Funcionamiento del motor de consultas de SPARQL en Neptune
- Cómo utilizar explain de SPARQL para analizar la ejecución de consultas de Neptune
- Ejemplos de invocación de explain de SPARQL en Neptune
- Operadores explain de SPARQL en Neptune
- Limitaciones de explain de SPARQL en Neptune

Funcionamiento del motor de consultas de SPARQL en Neptune

Para utilizar la información que proporciona la característica `explain` de SPARQL, debe comprender algunos detalles sobre el funcionamiento del motor de consultas de SPARQL de Amazon Neptune.

El motor traduce cada consulta SPARQL en una canalización de operadores. Empezando en el primer operador, las soluciones intermedias denominadas listas de enlaces fluyen a través de esta canalización de operadores. Una lista de enlaces se puede considerar una tabla en la que los encabezados son un subconjunto de las variables utilizadas en la consulta. Cada fila de la tabla representa un resultado, hasta el punto de evaluación.

Supongamos que se han definido dos prefijos de espacio de nombres para los datos:

```
@prefix ex:   <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

El siguiente sería un ejemplo de una lista de enlaces sencilla en este contexto:

```
?person       | ?firstName
------------------------------------------------------
ex:JaneDoe    | "Jane"
ex:JohnDoe    | "John"
ex:RichardRoe | "Richard"
```

Para cada una de las tres personas, la lista vincula la variable `?person` a un identificador de la persona y la variable `?firstName` al nombre de la persona.

En el caso general, las variables pueden permanecer sin vincular si, por ejemplo, hay una selección `OPTIONAL` de una variable en una consulta para la que no hay ningún valor presente en los datos.

El operador `PipelineJoin` es un ejemplo de un operador de motor de consultas de Neptune presente en la salida de `explain`. Toma como entrada un conjunto vinculante entrante del operador anterior y lo une a un patrón triple, por ejemplo `(?person, foaf:lastName, ?lastName)`. Esta operación usa los enlaces para la variable `?person` en su secuencia de entrada, los sustituye en el patrón triple y busca los triples en la base de datos.

Cuando se ejecuta en el contexto de los enlaces entrantes de la tabla anterior, `PipelineJoin` evaluará tres búsquedas, en concreto las siguientes:

```
(ex:JaneDoe,    foaf:lastName, ?lastName)
(ex:JohnDoe,    foaf:lastName, ?lastName)
(ex:RichardRoe, foaf:lastName, ?lastName)
```

Este enfoque se denomina evaluación según vinculación. Las soluciones de este proceso de evaluación se unen a las soluciones entrantes, rellenando el `?lastName` detectado en estas. Suponiendo que encuentra un apellido para las tres personas, el operador produciría una lista de enlaces saliente que tendría un aspecto similar al siguiente:

```
?person       | ?firstName | ?lastName
```

```
  ---------------------------------------
  ex:JaneDoe    | "Jane"     | "Doe"
  ex:JohnDoe    | "John"     | "Doe"
  ex:RichardRoe | "Richard"  | "Roe"
```

A continuación, esta lista de enlaces saliente sirve como entrada para el siguiente operador de la canalización. Al final, la salida del último operador de la canalización define el resultado de la consulta.

Las canalizaciones de operadores suelen ser lineales, en el sentido de que cada operador emite soluciones para un único operador conectado. Sin embargo, en algunos casos, pueden tener estructuras más complejas. Por ejemplo, un operador `UNION` de una consulta SPARQL se mapea a una operación `Copy`. Esta operación duplica los enlaces y reenvía las copias en dos subplanes, uno para el lado izquierdo y el otro para el lado derecho de la `UNION`.

Para obtener más información sobre los operadores, consulte Operadores `explain` de SPARQL en Neptune.

Cómo utilizar **`explain`** de SPARQL para analizar la ejecución de consultas de Neptune

La característica `explain` de SPARQL es una herramienta de autoservicio de Amazon Neptune que le ayuda a entender el enfoque de ejecución adoptado por el motor de Neptune. Para invocar `explain`, debe pasar un parámetro a una solicitud HTTP o HTTPS con el formato `explain=`*`mode`*.

El valor del modo puede ser: `static`, `dynamic` o `details`.

- En el modo estático, `explain` solo imprime la estructura estática del plan de consulta.
- En el modo dinámico, `explain` también incluye aspectos dinámicos del plan de consulta. Estos aspectos pueden incluir el número de enlaces intermedios que fluyen a través de los operadores, la proporción entre los enlaces entrantes y los enlaces salientes y el tiempo total que necesitan los operadores.
- En el modo de detalles, `explain` imprime la información mostrada en el modo `dynamic` más detalles adicionales como la cadena de consulta SPARQL real y el recuento de intervalo estimado para el patrón subyacente de un operador de unión.

Neptune admite el uso de `explain` con los tres protocolos de acceso a consultas de SPARQL que figuran en la especificación del protocolo W3C SPARQL 1.1, en concreto:

1. HTTP GET
2. HTTP POST con parámetros codificados en URL
3. HTTP POST con parámetros de texto

Para obtener información sobre el motor de consultas de SPARQL, consulte Funcionamiento del motor de consultas de SPARQL en Neptune.

Para obtener información sobre el tipo de salida producida al invocar a SPARQL `explain`, consulte Ejemplos de invocación de `explain` de SPARQL en Neptune.

Ejemplos de invocación de **`explain`** de SPARQL en Neptune

Los ejemplos de esta sección muestran los diferentes tipos de salida que puede producir al invocar la característica `explain` de SPARQL para analizar la ejecución de consultas en Amazon Neptune.

Temas

- Descripción de la salida de explain
- Ejemplo de salida del modo de detalles
- Ejemplo de salida del modo estático
- Distintas formas de codificar los parámetros
- Otros tipos de salida distintos de text/plain
- Ejemplo de salida explain de SPARQL cuando el DFE está habilitado

Descripción de la salida de explain

En este ejemplo, Jane Doe conoce a dos personas, John Doe y Richard Roe:

```
@prefix ex: <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:JaneDoe foaf:knows ex:JohnDoe .
ex:JohnDoe foaf:firstName "John" .
ex:JohnDoe foaf:lastName "Doe" .
ex:JaneDoe foaf:knows ex:RichardRoe .
ex:RichardRoe foaf:firstName "Richard" .
ex:RichardRoe foaf:lastName "Roe" .
.
```

Para determinar los nombres de todas las personas a las que conoce Jane Doe, puede escribir la siguiente consulta:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -H "Accept: text/csv"
```

Esta consulta sencilla devuelve lo siguiente:

```
firstName
John
Richard
```

A continuación, cambie el comando `curl` para invocar a `explain` añadiendo `-d "explain=dynamic"` y utilizando el tipo de salida predeterminado en lugar de `text/csv`:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=dynamic"
```

La consulta ahora devuelve la salida en formato ASCII bien escrito (tipo de contenido HTTP `text/plain`), que es el tipo de salida predeterminado:

```
########################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
########################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1          # 0.00  # 0         #
########################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1         # 2          # 2.00  # 1         #
#    #        #        #                   # joinType=join
            #          #         #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #         #           #       #           #
```

```
##################################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 1         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
##################################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 0         #
##################################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 1         #
##################################################################################################
```

Para obtener más información sobre las operaciones en la columna `Name` y sus argumentos, consulte Operadores explain.

A continuación, se describe la salida fila por fila:

1. El primer paso de la consulta principal siempre utiliza el operador `SolutionInjection` para inyectar una solución. A continuación, la solución se expande al resultado final a través del proceso de evaluación.

   En este caso, se inyecta lo que se denomina la solución universal `{ }`. En la presencia de cláusulas `VALUES` o un valor `BIND`, este paso también podría inyectar enlaces de variables más complejos para comenzar.

   La columna `Units Out` indica que esta solución única fluye del operador. La columna `Out #1` especifica el operador en el que este operador inyecta el resultado. En este ejemplo, todos los operadores están conectados al operador que se indica a continuación en la tabla.
2. El segundo paso es un `PipelineJoin`. Recibe como entrada la solución universal (sin restricciones) única producida por el operador anterior (`Units In := 1`). Lo une al patrón de tuplas definido por su argumento `pattern`. Esto corresponde a una búsqueda sencilla del patrón. En este caso, el patrón triple se define del modo siguiente:

   ```
   distinct( ex:JaneDoe, foaf:knows, ?person )
   ```

   El argumento `joinType := join` indica que esta es una unión normal (otros tipos incluyen uniones `optional`, `existence check`, etc.).

El argumento `distinct := true` dice que solo se extraen coincidencias distintas de la base de datos (sin duplicados) y que dichas coincidencias se enlazan a la variable `joinProjectionVars := ?person`, desduplicadas.

El hecho de que el valor de la columna `Units Out` sea 2 indica que hay dos soluciones fluyendo. En concreto, se trata de los enlaces para la variable `?person`, que reflejan las dos personas que los datos muestran que Jane Doe conoce:

```
?person
-------------
ex:JohnDoe
ex:RichardRoe
```

3. Las dos soluciones de la etapa 2 fluyen como entrada (`Units In := 2`) en la segunda `PipelineJoin`. Este operador une las dos soluciones anteriores con el siguiente patrón triple:

   ```
   distinct(?person, foaf:firstName, ?firstName)
   ```

   Se sabe que la variable `?person` está enlazada a `ex:JohnDoe` o a `ex:RichardRoe` por la solución entrante del operador. Teniendo en cuenta esto, `PipelineJoin` extrae los nombres, John y Richard. Las dos soluciones salientes (Units Out := 2) son las siguientes:

   ```
   ?person       | ?firstName
   --------------------------
   ex:JohnDoe    | John
   ex:RichardRoe | Richard
   ```

4. El siguiente operador de proyección toma como entrada la dos soluciones de la etapa 3 (`Units In := 2`) y las proyecta en la variable `?firstName`. Esto elimina el resto de enlaces de variables en los mapeos y pasa los dos enlaces (`Units Out := 2`):

   ```
   ?firstName
   ----------
   John
   Richard
   ```

5. Para mejorar el rendimiento, Neptune opera siempre que es posible con identificadores internos que asigna a términos como URI y literales de cadena, no con las propias cadenas. El operador

final, `TermResolution`, vuelve a mapear estos identificadores internos a las cadenas de términos correspondientes.

En una evaluación de consulta normal (que no es de explain), el resultado calculado por el último operador se serializa en el formato de serialización solicitado y se transmite en streaming al cliente.

Ejemplo de salida del modo de detalles

> **Note**
>
> El modo de detalles de explain de SPARQL está disponible a partir de la versión 1.0.2.1 del motor de Neptune.

Supongamos que ejecuta la misma consulta que la anterior en modo de detalles en lugar de hacerlo en modo dinámico:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=details"
```

Como muestra este ejemplo, la salida es la misma con algunos detalles adicionales como la cadena de consulta en la parte superior de la salida y el recuento de `patternEstimate` para el operador `PipelineJoin`:

```
Query:
PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://www.example.com/>
SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person foaf:firstName ?
firstName }

#############################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     # Units In # Units Out # Ratio # Time (ms) #
#############################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
            # -        # 0         # 1         # 0.00  # 0         #
```

```
#################################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)     # -        # 1        # 2         # 2.00  # 13        #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person]
            #          #          #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #          #           #       #           #
#################################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        # 2        # 2         # 1.00  # 3         #
#    #        #        #                   # joinType=join
            #          #          #           #       #           #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
            #          #          #           #       #           #
#    #        #        #                   # patternEstimate=2
            #          #          #           #       #           #
#################################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
            # retain   # 2        # 2         # 1.00  # 1         #
#################################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
            # id2value # 2        # 2         # 1.00  # 7         #
#################################################################################################
```

Ejemplo de salida del modo estático

Supongamos que ejecuta la consulta anterior en modo estático (el valor predeterminado) en lugar de en modo de detalles:

```
 curl http(s)://your_server:your_port/sparql \
   -d "query=PREFIX foaf: <https://xmlns.com/foaf/0.1/> PREFIX ex: <https://
www.example.com/> \
       SELECT ?firstName WHERE { ex:JaneDoe foaf:knows ?person . ?person
 foaf:firstName ?firstName }" \
   -d "explain=static"
```

Tal como muestra este ejemplo, el resultado es el mismo, salvo que omite las tres últimas columnas:

```
#################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments
            # Mode     #
```

```
##################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]
             # -        #
##################################################################################################
# 1  # 2      # -      # PipelineJoin      # pattern=distinct(ex:JaneDoe, foaf:knows, ?
person)      # -        #
#    #        #        #                   # joinType=join
             #          #
#    #        #        #                   # joinProjectionVars=[?person]
             #          #
##################################################################################################
# 2  # 3      # -      # PipelineJoin      # pattern=distinct(?person,
 foaf:firstName, ?firstName) # -        #
#    #        #        #                   # joinType=join
             #          #
#    #        #        #                   # joinProjectionVars=[?person, ?firstName]
             #          #
##################################################################################################
# 3  # 4      # -      # Projection        # vars=[?firstName]
             # retain   #
##################################################################################################
# 4  # -      # -      # TermResolution    # vars=[?firstName]
             # id2value #
##################################################################################################
```

Distintas formas de codificar los parámetros

Las siguientes consultas de ejemplo ilustran dos formas distintas de codificar parámetros al invocar la característica `explain` de SPARQL.

Uso de la codificación URL: en este ejemplo, se utiliza la codificación URL de parámetros y se especifica la salida dinámica:

```
curl -XGET "http(s)://your_server:your_port/sparql?query=SELECT%20*%20WHERE%20%7B
%20%3Fs%20%3Fp%20%3Fo%20%7D%20LIMIT%20%31&explain=dynamic"
```

Especificación de los parámetros directamente: es igual que en la consulta anterior, excepto que pasa los parámetros directamente a través de POST:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic"
```

Otros tipos de salida distintos de text/plain

Los ejemplos anteriores utilizan el tipo de salida `text/plain` predeterminado. Neptune también puede formatear la salida `explain` de SPARQL en otros dos formatos de tipo MIME, es decir, `text/csv` y `text/html`. Para invocarlos, establezca el encabezado HTTP `Accept` mediante la marca `-H` en `curl`, tal y como se indica a continuación:

```
  -H "Accept: output type"
```

Estos son algunos ejemplos:

Salida **`text/csv`**

Esta consulta solicita un salida CSV de tipo MIME especificando `-H "Accept: text/csv"`:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic" \
  -H "Accept: text/csv"
```

El formato CSV, que resulta útil para la importación en una hoja de cálculo o una base de datos, separa los campos de cada fila de `explain` mediante punto y coma (`;`), de la siguiente manera:

```
ID;Out #1;Out #2;Name;Arguments;Mode;Units In;Units Out;Ratio;Time (ms)
0;1;-;SolutionInjection;solutions=[{}];-;0;1;0.00;0
1;2;-;PipelineJoin;pattern=distinct(?s, ?p, ?o),joinType=join,joinProjectionVars=[?s, ?
p, ?o];-;1;6;6.00;1
2;3;-;Projection;vars=[?s, ?p, ?o];retain;6;6;1.00;2
3;-;-;Slice;limit=1;-;1;1;1.00;1
```

Salida **`text/html`**

Si se especifica `-H "Accept: text/html"`, `explain` genera una tabla HTML:

```
<!DOCTYPE html>
<html>
  <body>
    <table border="1px">
      <thead>
        <tr>
```

```
          <th>ID</th>
          <th>Out #1</th>
          <th>Out #2</th>
          <th>Name</th>
          <th>Arguments</th>
          <th>Mode</th>
          <th>Units In</th>
          <th>Units Out</th>
          <th>Ratio</th>
          <th>Time (ms)</th>
        </tr>
      </thead>

      <tbody>
        <tr>
          <td>0</td>
          <td>1</td>
          <td>-</td>
          <td>SolutionInjection</td>
          <td>solutions=[{}]</td>
          <td>-</td>
          <td>0</td>
          <td>1</td>
          <td>0.00</td>
          <td>0</td>
        </tr>

        <tr>
          <td>1</td>
          <td>2</td>
          <td>-</td>
          <td>PipelineJoin</td>
          <td>pattern=distinct(?s, ?p, ?o)<br>
              joinType=join<br>
              joinProjectionVars=[?s, ?p, ?o]</td>
          <td>-</td>
          <td>1</td>
          <td>6</td>
          <td>6.00</td>
          <td>1</td>
        </tr>

        <tr>
          <td>2</td>
```

```
          <td>3</td>
          <td>-</td>
          <td>Projection</td>
          <td>vars=[?s, ?p, ?o]</td>
          <td>retain</td>
          <td>6</td>
          <td>6</td>
          <td>1.00</td>
          <td>2</td>
        </tr>

        <tr>
          <td>3</td>
          <td>-</td>
          <td>-</td>
          <td>Slice</td>
          <td>limit=1</td>
          <td>-</td>
          <td>1</td>
          <td>1</td>
          <td>1.00</td>
          <td>1</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

El HTML representa en un navegador algo parecido a lo siguiente:

ID	Out #1	Out #2	Name	Arguments	Mode	Units In	Units Out	Ratio	Time (ms)
0	1	-	SolutionInjection	solutions=[{}]	-	0	1	0.00	0
1	2	-	PipelineJoin	pattern=distinct(?s, ?p, ?o) joinType=join joinProjectionVars=[?s, ?p, ?o]	-	1	6	6.00	1
2	3	-	Projection	vars=[?s, ?p, ?o]	retain	6	6	1.00	2
3	-	-	Slice	limit=1	-	1	1	1.00	1

Ejemplo de salida **`explain`** de SPARQL cuando el DFE está habilitado

El siguiente es un ejemplo de una salida `explain` de SPARQ cuando el motor de consultas alternativo DFE de Neptune está habilitado:

```
##################################################################################################
# ID # Out #1 # Out #2 # Name              # Arguments


 # Mode     # Units In # Units Out # Ratio # Time (ms) #
##################################################################################################
# 0  # 1      # -      # SolutionInjection # solutions=[{}]


 # -        # 0        # 1         # 0.00  # 0         #
##################################################################################################
# 1  # 2      # -      # HashIndexBuild    # solutionSet=solutionSet1


 # -        # 1        # 1         # 1.00  # 22        #
#    #        #        #                   # joinVars=[]


 #          #          #           #       #           #
#    #        #        #                   # sourceType=pipeline


 #          #          #           #       #           #
##################################################################################################
# 2  # 3      # -      # DFENode            # DFE Stats=


      # -        # 101      # 100       # 0.99  # 32        #
#    #        #        #                   # ====> DFE execution time (measured by
 DFEQueryEngine)

     #          #          #           #       #           #
#    #        #        #                   # accepted [micros]=127


 #          #          #           #       #           #
#    #        #        #                   # ready [micros]=2


 #          #          #           #       #           #
#    #        #        #                   # running [micros]=5627
```

```
 #          #          #           #       #           #
#    #        #        #                   # finished [micros]=0


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> DFE execution time (measured in
 DFENode)

      #          #          #           #       #           #
#    #        #        #                   # -> setupTime [ms]=1


 #          #          #           #       #           #
#    #        #        #                   # -> executionTime [ms]=14


 #          #          #           #       #           #
#    #        #        #                   # -> resultReadTime [ms]=0


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   #


 #          #          #           #       #           #
#    #        #        #                   # ===> Static analysis statistics


 #          #          #           #       #           #
#    #        #        #                   # --> 35907 micros spent in parser.
```

```
 #          #          #           #       #           #
#    #        #        #                   # --> 7643 micros spent in range count
 estimation

      #          #          #           #       #           #
#    #        #        #                      # --> 2895 micros spent in value resolution


 #          #          #           #       #           #
#    #        #        #                      #


 #          #          #           #       #           #
#    #        #        #                      # --> 39974925 micros spent in optimizer
 loop

    #          #          #           #       #           #
#    #        #        #                      #


 #          #          #           #       #           #
#    #        #        #                      #


 #          #          #           #       #           #
#    #        #        #                      # DFEJoinGroupNode[ children={


  #          #          #           #       #           #
#    #        #        #                      #   DFEPatternNode[(?1, TERM[117442062], ?
2, ?3) . project DISTINCT[?1, ?2] {rangeCountEstimate=100},

  #          #          #           #       #           #
#    #        #        #                      #     OperatorInfoWithAlternative[


  #          #          #           #       #           #
#    #        #        #                      #       rec=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                      #         type=INCREMENTAL_PIPELINE_JOIN,
```

```
 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0],

                                   #          #          #           #       #
 #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0002,comp=0.0000,mem=0]

                       #          #          #           #       #           #
#    #        #        #                   #       alt=OperatorInfo[


  #          #          #           #       #           #
#    #        #        #                   #         type=INCREMENTAL_HASH_JOIN,


 #          #          #           #       #           #
#    #        #        #                   #
 costEstimates=OperatorCostEstimates[

                                    #          #          #           #       #
  #
#    #        #        #                   #
 costEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=3212],

                                #          #          #           #       #           #
#    #        #        #                   #
 worstCaseCostEstimate=OperatorCostEstimate[in=1.0000,out=100.0000,io=0.0003,comp=0.0000,mem=32

                  #          #          #           #       #           #
#    #        #        #                   #   DFEPatternNode[(?1, TERM[150997262], ?
4, ?5) . project DISTINCT[?1, ?4] {rangeCountEstimate=100},

  #          #          #           #       #           #
#    #        #        #                   #     OperatorInfoWithAlternative[
```

```
  #          #           #           #       #           #
#    #        #        #                  #       rec=OperatorInfo[


  #          #           #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_HASH_JOIN,


 #          #          #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                   #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=6400],

                             #          #          #           #       #           #
#    #        #        #                  #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0003,comp=0.0000,mem=

                #          #          #           #       #           #
#    #        #        #                  #       alt=OperatorInfo[


  #          #           #           #       #           #
#    #        #        #                  #         type=INCREMENTAL_PIPELINE_JOIN,


 #          #          #           #       #           #
#    #        #        #                  #
 costEstimates=OperatorCostEstimates[

                                   #          #          #           #       #
  #
#    #        #        #                  #
 costEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=0],

                               #          #          #           #       #           #
#    #        #        #                  #
 worstCaseCostEstimate=OperatorCostEstimate[in=100.0000,out=100.0000,io=0.0010,comp=0.0000,mem=

                  #          #          #           #       #           #
```

```
#    #        #        #                # },


 #          #          #          #       #          #
#    #        #        #                # ]


 #          #          #          #       #          #
#    #        #        #                #


 #          #          #          #       #          #
#    #        #        #                # ===> DFE configuration:


 #          #          #          #       #          #
#    #        #        #                # solutionChunkSize=5000


 #          #          #          #       #          #
#    #        #        #                # ouputQueueSize=20


 #          #          #          #       #          #
#    #        #        #                # numComputeCores=3


 #          #          #          #       #          #
#    #        #        #                # maxParallelIO=10


 #          #          #          #       #          #
#    #        #        #                # numInitialPermits=12


 #          #          #          #       #          #
#    #        #        #                #


 #          #          #          #       #          #
#    #        #        #                #


 #          #          #          #       #          #
```

```
#    #        #        #                  # ====> DFE configuration (reported back)


 #          #          #           #       #           #
#    #        #        #                  # numComputeCores=3


 #          #          #           #       #           #
#    #        #        #                  # maxParallelIO=2


 #          #          #           #       #           #
#    #        #        #                  # numInitialPermits=12


 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # ===> Statistics & operator histogram


 #          #          #           #       #           #
#    #        #        #                  # ==> Statistics


 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3668 micros total elapsed (incl.
 wait / excl. wait)

 #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 3 millis total elapse (incl.
 wait / excl. wait)

    #          #          #           #       #           #
#    #        #        #                  # -> 3741 / 0 secs total elapsed (incl.
 wait / excl. wait)

     #          #          #           #       #           #
#    #        #        #                  # ==> Operator histogram


 #          #          #           #       #           #
```

```
#    #        #        #                  # -> 47.66% of total time (excl. wait):
 pipelineScan (2 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 10.99% of total time (excl. wait):
 merge (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 41.17% of total time (excl. wait):
 symmetricHashJoin (1 instances)

     #          #          #           #       #           #
#    #        #        #                  # -> 0.19% of total time (excl. wait): drain
 (1 instances)

 #          #          #           #       #           #
#    #        #        #                  #


 #          #          #           #       #           #
#    #        #        #                  # nodeId | out0   | out1 | opName
   | args                                          | rowsIn | rowsOut | chunksIn |
 chunksOut | elapsed* | outWait | outBlocked | ratio    | rate* [M/s] | rate [M/s] | %
    #          #          #           #       #           #
#    #        #        #                  # ------ | ------ | ---- | -----------------
 | ------------------------------------------------ | ------ | ------- | -------- |
 --------- | -------- | ------- | ---------- | -------- | ----------- | ---------- |
 ----- #          #          #           #       #           #
#    #        #        #                  # node_0 | node_2 | -    | pipelineScan
  | (?1, TERM[117442062], ?2, ?3) DISTINCT [?1, ?2]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_1 | node_2 | -    | pipelineScan
  | (?1, TERM[150997262], ?4, ?5) DISTINCT [?1, ?4]  | 0      | 100     | 0        | 1
        | 874      | 0       | 0          | Infinity | 0.1144      | 0.1144     | 23.83
 #          #          #           #       #           #
#    #        #        #                  # node_2 | node_4 | -    | symmetricHashJoin
 |                                                | 200    | 100     | 2        | 2
        | 1510     | 73      | 0          | 0.50     | 0.0662      | 0.0632     | 41.17
 #          #          #           #       #           #
#    #        #        #                  # node_3 | -      | -    | drain
  |                                                | 100    | 0       | 1        | 0
        | 7        | 0       | 0          | 0.00     | 0.0000      | 0.0000     | 0.19
 #          #          #           #       #           #
```

```
#    #        #        #                  # node_4 | node_3 | -    | merge
  |                                                | 100    | 100     | 2        | 1
        | 403      | 0       | 0          | 1.00     | 0.2481      | 0.2481     | 10.99
 #          #          #          #       #          #
##########################################################################################
# 3  # 4      # -      # HashIndexJoin     # solutionSet=solutionSet1


 # -        # 100      # 100       # 1.00  # 4         #
#    #        #        #                  # joinType=join


 #          #          #          #       #          #
##########################################################################################
# 4  # 5      # -      # Distinct          # vars=[?s, ?o, ?o1]


 # -        # 100      # 100       # 1.00  # 9         #
##########################################################################################
# 5  # 6      # -      # Projection        # vars=[?s, ?o, ?o1]


 # retain   # 100      # 100       # 1.00  # 2         #
##########################################################################################
# 6  # -      # -      # TermResolution    # vars=[?s, ?o, ?o1]


 # id2value # 100      # 100       # 1.00  # 11        #
##########################################################################################
```

Operadores **`explain`** de SPARQL en Neptune

En las siguientes secciones, se describen los operadores y los parámetros de la característica `explain` de SPARQL disponible actualmente en Amazon Neptune.

> **Important**
>
> La característica `explain` de SPARQL está en proceso de mejora. Los operadores y los parámetros que se documentan aquí pueden cambiar en versiones futuras.

Temas

- operador Aggregation
- operador ConditionalRouting
- operador Copy
- operador DFENode
- operador Distinct
- operador Federation
- operador Filter
- operador HashIndexBuild
- operador HashIndexJoin
- operador MergeJoin
- operador NamedSubquery
- operador PipelineJoin
- operador PipelineCountJoin
- operador PipelinedHashIndexJoin
- operador Projection
- operador PropertyPath
- operador TermResolution
- operador Slice
- operador SolutionInjection
- operador Sort
- operador VariableAlignment

operador **`Aggregation`**

Realiza una o varias agregaciones, implementando la semántica de los operadores de agregación de SPARQL como `count`, `max`, `min`, `sum`, etc.

`Aggregation` incluye una agrupación opcional que utiliza cláusulas `groupBy` y restricciones `having` opcionales.

Argumentos

- `groupBy`: (opcional) proporciona una cláusula `groupBy` que especifica la secuencia de expresiones según la cual se agrupan las soluciones entrantes.

- `aggregates`: (obligatorio) especifica una lista ordenada de expresiones de agregación.
- `having`: (opcional) añade restricciones para filtrar los grupos, tal y como se implica en la cláusula `having` de la consulta de SPARQL.

operador **`ConditionalRouting`**

Direcciona las soluciones entrantes en función de una condición determinada. Las soluciones que cumplen la condición se direccionan al ID de operador al que hace referencia `Out #1`, mientras que las soluciones que no la cumplen se direccionan al operador al que hace referencia `Out #2`.

Argumentos

- `condition`: (obligatorio) la condición de enrutamiento.

operador **`Copy`**

Delega el flujo de la solución tal y como indica el modo especificado.

Modos

- `forward`: reenvía las soluciones al operador posterior identificado por `Out #1`.
- `duplicate`: duplica las soluciones y las reenvía a cada uno de los dos operadores identificados mediante `Out #1` y `Out #2`.

`Copy` no tiene argumentos.

operador **`DFENode`**

Este operador es una abstracción del plan que ejecuta el motor de consultas alternativo del DFE. El plan detallado del DFE se describe en los argumentos de este operador. Actualmente, el argumento está sobrecargado para contener las estadísticas de tiempo de ejecución detalladas del plan del DFE. Contiene el tiempo empleado en los distintos pasos de la ejecución de la consulta por parte del DFE.

El árbol de sintaxis abstracta (AST) lógico optimizado para el plan de consultas del DFE se imprime con información sobre los tipos de operadores que se tuvieron en cuenta en la planificación y los costos de funcionamiento de los operadores, en el mejor y peor de los casos. Por el momento, el AST consta de los siguientes tipos de nodos:

- `DFEJoinGroupNode`: representa una unión de uno o más `DFEPatternNodes`.
- `DFEPatternNode`: encapsula un patrón subyacente mediante el cual las tuplas coincidentes se proyectan fuera de la base de datos subyacente.

La subsección, `Statistics & Operator histogram`, contiene detalles sobre el tiempo de ejecución del plan `DataflowOp` y el desglose del tiempo de CPU utilizado por cada operador. Debajo hay una tabla que muestra estadísticas detalladas de tiempo de ejecución del plan ejecutado por el DFE.

> **Note**
>
> Dado que el DFE es una característica experimental publicada en el modo de laboratorio, el formato exacto de la salida de `explain` está sujeto a cambios.

operador **`Distinct`**

Calcula la proyección distintiva en un subconjunto de las variables, eliminando los duplicados. Como resultado, el número de soluciones entrantes es mayor o igual que el número de soluciones salientes.

Argumentos

- `vars`: (obligatorio) las variables a las que se va a aplicar la proyección `Distinct`.

operador **`Federation`**

Pasa una consulta especificada a un punto de enlace SPARQL remoto especificado.

Argumentos

- `endpoint`: (obligatorio) la URL del punto de conexión de la instrucción `SERVICE` de SPARQL. Puede ser una cadena constante o si el punto de enlace de consulta se determina en función de una variable dentro de la misma consulta, puede ser el nombre de la variable.
- `query`: (obligatorio) la cadena de consulta reconstruida que se enviará al punto de conexión remoto. El motor añade prefijos predeterminados a esta consulta incluso cuando el cliente no especifica ninguno.

- `silent`: (obligatorio) un booleano que indica si la palabra clave `SILENT` ha aparecido después de la palabra clave. `SILENT` le dice al motor que no falle en toda la consulta, incluso si la parte `SERVICE` remota falla.

operador **`Filter`**

Filtra las soluciones entrantes. Solo las soluciones que satisfacen la condición de filtro se reenvían al operador anterior; todas las demás se descartan.

Argumentos

- `condition`: (obligatorio) el estado del filtro.

operador **`HashIndexBuild`**

Toma una lista de enlaces y los incorpora a un índice hash cuyo nombre viene definido por el argumento `solutionSet`. Normalmente, los operadores posteriores realizan uniones en este conjunto de soluciones, haciendo referencia a él por ese nombre.

Argumentos

- `solutionSet`: (obligatorio) el nombre del conjunto de soluciones de índice hash.
- `sourceType`: (obligatorio) el tipo de origen del que se obtienen los enlaces que se van a almacenar en el índice hash:
  - `pipeline`: reúne las soluciones entrantes desde el operador que hay más abajo en la canalización del operador en el índice hash.
  - `binding set`: coloca el conjunto de enlaces fijos especificado por el argumento `sourceBindingSet` en el índice hash.
- `sourceBindingSet`: (opcional) si el valor del argumento `sourceType` es `binding set`, este argumento especifica el conjunto de enlaces estáticos que se incluirá en el índice hash.

operador **`HashIndexJoin`**

Une las soluciones entrantes con el conjunto de soluciones del índice hash identificado por el argumento `solutionSet`.

Argumentos

- `solutionSet`: (obligatorio) nombre del conjunto de soluciones en el que se va a unir. Debe ser un índice hash que se haya construido en un paso anterior con el operador `HashIndexBuild`.
- `joinType`: (obligatorio) tipo de unión que se va a realizar:
  - `join`: unión normal, que requiere una coincidencia exacta de todas las variables compartidas.
  - `optional`: unión `optional` que utiliza la semántica del operador `OPTIONAL` de SPARQL.
  - `minus`: una operación `minus` conserva un mapeo para el que no existe ningún socio de unión, utilizando la semántica del operador `MINUS` de SPARQL.
  - `existence check`: comprueba si hay un socio de unión o no y vincula la variable `existenceCheckResultVar` al resultado de esta comprobación.
- `constraints`: (opcional) restricciones de unión adicionales que se tienen en cuenta durante la unión. Las uniones que no satisfacen estas restricciones se descartan.
- `existenceCheckResultVar`: (opcional) solo se usa para uniones donde `joinType` es igual a `existence check` (consulte el argumento `joinType` anterior).

operador **`MergeJoin`**

Una unión de fusión entre varios conjuntos de soluciones, que se indican en el argumento `solutionSets`.

Argumentos

- `solutionSets`: (obligatorio) conjuntos de la solución que van a unirse.

operador **`NamedSubquery`**

Activa la evaluación de la subconsulta indicada por el argumento `subQuery` e incorpora el resultado al conjunto de soluciones especificado por el argumento `solutionSet`. Las soluciones entrantes para el operador se reenvían a la subconsulta y, a continuación, al siguiente operador.

Argumentos

- `subQuery`: (obligatorio) nombre de la subconsulta que se va a evaluar. La subconsulta se representa explícitamente en la salida.
- `solutionSet`: (obligatorio) nombre del conjunto de soluciones en el que se almacenará el resultado de la subconsulta.

operador **`PipelineJoin`**

Recibe como entrada la salida del operador anterior y la une con el patrón de tuplas definido por el argumento `pattern`.

Argumentos

- `pattern`— (Obligatorio) El patrón, que adopta la forma de una tupla gráfica y subject-predicate-object, opcionalmente, una tupla gráfica que subyace a la unión. Si se especifica `distinct` para el patrón, la unión únicamente extrae soluciones distintas de las variables de proyección especificadas por el argumento `projectionVars`, en lugar de extraer todas las soluciones coincidentes.
- `inlineFilters`: (opcional) conjunto de filtros que se aplicarán a las variables del patrón. El patrón se evalúa junto con estos filtros.
- `joinType`: (obligatorio) tipo de unión que se va a realizar:
  - `join`: unión normal, que requiere una coincidencia exacta de todas las variables compartidas.
  - `optional`: unión `optional` que utiliza la semántica del operador `OPTIONAL` de SPARQL.
  - `minus`: una operación `minus` conserva un mapeo para el que no existe ningún socio de unión, utilizando la semántica del operador `MINUS` de SPARQL.
  - `existence check`: comprueba si hay un socio de unión o no y vincula la variable `existenceCheckResultVar` al resultado de esta comprobación.
- `constraints`: (opcional) restricciones de unión adicionales que se tienen en cuenta durante la unión. Las uniones que no satisfacen estas restricciones se descartan.
- `projectionVars`: (opcional) variables de proyección. Se utiliza junto con `distinct := true` para aplicar la extracción de proyecciones distintas en un conjunto de variables especificado.
- `cutoffLimit`: (opcional) límite del número de socios de unión extraídos. Aunque no existe ningún límite de forma predeterminada, puede establecer este en 1 al realizar uniones para implementar cláusulas `FILTER (NOT) EXISTS`, donde es suficiente con demostrar o refutar que hay un socio de unión.

operador **`PipelineCountJoin`**

Variante del operador `PipelineJoin`. En lugar de realizar la unión, solo cuenta los socios de unión coincidentes y vincula el recuento a la variable especificada por el argumento `countVar`.

Argumentos

- `countVar`: (obligatorio) variable a la que se debe vincular el resultado del recuento, es decir, el número de socios que se unen.
- `pattern`— (Obligatorio) El patrón, que adopta la forma de una tupla gráfica y subject-predicate-object, opcionalmente, una tupla gráfica que subyace a la unión. Si se especifica `distinct` para el patrón, la unión únicamente extrae soluciones distintas de las variables de proyección especificadas por el argumento `projectionVars`, en lugar de extraer todas las soluciones coincidentes.
- `inlineFilters`: (opcional) conjunto de filtros que se aplicarán a las variables del patrón. El patrón se evalúa junto con estos filtros.
- `joinType`: (obligatorio) tipo de unión que se va a realizar:
  - `join`: unión normal, que requiere una coincidencia exacta de todas las variables compartidas.
  - `optional`: unión `optional` que utiliza la semántica del operador `OPTIONAL` de SPARQL.
  - `minus`: una operación `minus` conserva un mapeo para el que no existe ningún socio de unión, utilizando la semántica del operador `MINUS` de SPARQL.
  - `existence check`: comprueba si hay un socio de unión o no y vincula la variable `existenceCheckResultVar` al resultado de esta comprobación.
- `constraints`: (opcional) restricciones de unión adicionales que se tienen en cuenta durante la unión. Las uniones que no satisfacen estas restricciones se descartan.
- `projectionVars`: (opcional) variables de proyección. Se utiliza junto con `distinct := true` para aplicar la extracción de proyecciones distintas en un conjunto de variables especificado.
- `cutoffLimit`: (opcional) límite del número de socios de unión extraídos. Aunque no existe ningún límite de forma predeterminada, puede establecer este en 1 al realizar uniones para implementar cláusulas `FILTER (NOT) EXISTS`, donde es suficiente con demostrar o refutar que hay un socio de unión.

operador **`PipelinedHashIndexJoin`**

Se trata de un índice hash de all-in-one creación y un operador de unión. Toma una lista de enlaces, los agrupa en un índice hash y, a continuación, une las soluciones entrantes con el índice hash.

Argumentos

- `sourceType`: (obligatorio) el tipo de origen del que se obtienen los enlaces que se van a almacenar en el índice hash. Es uno de los siguientes:

  - `pipeline`: hace que `PipelinedHashIndexJoin` agrupe las soluciones entrantes desde el operador que está más abajo en la canalización del operador en el índice hash.
  - `binding set`: hace que `PipelinedHashIndexJoin` agrupe el conjunto de enlaces fijos especificado por el argumento `sourceBindingSet` en el índice hash.
- `sourceSubQuery` : (opcional) si el valor del argumento `sourceType` es `pipeline`, este argumento especifica la subconsulta que se evalúa y se incluye en el índice hash.
- `sourceBindingSet` : (opcional) si el valor del argumento `sourceType` es `binding set`, este argumento especifica el conjunto de enlaces estáticos que se incluirá en el índice hash.
- `joinType`: (obligatorio) tipo de unión que se va a realizar:
  - `join`: unión normal, que requiere una coincidencia exacta de todas las variables compartidas.
  - `optional`: unión `optional` que utiliza la semántica del operador `OPTIONAL` de SPARQL.
  - `minus`: una operación `minus` conserva un mapeo para el que no existe ningún socio de unión, utilizando la semántica del operador `MINUS` de SPARQL.
  - `existence check`: comprueba si hay un socio de unión o no y vincula la variable `existenceCheckResultVar` al resultado de esta comprobación.
- `existenceCheckResultVar`: (opcional) solo se usa para uniones donde `joinType` es igual a `existence check` (consulte el argumento joinType anterior).

operador **`Projection`**

Realiza una proyección en un subconjunto de las variables. El número de soluciones entrantes es igual al número de soluciones salientes, pero la forma de la solución es diferente, en función de la configuración de modo.

Modos

- `retain`: en las soluciones se conservan únicamente las variables especificadas por el argumento `vars`.
- `drop`: se eliminan todas las variables especificadas por el argumento `vars`.

Argumentos

- `vars`: (obligatorio) las variables que se deben conservar o eliminar, según la configuración del modo.

operador **`PropertyPath`**

Habilita las rutas de propiedades recursivas como `+` o `*`. Neptune implementa un enfoque de iteración de punto fijo basado en una plantilla especificada por el argumento `iterationTemplate`. Las variables del lado izquierdo o del lado derecho conocidas se incluyen en la plantilla para cada iteración de punto fijo hasta que no se encuentren más soluciones nuevas.

Argumentos

- `iterationTemplate`: (obligatorio) nombre de la plantilla de subconsulta utilizada para implementar la iteración de punto fijo.
- `leftTerm`: (obligatorio) el término (variable o constante) que se encuentra en el lado izquierdo de la ruta de la propiedad.
- `rightTerm`: (obligatorio) el término (variable o constante) que se encuentra en el lado derecho de la ruta de la propiedad.
- `lowerBound`: (obligatorio) el límite inferior de la iteración de punto fijo (ya sea `0` para consultas `*` o `1` para consultas `+`).

operador **`TermResolution`**

Traduce de nuevo los valores de identificador de cadena internos a sus cadenas externas correspondientes o traduce las cadenas externas a valores de identificador de cadena internos, en función del modo.

Modos

- `value2id`: mapea términos como literales y URI a los valores de identificadores internos correspondientes (codificación en valores internos).
- `id2value`: mapea los valores de identificadores internos a los términos correspondientes, como literales y URI (decodificación de valores internos).

Argumentos

- `vars`: (obligatorio) especifica las variables cuyas cadenas o identificadores de cadenas internas deben mapearse.

operador **`Slice`**

Implementa un sector en el flujo de soluciones entrantes, utilizando la semántica de las cláusulas `LIMIT` y `OFFSET` de SPARQL.

Argumentos

- `limit`: (opcional) límite en las soluciones que se van a reenviar.
- `offset`: (opcional) desfase con el que se evalúan las soluciones para su reenvío.

operador **`SolutionInjection`**

No recibe ninguna entrada. Inyecta soluciones de forma estática en el plan de consulta y las registra en el argumento `solutions`.

Los planes de consulta siempre comienzan con esta inyección estática. Si las soluciones estáticas que se van a inyectar se pueden obtener de la propia consulta uniendo distintas fuentes de enlaces estáticos (por ejemplo, de las cláusulas `VALUES` o `BIND`), el operador `SolutionInjection` inyecta estas soluciones estáticas derivadas. En el caso más sencillo, estas soluciones reflejan enlaces que se encuentran implícitos en una cláusula `VALUES` exterior.

Si no se puede obtener ninguna solución estática de la consulta, `SolutionInjection` inyecta lo que se denominada la solución universal vacía, que se expande y multiplica en todo el proceso de evaluación de consultas.

Argumentos

- `solutions`: (obligatorio) secuencia de soluciones que inyecta el operador.

operador **`Sort`**

Ordena el conjunto de soluciones utilizando las condiciones de ordenación especificadas.

Argumentos

- `sortOrder`: (obligatorio) lista ordenada de variables, cada una de las cuales contiene un identificador `ASC` (ascendente) o `DESC` (descendente), que se utiliza secuencialmente para ordenar el conjunto de soluciones.

operador **`VariableAlignment`**

Inspecciona las soluciones de una en una, realizando la alineación en cada una de ellas con respecto a dos variables especificadas: `sourceVar` y `targetVar`.

Si `sourceVar` y `targetVar` tienen el mismo valor en una solución, las variables se consideran alineadas y la solución se reenvía con la variable `sourceVar` redundante proyectada.

Si las variables se vinculan a valores diferentes, la solución se excluye por completo.

Argumentos

- `sourceVar`: (obligatorio) variable de origen, que se va a comparar con la variable de destino. Si la alineación se realiza correctamente en una solución, lo que significa que las dos variables tienen el mismo valor, la variable de origen se proyecta.
- `targetVar`: (obligatorio) variable de destino con la que se compara la variable de origen. Se conserva incluso si la alineación se realiza correctamente.

Limitaciones de **`explain`** de SPARQL en Neptune

La versión de la característica `explain` de SPARQL en Neptune tiene las siguientes limitaciones.

Actualmente, Neptune solo admite explain en las consultas SELECT de SPARQL

Para obtener información sobre el proceso de evaluación para otros formatos de consulta, como por ejemplo las consultas `ASK`, `CONSTRUCT`, `DESCRIBE` y `SPARQL UPDATE`, puede transformar estas consultas en una consulta SELECT. A continuación, utilice `explain` para inspeccionar la consulta SELECT correspondiente.

Por ejemplo, para obtener información de `explain` sobre una consulta `ASK WHERE {...}`, ejecute la consulta `SELECT WHERE {...} LIMIT 1` correspondiente con `explain`.

Del mismo modo, para una consulta `CONSTRUCT {...} WHERE {...}`, elimine la parte `CONSTRUCT {...}` y ejecute una consulta `SELECT` con `explain` en la segunda cláusula `WHERE {...}`. Normalmente, la evaluación de la segunda cláusula `WHERE` revela los principales desafíos del procesamiento de la consulta `CONSTRUCT`, ya que las soluciones que fluyen de la segunda cláusula `WHERE` a la plantilla `CONSTRUCT` solo suelen requerir una sustitución sencilla.

Los operadores de explain pueden cambiar en versiones futuras

Los operadores de `explain` de SPARQL y sus parámetros pueden cambiar en versiones futuras.

La salida de explain puede cambiar en versiones futuras

Por ejemplo, los encabezados de columna pueden cambiar y es posible que se añadan más columnas a las tablas.

Consultas federadas de SPARQL en Neptune mediante la extensión `SERVICE`

Amazon Neptune admite totalmente la extensión de consulta federada de SPARQL que utiliza la palabra clave `SERVICE`. Para obtener más información, consulte Consulta federada de SPARQL 1.1.

> **Note**
>
> Esta característica está disponible a partir de Versión 1.0.1.0.200463.0 (15/10/2019).

La palabra clave `SERVICE` indica al motor de consultas SPARQL que ejecute una parte de la consulta en un punto de enlace SPARQL remoto y que componga el resultado final de la consulta. Solo son posibles las operaciones `READ`. Las operaciones `WRITE` y `DELETE` no se admiten. Neptune solo puede ejecutar consultas federadas con respecto a puntos de conexión de SPARQL a los que se pueda acceder desde su nube privada virtual (VPC). Sin embargo, también puede utilizar un proxy inverso en la VPC para hacer accesible un origen de datos externo dentro de la VPC.

> **Note**
>
> Cuando se utiliza `SERVICE` de SPARQ para federar una consulta en dos o más clústeres de Neptune situados en la misma VPC, los grupos de seguridad deben estar configurados de modo que permitan que todos esos clústeres de Neptune se comuniquen entre sí.

> **Important**
>
> La federación SPARQL 1.1 realiza solicitudes de servicio en su nombre al pasar consultas y parámetros a puntos de enlace SPARQL externos. Es su responsabilidad verificar que los puntos de enlace SPARQL externos satisfagan los requisitos de seguridad y gestión de datos de su aplicación.

Ejemplo de una consulta federada de Neptune

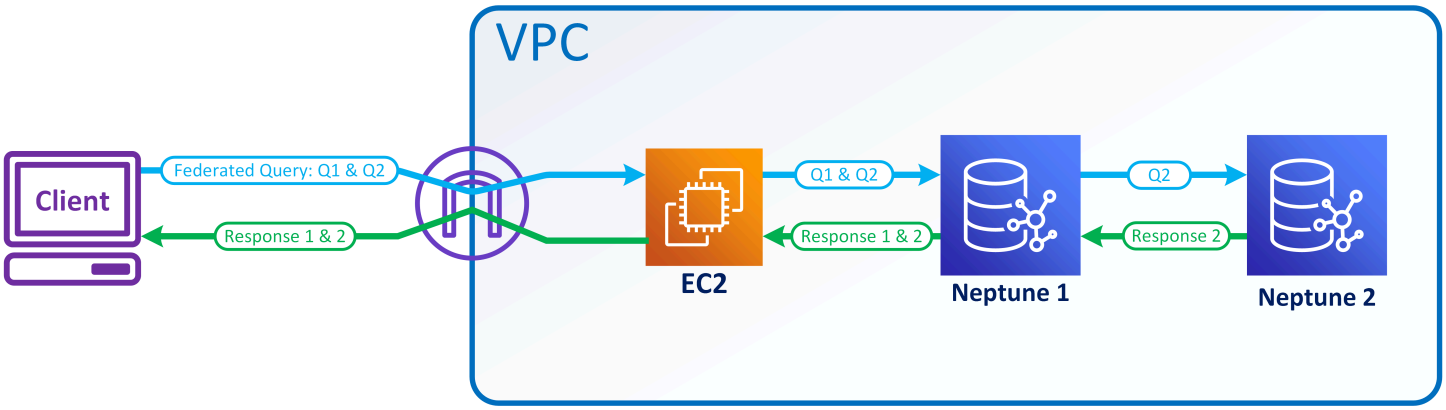
El siguiente ejemplo sencillo muestra cómo funcionan las consultas federadas SPARQL.

Supongamos que un cliente envía la siguiente consulta a Neptune-1 en `http://neptune-1:8182/sparql`.

```
SELECT * WHERE {
   ?person rdf:type foaf:Person .
   SERVICE <http://neptune-2:8182/sparql> {
       ?person foaf:knows ?friend .
    }
}
```

1. Neptune-1 evalúa el primer patrón de consulta (Q-1), que es `?person rdf:type foaf:Person`, utiliza los resultados para resolver `?person` en Q-2 (`?person foaf:knows ?friend`) y reenvía el patrón a Neptune-2 en `http://neptune-2:8182/sparql`.
2. Neptune-2 evalúa Q-2 y vuelve a enviar los resultados a Neptune-1.
3. Neptune-1 combina las soluciones para ambos patrones y vuelve a enviar los resultados al cliente.

Este flujo se muestra en el siguiente diagrama.



> **Note**
>
> De forma predeterminada, el optimizador determina en qué punto de la ejecución de la consulta se ejecuta la instrucción `SERVICE`. Puede anular esta ubicación utilizando la sugerencia de consulta joinOrder.

Control de acceso para consultas federadas en Neptune

Neptune usa AWS Identity and Access Management (IAM) para la autenticación y la autorización. El control de acceso de una consulta federada puede implicar más de una instancia de base de datos de Neptune. Estas instancias pueden tener diferentes requisitos de control de acceso. En determinadas circunstancias, esto puede limitar su capacidad para realizar una consulta federada.

Analice el sencillo ejemplo que se presenta en la sección anterior. Neptune-1 llama a Neptune-2 con las mismas credenciales con las que se ha llamado.

- Si Neptune-1 requiere autenticación y autorización de IAM, pero Neptune-2 no, lo único que necesita son los permisos de IAM adecuados para que Neptune-1 realice la consulta federada.
- Si Neptune-1 y Neptune-2 requieren autenticación y autorización de IAM, debe asociar permisos de IAM para ambas bases de datos para realizar la consulta federada. Ambos clústeres también deben estar en la misma AWS cuenta y en la misma región. Actualmente, no se admiten las arquitecturas de consultas federadas entre regiones o entre cuentas.
- Sin embargo, en el caso en el que Neptune-1 no disponga de IAM, pero Neptune-2 sí, no podrá realizar una consulta federada. El motivo es que Neptune-1 no puede recuperar sus credenciales de IAM y pasarlas a Neptune-2 para autorizar la segunda parte de la consulta.

Herramientas de visualización de gráficos para Neptune

Además de las capacidades de visualización integradas en los cuadernos gráficos de Neptune, también puede utilizar soluciones creadas por AWS socios y proveedores externos para visualizar los datos almacenados en Neptuno.

La sofisticada visualización de gráficos puede ayudar a los científicos de datos, los administradores y otros roles de una organización a explorar los datos de gráficos de forma interactiva, sin tener que saber escribir consultas complejas.

Temas

Graph-explorer de código abierto

Graph-explorer es una herramienta de exploración visual de código abierto y de bajo código para datos de gráficos, que está disponible con la licencia Apache-2.0. Permite explorar datos de gráficos de propiedades etiquetadas (LPG) o del marco de descripción de recursos (RDF) en una base de datos de gráficos sin tener que escribir consultas gráficas. Graph-explorer está diseñado para ayudar a los científicos de datos, analistas de negocio y otros roles de una organización a explorar los datos de los gráficos de forma interactiva sin tener que aprender un lenguaje de consulta de gráficos.

Graph-explorer proporciona una aplicación web basada en React que se puede implementar como un contenedor para visualizar los datos de los gráficos. Puede conectarse a Amazon Neptune o a otras bases de datos de gráficos que proporcionen un punto final Apache TinkerPop Gremlin o SPARQL 1.1.

- Puede ver rápidamente un resumen de los datos mediante los filtros por facetas o buscar los datos escribiendo texto en la barra de búsqueda.

- También puede explorar las conexiones de nodos y ejes de forma interactiva. Puede ver los nodos vecinos para comprobar cómo se relacionan los objetos entre sí y, a continuación, profundizar para inspeccionar visualmente los bordes y las propiedades.
- También puede personalizar el diseño del gráfico, los colores, los iconos y las propiedades predeterminadas que se mostrarán en los nodos y los bordes. En el caso de los gráficos RDF, también puede personalizar los espacios de nombres para las URI de los recursos.
- Para los informes y presentaciones que incluyan datos de gráficos, puede configurar y guardar las vistas que haya creado en formato PNG de alta resolución. También puede descargar los datos asociados a un archivo CSV o JSON para su posterior procesamiento.

Uso del explorador de gráficos en un cuaderno de gráficos de Neptuno

La forma más fácil de usar el explorador de gráficos con Neptuno es en un cuaderno de gráficos de Neptuno.

Si utiliza un entorno de trabajo de Neptune para alojar un cuaderno de Neptuno, el explorador de gráficos se implementa automáticamente con el cuaderno y se conecta a Neptuno.

Una vez que haya creado un cuaderno, vaya a la consola de Neptune para iniciar el explorador de gráficos:

1. Vaya a Neptune.
2. Seleccione su cuaderno en Cuadernos.
3. En Acciones, seleccione Abrir el explorador de gráficos.

Cómo ejecutar Graph-explorer de Amazon ECS en AWS Fargate y conectarse a Neptune

También puede crear la imagen Docker del explorador de gráficos y ejecutarla en una máquina local o en un servicio hospedado, como Amazon Elastic Compute Cloud (Amazon EC2) o AmazonElastic Container Service (Amazon ECS), tal y como se explica en la sección Introducción del proyecto read-me in the graph-explorer. GitHub

A modo de ejemplo, en esta sección se proporcionan step-by-step instrucciones para ejecutar el explorador de gráficos en Amazon ECS en: AWS Fargate

1. Cree un nuevo rol de IAM y asóciele estas políticas:

- AmazonECS TaskExecution RolePolicy
- CloudWatchLogsFullAcceso

Tenga a mano el nombre del rol, ya que lo utilizará dentro de un momento.

2. Cree un clúster de Amazon ECS con la infraestructura configurada en FARGATE y las siguientes opciones de red:

   - VPC: se establece en la VPC en la que se encuentra la base de datos de Neptune.
   - Subnets: se establece en las subredes públicas de esa VPC (se eliminan todas las demás).

3. Cree una nueva definición de tarea JSON de la siguiente manera:

```
{
  "family": "explorer-test",
  "containerDefinitions": [
    {
      "name": "graph-explorer",
      "image": "public.ecr.aws/neptune/graph-explorer:latest",
      "cpu": 0,
      "portMappings": [
        {
          "name": "graph-explorer-80-tcp",
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp",
          "appProtocol": "http"
        },
        {
          "name": "graph-explorer-443-tcp",
          "containerPort": 443,
          "hostPort": 443,
          "protocol": "tcp",
          "appProtocol": "http"
        }
      ],
      "essential": true,
      "environment": [
        {
          "name": "HOST",
          "value": "localhost"
        }
```

```
      ],
      "mountPoints": [],
      "volumesFrom": [],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "true",
          "awslogs-group": "/ecs/graph-explorer",
          "awslogs-region": "{region}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "taskRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "executionRoleArn": "arn:aws:iam::{account_no}:role/{role_name_from_step_1}",
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072",
  "runtimePlatform": {
    "cpuArchitecture": "X86_64",
    "operatingSystemFamily": "LINUX"
  }
}
```

4. Inicie una nueva tarea con la configuración predeterminada, excepto en los campos siguientes:

   - Entorno
     - Opciones de computación => Tipo de lanzamiento
   - Configuración de implementación
     - Tipo de aplicación => Tarea
     - Familia => *(su nueva definición de tarea JSON)*
     - Revisión => *(más reciente)*
   - Redes
     - VPC => *(la VPC de Neptune a la que quiere conectarse)*
     - Subredes => *(SOLO las subredes públicas de la VPC; elimine todas las demás)*

- Grupo de seguridad => Crear un nuevo grupo de seguridad
- Nombre del grupo de seguridad => graph-explorer
- Descripción del grupo de seguridad = Grupo de seguridad para acceder a graph-explorer
- Reglas de entrada para grupos de seguridad =>
  1. 80 Anywhere
  2. 443 Anywhere

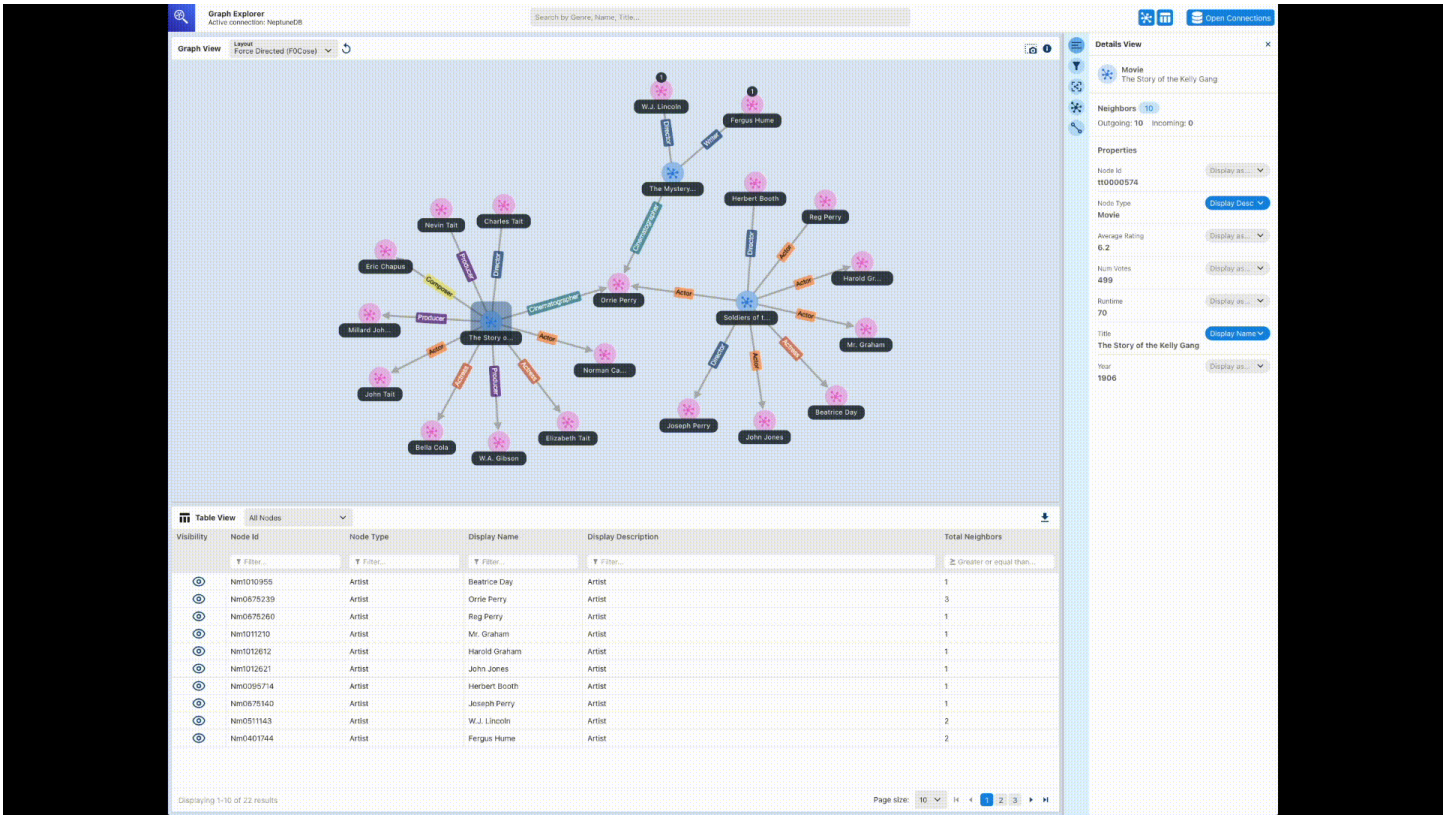
5. Seleccione Crear.
6. Una vez iniciada la tarea, copie la IP pública de la tarea en ejecución y navegue hasta: `https://(your public IP)/explorer`.
7. Acepte el riesgo de utilizar el certificado no reconocido que se ha generado o añádalo a su cadena de claves.
8. Ahora puede añadir una conexión a Neptune. Cree una nueva conexión, ya sea para un gráfico de propiedades (LPG) o para un RDF, y defina los siguientes campos:

```
Using proxy server => true
Public or Proxy Endpoint => https://(your public IP address)
Graph connection URL => https://(your Neptune endpoint):8182
```

Ahora debería estar conectado.

Demostración de Graph-explorer

Este breve vídeo le da una idea de cómo puede ver fácilmente los datos de sus gráficos con Graph-explorer:



Software Tom Sawyer

Tom Sawyer Perspectives es una plataforma de desarrollo de análisis y visualización de gráficos y datos de bajo código para los datos almacenados en Amazon Neptune. Las interfaces integradas de diseño y vista previa y las amplias bibliotecas de API le permiten crear rápidamente aplicaciones de visualización personalizadas y con calidad de producción. Con una interfaz de point-and-click diseño y 30 algoritmos de análisis integrados, puede diseñar y desarrollar aplicaciones para obtener información sobre los datos federados de docenas de fuentes.

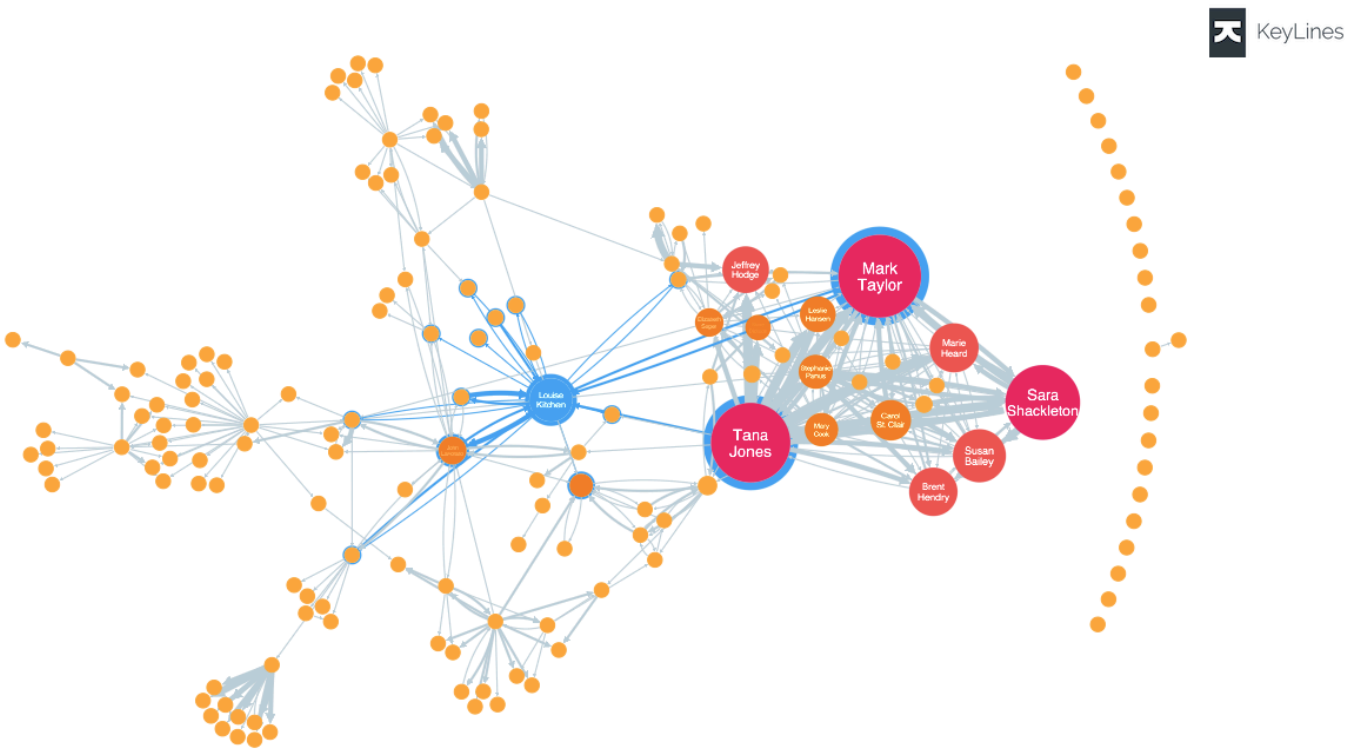
Tom Sawyer Graph Database Browser facilita la visualización y el análisis de los datos en Amazon Neptune. Puede ver y comprender las conexiones en sus datos sin tener un conocimiento exhaustivo del lenguaje o esquema de consulta. Puede interactuar con los datos sin conocimientos técnicos, simplemente cargando los nodos vecinos de los nodos seleccionados y creando la visualización en la dirección que necesite. También puede aprovechar cinco diseños de gráficos exclusivos para mostrar el gráfico de una manera que proporcione más datos, y puede aplicar análisis de centralidad, agrupamiento y búsqueda de rutas para revelar patrones que antes no se habían detectado. Para ver un ejemplo de la integración de Graph Database Browser con Neptune, consulte esta publicación del blog. Para empezar con una versión de prueba gratuita de Graph Database Browser, visite el AWS Marketplace.

Cambridge Intelligence

Cambridge Intelligence proporciona tecnologías de visualización de datos para explorar y comprender los datos de Amazon Neptune. Los kits de herramientas de visualización de gráficos (KeyLinespara JavaScript desarrolladores y ReGraphdesarrolladores de React) ofrecen una forma sencilla de crear herramientas altamente interactivas y personalizables para aplicaciones web. Estos kits de herramientas utilizan WebGL y HTML5 Canvas para conseguir un rendimiento rápido, admiten funciones avanzadas de análisis de gráficos y combinan flexibilidad y escalabilidad con una arquitectura segura y sólida. Estos SDK funcionan con datos de Gremlin y RDF de Neptune.

Consulte estos tutoriales de integración para datos de Gremlin, datos de SPARQL y la arquitectura de Neptune.

He aquí un ejemplo de KeyLines visualización:

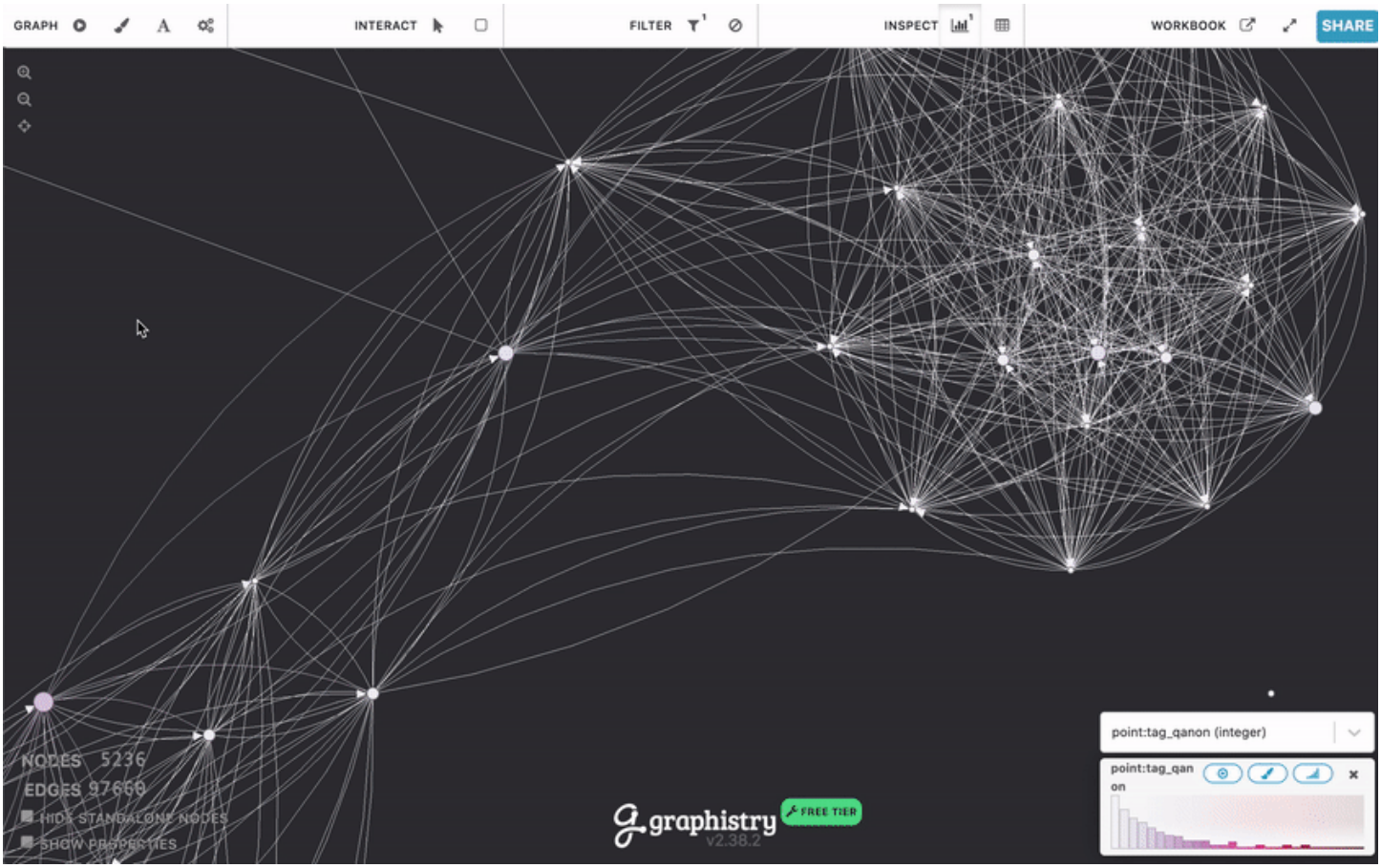


Graphistry

Graphistry es una plataforma de inteligencia gráfica visual que aprovecha la aceleración de la GPU para ofrecer experiencias visuales enriquecidas. Los equipos pueden colaborar en Graphistry utilizando una variedad de características, desde la exploración sin código de archivos y bases de datos, hasta el uso compartido de cuadernos de Jupyter y paneles de Streamlit, o el uso de la API integrada en sus propias aplicaciones.

Puede empezar con paneles totalmente interactivos y de bajo código simplemente configurando e iniciando el graph-app-kit y modificando solo unas pocas líneas de código. Consulte esta publicación del blog para ver un tutorial sobre cómo crear su primer panel de control con Graphistry y Neptune. También puedes probar la demo de Neptune PyGraphistry. PyGraphistry es una biblioteca de análisis de gráficos visuales de Python para cuadernos. Consulta este cuaderno tutorial para ver una demostración de Neptune PyGraphistry .

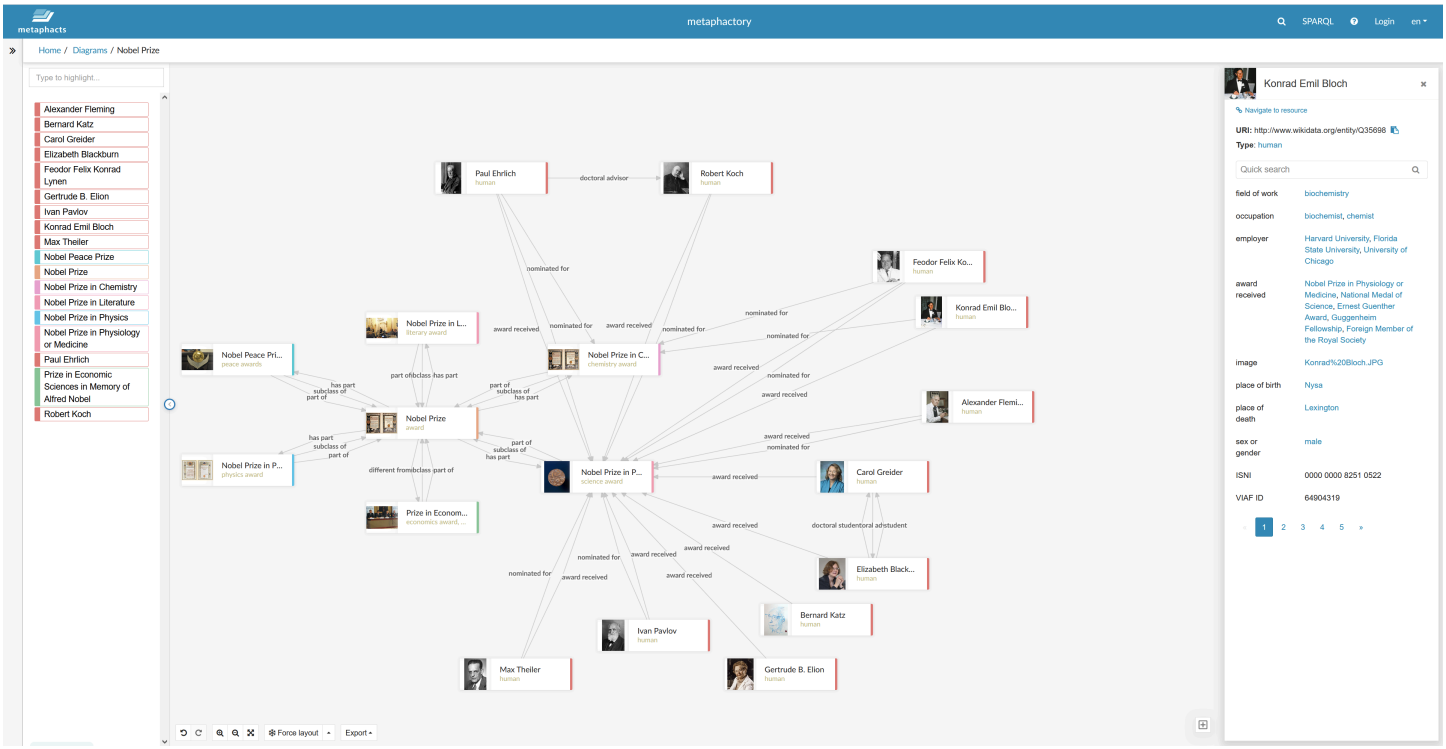
Para empezar, visita Graphistry in the AWS Marketplace.



metaphacts

metaphacts ofrece una plataforma abierta y flexible para describir y consultar datos de gráficos y para visualizar gráficos de conocimiento e interactuar con ellos. Con metaphactory, puede crear aplicaciones web interactivas, como visualizaciones y paneles, sobre gráficos de conocimiento de Neptune utilizando el modelo de datos RDF. La plataforma de metaphactory ofrece una experiencia de desarrollo de bajo código, con una interfaz de usuario para cargar datos, una interfaz visual de modelado de ontología compatible con OWL y SHACL, una IU y un catálogo de consultas de SPARQL, y un amplio conjunto de componentes web para la exploración, visualización, búsqueda y creación de gráficos.

A continuación se ofrece un ejemplo de visualización de metaphactory:



La plataforma está diseñada para utilizarse de manera productiva en ingeniería, fabricación, farmacia, ciencias biológicas, finanzas, seguros y mucho más. Para ver un ejemplo de la arquitectura de una solución, consulte esta publicación del blog.

Para empezar una versión de prueba gratuita de metaphactory, visite el AWS Marketplace.

G.V( )

G.V () es una poderosa herramienta de entorno de desarrollo integrado (IDE) de Gremlin para desarrolladores y analistas de datos. Con ella, puede consultar, visualizar y actualizar datos de gráficos de forma interactiva en Neptune. G.V () ofrece una funcionalidad de autocompletado integrada en el lenguaje Gremlin, que proporciona sugerencias y documentación a medida que escribe la consulta, en función del modelo de datos de su gráfico.

También puede utilizar la característica de depuración de consultas de Gremlin para escribir, depurar, probar y analizar en profundidad los procesos de recorrido de gráficos.

Con el procesamiento del lenguaje natural impulsado por OpenAI, G.V () puede generar consultas Gremlin con precisión para el esquema de datos de su gráfico a partir de un mensaje de texto, para consultar sus datos mediante un lenguaje natural.

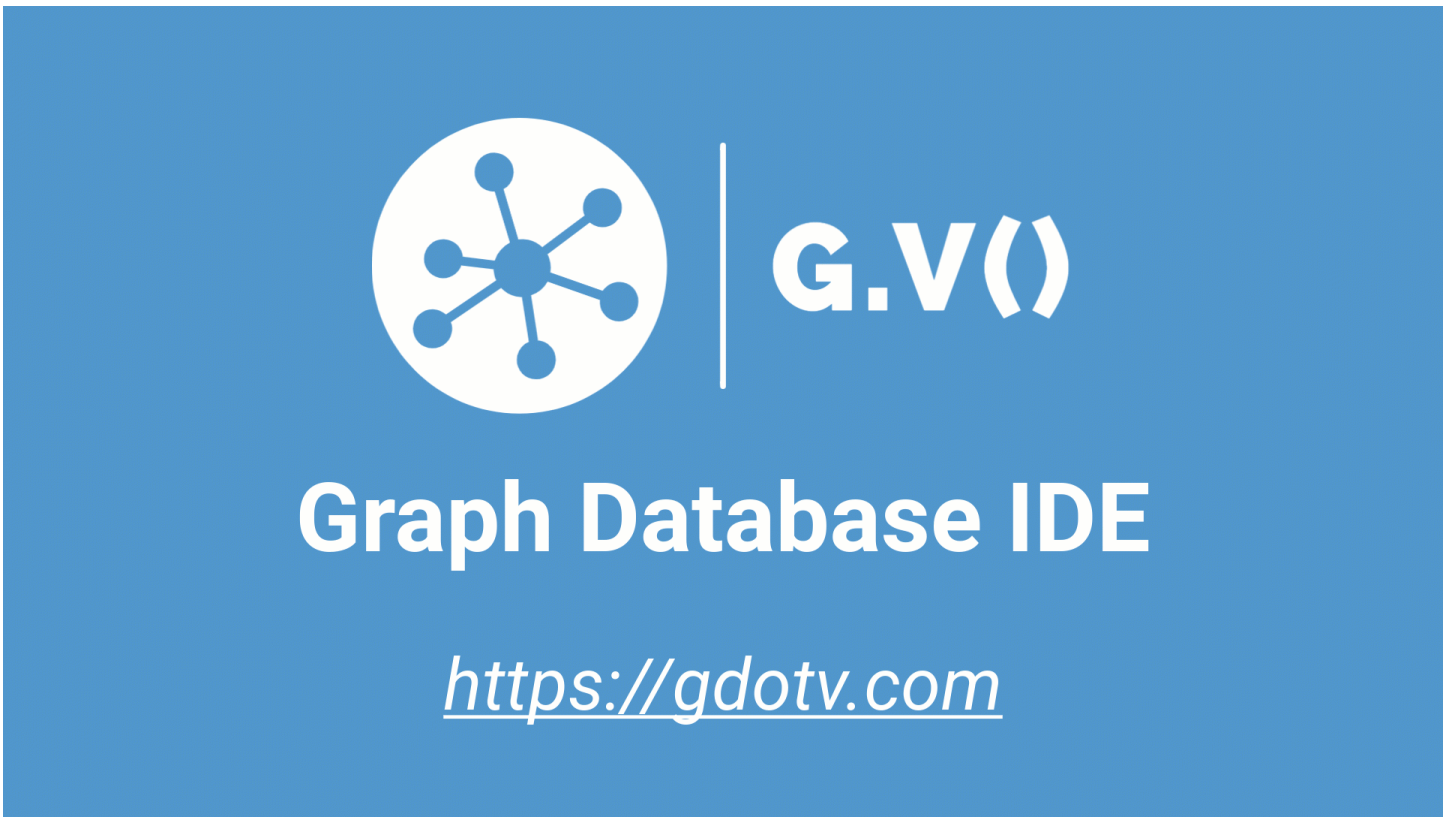
El explorador de datos gráficos te permite navegar por tu gráfico y modificarlo para diseñar rápidamente nuevas estructuras gráficas y mantener las existentes.

G.V () ofrece varios formatos de visualización para los resultados de las consultas que le ayudan a interpretar el resultado de la consulta y a navegar por el gráfico de forma interactiva. Entre ellos, se incluyen los formatos de salida de tabla, gráfico, JSON y la consola de Gremlin.

G.V () es totalmente compatible con Amazon Neptune y ofrece muchas funciones adicionales específicas para Amazon Neptune, como información sobre consultas lentas o registros de auditoría y compatibilidad con la autenticación de IAM. Para obtener más información, consulte la documentación.

G.V () está en continua evolución y recibe nuevas funciones todos los meses. Para obtener más información sobre G.V (), comience con una prueba gratuita visitando el sitio web de G.V ().

Vea a continuación una demostración de G.V () en acción:



Linkurious

Linkurious ofrece diferentes soluciones de inteligencia gráfica para usuarios técnicos y no técnicos y una variedad de casos de uso.

Linkurious Enterprise Explorer es un software de visualización y análisis de off-the-shelf gráficos creado para equipos que puede mantenerse al día con las exigencias de sus day-to-day actividades y ayuda a los profesionales que se basan en los datos a hacer grandes cosas de forma sencilla. Totalmente configurable y fácil de usar, se adapta fácilmente a sus necesidades y permite a los usuarios principiantes o avanzados visualizar rápidamente los datos en AWS Neptune, explorar intuitivamente su conjunto de datos sin importar el tamaño o la complejidad de sus datos y colaborar sin problemas a nivel empresarial o de equipo.

Linkurious Enterprise Watchtower aprovecha la potencia de Linkurious Enterprise Explorer e incorpora capacidades innovadoras de detección y gestión de casos para ofrecer un software integrado de detección e investigación basado en tecnología gráfica. Por un lado, le permite configurar alertas que aprovechan Neptune Database y Neptune Analytics para mostrar automáticamente anomalías o patrones en datos conectados complejos. Por otro lado, combina la gestión de casos y las funciones de colaboración para ayudar a los equipos a gestionar de forma eficiente sus flujos de trabajo de investigación.

Ogma es una JavaScript biblioteca comercial que le ayuda a desarrollar visualizaciones gráficas interactivas potentes y a gran escala para sus aplicaciones. Aprovecha el renderizado WebGL y los diseños de alto rendimiento para permitir a los usuarios mostrar e interactuar con miles de nodos y bordes en cuestión de segundos. También proporciona una variedad de funciones para personalizar su aplicación y crear experiencias de usuario enriquecedoras. Por último, viene equipado con documentación y herramientas completas, como tutoriales, docenas de ejemplos y un parque de juegos interactivo.

Para empezar, solicite una prueba gratuita de 30 días de Linkurious Enterprise u Ogma.

Exportación de datos desde un clúster de base de datos de Neptune

Existen varias formas eficaces de exportar datos desde un clúster de base de datos de Neptune:

- Para cantidades pequeñas de datos, simplemente utilice los resultados de una o varias consultas.
- En el caso de los datos RDF, el protocolo de almacenamiento de gráficos (GSP) facilita la exportación. Por ejemplo:

```
curl --request GET \
  'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/graph'
```

- También existe una herramienta de código abierto potente y flexible para exportar datos de Neptune que se denomina `neptune-export`. En las siguientes secciones, se describen las características de esta herramienta y cómo utilizarla.

Temas

Uso **neptune-export**

Puede usar la herramienta neptune-export de código abierto de dos maneras diferentes:

- Como el servicio Neptune-Export. Cuando exporta datos de Neptune mediante el servicio Neptune-Export, activa y monitoriza los trabajos de exportación a través de una API de REST.
- Como la utilidad de línea de comandos de Java **neptune-export**. Para utilizar esta herramienta de línea de comandos para exportar datos de Neptune, debe ejecutarla en un entorno en el que se pueda acceder al clúster de base de datos de Neptune.

Tanto el servicio Neptune-Export como la herramienta de línea de comandos neptune-export publican datos en Amazon Simple Storage Service (Amazon S3), que se cifran mediante cifrado del lado del servidor de Amazon S3 (SSE-S3).

> **Note**
>
> Se recomienda habilitar el registro de acceso en todos los buckets de Amazon S3 para poder auditar todos los accesos a esos buckets.

Si intenta exportar datos de un clúster de base de datos de Neptune cuyos datos cambian mientras se realiza la exportación, no se garantiza la coherencia de los datos exportados. Es decir, si su clúster atiende el tráfico de escritura mientras se está realizando un trabajo de exportación, es posible que haya incoherencias en los datos exportados. Esto es cierto tanto si realiza la exportación desde la instancia principal del clúster como desde una o más réplicas de lectura.

Para garantizar la coherencia de los datos exportados, es mejor exportarlos desde un clon del clúster de base de datos. Esto proporciona a la herramienta de exportación una versión estática de los datos y garantiza que el trabajo de exportación no ralentice las consultas en el clúster de base de datos original.

Para facilitar esta tarea, puede indicar que desea clonar el clúster de base de datos de origen al activar un trabajo de exportación. De esta manera, el proceso de exportación crea automáticamente el clon, lo usa para la exportación y, después, lo elimina cuando finaliza la exportación.

Uso del servicio Neptune-Export para exportar datos de Neptune

Utilice estos pasos para exportar datos de su clúster de base de datos de Neptune a Amazon S3 mediante el servicio Neptune-Export:

Instalación del servicio Neptune-Export.

Use una plantilla de AWS CloudFormation para crear la pila:

Para instalar el servicio Neptune-Export

1. Inicie la pila AWS CloudFormation en la consola de AWS CloudFormation con uno de los botones Lanzar pila de la tabla siguiente:

Region	Visualización	Ver en Designer	Lanzar
Este de EE. UU. (Norte de Virginia)	Ver	Ver en Designer	
Este de EE. UU. (Ohio)	Ver	Ver en Designer	
Oeste de EE. UU. (Norte de California)	Ver	Ver en Designer	
Oeste de EE. UU. (Oregón)	Ver	Ver en Designer	
Canadá (Centro)	Ver	Ver en Designer	
América del Sur (São Paulo)	Ver	Ver en Designer	
Europa (Estocolmo)	Ver	Ver en Designer	
Europa (Irlanda)	Ver	Ver en Designer	

Region	Visualización	Ver en Designer	Lanzar
Europa (Londres)	Ver	Ver en Designer	
Europa (París)	Ver	Ver en Designer	
Europa (Fráncfort)	Ver	Ver en Designer	
Medio Oriente (Baréin)	Ver	Ver en Designer	
Medio Oriente (EAU)	Ver	Ver en Designer	
Israel (Tel Aviv)	Ver	Ver en Designer	
África (Ciudad del Cabo)	Ver	Ver en Designer	
Asia-Pacífico (Hong Kong)	Ver	Ver en Designer	
Asia-Pacífico (Tokio)	Ver	Ver en Designer	
Asia-Pacífico (Seúl)	Ver	Ver en Designer	
Asia-Pacífico (Singapur)	Ver	Ver en Designer	
Asia-Pacífico (Sídney)	Ver	Ver en Designer	
Asia-Pacífico (Bombay)	Ver	Ver en Designer	
China (Pekín)	Ver	Ver en Designer	

Region	Visualización	Ver en Designer	Lanzar
China (Ningxia)	Ver	Ver en Designer	Launch Stack
AWS GovCloud (Oeste de EE. UU.)	Ver	Ver en Designer	Launch Stack
AWS GovCloud (Este de EE. UU.)	Ver	Ver en Designer	Launch Stack

2. En la página Select Template, elija Next.
3. En la página Especificar detalles de la plantilla, defina estos parámetros:

   - **`VPC`**: la forma más sencilla de configurar el servicio Neptune-Export es instalarlo en la misma Amazon VPC que la base de datos de Neptune. Si desea instalarlo en una VPC diferente, puede utilizar la interconexión con VPC para establecer la conectividad entre la VPC del clúster de base de datos de Neptune y la VPC del servicio Neptune-Export.
   - **`Subnet1`**: el servicio Neptune-Export debe estar instalado en una subred de la VPC que permita el tráfico HTTPS IPv4 saliente de la subred a Internet. Esto sirve para que el servicio Neptune-Export pueda llamar a la API AWS Batch para crear y ejecutar un trabajo de exportación.

     Si creó el clúster de Neptune con la plantilla de CloudFormation en la página Creación de un clúster de base de datos de la documentación de Neptune, puede utilizar las salidas `PrivateSubnet1` y `PrivateSubnet2` de esa pila para rellenar este parámetro y el siguiente.
   - **`Subnet2`**: una segunda subred de la VPC que permite el tráfico HTTPS IPv4 saliente de la subred a Internet.
   - **`EnableIAM`**: configúrelo en `true` para proteger la API de Neptune-Endpoint mediante AWS Identity and Access Management (IAM). Recomendamos que lo haga.

     Si habilita la autenticación de IAM, debe firmar con `Sigv4` todas las solicitudes HTTPS que se envíen al punto de conexión. Puede usar una herramienta como awscurl para firmar las solicitudes en su nombre.
   - **`VPCOnly`**: si lo configura en `true`, el punto de conexión de exportación será solo para VPC, de forma que solo se puede acceder a él desde la VPC en la que está instalado el servicio

Neptune-Export. Esto restringe el uso de la API de Neptune-Export únicamente desde esa VPC.

Es recomendable que defina `VPCOnly` como `true`.

- **`NumOfFilesULimit`** : especifique un valor entre 10 000 y 1 000 000 000 para `nofile` en la propiedad del contenedor `ulimits`. El valor predeterminado es 10 000 y se recomienda mantenerlo a menos que el gráfico contenga un gran número de etiquetas únicas.
- **`PrivateDnsEnabled`** (booleano): indica si se asocia o no una zona alojada privada a la VPC especificada. El valor predeterminado es `true`.

  Cuando se crea un punto de conexión de VPC con este indicador habilitado, todo el tráfico de API Gateway se enruta a través del punto de conexión de VPC y las llamadas a puntos de conexión de API Gateway públicos se deshabilitan. Si configura `PrivateDnsEnabled` en `false`, el punto de conexión de API Gateway público se habilita, pero el servicio de exportación de Neptune no se puede conectar a través del punto de conexión de DNS privado. A continuación, puede usar un punto de conexión de DNS público para que el punto de conexión de VPC llame al servicio de exportación, como se detalla aquí.

4. Elija Siguiente.
5. En la página Opciones, seleccione Siguiente.
6. En la página Revisar, seleccione la primera casilla para confirmar que AWS CloudFormation debe crear los recursos de IAM. Seleccione la segunda casilla para confirmar `CAPABILITY_AUTO_EXPAND` para la nueva pila.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` confirma explícitamente que las macros se expandirán al crear la pila, sin revisión previa. Los usuarios suelen crear un conjunto de cambios a partir de una plantilla procesada para que los cambios realizados por las macros puedan revisarse antes de crear la pila. Para obtener más información, consulte la API CreateStack de AWS CloudFormation.

A continuación, elija Crear.

Habilite el acceso a Neptune desde Neptune-Export

Una vez finalizada la instalación de Neptune-Export, actualice el grupo de seguridad de VPC de Neptune para permitir el acceso desde Neptune-Export. Cuando se ha creado la pila AWS CloudFormation de Neptune-Export, la pestaña Salidas incluye un ID de `NeptuneExportSecurityGroup`. Actualice su grupo de seguridad de VPC de Neptune para permitir el acceso desde este grupo de seguridad de Neptune-Export.

Habilite el acceso al punto de conexión de Neptune-Export desde una instancia de EC2 basada en VPC

Si hace que su punto de conexión de Neptune-Export sea solo para VPC, solo podrá acceder a él desde la VPC en la que esté instalado el servicio Neptune-Export. Para permitir la conectividad desde una instancia de Amazon EC2 en la VPC desde la que puede realizar llamadas a la API de Neptune-Export, asocie el `NeptuneExportSecurityGroup` creado por la pila AWS CloudFormation a esa instancia de Amazon EC2.

Ejecute un trabajo de Neptune-Export mediante la API de Neptune-Export

La pestaña Salidas de la pila AWS CloudFormation también incluye `NeptuneExportApiUri`. Utilice este URI siempre que envíe una solicitud al punto de conexión de Neptune-Export.

Ejecute un trabajo de exportación

- Asegúrese de que se haya concedido el permiso `execute-api:Invoke` al usuario o rol con el que se ejecuta la exportación.
- Si ha configurado el parámetro `EnableIAM` en `true` en la pila AWS CloudFormation al instalar Neptune-Export, tendrá que firmar con `Sigv4` todas las solicitudes en la API de Neptune-Export. Recomendamos usar awscurl para realizar solicitudes a la API. En todos los ejemplos que aparecen aquí, se asume que la autenticación de IAM está habilitada.
- Si ha establecido el parámetro `VPCOnly` en `true` en la pila AWS CloudFormation cuando instaló Neptune-Export, debe llamar a la API de Neptune-Export desde la VPC, normalmente desde una instancia de Amazon EC2 ubicada en la VPC.

Para empezar a exportar datos, envíe una solicitud al punto de conexión de `NeptuneExportApiUri` con los parámetros de solicitud `command` y `outputS3Path` y un parámetro de exportación `endpoint`.

El siguiente mensaje es un ejemplo de una solicitud que exporta datos de gráficos de propiedades de Neptune y los publica en Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Del mismo modo, a continuación se muestra un ejemplo de una solicitud que exporta datos RDF de Neptune a Amazon S3:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": { "endpoint": "(your Neptune endpoint DNS name)" }
      }'
```

Si omite el parámetro de la solicitud `command`, Neptune-Export intentará exportar de forma predeterminada los datos del gráfico de propiedades de Neptune.

Si el comando anterior se ejecutó correctamente, el resultado tendría el siguiente aspecto:

```
{
  "jobName": "neptune-export-abc12345-1589808577790",
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f"
}
```

Monitorice el trabajo de exportación que acaba de iniciar

Para monitorizar un trabajo en ejecución, añada su jobID a su `NeptuneExportApiUri`, de una forma parecida a lo siguiente:

```
curl \
  (your NeptuneExportApiUri)(the job ID)
```

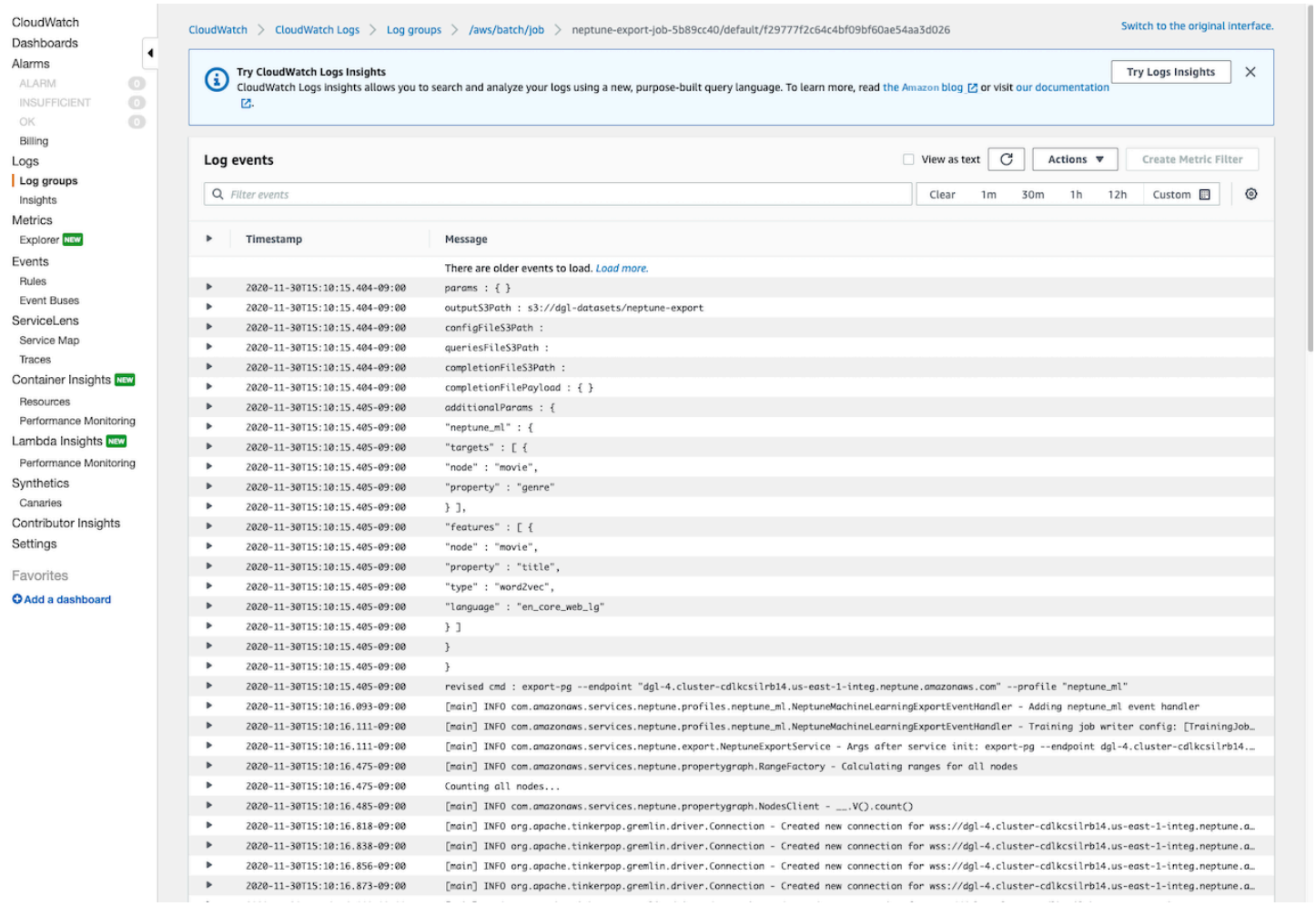
Si el servicio aún no hubiera iniciado el trabajo de exportación, la respuesta tendría el siguiente aspecto:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "pending"
}
```

Al repetir el comando una vez iniciado el trabajo de exportación, la respuesta tendría el siguiente aspecto:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "running",
  "logs": "https://us-east-1.console.aws.amazon.com/cloudwatch/home?..."
}
```

Si abre los registros en Registros de CloudWatch con el URI proporcionado por la llamada de estado, podrá monitorizar el progreso de la exportación en detalle:



Cancele un trabajo de exportación en ejecución

Para cancelar un trabajo de exportación en ejecución mediante la AWS Management Console

1. Abra la consola AWS Batch en https://console.aws.amazon.com/batch/.
2. Elija Jobs (Trabajos).
3. Localice el trabajo en ejecución que desee cancelar, en función de su `jobID`.
4. Seleccione Cancelar trabajo.

Para cancelar un trabajo de exportación en ejecución mediante la API de exportación de Neptune:

Envíe una solicitud `HTTP DELETE` al `NeptuneExportApiUri` con el `jobID` adjunto de la siguiente manera:

```
curl -X DELETE \
```

```
  (your NeptuneExportApiUri) (the job ID)
```

Uso de la herramienta de línea de comandos **neptune-export** para exportar datos de Neptune

Puede seguir estos pasos para exportar datos de su clúster de base de datos de Neptune a Amazon S3 mediante la utilidad de línea de comandos `neptune-export`:

Requisitos previos para usar la utilidad de línea de comandos **neptune-export**

Antes de comenzar

- Disponer de la versión 8 del JDK: necesita tener instalada la versión 8 del Java SE Development Kit (JDK).
- Descargue la utilidad neptune-export: descargue e instale el archivo neptune-export.jar.
- Asegúrese de que **neptune-export** tiene acceso a su VPC de Neptune: ejecute neptune-export desde una ubicación en la que pueda acceder a la VPC en la que se encuentra su clúster de base de datos de Neptune.

  Por ejemplo, puede ejecutarlo en una instancia de Amazon EC2 dentro de la VPC de Neptune, en una VPC independiente que esté interconectada con la VPC de Neptune, o en un host bastión independiente.
- Asegúrese de que los grupos de seguridad de VPC concedan acceso a **neptune-export**: compruebe que los grupos de seguridad de VPC conectados a la VPC de Neptune permiten el acceso a su clúster de base de datos desde la dirección IP o el grupo de seguridad asociado al entorno `neptune-export`.
- Configure los permisos de IAM necesarios: si su base de datos tiene habilitada la autenticación de bases de datos de AWS Identity and Access Management (IAM), asegúrese de que la función con la que `neptune-export` se ejecuta esté asociada a una política de IAM que permita las conexiones a Neptune. Para obtener información sobre las políticas de Neptune, consulte Uso de políticas de IAM.

  Si desea utilizar el parámetro de exportación `clusterId` en sus solicitudes de consulta, el rol en el que `neptune-export` se ejecuta requiere los siguientes permisos de IAM:
  - `rds:DescribeDBClusters`
  - `rds:DescribeDBInstances`
  - `rds:ListTagsForResource`

Si quiere exportar desde un clúster clonado, el rol en el que `neptune-export` se ejecuta requiere los siguientes permisos de IAM:

- `rds:AddTagsToResource`
- `rds:DescribeDBClusters`
- `rds:DescribeDBInstances`
- `rds:ListTagsForResource`
- `rds:DescribeDBClusterParameters`
- `rds:DescribeDBParameters`
- `rds:ModifyDBParameterGroup`
- `rds:ModifyDBClusterParameterGroup`
- `rds:RestoreDBClusterToPointInTime`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterParameterGroup`
- `rds:DeleteDBParameterGroup`
- `rds:DeleteDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterParameterGroup`
- `rds:CreateDBParameterGroup`

Para publicar los datos exportados en Amazon S3, el rol en el que `neptune-export` se ejecuta requiere los siguientes permisos de IAM para las ubicaciones de Amazon S3:

- `s3:PutObject`
- `s3:PutObjectTagging`
- `s3:GetObject`

- Defina la variable de entorno **`SERVICE_REGION`**: defina la variable de entorno `SERVICE_REGION` para identificar la región en la que se encuentra el clúster de base de datos (consulte Connecting to Neptune para obtener una lista de identificadores de región).

Ejecución de la utilidad **neptune-export** para iniciar una operación de exportación

Utilice el siguiente comando para ejecutar neptune-export desde la línea de comandos e iniciar una operación de exportación:

```
java -jar neptune-export.jar nesvc \
  --root-path (path to a local directory) \
  --json (the JSON file that defines the export)
```

El comando tiene dos parámetros:

Parámetros para neptune-export al iniciar una exportación

- **--root-path**: ruta a un directorio local donde se escriben los archivos de exportación antes de publicarlos en Amazon S3.
- **--json**: un objeto JSON que define la exportación.

Ejemplos de comandos que utilizan la utilidad de línea de comandos **neptune-export**

Para exportar los datos del gráfico de propiedades directamente desde el clúster de base de datos de origen:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Para exportar los datos de RDF directamente desde el clúster de base de datos de origen:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
```

```
  --json '{
            "command": "export-rdf",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
            }
          }'
```

Si omite el parámetro de la solicitud `command`, la utilidad `neptune-export` exporta de forma predeterminada los datos del gráfico de propiedades de Neptune.

Para exportar desde un clon de su clúster de base de datos:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)",
              "cloneCluster" : true
            }
          }'
```

Para exportar desde su clúster de base de datos mediante la autenticación de IAM:

```
java -jar neptune-export.jar nesvc \
  --root-path /home/ec2-user/neptune-export \
  --json '{
            "command": "export-pg",
            "outputS3Path" : "s3://(your Amazon S3 bucket)/neptune-export",
            "params": {
              "endpoint" : "(your neptune DB cluster endpoint)"
              "useIamAuth" : true
            }
          }'
```

Archivos exportados por Neptune-Export y **neptune-export**

Cuando se completa la exportación, los archivos de exportación se publican en la ubicación de Amazon S3 que haya especificado. Todos los archivos publicados en Amazon S3 se cifran mediante el cifrado del lado del servidor de Amazon S3 (`SSE-S3`). Las carpetas y los archivos publicados en Amazon S3 varían en función de si exporta datos de gráficos de propiedades o datos de RDF. Si abre la ubicación de Amazon S3 en la que se publican los archivos, verá el siguiente contenido:

Ubicaciones de los archivos exportados en Amazon S3

- **`nodes/`**: esta carpeta contiene archivos de datos de nodos en un formato de valores separados por comas (CSV) o JSON.

  En Neptune, los nodos pueden tener una o más etiquetas. Los nodos con diferentes etiquetas individuales (o diferentes combinaciones de varias etiquetas) se escriben en archivos diferentes, lo que significa que ningún archivo individual contiene datos de nodos con diferentes combinaciones de etiquetas. Si un nodo tiene varias etiquetas, estas se ordenan alfabéticamente antes de asignarlas a un archivo.
- **`edges/`**: esta carpeta contiene archivos de datos de bordes en un formato de valores separados por comas (CSV) o JSON.

  Al igual que con los archivos de nodos, los datos de bordes se escriben en diferentes archivos en función de una combinación de sus etiquetas. Para entrenar modelos, los datos de bordes se asignan a diferentes archivos en función de una combinación de la etiqueta de borde más las etiquetas de los nodos de inicio y final del borde.
- **`statements/`**: esta carpeta contiene archivos de datos RDF en formato Turtle, N-Quads, N-Triples o JSON.
- **`config.json`**: este archivo contiene el esquema del gráfico tal como se deduce del proceso de exportación.
- **`lastEventId.json`**: este archivo contiene el `commitNum` y `opNum` del último evento de los flujo de Neptune de la base de datos. El proceso de exportación solo incluye este archivo si establece el parámetro de exportación `includeLastEventId` en `true` y la base de datos desde la que exporta los datos tiene habilitados flujos de Neptune.

Parámetros utilizados para controlar el proceso de exportación de Neptune

Tanto si utiliza el servicio Neptune-Export como la utilidad de línea de comandos `neptune-export`, los parámetros que utiliza para controlar la exportación son prácticamente los mismos. Contienen un objeto JSON que se pasa al punto de conexión de Neptune-Export o a `neptune-export` en la línea de comandos.

El objeto transferido al proceso de exportación tiene hasta cinco campos de nivel superior:

```
-d '{
      "command" : "(either export-pg or export-rdf)",
      "outputS3Path" : "s3:/(your Amazon S3 bucket)/(path to the folder for exported
 data)",
      "jobsize" : "(for Neptune-Export service only)",
      "params" : { (a JSON object that contains export-process parameters) },
      "additionalParams": { (a JSON object that contains parameters for training
 configuration) }
   }'
```

Contenido

Parámetro **`command`**

El parámetro de nivel superior `command` determina si se deben exportar datos de gráficos de propiedades o datos RDF. Si omite el parámetro `command`, el proceso de exportación exportará de forma predeterminada los datos del gráfico de propiedades.

- **`export-pg`**: exporta datos de gráficos de propiedades.
- **`export-rdf`**: exporta los datos RDF.

Parámetro **`outputS3Path`**

El parámetro de nivel superior `outputS3Path` es obligatorio y debe contener el URI de una ubicación de Amazon S3 en la que se puedan publicar los archivos exportados:

```
  "outputS3Path" : "s3://(your Amazon S3 bucket)/(path to output folder)"
```

El valor debe empezar por `s3://`, seguido de un nombre de bucket válido y, opcionalmente, de una ruta de carpeta dentro del bucket.

Parámetro **`jobSize`**

El parámetro de nivel superior `jobSize` solo se usa con el servicio Neptune-Export, no con la utilidad de línea de comandos `neptune-export`, y es opcional. Le permite caracterizar el tamaño del trabajo de exportación que está iniciando, lo que ayuda a determinar la cantidad de recursos de computación dedicados al trabajo y su nivel máximo de simultaneidad.

```
  "jobsize" : "(one of four size descriptors)"
```

Los cuatro descriptores de tamaño válidos son:

- `small`: simultaneidad máxima: 8. Adecuado para volúmenes de almacenamiento de hasta 10 GB.
- `medium`: simultaneidad máxima: 32. Adecuado para volúmenes de almacenamiento de hasta 100 GB.
- `large`: simultaneidad máxima: 64. Adecuado para volúmenes de almacenamiento superiores a 100 GB pero inferiores a 1 TB.
- `xlarge`: simultaneidad máxima: 96. Adecuado para volúmenes de almacenamiento superiores a 1 TB.

De forma predeterminada, una exportación iniciada en el servicio Neptune-Export se ejecuta como un trabajo `small`.

El rendimiento de una exportación depende no solo de la configuración de `jobSize`, sino también del número de instancias de base de datos desde las que se exporta, del tamaño de cada instancia y del nivel de simultaneidad efectivo del trabajo.

Para las exportaciones de gráficos de propiedades, puede configurar el número de instancias de base de datos mediante el parámetro cloneClusterReplicaContar y puede configurar el nivel de simultaneidad efectivo del trabajo mediante el parámetro concurrency.

El objeto **`params`**

El parámetro de nivel superior `params` es un objeto JSON que contiene parámetros que se utilizan para controlar el propio proceso de exportación, como se explica en Exporte campos de parámetros en el objeto JSON de nivel superior `params`. Algunos de los campos del objeto `params` son específicos de las exportaciones de gráficos de propiedades y otros de RDF.

El objeto **`additionalParams`**

El parámetro de nivel superior `additionalParams` es un objeto JSON que contiene parámetros que puede utilizar para controlar las acciones que se aplican a los datos una vez exportados. En la actualidad, `additionalParams` solo se usa para exportar datos de entrenamiento para Neptune ML.

Exporte campos de parámetros en el objeto JSON de nivel superior **`params`**

El objeto JSON `params` de exportación de Neptune le permite controlar la exportación, incluidos el tipo y el formato de los datos exportados.

Lista de campos posibles en el objeto **`params`** de parámetros de exportación

A continuación, se enumeran todos los posibles campos de nivel superior que pueden aparecer en un objeto `params`. Solo un subconjunto de estos campos aparece en un objeto.

Lista de campos comunes a todos los tipos de exportaciones

- `cloneCluster`
- `cloneClusterInstanceType`
- `cloneClusterReplicaCount`
- `clusterId`
- `endpoint`
- `endpoints`
- `profile`
- `useIamAuth`
- `includeLastEventId`

Lista de campos para la exportación de gráficos de propiedades

- `concurrency`
- `edgeLabels`
- `filter`
- `filterConfigFile`
- `gremlinFilter`
- `gremlinNodeFilter`
- `gremlinEdgeFilter`
- `format`
- `nodeLabels`

- `scope`

Lista de campos para exportaciones RDF

- `format`
- `rdfExportScope`
- `sparql`
- `namedGraph`

Campos comunes a todos los tipos de exportaciones

Campo **`cloneCluster`** en **`params`**

(Opcional). Predeterminado: `false`.

Si el parámetro `cloneCluster` está establecido en `true`, el proceso de exportación utiliza un clon rápido del clúster de base de datos:

```
  "cloneCluster" : true
```

De forma predeterminada, el proceso de exportación exporta los datos del clúster de base de datos que especifique mediante los parámetros `endpoint`, `endpoints` o `clusterId`. Sin embargo, si el clúster de base de datos está en uso mientras se realiza la exportación y los datos cambian, el proceso de exportación no puede garantizar la coherencia de los datos que se exportan.

Para asegurarse de que los datos exportados son coherentes, utilice el parámetro `cloneCluster` para exportar desde un clon estático del clúster de base de datos.

El clúster de base de datos clonado se crea en la misma VPC que el clúster de base de datos de origen y hereda la configuración de autenticación del grupo de seguridad, el grupo de subredes y la base de datos de IAM del origen. Cuando se completa la exportación, Neptune elimina el clúster de base de datos clonado.

De forma predeterminada, un clúster de base de datos clonado consta de una sola instancia del mismo tipo que la instancia principal del clúster de base de datos de origen. Puede cambiar el tipo de instancia utilizado para el clúster de base de datos clonado especificando uno diferente mediante `cloneClusterInstanceType`.

> **Note**
>
> Si no utiliza la opción `cloneCluster` y exporta directamente desde su clúster de base de datos principal, es posible que necesite aumentar el tiempo de espera de las instancias desde las que se exportan los datos. En el caso de conjuntos de datos de gran tamaño, el tiempo de espera debe establecerse en varias horas.

Campo **`cloneClusterInstanceType`** en **`params`**

(Opcional).

Si el parámetro `cloneCluster` está presente y establecido en `true`, puede usar el parámetro `cloneClusterInstanceType` para especificar el tipo de instancia utilizado para el clúster de base de datos clonado:

De forma predeterminada, un clúster de base de datos clonado consta de una sola instancia del mismo tipo que la instancia principal del clúster de base de datos de origen.

```
  "cloneClusterInstanceType" : "(for example, r5.12xlarge)"
```

Campo **`cloneClusterReplicaCount`** en **`params`**

(Opcional).

Si el parámetro `cloneCluster` está presente y establecido en `true`, puede usar el parámetro `cloneClusterReplicaCount` para especificar el número de instancias de réplica de lectura creadas en el clúster de base de datos clonado:

```
  "cloneClusterReplicaCount" : (for example, 3)
```

De forma predeterminada, un clúster de base de datos clonado consta de una única instancia principal. El parámetro `cloneClusterReplicaCount` le permite especificar cuántas instancias de réplica de lectura adicionales se deben crear.

Campo **`clusterId`** en **`params`**

(Opcional).

El parámetro `clusterId` especifica el ID del clúster de base de datos que se va a utilizar:

```
"clusterId" : "(the ID of your DB cluster)"
```

Si utiliza el parámetro `clusterId`, el proceso de exportación utiliza todas las instancias disponibles en ese clúster de base de datos para extraer los datos.

> **Note**
>
> Los parámetros `endpoint`, `endpoints` y `clusterId` son mutuamente excluyentes. Utilice una y solo una de ellas.

Campo **`endpoint`** en **`params`**

(Opcional).

Utilice `endpoint` para especificar un punto de conexión de una instancia de Neptune en su clúster de base de datos que el proceso de exportación pueda consultar para extraer datos (consulte Conexiones de punto de enlace). Este es solo el nombre de DNS y no incluye el protocolo ni el puerto:

```
"endpoint" : "(a DNS endpoint of your DB cluster)"
```

Utilice un punto de conexión de clúster o instancia, pero no el punto de conexión del lector principal.

> **Note**
>
> Los parámetros `endpoint`, `endpoints` y `clusterId` son mutuamente excluyentes. Utilice una y solo una de ellas.

Campo **`endpoints`** en **`params`**

(Opcional).

Utilice `endpoints` para especificar una matriz JSON de puntos de conexión en su clúster de base de datos que el proceso de exportación pueda consultar para extraer datos (consulte Conexiones de punto de enlace). Este es solo el nombre de DNS y no incluye el protocolo ni el puerto:

```
"endpoints": [
```

```
    "(one endpoint in your DB cluster)",
    "(another endpoint in your DB cluster)",
    "(a third endpoint in your DB cluster)"
    ]
```

Si tiene varias instancias en el clúster (una principal y una o más réplicas de lectura), puede mejorar el rendimiento de la exportación mediante el uso del parámetro `endpoints` para distribuir las consultas en una lista de esos puntos de conexión.

> **Note**
>
> Los parámetros `endpoint`, `endpoints` y `clusterId` son mutuamente excluyentes. Utilice una y solo una de ellas.

Campo **`profile`** en **`params`**

(Necesario para exportar los datos de entrenamiento de Neptune ML, a menos que el campo *`neptune_ml`* esté presente en el campo*`additionalParams`*).

El parámetro `profile` proporciona conjuntos de parámetros preconfigurados para cargas de trabajo específicas. En la actualidad, el proceso de exportación solo admite el perfil `neptune_ml`

Si va a exportar datos de entrenamiento para Neptune ML, añada el siguiente parámetro al objeto `params`:

```
"profile" : "neptune_ml"
```

Campo **`useIamAuth`** en **`params`**

(Opcional). Predeterminado: `false`.

Si la base de datos desde la que exporta los datos tiene habilitada la autenticación de IAM, debe incluir el conjunto de parámetros `useIamAuth` en `true`:

```
"useIamAuth" : true
```

Campo **`includeLastEventId`** en **`params`**

Si establece `includeLastEventId` en true y la base de datos desde la que se exportan los datos tiene los flujos de Neptune habilitados, el proceso de exportación escribe un archivo

`lastEventId.json` en la ubicación de exportación especificada. Este archivo contiene el `commitNum` y opNum del último evento del flujo.

```
  "includeLastEventId" : true
```

Una base de datos clonada que se ha creado por medio del proceso de exportación hereda la configuración de flujos de la principal. Si la principal tiene habilitados los flujos, el clon también los tendrá. El contenido del flujo del clon reflejará el contenido de la principal (incluidos los mismos ID de evento) en el momento en que se creó el clon.

Campos para la exportación de gráficos de propiedades

Campo **`concurrency`** en **`params`**

(Opcional). Predeterminado: `4`.

El parámetro `concurrency` especifica el número de consultas paralelas que debe utilizar el proceso de exportación:

```
  "concurrency" : (for example, 24)
```

Una buena pauta es establecer el nivel de simultaneidad que sea el doble del número de vCPU en todas las instancias desde las que se exportan datos. Una instancia r5.xlarge, por ejemplo, tiene 4 vCPU. Si exporta desde un clúster de 3 instancias de r5.xlarge, puede establecer el nivel de simultaneidad en 24 (= 3 x 2 x 4).

Si utiliza el servicio Neptune-Export, el nivel de simultaneidad está limitado por la configuración de jobSize. Un trabajo pequeño, por ejemplo, admite un nivel de simultaneidad de 8. Si intenta especificar un nivel de simultaneidad de 24 para un trabajo pequeño mediante el parámetro `concurrency`, el nivel efectivo permanece en 8.

Si exporta desde un clúster clonado, el proceso de exportación calcula un nivel de simultaneidad adecuado en función del tamaño de las instancias clonadas y del tamaño del trabajo.

Campo **`edgeLabels`** en **`params`**

(Opcional).

Utilice `edgeLabels` para exportar solo los bordes que tengan las etiquetas que especifique:

```
  "edgeLabels" : ["(a label)", "(another label"]
```

Cada etiqueta de la matriz JSON debe ser una etiqueta única y sencilla.

El parámetro `scope` tiene prioridad sobre el parámetro `edgeLabels`, por lo que si el valor de `scope` no incluye bordes, el parámetro `edgeLabels` no tiene ningún efecto.

Campo **`filter`** en **`params`**

(Opcional).

Utilice `filter` para especificar que solo se deben exportar los nodos o bordes con etiquetas específicas y para filtrar las propiedades que se exportan para cada nodo o borde.

La estructura general de un objeto `filter`, ya sea en línea o en un archivo de configuración de filtros, es la siguiente:

```
"filter" : {
  "nodes": [ (array of node label and properties objects) ],
  "edges": [ (array of edge definition an properties objects) ]
}
```

- **`nodes`**: contiene una matriz JSON de nodos y propiedades de nodos con el siguiente formato:

  ```
  "nodes : [
    {
      "label": "(node label)",
      "properties": [ "(a property name)", "(another property name)", ( ... ) ]
    }
  ]
  ```

  - `label`: la etiqueta o etiquetas del gráfico de propiedades del nodo.

    Toma un único valor o, si el nodo tiene varias etiquetas, una matriz de valores.
  - `properties`: contiene una matriz con los nombres de las propiedades del nodo que desea exportar.
- **`edges`**: contiene una matriz JSON de definiciones de borde con el siguiente formato:

  ```
  "edges" : [
    {
      "label": "(edge label)",
      "properties": [ "(a property name)", "(another property name)", ( ... ) ]
    }
  ```

```
    ]
```

- `label`: la etiqueta del gráfico de propiedades de borde. Toma un único valor.
- `properties`: contiene una matriz con los nombres de las propiedades del borde que desea exportar.

Campo **`filterConfigFile`** en **`params`**

(Opcional).

Se utiliza `filterConfigFile` para especificar un archivo JSON que contenga una configuración de filtro con el mismo formato que el parámetro `filter`:

```
  "filterConfigFile" : "s3://(your Amazon S3 bucket)/neptune-export/(the name of the
JSON file)"
```

Consulte filter para conocer el formato del archivo `filterConfigFile`.

Campo **`format`** que se utiliza para los datos del gráfico de propiedades en **`params`**

(Opcional). Predeterminado: `csv` (valores separados por comas):

El parámetro `format` especifica el formato de salida de los datos del gráfico de propiedades exportados:

```
  "format" : (one of: csv, csvNoHeaders, json, neptuneStreamsJson)
```

- **`csv`**: salida con formato de valores separados por comas (CSV), con encabezados de columna con el formato de datos de carga de Gremlin.
- **`csvNoHeaders`**: datos en formato CSV sin encabezados de columna.
- **`json`**: datos con formato JSON.
- **`neptuneStreamsJson`**: los datos con formato JSON que utilizan el formato de serialización de cambios GREMLIN_JSON.

Campo **`gremlinFilter`** en **`params`**

(Opcional).

El parámetro `gremlinFilter` permite proporcionar un fragmento de Gremlin, como un paso `has()`, que se utiliza para filtrar tanto los nodos como los bordes:

```
"gremlinFilter" : (a Gremlin snippet)
```

Los nombres de los campos y los valores de las cadenas deben ir entre comillas dobles con caracteres de escape. Para las fechas y horas, puede utilizar el método datetime.

En el siguiente ejemplo, se exportan solo los nodos y bordes con una propiedad de fecha de creación cuyo valor sea superior a 2021-10-10:

```
"gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
```

Campo **`gremlinNodeFilter`** en **`params`**

(Opcional).

El parámetro `gremlinNodeFilter` permite proporcionar un fragmento de Gremlin, como un paso `has()`, que se utiliza para filtrar los nodos:

```
"gremlinNodeFilter" : (a Gremlin snippet)
```

Los nombres de los campos y los valores de las cadenas deben ir entre comillas dobles con caracteres de escape. Para las fechas y horas, puede utilizar el método datetime.

El siguiente ejemplo exporta solo los nodos con una propiedad booleana `deleted` cuyo valor es `true`:

```
"gremlinNodeFilter" : "has(\"deleted\", true)"
```

Campo **`gremlinEdgeFilter`** en **`params`**

(Opcional).

El parámetro `gremlinEdgeFilter` permite proporcionar un fragmento de Gremlin, como un paso `has()`, que se utiliza para filtrar los bordes:

```
"gremlinEdgeFilter" : (a Gremlin snippet)
```

Los nombres de los campos y los valores de las cadenas deben ir entre comillas dobles con caracteres de escape. Para las fechas y horas, puede utilizar el método datetime.

En el siguiente ejemplo, se exportan únicamente los bordes con una propiedad numérica `strength` cuyo valor es 5:

```
"gremlinEdgeFilter" : "has(\"strength\", 5)"
```

Campo **`nodeLabels`** en **`params`**

(Opcional).

Utilice `nodeLabels` para exportar solo los nodos que tengan las etiquetas que especifique:

```
"nodeLabels" : ["(a label)", "(another label"]
```

Cada etiqueta de la matriz JSON debe ser una etiqueta única y sencilla.

El parámetro `scope` tiene prioridad sobre el parámetro `nodeLabels`, por lo que si el valor de `scope` no incluye nodos, el parámetro `nodeLabels` no tiene ningún efecto.

Campo **`scope`** en **`params`**

(Opcional). Predeterminado: `all`.

El parámetro `scope` especifica si se van a exportar solo los nodos, o solo los bordes, o tanto los nodos como los bordes:

```
"scope" : (one of: nodes, edges, or all)
```

- `nodes`: se exportan únicamente los nodos y sus propiedades.
- `edges`: se exportan únicamente los bordes y sus propiedades.
- `all`: se exportan tanto los nodos como los bordes y sus propiedades (opción predeterminada).

Campos para la exportación RDF

Campo **`format`** utilizado para los datos RDF en **`params`**

(Opcional). Valor predeterminado: `turtle`

El parámetro `format` especifica el formato de salida de los datos RDF exportados:

```
"format" : (one of: turtle, nquads, ntriples, neptuneStreamsJson)
```

- **`turtle`**: salida en formato Turtle.
- **`nquads`**: datos en formato N-Quads sin encabezados de columna.
- **`ntriples`**: datos en formato N-Triples.
- **`neptuneStreamsJson`**: los datos con formato JSON que utilizan el formato de serialización de cambios NQUADS de SPARQL.

Campo **`rdfExportScope`** en **`params`**

(Opcional). Predeterminado: `graph`.

El parámetro `rdfExportScope` especifica el alcance de la exportación de RDF:

```
"rdfExportScope" : (one of: graph, edges, or query)
```

- `graph`: exporta todos los datos RDF.
- `edges`: exporta solo los triples que representan bordes.
- `query`: exporta los datos recuperados por una consulta SPARQL que se proporciona mediante el campo `sparql`.

Campo **`sparql`** en **`params`**

(Opcional).

El parámetro `sparql` le permite especificar una consulta SPARQL para recuperar los datos que se van a exportar:

```
"sparql" : (a SPARQL query)
```

Si proporciona una consulta mediante el campo `sparql`, también debe establecer el campo `rdfExportScope` en `query`.

Campo **`namedGraph`** en **`params`**

(Opcional).

El `namedGraph` parámetro permite especificar un IRI para limitar la exportación a un único gráfico con nombre:

```
  "namedGraph" : (Named graph IRI)
```

El `namedGraph` parámetro solo se puede usar con el `rdfExportScope` campo establecido en`graph`.

Ejemplos de filtrado de lo que se exporta

A continuación, se muestran ejemplos que ilustran las formas de filtrar los datos que se exportan.

Filtrado de la exportación de datos de gráficos de propiedades

Ejemplo de uso de **`scope`** para exportar solo bordes

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "scope": "edges"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Ejemplo de uso de **`nodeLabels`** y **`edgeLabels`** para exportar únicamente nodos y bordes con etiquetas específicas

El parámetro `nodeLabels` del siguiente ejemplo especifica que solo se deben exportar los nodos que tengan una etiqueta `Person` o una etiqueta `Post`. El parámetro `edgeLabels` especifica que solo se deben exportar los bordes con una etiqueta `likes`:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "nodeLabels": ["Person", "Post"],
    "edgeLabels": ["likes"]
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Ejemplo de uso de**`filter`** para exportar solo nodos, bordes y propiedades específicos

El objeto `filter` de este ejemplo exporta los nodos `country` con sus propiedades `type`, `code` y `desc`, y también los bordes `route` con sus propiedades `dist`.

```
{
  "command": "export-pg",
  "params": {
```

```
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    }
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

En este ejemplo, se utiliza **`gremlinFilter`**.

En este ejemplo, se utiliza `gremlinFilter` para exportar los nodos y bordes creados después del 10 de octubre de 2021 (es decir, con una propiedad `created` cuyo valor sea superior a 2021-10-10):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

En este ejemplo, se utiliza **`gremlinNodeFilter`**.

En este ejemplo, se utiliza `gremlinNodeFilter` para exportar los nodos eliminados (nodos con una propiedad `deleted` booleana cuyo valor es `true`):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinNodeFilter" : "has(\"deleted\", true)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

En este ejemplo, se utiliza **`gremlinEdgeFilter`** .

En este ejemplo, se utiliza `gremlinEdgeFilter` para exportar los bordes con una propiedad numérica `strength` cuyo valor es 5:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinEdgeFilter" : "has(\"strength\", 5)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Combinación de **`filter`**, **`gremlinNodeFilter`**, **`nodeLabels`**, **`edgeLabels`** y **`scope`**

El objeto `filter` de este ejemplo exporta:

- Nodos `country` con sus propiedades `type`, `code` y `desc`
- Nodos `airport` con sus propiedades `code`, `icao` y `runways`
- Bordes `route` con su propiedad `dist`

El parámetro `gremlinNodeFilter` filtra los nodos para que solo se exporten los que tienen una propiedad `code` cuyo valor comience por A.

Los parámetros `nodeLabels` y `edgeLabels` restringen aún más la salida, de modo que solo se exportan los nodos `airport` y los bordes `route`.

Por último, el parámetro `scope` elimina los bordes de la exportación, por lo que solo quedan los nodos `airport` designados en la salida.

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "filter": {
      "nodes": [
        {
          "label": "airport",
          "properties": [
            "code",
            "icao",
            "runways"
          ]
        },
        {
          "label": "country",
          "properties": [
            "type",
            "code",
            "desc"
          ]
        }
      ],
      "edges": [
        {
          "label": "route",
          "properties": [
            "dist"
          ]
        }
      ]
    },
    "gremlinNodeFilter": "has(\"code\", startingWith(\"A\"))",
    "nodeLabels": [
      "airport"
    ],
    "edgeLabels": [
      "route"
    ],
    "scope": "nodes"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Filtrado de la exportación de datos RDF

Uso de **`rdfExportScope`** y **`sparql`** para exportar bordes específicos

En este ejemplo, se exportan triples cuyo predicado es <http://kelvinlawrence.net/air-routes/objectProperty/route> y cuyo objeto no es un literal:

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "query",
    "sparql": "CONSTRUCT { ?s <http://kelvinlawrence.net/air-routes/objectProperty/
route> ?o } WHERE { ?s ?p ?o . FILTER(!isLiteral(?o)) }"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Se utiliza **`namedGraph`** para exportar un gráfico con un solo nombre

En este ejemplo, se exportan los triples que pertenecen al grafo indicado < http://aws.amazon.com/neptune/vocab/v01/ >: DefaultNamedGraph

```
{
  "command": "export-rdf",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "rdfExportScope": "graph",
    "namedGraph": "http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Solución de problemas del proceso de exportación de Neptune

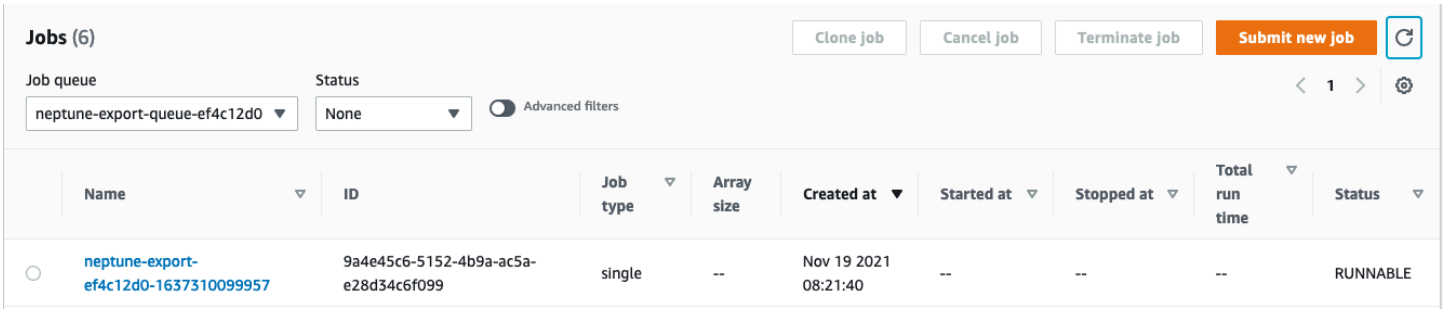
El proceso de exportación de Amazon Neptune utiliza AWS Batch para aprovisionar los recursos de computación y de almacenamiento necesarios para exportar los datos de Neptune. Cuando se está realizando una exportación, puede utilizar el enlace del campo `logs` para acceder a los registros de CloudWatch correspondientes al trabajo de exportación.

Sin embargo, los registros de CloudWatch del trabajo AWS Batch que realiza la exportación solo están disponibles cuando el trabajo AWS Batch está en ejecución. Si la exportación de Neptune informa que una exportación está pendiente, no habrá un enlace de registros a través del cual pueda acceder a los registros de CloudWatch. Si un trabajo de exportación permanece en el estado `pending` durante más de unos minutos, es posible que se produzca un problema al aprovisionar los recursos de AWS Batch subyacentes.

Cuando el trabajo de exportación deja de estar pendiente, puedes comprobar su estado de la siguiente manera:

Para comprobar el estado de un trabajo AWS Batch

1. Abra la consola de AWS Batch en https://console.aws.amazon.com/batch/.
2. Seleccione la cola de trabajo neptune-export.
3. Busque el trabajo cuyo nombre coincida con el `jobName` devuelto por la exportación de Neptune al iniciar la exportación.

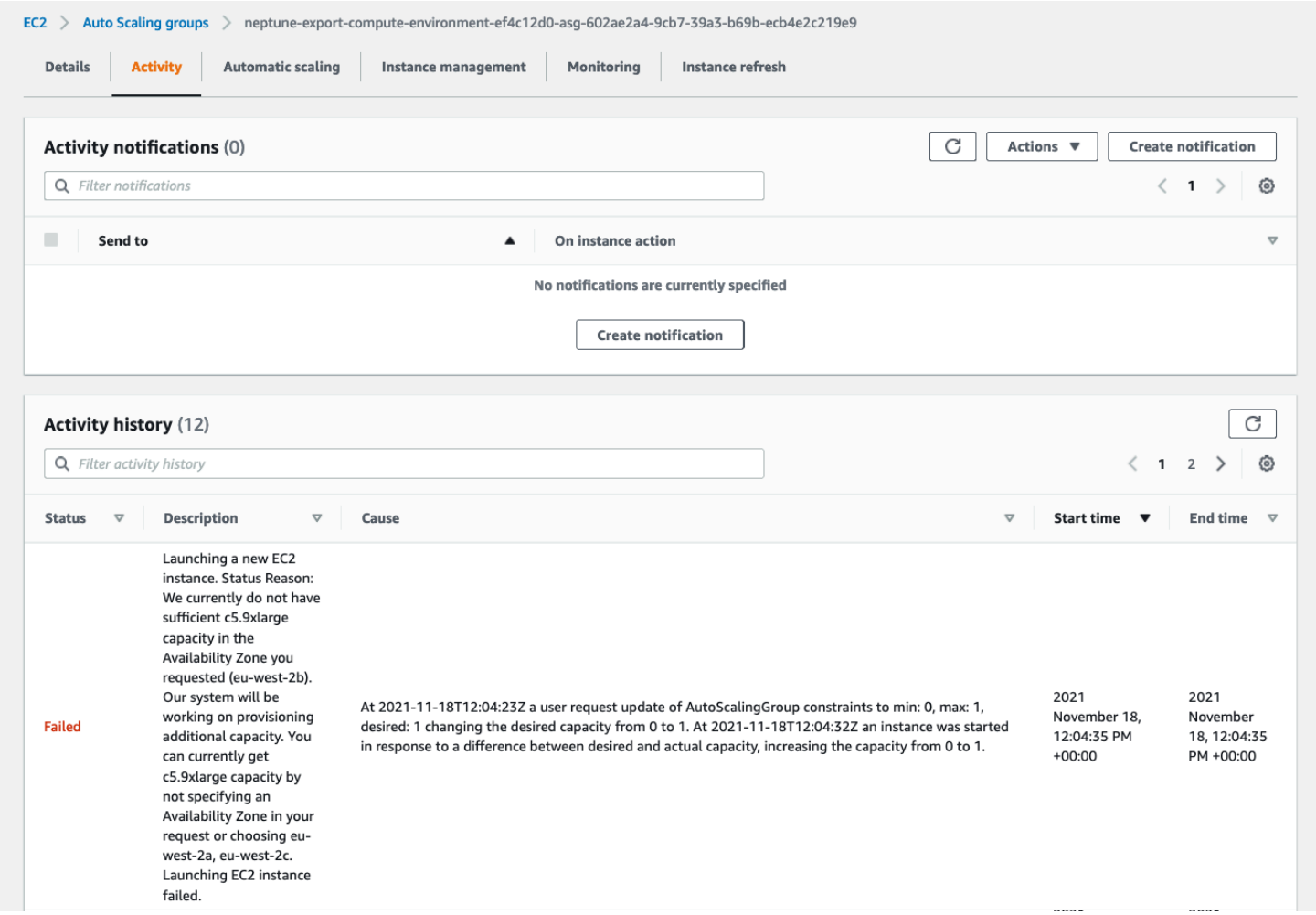


Si el trabajo sigue estancado en un estado `RUNNABLE`, esto puede deberse a problemas de red o seguridad que impidan que la instancia de contenedor se una al clúster subyacente de Amazon Elastic Container Service (Amazon ECS). Consulte la sección sobre la verificación de la configuración de red y seguridad del entorno de computación en este artículo de soporte.

Otra cosa que puede comprobar es si hay problemas con el escalado automático:

Para comprobar el grupo de escalado automático de Amazon EC2 para el entorno de computación AWS Batch

1. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. Seleccione el grupo Escalado automático para el entorno de computación de neptune-export.
3. Abra la pestaña Actividad y consulte el historial de actividades para ver si hay eventos que han fallado.



Errores comunes de Neptune Export

`org.eclipse.rdf4j.query.QueryEvaluationException: Tag mismatch!`

Si un trabajo `export-rdf` suele fallar con una `QueryEvaluationException` de `Tag mismatch!`, la instancia de Neptune no tiene el tamaño adecuado para las consultas grandes y de larga duración que utiliza Neptune Export.

Puede evitar que se produzca este error escalando verticalmente a una instancia de Neptune más grande o configurando el trabajo para que se exporte desde un clúster clonado de gran tamaño, de la siguiente manera:

```
'{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "cloneCluster": True,
    "cloneClusterInstanceType" : "r5.24xlarge"
  }
}'
```

Administración de la base de datos de Amazon Neptune

En esta sección se explica cómo administrar y mantener el clúster de base de datos de Neptune a través de la AWS Management Console y la AWS CLI.

Neptune funciona en clústeres de servidores de base de datos que están conectados en una topología de replicación. Por lo tanto, la administración de Neptune suele requerir la implementación de cambios en varios servidores y asegurarse de que todas las réplicas de Neptune mantengan el ritmo del servidor principal.

Dado que Neptune escala de manera transparente el almacenamiento subyacente a medida que crecen sus datos, la administración de Neptune requiere relativamente poco esfuerzo administrativo del almacenamiento en disco. Del mismo modo, dado que Neptune realiza automáticamente copias de seguridad continuas, el clúster de Neptune no requiere una planificación extensa ni tiempo de inactividad a la hora de realizarlas.

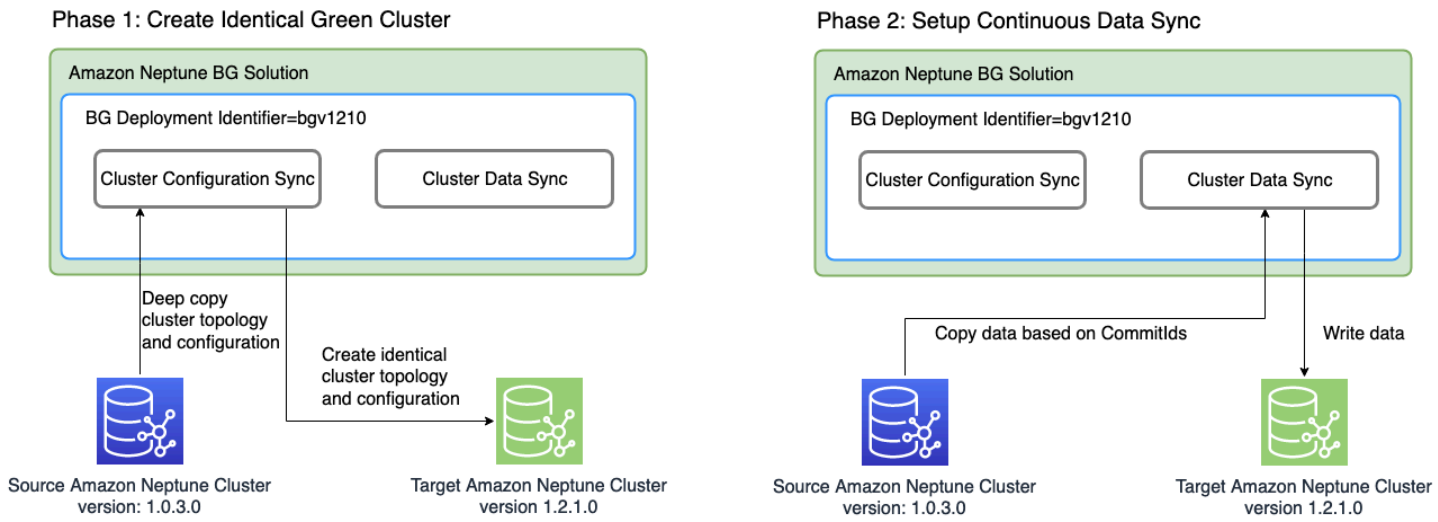
Temas

Uso de la solución azul/verde de Neptune para realizar actualizaciones azul/verde

Las actualizaciones del motor de Amazon Neptune pueden requerir un tiempo de inactividad de las aplicaciones, ya que la base de datos no está disponible mientras se instalan y verifican las actualizaciones. Esto se aplica tanto si se inician de forma manual como automática.

Neptune proporciona una solución de implementación azul/verde que se puede ejecutar mediante una pila de AWS CloudFormation y que reduce considerablemente ese tiempo de inactividad. Crea un entorno de almacenamiento provisional verde que se sincroniza con el entorno de producción azul. A continuación, puede actualizar ese entorno de almacenamiento provisional para realizar una actualización principal o secundaria de la versión del motor, un cambio en el modelo de datos de gráficos o una actualización del sistema operativo, y comprobar el resultado. Por último, puede cambiarlo rápidamente para que se convierta en su entorno de producción, con muy poco tiempo de inactividad.

La solución azul/verde de Neptune pasa por dos fases, tal y como se muestra en este diagrama:



La fase 1 crea un clúster de base de datos verde idéntico al clúster de producción

La solución crea un clúster de base de datos con un identificador único de implementación azul/verde y con la misma topología de clúster que el de producción. Es decir, tiene el mismo número y tamaño de instancias de base de datos, los mismos grupos de parámetros y todas las mismas configuraciones que el clúster de base de datos de producción (azul), excepto que se ha actualizado a la versión de motor de destino que especificó, que debe ser posterior a la versión de motor actual (azul). Puede especificar una versión de motor principal y secundaria del motor para el destino. Si es

necesario, la solución realizará las actualizaciones intermedias necesarias para alcanzar la versión especificada del motor de destino. Este nuevo clúster se convierte en el entorno de almacenamiento provisional verde.

La fase 2 establece la sincronización continua de datos

Una vez que el entorno verde se ha preparado por completo, la solución establece una replicación continua entre el clúster de origen (azul) y el clúster de destino (verde) mediante transmisiones de Neptune. Cuando la diferencia de replicación entre ellos llegue a cero, el entorno de almacenamiento provisional estará listo para las pruebas. En ese momento, debe hacer una pausa en la escritura en el clúster azul para evitar más retrasos en la replicación.

La versión del motor de destino puede tener nuevas características o dependencias que afecten a las aplicaciones. Consulte la página Versión del motor de destino y las páginas Intervención de las versiones del motor en Versiones del motor para ver qué ha cambiado desde la versión actual del motor. Le recomendamos realizar pruebas de integración o verificar las aplicaciones manualmente en el clúster verde antes de pasarlas al entorno de producción.

Una vez que haya probado y calificado los cambios en el clúster verde, tan solo tiene que cambiar el punto de conexión de la base de datos de las aplicaciones del clúster azul al verde.

Tras la transición, la solución azul/verde de Neptune no elimina el antiguo entorno de producción azul. Seguirá teniendo acceso a él para realizar validaciones y pruebas adicionales si es necesario. Se aplican cargos de facturación estándar a sus instancias hasta que las elimine. La solución azul/verde también utiliza otros servicios deAWS, cuyos costos se facturan a precios normales. Podrá encontrar información sobre cómo eliminar la solución cuando haya terminado de utilizarla en la sección de limpieza.

Requisitos previos para ejecutar la pila azul/verde de Neptune

Antes de lanzar la pila azul y verde de Neptune:

- Asegúrese de habilitar las transmisiones de Neptune en el clúster de producción (azul).
- Todas las instancias del clúster azul deben estar en el estado disponible. Puede comprobar los estados de las instancias en la consola de Neptune o mediante la API de describe-db-instances.
- Todas las instancias también deben estar sincronizadas con el grupo de parámetros del clúster de base de datos.

- La solución azul/verde de Neptune requiere un punto de conexión de VPC de DynamoDB en la VPC donde se encuentra el clúster azul. Consulte Uso de puntos de conexión de VPC para tener acceso a DynamoDB.
- Seleccione el momento en el que desee ejecutar la solución cuando la carga de trabajo de escritura del clúster de base de datos de producción azul sea lo más reducida posible. Evite, por ejemplo, ejecutar la solución cuando se produzca una carga masiva o cuando sea probable que haya un gran número de operaciones de escritura por cualquier otro motivo.

Uso de una plantilla de AWS CloudFormation para ejecutar la solución azul/verde de Neptune

Puede utilizar AWS CloudFormation para implementar la solución azul/verde de Neptune. La plantilla CloudFormation crea una instancia de Amazon EC2 en la misma VPC que la base de datos de Neptune de origen azul, instala la solución allí y la ejecuta. Puede supervisar su progreso en los registros de CloudWatch, tal y como se explica en Supervisión del progreso.

Puede utilizar estos enlaces para revisar la plantilla de la solución o seleccionar el botón Lanzar pila para lanzarla en la consola de AWS CloudFormation:

Ver	Ver en Designer	

En la consola, elija la región de AWS en la que desee ejecutar la solución en el menú desplegable situado en la parte superior derecha de la ventana.

Establezca los parámetros de pila, tal y como se indica a continuación:

- **`DeploymentID`**: un identificador único para cada implementación azul/verde de Neptune.

  Se utiliza como identificador del clúster de base de datos verde y como prefijo para asignar nombres a los nuevos recursos creados durante la implementación.
- **`NeptuneSourceClusterId`**: el identificador del clúster de base de datos azul que desea actualizar.
- **`NeptuneTargetClusterVersion:`** la versión del motor de Neptune a la que desea actualizar el clúster de base de datos azul.

  Este valor debe ser posterior al de la versión del motor del clúster de la base de datos azul actual.

- **`DeploymentMode`**: indica si se trata de una implementación nueva o de un intento de reanudar una implementación anterior. Si utiliza el mismo `DeploymentID` que una implementación anterior, establezca `DeploymentMode` en `resume`.

  Los valores válidos son: `new` (valor predeterminado) y `resume`.

- **`GraphQueryType`**: el tipo de datos del gráfico de la base de datos.

  Los valores válidos son: `propertygraph` (valor predeterminado) y `rdf`.

- **`SubnetId`**: un ID de subred de la misma VPC en la que se encuentra el clúster de base de datos azul (consulte Conexión a un clúster de base de datos de Neptune desde una instancia de Amazon EC2 en la misma VPC).

  Proporcione el ID de una subred pública si desea utilizar SSH en la instancia a través de EC2 Connect.

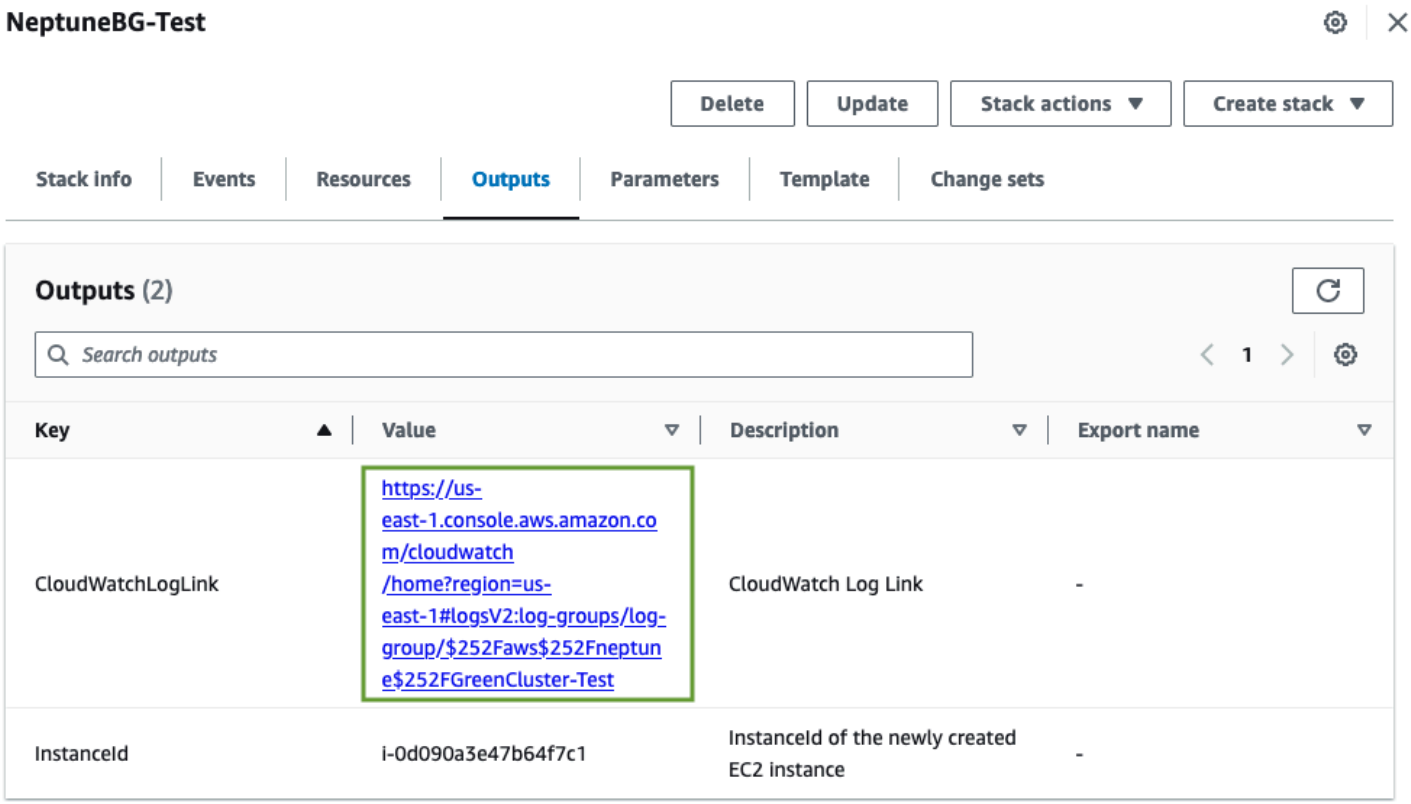
- **`InstanceSecurityGroup`**: un grupo de seguridad para la instancia de Amazon EC2.

  El grupo de seguridad debe tener acceso al clúster de base de datos azul, y debe ser posible conectar SSH a la instancia. Consulte Cree un grupo de seguridad con la consola de VPC.

Espere a que se complete la pila. En cuanto finalice, se iniciará la solución. A continuación, puede supervisar el proceso de implementación mediante los registros de CloudWatch, tal y como se describe en la siguiente sección.

Supervisión del progreso de una implementación azul/verde de Neptune

Para supervisar el progreso de la solución azul/verde de Neptune, vaya a la consola de CloudWatch y consulte los registros del grupo de registro `/aws/neptune/`*`(Neptune Blue/Green deployment ID)`* de CloudWatch. Puede encontrar un enlace a los registros de CloudWatch en los resultados de la pila de AWS CloudFormation de la solución:



Si ha proporcionado una subred pública como parámetro de pila, también puede conectar SSH a la instancia de Amazon EC2 creada como parte de la pila y consultar el registro en `/var/log/cloud-init-output.log`.

El registro muestra las acciones realizadas por la solución azul/verde de Neptune, tal y como se muestra en esta captura de pantalla:

```
============================================================

Neptune Blue Green Deployment Solution Version: 0.1.06012023

============================================================

Checking whether cluster with id = bg-06-01-14-20-29test-bg1-bgInt already exists.

BlueGreen deployment_mode = new

Didn't find any cluster with id bg-06-01-14-20-29test-bg1-bgInt

Cloned_cluster_id: bg-06-01-14-20-29test-bg1-bgInt

Replication_stack_name: bg-06-01-14-20-29test-bg1-bgInt-replication

DescribeDbClusters response for test-bg1-bgIntegTest-06-01-14-20-29: {'AllocatedStorage': 1,
'AvailabilityZones': ['us-east-1b', 'us-east-1c', 'us-east-1f'], 'BackupRetentionPeriod': 1,
'DBClusterIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29', 'DBClusterParameterGroup': 'green-     -blue-
green-deployment-test-123456789012345-pg-tes710', 'DBSubnetGroup': 'default', 'Status': 'available',
'EarliestRestorableTime': datetime.datetime(2023, 6, 1, 8, 51, 23, 394000, tzinfo=tzlocal()), 'Endpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-critvszpmydm.us-east-1.neptune.amazonaws.com', 'ReaderEndpoint':
'test-bg1-bgintegtest-06-01-14-20-29.cluster-ro-critvszpmydm.us-east-1.neptune.amazonaws.com', 'MultiAZ':
False, 'Engine': 'neptune', 'EngineVersion': '1.2.0.0', 'LatestRestorableTime': datetime.datetime(2023, 6, 1,
8, 51, 23, 394000, tzinfo=tzlocal()), 'Port': 8182, 'MasterUsername': 'admin', 'PreferredBackupWindow':
'06:33-07:03', 'PreferredMaintenanceWindow': 'fri:09:44-fri:10:14', 'ReadReplicaIdentifiers': [],
'DBClusterMembers': [{'DBInstanceIdentifier': 'test-bg1-bgintegtest-06-01-14-20-29i-1', 'IsClusterWriter':
True, 'DBClusterParameterGroupStatus': 'in-sync', 'PromotionTier': 1}], 'VpcSecurityGroups':
```

Los mensajes de registro muestran el estado de sincronización entre el clúster azul y el verde:

```
DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611142127'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-anl   -234567899-replication'}}

Time difference for last checkpoint and last stream event: 5841351

Stream eventId difference for last replication checkpoint and last stream event on the Source cluster: 0:0

Found region : us-east-1

Cloudwatch Log Url for blue green solution is https://us-east-1.console.aws.amazon.com/cloudwatch
/home?region=us-east-1#logsV2:log-groups/log-group/aws/neptune/bg

Cloudwatch dashboard url for replication is https://console.aws.amazon.com/cloudwatch/home?region=us-
east-1#dashboards:name=neptune-stream-poller-bg-an   -234567899-replication

Replication poller lambda arn is arn:aws:lambda:us-east-1:451235071234:function:bg-an   -234567899-replic-
NeptuneStreamPollerLambd-B6V1ytULgmSP. Look for CW log the poller lambda for more troubleshooting.

Stream Last EventId {'commitNum': 1, 'opNum': 6} on cluster : database-d61852469-t        -experiment.cluster-
critvszpmydm.us-east-1.neptune.amazonaws.com:8182

DDB checkpoint {'S': '1'}, {'S': '6'}

DDB Checkpoint: {'checkpointSubSequenceNumber': {'S': '6'}, 'lastUpdateTime': {'N': '1685611207245'},
'leaseOwner': {'S': 'nobody'}, 'checkpoint': {'S': '1'}, 'leaseKey': {'S': 'bg-ankig-234567899-replication'}}
```

El proceso de sincronización comprueba el retraso de replicación calculando la diferencia entre el `eventID` de la última transmisión del clúster azul y el punto de comprobación de replicación presente en la tabla de puntos de comprobación de DynamoDB creada por la pila de replicación de Neptune a Neptune. Con estos mensajes, puede supervisar la diferencia de replicación actual.

Pasar del clúster azul de producción al clúster verde actualizado

Antes de pasar el clúster verde a producción, asegúrese de que la diferencia de confirmación entre el clúster azul y el verde sea cero y, a continuación, deshabilite todo el tráfico de escritura dirigido al clúster azul. Si se sigue escribiendo en el clúster azul mientras se cambia el punto de conexión de la base de datos al clúster verde, se pueden dañar los datos al escribir datos parciales en ambos clústeres. Es posible que aún no necesite deshabilitar el tráfico de lectura.

Si ha habilitado la autenticación de IAM en el clúster de origen (azul), asegúrese de actualizar las políticas de IAM utilizadas en las aplicaciones para que apunten al clúster verde (para ver un ejemplo de estas políticas, consulte esta política de acceso sin restricciones).

Tras deshabilitar el tráfico de escritura, espere a que finalice la replicación y, a continuación, habilite el tráfico de escritura en el clúster verde (pero no en el azul). Cambie también el tráfico de lectura del clúster azul al verde.

Limpieza una vez completada la solución azul/verde de Neptune

Una vez que haya promocionado el clúster de almacenamiento provisional (verde) a producción, limpie los recursos creados por la solución azul/verde de Neptune:

- Elimine la instancia de Amazon EC2 que se creó para ejecutar la solución.
- Elimine las plantillas de AWS CloudFormation para la replicación basada en transmisiones de Neptune que mantenían el clúster verde sincronizado con el clúster azul. La principal tiene el nombre de pila que proporcionó anteriormente y la otra consta del ID de implementación seguido de “-replication”, es decir, *(DeploymentID)*`-replication`.

Al eliminar las plantillas de AWS CloudFormation, no se eliminan los propios clústeres. Una vez que haya comprobado que el clúster verde funciona según lo previsto, si lo desea, puede tomar una instantánea antes de eliminar manualmente el clúster azul.

Prácticas recomendadas de la solución azul/verde de Neptune

- Antes de pasar el clúster verde a producción, merece la pena comprobar minuciosamente que funciona de forma correcta. Compruebe la coherencia de los datos y la configuración de la base de datos. Es posible que algunas de las nuevas versiones del motor también requieran actualizaciones del cliente. Consulte las notas de la versión del motor antes de realizar la actualización. Vale la pena probar todo esto en los entornos de desarrollo, pruebas y preproducción antes de iniciar una actualización azul/verde en la fase de producción.
- Le recomendamos realizar el cambio del servidor azul al verde durante el periodo de mantenimiento.
- Para garantizar que todo funcione correctamente tras la actualización y la sincronización, merece la pena conservar el clúster original durante un tiempo antes de eliminarlo. Podría resultar útil si surge algún problema imprevisto.
- Evite las operaciones con mucha escritura, como las cargas masivas, al ejecutar la solución azul/verde de Neptune, ya que pueden provocar un retraso en la replicación que, a su vez, origina un considerable tiempo de inactividad. Lo ideal es que el tiempo que transcurra entre la desactivación de las escrituras en el clúster azul y su activación en el clúster verde sea de solo unos instantes.

Solución de problemas de la solución azul/verde de Neptune

Errores generados por la solución azul/verde de Neptune

- **`Cluster with id = (blue_green_deployment_id) already exists`**: hay un clúster con el identificador *`(blue_green_deployment_id)`*.

  Proporcione un nuevo ID de implementación o establezca el modo de implementación en `resume` si el clúster se creó en una ejecución anterior de la solución azul/verde de Neptune.
- **`Streams should be enabled on the source Cluster for Blue Green Deployment`**: permite habilitar las transmisiones de Neptune en el clúster azul (origen).
- **`No Bulkload should be in progress on source cluster: (cluster_id)`**: la solución azul/verde de Neptune finaliza si identifica una carga masiva en curso.

  Esto es para garantizar que el proceso de sincronización pueda ponerse al día con las escrituras que se están realizando. Evite o cancele cualquier trabajo de carga masiva en curso antes de iniciar la solución azul/verde de Neptune.
- **`Blue Green deployment requires instances to be in sync with db cluster parameter group`**: cualquier cambio en el grupo de parámetros del clúster debe estar sincronizado en todo el clúster de base de datos. Consulte Grupos de parámetros de Amazon Neptune.
- **`Invalid target engine version for Blue Green Deployment`**: la versión del motor de destino debe figurar como activa en Versiones del motor para Amazon Neptune y debe ser posterior a la versión actual del motor del clúster de origen (azul).

Creación de un usuario de IAM con permisos para Neptune

Para acceder a la consola de Neptune para crear y administrar un clúster de base de datos Neptune, debe crear un usuario de IAM con todos los permisos necesarios.

El primer paso es crear una política de rol vinculado a un servicio de Neptune:

Creación de una política de rol vinculado a un servicio de Amazon Neptune

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la izquierda, elija Policies (Políticas).
3. En la página Políticas, seleccione Crear una política.
4. En la página Crear política, seleccione la pestaña JSON y copia la siguiente política de rol vinculado a un servicio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
      "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
      }
    }
  ]
}
```

5. Seleccione Siguiente: Etiquetas y en la página Añadir etiquetas, seleccione Siguiente: Revisar.
6. En la página Revisar política, asigne el nombre “NeptuneServiceLinked” a la nueva política.

Para obtener más información acerca de los roles vinculados a servicios, consulte Uso de roles vinculados a servicios para Neptune.

Creación de un nuevo usuario de IAM con todos los permisos necesarios

A continuación, cree el nuevo usuario de IAM con las políticas administradas adecuadas que le concederán los permisos que necesite, junto con la política de rol vinculada a un servicio que ha creado (denominada aquí `NeptuneServiceLinked`):

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la izquierda, elija Usuarios y en la página Usuarios, elija Añadir usuarios.
3. En la página Añadir usuario, introduzca un nombre para el nuevo usuario de IAM, elija Clave de acceso: acceso programático para el tipo de credencial de AWS y elija Siguiente: Permisos.
4. En la página Establecer permisos, en el cuadro Filtrar políticas, escriba “Neptune”. Ahora seleccione lo siguiente en las políticas que aparecen en la lista:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked (suponiendo que es el nombre que asignó a la política de rol vinculado a un servicio que creó anteriormente).
5. A continuación, escriba “VPC” en el cuadro Filtrar políticas en lugar de “Neptune”. Seleccione AmazonVPCFullAccess en las políticas que aparecen en la lista.
6. Seleccione Siguiente: Etiquetas y en la página Añadir etiquetas, seleccione Siguiente: Revisar.
7. En la página Revisar, compruebe que todas las políticas siguientes estén ahora asociadas al nuevo usuario:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked
   - AmazonVPCFullAccess

   A continuación, seleccione Crear usuario.
8. Por último, descargue y guarde el ID de clave de acceso y la clave de acceso secreta del nuevo usuario.

Para interactuar con otros servicios, como Amazon Simple Storage Service (Amazon S3), tendrá que añadir más permisos y relaciones de confianza.

Grupos de parámetros de Amazon Neptune

Use los parámetros de un grupo de parámetros para administrar la configuración de la base de datos en Amazon Neptune. Los grupos de parámetros sirven de contenedor para los valores de configuración del motor que se aplican a una o varias instancias de bases de datos.

Existen dos tipos de grupos de parámetros: los grupos de parámetros de clúster de base de datos y los grupos de parámetros de base de datos.

- Los grupos de parámetros de base de datos se aplican en el nivel de instancia y suelen estar asociados con la configuración del motor de gráficos de Neptune, como, por ejemplo, el parámetro `neptune_query_timeout`.
- Los grupos de parámetros de clúster de base de datos se aplican a todas las instancias del clúster y suelen tener una configuración más extensa. Cada clúster de Neptune se asocia a un grupo de parámetros del clúster de base de datos. Cada instancia de base de datos dentro de ese clúster hereda los valores de configuración del motor incluidos en el grupo de parámetros de clúster de base de datos.

Los valores de configuración que modifique en el grupo de parámetros de clúster de base de datos anulan los valores predeterminados en el grupo de parámetros de base de datos. Si edita los valores correspondientes en el grupo de parámetros de base de datos, dichos valores anulan la configuración del grupo de parámetros de clúster de base de datos.

Si se crea una instancia de base de datos sin especificar un grupo de parámetros de base de datos personalizado, se usa un grupo de parámetros de base de datos predeterminado. La configuración de los parámetros de un grupo de parámetros de base de datos predeterminado no se puede modificar. En su lugar, para cambiar la configuración predeterminada de parámetros, debe crear un nuevo grupo de parámetros de base de datos. No todos los parámetros del motor de base de datos pueden cambiarse en el grupo de parámetros de base de datos que cree.

Los grupos de parámetros se crean en familias que son compatibles con diferentes versiones del motor de Neptune. La familia predeterminada de grupos de parámetros es `neptune1`, que es compatible con todas las versiones de motores anteriores a `1.2.0.0`. A partir de Versión: 1.2.0.0 (21/07/2022), debe utilizarse la familia de grupos de parámetros `neptune1.2`. Esto significa que, al actualizar a `1.2.0.0` o una versión posterior, primero debe volver a crear todos los grupos de parámetros personalizados de la familia `neptune1.2` para poder asociarlos al actualizar.

Algunos parámetros de Neptune son estáticos y otros son dinámicos. Las diferencias son las siguientes.

Parámetros estáticos

- Un parámetro estático es aquel que solo se aplica después de reiniciar una instancia de base de datos. Dicho de otro modo, cuando se cambia un parámetro estático y se guarda el grupo de parámetros de base de datos de la instancia, debe reiniciar manualmente la instancia de base de datos para que se aplique el cambio de parámetro. Por el momento, todos los parámetros de nivel de instancia de Neptune (en un grupo de parámetros de base de datos en lugar de en un grupo de parámetros de clúster de base de datos) son estáticos.
- Cuando se cambia un parámetro estático de nivel de clúster y se guarda el grupo de parámetros de clúster de base de datos, el cambio de parámetros se aplicará después de reiniciar manualmente cada una de las instancias de base de datos en el clúster.

Parámetros dinámicos

- Un parámetro dinámico es aquel que se aplica casi inmediatamente después de actualizar el parámetro en su grupo de parámetros. En otras palabras, no es necesario reiniciar una instancia de base de datos después de actualizar un parámetro dinámico para que se aplique el cambio de parámetro.
- Espere un pequeño retraso para que un cambio de parámetro dinámico del clúster se aplique a todas las instancias de base de datos.
- Un valor de parámetro dinámico actualizado no se aplica a las solicitudes que se estén ejecutando actualmente, sino solo a las que se envíen después de que se haya realizado el cambio.
- Al cambiar un parámetro de nivel de clúster dinámico, el cambio de parámetros se aplica, de forma predeterminada, al clúster de base de datos inmediatamente, sin necesidad de reiniciar. Para aplazar el cambio de parámetros hasta después de reiniciar las instancias de base de datos de un clúster de base de datos asociado, puede utilizar la AWS CLI para establecer el `ApplyMethod` en `pending-reboot` para el cambio de parámetros.

Por el momento, todos los parámetros son estáticos, excepto los siguientes parámetros de clúster nuevos:

- `neptune_enable_slow_query_log` (en el nivel de clúster)
- `neptune_slow_query_log_threshold` (en el nivel de clúster)

Estos son algunos puntos importantes que debe tener en cuenta para utilizar los parámetros de un grupo de parámetros de base de datos:

- Si se configuran de forma incorrecta los parámetros de un grupo de parámetros de base de datos, pueden producirse efectos adversos no deseados, como la degradación del rendimiento y la inestabilidad del sistema. Realice siempre cualquier modificación de los parámetros de base de datos con cuidado y haga una copia de seguridad de los datos antes de modificar un grupo de parámetros de base de datos. Pruebe los cambios de configuración de los grupos de parámetros en una instancia de base de datos de prueba antes de aplicar dichos cambios a una instancia de base de datos de producción.
- Cuando se cambia el grupo de parámetros de base de datos asociado a una instancia de base de datos, se debe reiniciar manualmente la instancia para que esta utilice el nuevo grupo de parámetros de base de datos.

> **Note**
>
> Antes de Versión: 1.2.0.0 (21/07/2022), todas las instancias de réplicas de lectura de un clúster de base de datos se reiniciaban automáticamente cuando se reiniciaba la instancia principal (escritor).
> A partir de Versión: 1.2.0.0 (21/07/2022), el reinicio de la instancia principal no provoca el reinicio de ninguna de las instancias de réplica. Esto significa que si va a cambiar un parámetro de nivel de clúster, debe reiniciar cada instancia por separado para detectar el cambio de parámetro.

Edición de un grupo de parámetros de clúster de base de datos o de un grupo de parámetros de base de datos

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Elija Parameter groups (Grupos de parámetros) en el panel de navegación.
3. Elija el enlace Name (Nombre) para el grupo de parámetros de base de datos que desea editar.

   (Opcional) Elija Create parameter group (Crear grupo de parámetros) para crear un grupo de parámetros de clúster y crear el grupo. A continuación, elija Name (Nombre) para el nuevo grupo de parámetros.

> **Important**
>
> Este paso es necesario si solo tiene el grupo de parámetros de clúster de base de datos predeterminado, ya que dicho grupo no se puede modificar.

4. Busque el parámetro y haga clic en el campo Valor situado junto a la columna Nombre.
5. Introduzca el valor permitido y seleccione la casilla situada junto al campo de valor.
6. Elija Guardar cambios.
7. Reinicie todas las instancias de base de datos del clúster de Neptune si está cambiando un parámetro del clúster de base de datos, o una o varias instancias específicas si está cambiando un parámetro de instancia de base de datos.

Creación de un grupo de parámetros de base de datos o de un grupo de parámetros de clúster de base de datos

Puede utilizar la consola de Neptune para crear un nuevo grupo de parámetros:

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Seleccione Parameter groups (Grupos de parámetros) en el panel de navegación izquierdo.
3. Elija Create DB parameter group (Crear grupo de parámetros de base de datos).

   Aparece la página Create DB Parameter Group (Crear grupo de parámetros de base de datos).
4. En la lista Familia de grupos de parámetros, elija neptune1 o, si tiene la versión 1.2.0.0 del motor como destino o una versión posterior, elija neptune1.2.
5. En la lista Type (Tipo), elija DB Parameter Group (Grupo de parámetros de base de datos) o DB Cluster Parameter Group (Grupo de parámetros de clúster de base de datos).
6. En el cuadro Group name (Nombre de grupo), escriba el nombre del nuevo grupo de parámetros de base de datos.
7. En el cuadro Description (Descripción) escriba una descripción para el nuevo grupo de parámetros de base de datos.
8. Seleccione Crear.

También puede crear un nuevo grupo de parámetros mediante la AWS CLI:

```
aws neptune create-db-parameter-group \
  --db-parameter-group-name (a name for the new DB parameter group) \
  --db-parameter-group-family (either neptune1 or neptune1.2, depending on the engine
 version) \
  --description (a description for the new DB parameter group)
```

Parámetros de Amazon Neptune

Use los parámetros de los grupos de parámetros para administrar la configuración de la base de datos en Amazon Neptune. Están disponibles los siguientes parámetros para configurar la base de datos de Neptune:

Parámetros de nivel de clúster

- neptune_enable_audit_log
- neptune_enable_slow_query_log
- neptune_slow_query_log_threshold
- neptune_lab_mode
- neptune_query_timeout
- neptune_streams
- neptune_streams_expiry_days
- neptune_lookup_cache
- neptune_autoscaling_config
- neptune_ml_iam_role
- neptune_ml_endpoint

Parámetros de nivel de instancia

- neptune_dfe_query_engine
- neptune_query_timeout
- neptune_result_cache

Parámetros no disponibles

- neptune_enforce_ssl

neptune_enable_audit_log (parámetro de nivel de clúster)

Este parámetro cambia el registro de auditoría de Neptune.

Los valores permitidos son `0` (deshabilitado) y `1` (habilitado). El valor predeterminado es `0`.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Puede publicar registros de auditoría en Amazon CloudWatch, tal y como se describe en Uso de la CLI para publicar los registros de auditoría de Neptune en Logs CloudWatch .

neptune_enable_slow_query_log (parámetro de nivel de clúster)

Utilice este parámetro para habilitar o deshabilitar la característica registro de consultas lentas de Neptune.

Se trata de un parámetro dinámico, lo que significa que cambiar su valor no requiere ni provoca el reinicio del clúster de base de datos.

Los valores permitidos son:

- **`info`**: permite el registro de consultas lentas y registra los atributos seleccionados que podrían dando lugar a un rendimiento lento.
- **`debug`**: permite el registro de consultas lentas y registra todos los atributos disponibles de la ejecución de la consulta.
- **`disable`**: deshabilita el registro de consultas lentas.

El valor predeterminado es `disable`.

Puede publicar registros de consultas lentas en Amazon CloudWatch, tal y como se describe en Uso de la CLI para publicar registros de consultas lentas de Neptune en Logs CloudWatch .

neptune_slow_query_log_threshold (parámetro de nivel de clúster)

Este parámetro especifica el umbral de tiempo de ejecución, en milisegundos, tras el cual una consulta se considera lenta. Si el registro de consultas lentas está habilitado, las consultas que superen este umbral se registrarán junto con algunos de sus atributos.

El valor predeterminado es de 5000 milisegundos (5 segundos).

Se trata de un parámetro dinámico, lo que significa que cambiar su valor no requiere ni provoca el reinicio del clúster de base de datos.

`neptune_lab_mode` (parámetro de nivel de clúster)

Cuando se establece, este parámetro habilita características experimentales específicas de Neptune. Consulte Modo de laboratorio de Neptune para ver las características experimentales que están disponibles actualmente.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Para habilitar o deshabilitar una característica experimental, incluya *(nombre de la característica)*=enabled o *(nombre de la característica)*=disabled en este parámetro. Puede habilitar o deshabilitar varias características separándolas con comas; por ejemplo:

(nombre de característica 1)=enabled,*(nombre de característica 2)*=enabled

Las características del modo de laboratorio suelen estar deshabilitadas de forma predeterminada. Una excepción es la característica `DFEQueryEngine`, que se habilitó de forma predeterminada para su uso con sugerencias de consulta (`DFEQueryEngine=viaQueryHint`) a partir de la versión 1.0.5.0 del motor de Neptune. A partir de la versión 1.1.1.0 del motor de Neptune, el motor DFE ya no está en modo de laboratorio y ahora se controla mediante el parámetro de instancia neptune_dfe_query_engine del grupo de parámetros de base de datos de una instancia.

`neptune_query_timeout` (parámetro de nivel de clúster)

Especifica un determinado tiempo de espera para las consultas de gráficos, en milisegundos.

Los valores permitidos van desde `10` hasta `2,147,483,647` (2^{31} - 1). El valor predeterminado es `120,000` (2 minutos).

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

> **Note**
>
> Es posible incurrir en costos inesperados si se establece un valor de tiempo de espera de consulta demasiado alto, especialmente en una instancia sin servidor. Sin una configuración de tiempo de espera razonable, es posible que, sin darse cuenta, emita una consulta que

siga ejecutándose durante mucho más tiempo del esperado, lo que dará lugar a costos que no había previsto. Esto ocurre especialmente en una instancia sin servidor que podría escalarse verticalmente hasta convertirse en un tipo de instancia grande y caro mientras se ejecuta la consulta.
Para evitar gastos inesperados de este tipo, utilice un valor de tiempo de espera de consulta que se adapte a la mayoría de las consultas y que solo provoque que se agoten inesperadamente las que se ejecutan durante mucho tiempo.

`neptune_streams` (parámetro de nivel de clúster)

Habilita o deshabilita Flujos de Neptune.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Los valores permitidos son `0` (deshabilitado, que es el valor predeterminado) y `1` (habilitado).

`neptune_streams_expiry_days` (parámetro de nivel de clúster)

Especifica cuántos días deben transcurrir antes de que el servidor elimine los registros de transmisión.

Los valores permitidos son de `1` a `90`, ambos incluidos. El valor predeterminado es `7`.

Este parámetro se introdujo en la versión 1.2.0.0 del motor.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

`neptune_lookup_cache` (parámetro de nivel de clúster)

Deshabilita o vuelve a habilitar la caché de búsqueda de Neptune en instancias `R5d`.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Los valores permitidos son `enabled` y `disabled`. El valor predeterminado es `disabled`, pero siempre que se crea una instancia `R5d` en el clúster de base de datos, el parámetro `neptune_lookup_cache` se establece automáticamente en `enabled` y se crea una caché de búsqueda en esa instancia.

`neptune_autoscaling_config` (parámetro de nivel de clúster)

Establece los parámetros de configuración de las instancias de réplica de lectura que el escalado automático de Neptune crea y administra.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Con una cadena JSON que establezca como valor del parámetro `neptune_autoscaling_config`, puede especificar:

- El tipo de instancia que utiliza el escalado automático de Neptune para todas las nuevas instancias de réplica de lectura que crea.
- Los periodos de mantenimiento asignados a esas réplicas de lectura.
- Las etiquetas que se asociarán a todas las réplicas de lectura nuevas.

La cadena JSON tiene una estructura como esta:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Tenga en cuenta que a todas las comillas de la cadena se les debe aplicar una secuencia de escape con un carácter de barra diagonal inversa (\).

Cualquiera de los tres ajustes de configuración no especificados en el parámetro `neptune_autoscaling_config` se copia de la configuración de la instancia de escritor principal del clúster de base de datos.

`neptune_ml_iam_role` (parámetro de nivel de clúster)

Especifica el ARN del rol de IAM utilizado en Neptune ML. El valor puede ser cualquier ARN válido de rol de IAM.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Puede especificar el ARN predeterminado del rol de IAM para el machine learning en los gráficos.

`neptune_ml_endpoint` (parámetro de nivel de clúster)

Especifica el punto de conexión utilizado para Neptune ML. El valor puede ser cualquier nombre de punto de conexión de SageMaker válido.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Puede especificar el punto de conexión predeterminado de SageMaker para el machine learning en los gráficos.

`neptune_dfe_query_engine` (parámetro de nivel de instancia)

A partir de la versión 1.1.1.0 del motor Neptune, este parámetro de instancia de base de datos se utiliza para controlar el uso del motor de consultas DFE. Los valores permitidos son los siguientes:

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

- **`enabled`**: hace que se utilice el motor DFE siempre que sea posible, excepto cuando la sugerencia de consulta `useDFE` esté presente y establecida en `false`.
- **`viaQueryHint`** (valor predeterminado): hace que el motor DFE se utilice únicamente para consultas que incluyen explícitamente la sugerencia de consulta `useDFE` establecida en `true`.

Si este parámetro no se ha establecido de forma explícita, se utiliza el valor predeterminado, `viaQueryHint`, al iniciar la instancia.

> **Note**
>
> El motor DFE ejecuta todas las consultas de openCypher, independientemente de cómo esté establecido este parámetro.

Antes de la versión 1.1.1.0, era un parámetro de modo laboratorio y no un parámetro de instancia de base de datos.

`neptune_query_timeout` (parámetro de nivel de instancia)

Este parámetro de instancia de base de datos especifica el tiempo de espera para las consultas de gráficos, en milisegundos, para una instancia.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Los valores permitidos van desde `10` hasta `2,147,483,647` (2^{31} - 1). El valor predeterminado es `120,000` (2 minutos).

> **Note**
>
> Es posible incurrir en costos inesperados si se establece un valor de tiempo de espera de consulta demasiado alto, especialmente en una instancia sin servidor. Sin una configuración de tiempo de espera razonable, es posible que, sin darse cuenta, emita una consulta que siga ejecutándose durante mucho más tiempo del esperado, lo que dará lugar a costos que no había previsto. Esto ocurre especialmente en una instancia sin servidor que podría escalarse verticalmente hasta convertirse en un tipo de instancia grande y caro mientras se ejecuta la consulta.
> Para evitar gastos inesperados de este tipo, utilice un valor de tiempo de espera de consulta que se adapte a la mayoría de las consultas y que solo provoque que se agoten inesperadamente las que se ejecutan durante mucho tiempo.

`neptune_result_cache` (parámetro de nivel de instancia)

`neptune_result_cache`: este parámetro de instancia de base de datos habilita o deshabilita Almacenamiento en caché de resultados de las consultas.

Este parámetro es estático, lo que significa que los cambios que se realicen en él no se aplican en ninguna instancia hasta que se haya reiniciado.

Los valores permitidos son `0` (deshabilitado, que es el valor predeterminado) y 1 (habilitado).

`neptune_enforce_ssl`(Parámetro de nivel de clúster NO DISPONIBLE)

(No disponible) Solía haber regiones que permitían las conexiones HTTP a Neptune, y este parámetro se utilizó para forzar a todas las conexiones a usar HTTPS cuando estaba establecida en

1. Sin embargo, este parámetro ya no es relevante, ya que Neptune ahora solo acepta conexiones HTTPS en todas las regiones.

Lanzamiento de un clúster de base de datos de Neptune mediante la AWS Management Console

La forma más sencilla de lanzar un nuevo clúster de base de datos de Neptune es utilizar una plantilla de AWS CloudFormation que cree todos los recursos necesarios, tal y como se explica en Creación de un clúster de base de datos.

Si lo prefiere, también puede utilizar la consola de Neptune para lanzar un nuevo clúster de base de datos manualmente, tal y como se explica aquí.

Para acceder a la consola de Neptune para crear y administrar un clúster de Neptune, debe crear un usuario de IAM con todos los permisos necesarios, tal y como se explica en Creación de un usuario de IAM con permisos para Neptune.

A continuación, inicie sesión AWS Management Console como ese usuario de IAM y siga los pasos que se indican a continuación para crear un nuevo clúster de base de datos:

Para lanzar un clúster de base de datos de Neptune mediante la consola

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. Vaya a la página Bases de datos y seleccione Crear base de datos, lo que abrirá la página Crear base de datos.
3. En Opciones de motor, el tipo de motor es `neptune` y puede elegir una versión de motor específica o aceptar la predeterminada.
4. En Configuración, introduzca un nombre para el nuevo clúster de base de datos o acepte el nombre predeterminado que se proporciona allí. Este nombre se utiliza en la dirección del punto de conexión de la instancia y debe cumplir las siguientes restricciones:
   - Debe contener de 1 a 63 caracteres alfanuméricos o guiones.
   - El primer carácter debe ser una letra.
   - No puede terminar con un guion ni contener dos guiones consecutivos.
   - Debe ser único en todas las instancias de base de datos de su cuenta de AWS en una determinada región de AWS.
5. En Plantillas, elija Producción o Desarrollo y pruebas.

6. En Tamaño de la instancia de base de datos, elija un tamaño de instancia. Esto determinará el procesamiento y la capacidad de memoria de la instancia de escritura principal del nuevo clúster de base de datos.

   Si ha seleccionado la plantilla Producción, solo podrá elegir entre las clases de memoria optimizada disponibles de la lista, pero si ha seleccionado Desarrollo y pruebas, también puede elegir entre las clases por ráfagas más económicas (consulte Instancias con ráfagas T3 para obtener información sobre las clases ampliables).

   > **Note**
   >
   > A partir de la versión 1.1.0.0 del motor de Neptune, Neptune ya no admite los tipos de instancias R4.

7. En Disponibilidad y durabilidad, puede elegir si desea habilitar o no la implementación multi-AZ en varias zonas de disponibilidad. La plantilla de producción permite la implementación multi-AZ de forma predeterminada, mientras que la plantilla de desarrollo y pruebas no lo permite. Si la implementación Multi-AZ está habilitada, Neptune localiza las instancias de réplica de lectura que cree en diferentes zonas de disponibilidad (AZ) para mejorar la disponibilidad.
8. En Conectividad, seleccione la nube privada virtual (VPC) que alojará el nuevo clúster de base de datos entre las opciones disponibles. Aquí puede elegir Crear nueva VPC si desea que Neptune cree la VPC por usted. Debe crear una instancia de Amazon EC2 en esta misma VPC para acceder a la instancia de Neptune (para obtener más información, consulte Cada clúster de base de datos de Amazon Neptune reside en una Amazon VPC). Tenga en cuenta que no puede cambiar la VPC después de crear el clúster de base de datos.

   Si lo necesita, puede configurar aún más la conectividad del clúster en Configuración de conectividad adicional:

   a. En Grupo de subredes, puede elegir el grupo de subredes de base de datos de Neptune que se debe usar para el nuevo clúster de base de datos. Si la VPC no tiene aún ningún grupo de subred, Neptune crea un grupo de subred de base de datos (consulte Cada clúster de base de datos de Amazon Neptune reside en una Amazon VPC).
   b. En Grupos de seguridad de la VPC, elija uno o varios grupos de seguridad de la VPC existentes para proteger el acceso de red al nuevo clúster de base de datos o elija Crear nuevo si desea que Neptune cree uno por usted y, a continuación, proporcione un nombre para el nuevo grupo de seguridad de la VPC (consulte Cree un grupo de seguridad con la consola de VPC).

c. En Puerto de base de datos, introduzca el puerto TCP/IP que utilizará la base de datos para las conexiones de las aplicaciones. Neptune usa el número de puerto 8182 de forma predeterminada.

9. En Configuración de cuaderno, seleccione Crear cuaderno si desea que Neptune cree cuadernos de Jupyter en el entorno de trabajo de Neptune (consulte Uso de los cuadernos de gráficos de Neptune para empezar rápidamente y Uso del entorno de trabajo de Neptune para alojar los cuadernos de Neptune). A continuación, puede elegir cómo deben configurarse los nuevos cuadernos:

   a. En Tipo de instancia de cuaderno, elija una de las clases de instancias disponibles para el cuaderno.

   b. En Nombre del cuaderno, escriba un nombre para el cuaderno.

   c. Si lo desea, también puedes introducir una descripción del cuaderno en Descripción: opcional.

   d. En el nombre del rol de IAM, elija que Neptune cree un rol de IAM para el cuaderno e introduzca un nombre para el nuevo rol, o elija un rol de IAM existente entre los roles disponibles.

   e. Por último, elija si el cuaderno se conecta a Internet directamente o a través de Amazon SageMaker o a través de una VPC con una puerta de enlace NAT. Consulte Conexión de una instancia de cuaderno a los recursos de una VPC para obtener más información.

10. En Etiquetas, puede asociar hasta 50 etiquetas al nuevo clúster de base de datos.

11. En Configuración adicional, puede realizar más ajustes para el nuevo clúster de base de datos (en muchos casos, puede omitirlos y aceptar, por ahora, los valores predeterminados):

Opción	Qué puede hacer
DB Instance Identifier (Identificador de instancias de bases de datos)	Puede proporcionar un nombre para la instancia del escritor del clúster. Si no lo hace, se utilizará un identificador predeterminado basado en el nombre del clúster. Si lo hace, escriba un nombre que sea único para todas las instancias de base de datos pertenecientes a su cuenta de AWS en la región actual. El identificador de instancias de bases de datos no distingue entre

Opción	Qué puede hacer
	mayúsculas y minúsculas, pero se almacena con todas las letras en minúsculas.
Grupo de parámetros de clúster de base de datos	Seleccione un grupo de parámetros de clúster de base de datos para definir la configuración predeterminada de todas las instancias de base de datos del clúster. A no ser que elija lo contrario, Neptune utilizará un grupo de parámetros de clúster de base de datos predeterminado. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros de Amazon Neptune.
Grupo de parámetros de base de datos	Seleccione un grupo de parámetros de clúster de base de datos para definir la configuración predeterminada de la instancia de base de datos principal del clúster. A no ser que elija lo contrario, Neptune utilizará un grupo de parámetros predeterminado. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros.

Opción	Qué puede hacer
IAM DB authentication (Autenticación de base de datos de IAM)	Si marca Habilitar autenticación de bases de datos de IAM, todos los accesos a la base de datos se autenticarán mediante AWS Identity and Access Management (IAM). **Important** Para ello, es necesario que use AWS Signature Version 4 para firmar todas las solicitudes. Para obtener más información, consulte Descripción general de AWS Identity and Access Management (IAM) en Amazon Neptune.
Failover priority (Prioridad de conmutación por error)	Elija `No preference` o un nivel de prioridad para la conmutación por error. Si elige un nivel y este presenta contención, se selecciona la réplica que tenga el mismo tamaño que la instancia principal.
Backup retention period (Periodo de retención de copia de seguridad)	Elija el periodo de tiempo, de 1 a 35 días, durante el que Neptune debe conservar las copias de seguridad automáticas de esta instancia de base de datos. Solo puede realizar una restauración puntual (PITR) en un momento dentro del periodo de retención de la copia de seguridad.
Copy tags to snapshots (Copiar etiquetas en instantáneas)	(Habilitada de forma predeterminada) Esta opción hace que todas las etiquetas asociadas al clúster de base de datos se copien en cualquier instantánea del mismo.

Opción	Qué puede hacer
Enable encryption	(Habilitada de forma predeterminada) Esta opción hace que los datos del clúster de base de datos se cifren en reposo. Si lo hace, elija la clave principal utilizada para proteger la clave que se usa para cifrar este volumen de base de datos. Puede seleccionar la clave `aws/rds` predeterm inada, elegir una de las claves principales de su cuenta, o escribir el ARN de una clave de otra cuenta. Para crear una clave de cifrado principal nueva, vaya a la pestaña Claves de cifrado de la consola de IAM. Para obtener más información, consulte Cifrado de los recursos de Neptune en reposo.
Registro de auditoría	Seleccione esta opción si desea que los registros de auditoría del clúster de base de datos se publiquen en los registros de CloudWatch.
Enable auto minor version upgrade (Habilita r la actualización automática de la versión secundaria)	(Habilitada de forma predeterminada) Esta opción hace que el clúster de base de datos se actualice automáticamente a nuevas versiones secundarias del motor una vez publicadas. Las actualizaciones automátic as se producen durante la ventana de mantenimiento de la base de datos. Consulte Uso de `AutoMinorVersionUpgrade` .

Opción	Qué puede hacer
Periodo de mantenimiento	Puede seleccionar un periodo específico durante el que desee que se produzcan las modificaciones pendientes en el clúster de base de datos, como, por ejemplo, un cambio en una clase de instancia de base de datos o un parche automático del motor. Todas estas operaciones de mantenimi ento se inician y se completan en el periodo seleccionado. Si no selecciona un período, Neptune asignará un periodo de mantenimi ento de forma arbitraria.
Enable deletion protection (Habilitar la protección contra la eliminación	(Habilitada de forma predeterminada) La protección ante eliminaciones impide la eliminación del clúster de base de datos. Debe deshabilitarla de forma explícita para eliminar el clúster de base de datos.

12. Elija Crear base de datos para lanzar su nuevo clúster de base de datos de Neptune y su instancia principal.

    En la consola de Amazon Neptune, el nuevo clúster de base de datos aparece en la lista de bases de datos. El clúster de base de datos tendrá el estado Creating (Creándose) hasta que se cree y se pueda usar. Cuando el estado cambie a Available (Disponible), podrá conectarse a la instancia principal de su clúster de base de datos. Dependiendo de la clase de instancia de base de datos y del almacenamiento asignado, es posible que las nuevas instancias tarden varios minutos en estar disponibles.

    Para ver el clúster que acaba de crear, elija la vista Bases de datos en la consola de Neptune.

    > **Note**
    >
    > Si elimina todas las instancias de base de datos de Neptune de un clúster de base de datos con la AWS Management Console, esta eliminará de forma automática el propio clúster de base de datos. Si utiliza la AWS CLI o el SDK, debe eliminar el clúster de base de datos de forma manual después de eliminar su última instancia.

Anote el valor de Punto de conexión de clúster. Lo necesitará para conectarse al clúster de base de datos de Neptune.

Detención e inicio de un clúster de base de datos de Amazon Neptune

Detener e iniciar los clústeres de Amazon Neptune le ayuda a administrar los costos de entornos de desarrollo y pruebas. Puede detener temporalmente todas las instancias de base de datos su clúster, en lugar de configurar y eliminar todas las instancias de base de datos cada vez que use el clúster.

Temas

Información general sobre la detención e inicio de un clúster de base de datos de Neptune

Durante los periodos en los que no necesite un clúster de Neptune, puede detener a la vez todas las instancias de dicho clúster. Puede volver a iniciar el clúster en cualquier momento que necesite usarlo. El inicio y la detención simplifican los procesos de configuración y eliminación de clústeres usados para desarrollo, pruebas o actividades similares que no requieren disponibilidad continua. Puede hacerlo en la AWS Management Console con una única acción, independientemente del número de instancias que haya en el clúster.

Mientras su instancia de base de datos esté detenida, solo se le cobrará el almacenamiento del clúster, las instantáneas manuales y el almacenamiento de la copia de seguridad automática dentro de su intervalo de retención especificado. No se le cobrarán las horas de ninguna instancia de base de datos.

Después de siete días, Neptune vuelve a iniciar automáticamente el clúster de base de datos para asegurarse de que no se quede atrás con las actualizaciones de mantenimiento necesarias.

Para minimizar los cargos de un clúster de Neptune de carga ligera, puede detener el clúster en lugar de eliminar todas sus réplicas de lectura. Para clústeres con más de una o dos instancias, eliminar y volver a crear con frecuencia las instancias de base de datos solo es práctico con la AWS CLI o la API de Neptune, y las eliminaciones también pueden ser difíciles de realizar en el orden correcto. Por ejemplo, debe eliminar todas las réplicas de lectura antes de eliminar la instancia principal para evitar activar el mecanismo de conmutación por error.

No utilice la opción de iniciar y detener si necesita mantener el clúster de base de datos en ejecución, pero desea reducir la capacidad. Si el clúster es demasiado costoso o no está muy ocupado, puede eliminar una o más instancias de base de datos o cambiar las instancias de base de datos para utilizar una clase de instancia más pequeña, pero no puede detener una instancia de base de datos individual.

Detención de un clúster de base de datos de Neptune

Cuando no lo vaya a utilizar durante un tiempo, puede detener un clúster de base de datos de Neptune en ejecución y, a continuación, volver a iniciarlo cuando lo necesite. Mientras el clúster esté detenido, se le cobrará el almacenamiento del clúster, las instantáneas manuales y el almacenamiento de la copia de seguridad automática dentro de su intervalo de retención especificado, pero no se le cobrarán las horas de instancia de base de datos.

La operación de detención detiene todas las instancias de réplica de lectura del clúster antes de detener la instancia principal, para evitar activar el mecanismo de conmutación por error.

Detención de un clúster de base de datos con la AWS Management Console

Para utilizar la AWS Management Console para detener un clúster de Neptune

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija un clúster. Puede realizar la operación de detención desde esta página o navegar a la página de detalles del clúster de bases de datos que desea detener.
3. En Actions (Acciones), elija Stop (Detener).

Detención de un clúster de base de datos con la AWS CLI

Para detener una instancia de base de datos mediante la AWS CLI, llame al comando stop-db-cluster con el parámetro `--db-cluster-identifier` para identificar el clúster de base de datos que desea detener.

Example

```
aws neptune stop-db-cluster --db-cluster-identifier mydbcluster
```

Detención de un clúster de base de datos con la API de administración de Neptune

Para detener una instancia de base de datos con la API de administración de Neptune, llame a la API StopDBCluster y utilice el parámetro `DBClusterIdentifier` para identificar el clúster de base de datos que desea detener.

Qué puede suceder mientras un clúster de base de datos está detenido

- Se puede restaurar a partir de una instantánea (consulte Restauración de una instantánea de clúster de base de datos).
- No se puede modificar la configuración del clúster de base de datos ni de ninguna de sus instancias de base de datos.
- No se pueden añadir ni eliminar instancias de base de datos del clúster.
- No se puede eliminar el clúster si todavía tiene instancias de base de datos asociadas.
- En general, debe volver a iniciar un clúster de base de datos detenido para realizar la mayoría de las acciones administrativas.
- Neptune aplica cualquier mantenimiento programado al clúster detenido en cuanto se vuelva a iniciar. Recuerde que, al cabo de siete días, Neptune vuelve a iniciar automáticamente un clúster detenido para que no se quede demasiado rezagado en el estado de mantenimiento.
- Neptune no realiza ninguna copia de seguridad automática de un clúster de base de datos detenido, porque los datos subyacentes no pueden cambiar mientras el clúster esté detenido.
- Neptune no amplía el periodo de retención de copia de seguridad del clúster de base de datos mientras esté detenido.

Inicio de un clúster de base de datos de Neptune detenido

Solo se puede iniciar un clúster de base de datos de Neptune que se encuentre en estado detenido. Cuando inicia el clúster, todas sus instancias de base de datos se vuelven disponibles otra vez. El clúster conserva sus ajustes de configuración como puntos de enlace, grupos de parámetros y grupos de seguridad de VPC.

Inicio de un clúster de base de datos detenido con la AWS Management Console

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.

2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija un clúster. Puede realizar la operación de inicio desde esta página o acceder a la página de detalles del clúster de base de datos y empezar ahí.
3. En Actions (Acciones), elija Start (Iniciar).

Inicio de un clúster de base de datos detenido con la AWS CLI

Para iniciar un clúster de base de datos detenido mediante la AWS CLI, llame al comando start-db-cluster mediante el parámetro `--db-cluster-identifier` para especificar el clúster de base de datos detenido que desea iniciar. Indique el nombre del clúster que haya elegido al crear el clúster de base de datos o utilice un nombre de instancia de base de datos que haya elegido, con `-cluster` anexado al final del mismo.

Example

```
aws neptune start-db-cluster --db-cluster-identifier mydbcluster
```

Inicio de un clúster de base de datos detenido mediante la API de administración de Neptune

Para iniciar un clúster de base de datos de Neptune con la API de administración de Neptune, llame a la API StartDBCluster mediante el parámetro `DBCluster` para especificar el clúster de base de datos detenido que desea iniciar. Indique el nombre del clúster que haya elegido al crear el clúster de base de datos o utilice un nombre de instancia de base de datos que haya elegido, con `-cluster` anexado al final del mismo.

Vaciado de un clúster de base de datos de Amazon Neptune con la API de restablecimiento rápido

La API de REST de restablecimiento rápido de Neptune le permite restablecer un gráfico de Neptune de forma rápida y sencilla, eliminando todos sus datos.

Puede hacerlo en un cuaderno de Neptune con el comando mágico de línea %db_reset.

> **Note**
>
> Esta característica está disponible a partir de la versión 1.0.4.0 del motor de Neptune.

- En la mayoría de los casos, la operación de restablecimiento rápido se completa en un par de minutos. La duración puede variar un poco en función de la carga del clúster cuando se inicia la operación.
- Una operación de restablecimiento rápido no genera E/S adicionales.
- El tamaño del volumen de almacenamiento no se reduce después de un restablecimiento rápido. En cambio, el almacenamiento se vuelve a utilizar a medida que se introducen nuevos datos. Esto significa que los tamaños de volumen de las instantáneas tomadas antes y después de una operación de restablecimiento rápido serán los mismos. Los tamaños de volumen de los clústeres restaurados con las instantáneas creadas antes y después de una operación de restablecimiento rápido también serán los mismos
- Como parte de la operación de restablecimiento, se reinician todas las instancias en el clúster de base de datos.

  > **Note**
  >
  > En casos excepcionales, estos reinicios del servidor también pueden provocar una conmutación por error del clúster.

> **Important**
>
> El uso del restablecimiento rápido puede interrumpir la integración del clúster de base de datos de Neptune con otros servicios. Por ejemplo:

- El restablecimiento rápido elimina todos los datos de transmisión de la base de datos y restablece completamente las transmisiones. Esto significa que es posible que los consumidores de transmisiones ya no funcionen sin una nueva configuración.
- El restablecimiento rápido elimina todos los metadatos sobre los recursos de SageMaker que utiliza Neptune ML, incluidos los trabajos y los puntos de conexión. Siguen existiendo en SageMaker, y puede seguir utilizando los puntos de conexión de SageMaker existentes para las consultas de inferencia de Neptune ML, pero las API de administración de Neptune ML ya no funcionan con ellos.
- Las integraciones, como, por ejemplo, la integración de búsqueda de texto completo con ElasticSearch, también se eliminan con el restablecimiento rápido y se tienen que restablecer manualmente antes de que sea posible volver a utilizarlas.

Para eliminar todos los datos de un clúster de base de datos de Neptune con la API

1. En primer lugar, se genera un token que, a continuación, se puede utilizar para restablecer la base de datos. El objetivo de este paso es evitar que alguien restablezca por error una base de datos.

   Para ello, debe enviar una solicitud `HTTP POST` al punto de conexión `/system` de la instancia de escritor del clúster de base de datos para especificar la acción `initiateDatabaseReset`.

   El comando `curl` con el tipo de contenido JSON sería:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{ "action" : "initiateDatabaseReset" }'
   ```

   O bien, con el tipo de contenido `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=initiateDatabaseReset '
   ```

La solicitud `initiateDatabaseReset` devuelve el token de restablecimiento en su respuesta JSON, tal y como se indica a continuación:

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

El token sigue siendo válido durante una hora (60 minutos) después de su emisión.

Si envía la solicitud a una instancia de lector o al punto de conexión de estado, Neptune lanzará una `ReadOnlyViolationException`.

Si envía varias solicitudes `initiateDatabaseReset`, solo el último token generado será válido para el segundo paso, en el que realmente realiza el restablecimiento.

Si el servidor se reinicia justo después de la solicitud `initiateDatabaseReset`, el token generado deja de ser válido y tendrá que enviar una nueva solicitud para obtener uno nuevo.

2. A continuación, envíe una solicitud `performDatabaseReset` con el token que ha recibido de `initiateDatabaseReset` a punto de conexión `/system` de la instancia de escritor del clúster de base de datos. De este modo, se eliminan todos los datos del clúster de base de datos.

   El comando `curl` con el tipo de contenido JSON es:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{
           "action" : "performDatabaseReset",
           "token" : "token_guid"
         }'
   ```

   O bien, con el tipo de contenido `x-www-form-urlencoded`:

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
   ```

```
  -d 'action=performDatabaseReset&token=token_guid'
```

La solicitud devuelve una respuesta JSON. Si se acepta la solicitud, la respuesta es:

```
{
  "status" : "200 OK"
}
```

Si el token que envió no coincide con el que se emitió, la respuesta tendrá el siguiente aspecto:

```
{
  "code" : "InvalidParameterException",
  "requestId":"token_guid",
  "detailedMessage" : "System command parameter 'token' : 'token_guid' does not
 match database reset token"
}
```

Si se acepta la solicitud y comienza el restablecimiento, el servidor se reinicia y elimina los datos. No puede enviar ninguna otra solicitud al clúster de base de datos mientras se está restableciendo.

Uso de la API de restablecimiento rápido con IAM-Auth

Si ha habilitado IAM-auth en el clúster de base de datos, puede utilizar awscurl para enviar comandos de restablecimiento rápido que se autentiquen mediante IAM-Auth:

Uso de awscurl para enviar solicitudes de restablecimiento rápido con IAM-Auth

1. Configure las variables de entorno `AWS_ACCESS_KEY_ID` y `AWS_SECRET_ACCESS_KEY` correctamente (y también `AWS_SECURITY_TOKEN` si utiliza una credencial temporal).
2. Una solicitud `initiateDatabaseReset` tiene este aspecto:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "initiateDatabaseReset" }'
```

3. Una solicitud `performDatabaseReset` tiene este aspecto:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
```

```
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "performDatabaseReset" }'
```

Uso del comando mágico de línea **`%db_reset`** del entorno de trabajo de Neptune para restablecer un clúster de base de datos

El entorno de trabajo de Neptune admite un comando mágico de línea `%db_reset` que le permite restablecer rápidamente la base de datos en un cuaderno de Neptune.

Si invoca el comando mágico sin ningún parámetro, verá una pantalla en la que se le preguntará si desea eliminar todos los datos del clúster, con una casilla de verificación en la que se le pide que confirme que los datos del clúster dejarán de estar disponibles después de eliminarlos. En ese momento, puede elegir entre eliminar los datos o cancelar la operación.

Una opción más peligrosa es invocar `%db_reset` con la opción `--yes` o `-y`, lo que hace que la eliminación se realice sin más solicitudes.

También puedes realizar el restablecimiento en dos pasos, igual que con la API de REST:

```
%db_reset --generate-token
```

La respuesta es:

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

Haga lo siguiente:

```
%db_reset --token new_token_guid
```

La respuesta es:

```
{
```

```
  "status" : "200 OK"
}
```

Códigos de error comunes para las operaciones de restablecimiento rápido

Código de error de Neptune	Estado HTTP	Mensaje	Ejemplo
`InvalidParameterException`	400	El parámetro del comando del sistema “*acción*” tiene el valor no admitido “*XXX*”	Parámetro no válido
`InvalidParameterException`	400	Se han proporcionado demasiados valores para: *acción*	Una solicitud de restablecimiento rápido con más de una acción enviada con el encabezado “Content-type:application/x-www-form-urlencoded”
`InvalidParameterException`	400	Campo duplicado “acción”	Una solicitud de restablecimiento rápido con más de una acción enviada con el encabezado “Content-Type: application/json”
`MethodNotAllowedException`	400	Ruta incorrecta a: /*bad_endpoint*	Solicitud enviada a un punto de conexión incorrecto
`MissingParameterException`	400	Faltan parámetros obligatorios: [acción]	Una solicitud de restablecimiento rápido no incluye el

Código de error de Neptune	Estado HTTP	Mensaje	Ejemplo
			parámetro “acción” necesario
`ReadOnlyViolationException`	400	No se permiten escritores en una instancia de réplica de lectura	Se ha enviado una solicitud de restablecimiento rápido a un lector o punto de conexión de estado
`AccessDeniedException`	403	Falta token de autenticación	Se ha enviado una solicitud de restablecimiento rápido sin las firmas correctas a un punto de conexión de base de datos con la opción IAM-Auth habilitada
`ServerShutdownException`	500	El restablecimiento de la base de datos está en curso. Vuelva a intentar la consulta cuando el clúster esté disponible.	Cuando comienza el restablecimiento rápido, se produce un error en las consultas de Gremlin/Sparql existentes y entrantes.

Adición de instancias de lector de Neptune a un clúster de base de datos

En clústeres de base de datos de Neptune, hay una instancia de base de datos principal y hasta 15 instancias de lector de Neptune. La instancia principal de base de datos admite operaciones de lectura y escritura y realiza todas las modificaciones de los datos en el volumen del clúster. Las instancias de lector de Neptune se conectan con el mismo volumen de almacenamiento que la instancia de base de datos principal y solo admite operaciones de lectura.

Utilice las instancias de lector para descargar las cargas de trabajo de lectura desde la instancia de base de datos principal.

Le recomendamos que distribuya la instancia principal y los lectores de Neptune del clúster de base de datos entre varias zonas de disponibilidad para mejorar la disponibilidad del clúster de base de datos.

En la siguiente sección se describe cómo crear una instancia de lector en el clúster de base de datos.

Creación de una instancia de lector de Neptune con la consola

Tras crear la instancia principal para el clúster de base de datos de Neptune, puede añadir instancias de lector de Neptune adicionales con la consola de Neptune.

Para crear una instancia de lector de Neptune con la AWS Management Console

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione el clúster de base de datos en el que desee crear la instancia de lector.
4. Elija Acciones y, a continuación, Añadir lector.
5. En la página Crear instancia de base de datos de réplica, especifique las opciones de la réplica de Neptune. En la siguiente tabla se muestra la configuración de una réplica de lectura de Neptune.

Para esta opción...	Haga lo siguiente
DB instance class (Clase de instancia de base de datos)	Elija una clase de instancia de base de datos que defina los requisitos de procesamiento y memoria de la réplica de Neptune. Para obtener un listado actual de las clases de instancia de base de datos que Neptune ofrece en diferentes regiones, consulte la página de precios de Neptune.
Zona de disponibilidad	Especifique una zona de disponibilidad. Elija una zona diferente de la instancia de base de datos principal. La lista incluye únicamente las zonas de disponibilidad mapeadas por el grupo de subredes de base de datos del clúster de base de datos.
Encryption (Cifrado)	Habilite o deshabilite el cifrado.
Read replica source (Origen de réplica de lectura)	Seleccione el identificador de la instancia principal para la que se debe crear una réplica de Neptune.

Para esta opción...	Haga lo siguiente
DB Instance Identifier (Identificador de instancias de bases de datos)	Escriba un nombre para la instancia que sea único para su cuenta en la región que ha seleccionado. Puede optar por añadir al nombre información como la zona de disponibilidad seleccionada, por ejemplo, `neptune-us-east-1c` .
Puerto de base de datos	El número de puerto en el que la base de datos acepta conexiones.
DB Parameter Group (Grupo de parámetros de base de datos)	El grupo de parámetros para esta instancia.
Log exports (Exportaciones de registros)	Elija los registros que desee publicar, si los hay.
Auto Minor Version Upgrade	Seleccione Sí si desea habilitar la réplica de Neptune para recibir automáticamente actualizaciones de las versiones secundarias del motor de base de datos de Neptune cuando estén disponibles. La opción Auto minor version upgrade (Actualización automática de versión secundaria) solo se aplica a las versiones secundarias. No se aplica a los parches de mantenimiento del motor, que siempre se aplican automáticamente para mantener la estabilidad del sistema.

6. Elija Crear réplica de lectura para crear la instancia de réplica de Neptune.

Para eliminar una instancia de lector de Neptune de un clúster de base de datos, siga las instrucciones de Eliminar una instancia de base de datos en Amazon Neptune.

Modificación de un clúster de base de datos de Neptune con la consola

Cuando modifique una instancia de base de datos mediante la AWS Management Console, puede optar por aplicar los cambios mediante la selección de la opción Apply Immediately (Aplicar inmediatamente). Si opta por aplicar los cambios inmediatamente, se aplican los nuevos cambios y cualquier cambio de la cola de modificaciones pendientes a la vez.

Si decide no aplicar los cambios inmediatamente, estos se colocan en la cola de modificaciones pendientes. Los cambios pendientes en la cola se aplican durante el siguiente periodo de mantenimiento.

Important

Si alguna modificación pendiente requiere un tiempo de inactividad, elegir aplicarlas inmediatamente puede producir un tiempo de inactividad inesperado para la instancia de base de datos en cuestión. No hay tiempo de inactividad para el resto de instancias de base de datos en el clúster de base de datos.

Note

Al modificar un clúster de base de datos en Neptune, el ajuste Aplicar inmediatamente solo afecta a los cambios realizados en el Identificador de clúster de base de datos, Autenticación de base de datos IAM. Todas las demás modificaciones se aplican de inmediato, con independencia del valor del ajuste aplicar inmediatamente.

Para modificar un clúster de base de datos mediante la consola

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Clusters (Clústeres) y elija el clúster de base de datos que desea modificar.
3. Elija Actions (Acciones) y, a continuación, Modify cluster (Modificar clúster). Aparece la página Modificar clúster de base de datos.

4. Cambie los parámetros que desee.

> **Note**
>
> En la consola, algunos cambios en el nivel de instancia solo se aplican a la instancia actual de la base de datos, mientras que otros se aplican a la totalidad del clúster de base de datos. Para cambiar una configuración que modifique todo el clúster de base de datos en el nivel de la instancia en la consola, siga las instrucciones de Modificación de una instancia de base de datos en un clúster de base de datos.

5. Cuando haya realizado todos los cambios que desee, elija Continue (Continuar) y compruebe el resumen.
6. Para aplicar los cambios inmediatamente, seleccione Apply immediately.
7. En la página de confirmación, revise los cambios. Si son correctos, elija Modify cluster (Modificar clúster) para guardarlos.

   Para editar los cambios elija Back (Atrás) o para cancelar sus cambios elija Cancel (Cancelar).

Modificación de una instancia de base de datos en un clúster de base de datos

Para modificar una instancia de base de datos en un clúster de base de datos con la consola.

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Instances (Instancias) y, a continuación, seleccione la instancia de base de datos que desea modificar.
3. Elija Instance actions (Acciones de instancias) y, a continuación, Modify (Modificar). Aparece la página Modificar instancia de base de datos.
4. Cambie los parámetros que desee.

> **Note**
>
> Algunos ajustes se aplican a todo el clúster de la base de datos y deben cambiarse a nivel del clúster. Para cambiar dichos ajustes, siga las instrucciones de Modificación de un clúster de base de datos de Neptune con la consola.

En la AWS Management Console, algunos cambios en el nivel de instancia solo se aplican a la instancia actual de la base de datos, mientras que otros se aplican a la totalidad del clúster de base de datos.

5. Cuando haya realizado todos los cambios que desee, elija Continue (Continuar) y compruebe el resumen.
6. Para aplicar los cambios inmediatamente, seleccione Apply immediately.
7. En la página de confirmación, revise los cambios. Si son correctos, elija Modify DB Instance para guardarlos.

   Para editar los cambios elija Back (Atrás) o para cancelar sus cambios elija Cancel (Cancelar).

Rendimiento y escalado en Amazon Neptune

Los clústeres de base de datos y las instancias de Neptune se escalan en tres niveles diferentes:

- Escalado del almacenamiento
- Escalado de instancia;
- Escalado de lectura

Escalado del almacenamiento en Neptune

El almacenamiento de Neptune se escala automáticamente con los datos del volumen de clúster. A medida que los datos aumentan, también lo hace el almacenamiento de volumen del clúster, hasta alcanzar 128 TiB en todas las regiones compatibles, excepto en China y GovCloud, donde está limitado a 64 TiB.

El tamaño del volumen de clúster se comprueba cada hora para determinar los costos de almacenamiento.

El almacenamiento consumido por la base de datos de Neptune se factura en incrementos mensuales en GB y las E/S se facturan en incrementos de solicitudes por millón. Solo tiene que pagar por el almacenamiento y las E/S que consuma la base de datos de Neptune y no tiene que hacerlo con antelación.

Para obtener información acerca de los precios, consulte la página de producto de Neptune.

Escalado de instancias en Neptune

Puede escalar el clúster de base de datos de Neptune como considere necesario modificando la clase de instancia de base de datos para cada instancia de base de datos del clúster de base de datos. Neptune admite varias clases de instancias de bases de datos optimizadas.

Escalado de lectura en Neptune

Puede realizar el escalado de lectura del clúster de base de datos de Neptune creando un máximo de 15 réplicas de Neptune en el clúster de base de datos. Cada réplica de Neptune devuelve los mismos datos desde el volumen de clúster con un retardo de réplica mínimo, a menudo mucho menos de 100 milisegundos una vez que la instancia principal haya escrito una actualización. A medida que el tráfico de lectura aumenta, puede crear réplicas de Neptune adicionales y conectarlas

directamente para distribuir la carga de lectura del clúster de base de datos. Las réplicas de Neptune no tienen que ser de la misma clase de instancia de base de datos que la instancia principal.

Para obtener más información acerca de la adición de réplicas de Neptune a un clúster de base de datos, consulte Adición de instancias de lector.

Escalado automático del número de réplicas de un clúster de bases de datos de Amazon Neptune

Puede usar el escalado automático de Neptune para ajustar automáticamente el número de réplicas de Neptune en un clúster de base de datos para cumplir con los requisitos de conectividad y carga de trabajo. El escalado automático permite que el clúster de base de datos de Neptune gestione los aumentos de carga de trabajo y, luego, cuando la carga de trabajo disminuye, elimina las réplicas innecesarias, por lo que no tiene que pagar por la capacidad no utilizada.

Solo puede usar el escalado automático con un clúster de base de datos de Neptune que ya tenga una instancia de escritor principal y al menos una instancia de réplica de lectura (consulte Clústeres e instancias de base de datos de Amazon Neptune). Además, todas las instancias de réplica de lectura del clúster deben estar en un estado disponible. Si alguna réplica de lectura está en un estado distinto al disponible, el ajuste de escalado automático de Neptune no hace nada hasta que estén disponibles todas las réplicas de lectura del clúster.

Si necesita crear un nuevo clúster, consulte Creación de un clúster de base de datos.

Con el AWS CLI, se define y se aplica una política de escalado al clúster de base de datos. También puedes utilizarla AWS CLI para editar o eliminar tu política de autoscalamiento. La política especifica los siguientes parámetros de escalado automático:

- El número mínimo y máximo de réplicas que se van a tener en el clúster.
- Un `ScaleOutCooldown` intervalo entre la actividad de escalado de adición de réplicas y un `ScaleInCooldown` intervalo entre la actividad de escalado de eliminación de réplicas.
- La CloudWatch métrica y el valor de activación de la métrica para escalar hacia arriba o hacia abajo.

La frecuencia de las acciones de escalado automático de Neptune se reduce de varias maneras:

- Inicialmente, para que el escalado automático añada o elimine un lector, la alarma de umbral máximo `CPUUtilization` debe superarse durante al menos tres minutos o la alarma de umbral mínimo debe superarse durante al menos 15 minutos.
- Tras la primera adición o eliminación, la frecuencia de las siguientes acciones de escalado automático de Neptune queda limitada por la configuración `ScaleOutCooldown` y `ScaleInCooldown` de la política de escalado automático.

Si la CloudWatch métrica que utiliza alcanza el umbral máximo que especificó en su política, si el `ScaleOutCooldown` intervalo ha transcurrido desde la última acción de autoescalado y si su clúster de base de datos aún no tiene el número máximo de réplicas que estableció, el autoescalado de Neptune crea una nueva réplica con el mismo tipo de instancia que la instancia principal del clúster de base de datos.

Del mismo modo, si la métrica alcanza el umbral mínimo que especificó y si el intervalo `ScaleInCooldown` ha transcurrido desde la última acción de escalado automático, y si el clúster de base de datos tiene más réplicas que el número mínimo que especificó, el escalado automático de Neptune elimina una de las réplicas.

> **Note**
>
> Escalado automático de Neptune solo elimina las réplicas que ha creado. No elimina las réplicas preexistentes.

Con el parámetro de clúster de base de datos neptune_autoscaling_config, también puede especificar el tipo de instancia de las nuevas réplicas de lectura que crea el escalado automático de Neptune, los periodos de mantenimiento de esas réplicas de lectura y las etiquetas que se asociarán a cada una de las nuevas réplicas de lectura. Estos ajustes de configuración se proporcionan en una cadena JSON como valor del parámetro `neptune_autoscaling_config`, tal y como se muestra a continuación:

```
"{
  \"tags\": [
    { \"key\" : \"reader tag-0 key\", \"value\" : \"reader tag-0 value\" },
    { \"key\" : \"reader tag-1 key\", \"value\" : \"reader tag-1 value\" },
  ],
  \"maintenanceWindow\" : \"wed:12:03-wed:12:33\",
  \"dbInstanceClass\" : \"db.r5.xlarge\"
}"
```

Tenga en cuenta que a todas las comillas de la cadena JSON se les debe aplicar una secuencia de escape con un carácter de barra diagonal inversa (\). Todos los espacios en blanco de la cadena son opcionales, como de costumbre.

Cualquiera de los tres ajustes de configuración no especificados en el parámetro `neptune_autoscaling_config` se copia de la configuración de la instancia de escritor principal del clúster de base de datos.

Cuando el escalado automático añade una nueva instancia de réplica de lectura, antepone `autoscaled-reader` al ID de instancia de base de datos (por ejemplo, `autoscaled-reader-7r7t7z3lbd-20210828`). También añade una etiqueta a cada réplica de lectura que cree con la clave `autoscaled-reader` y un valor de TRUE. Puede verla en la pestaña Etiquetas de la página de detalles de la instancia de base de datos de la AWS Management Console.

```
"key" : "autoscaled-reader",  "value" : "TRUE"
```

El nivel de promoción de todas las instancias de réplica de lectura creadas mediante el escalado automático es el de menor prioridad, que, de forma predeterminada es 15. Esto significa que durante una conmutación por error, cualquier réplica con una mayor prioridad, como una creada manualmente, se promocionaría primero. Consulte Tolerancia a errores para un clúster de base de datos de Neptune.

El autoescalado de Neptuno se implementa mediante Application Auto Scaling con una política de escalado de seguimiento de objetivos que utiliza una métrica de `CPUUtilization` CloudWatch Neptuno como métrica predefinida.

Uso del escalado automático en un clúster de base de datos de Neptune sin servidor

Neptune sin servidor responde mucho más rápido que el escalado automático de Neptune cuando la demanda supera la capacidad de una instancia y escala verticalmente la instancia en lugar de añadir otra. Mientras que el escalado automático se ha diseñado para adaptarse a aumentos o disminuciones relativamente estables de la carga de trabajo, la tecnología sin servidor destaca a la hora de gestionar los picos y fluctuaciones rápidos de la demanda.

Al comprender sus puntos fuertes, puede combinar el escalado automático y la tecnología sin servidor para crear una infraestructura flexible que gestione los cambios en la carga de trabajo de forma eficaz y haga frente a la demanda y, al mismo tiempo, minimice los costos.

Para permitir que el escalado automático funcione de forma eficaz junto con la tecnología sin servidor, es importante establecer la configuración `maxNCU` del clúster sin servidor lo suficientemente alta como para adaptarse a los picos y los cambios breves en la demanda. De lo contrario, los cambios transitorios no desencadenan el escalado sin servidor, lo que puede provocar que el

escalado automático genere muchas instancias adicionales innecesarias. Si la opción `maxNCU` está establecida en un nivel lo suficientemente alto, el escalado sin servidor puede gestionar esos cambios de forma más rápida y económica.

Cómo habilitar el escalado automático para Amazon Neptune

El escalado automático solo se puede habilitar para un clúster de base de datos de Neptune con la AWS CLI. No se puede habilitar el escalado automático con la AWS Management Console.

Además, el escalado automático no se admite en las siguientes regiones de Amazon:

- África (Ciudad del Cabo): `af-south-1`
- Medio Oriente (EAU): `me-central-1`
- AWS GovCloud (Este de EE. UU.): `us-gov-east-1`
- AWS GovCloud (EEUU-Oeste): `us-gov-west-1`

La habilitación del escalado automático para un clúster de base de datos de Neptune consta de tres pasos:

1. Registro de un clúster de base de datos con Application Auto Scaling

El primer paso para habilitar el escalado automático de un clúster de base de datos de Neptune consiste en registrar el clúster con Application Auto Scaling, con la AWS CLI o uno de los SDK de Application Auto Scaling. El clúster ya debe tener una instancia principal y al menos una instancia de lectura de réplica:

Por ejemplo, para registrar un clúster para que se escale automáticamente con de una a ocho réplicas adicionales, puede usar el AWS CLI `register-scalable-target`comando de la siguiente manera:

```
aws application-autoscaling register-scalable-target \
  --service-namespace neptune \
  --resource-id cluster:(your DB cluster name) \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --min-capacity 1 \
  --max-capacity 8
```

Esto equivale a usar la operación de la API de Application Auto Scaling `RegisterScalableTarget`.

El comando AWS CLI `register-scalable-target` usa los siguientes parámetros:

- **`service-namespace`**: se establece en `neptune`.

  Este parámetro equivale al parámetro `ServiceNamespace` de la API de Application Auto Scaling.

- **`resource-id`**: configure esta opción en el identificador de recursos del clúster de base de datos de Neptune. El tipo de recurso es `cluster`, seguido de dos puntos (“`:`”) y, a continuación, el nombre del clúster de base de datos.

  Este parámetro equivale al parámetro `ResourceID` de la API de Application Auto Scaling.

- **`scalable-dimension`**: la dimensión escalable en este caso es el número de instancias de réplica en el clúster de base de datos, por lo que este parámetro está establecido en `neptune:cluster:ReadReplicaCount`.

  Este parámetro equivale al parámetro `ScalableDimension` de la API de Application Auto Scaling.

- **`min-capacity`**: el número mínimo de instancias de réplica de base de datos de lector que Application Auto Scaling va a administrar. Este valor debe establecerse en el rango comprendido entre 0 y 15 y debe ser igual o inferior al valor especificado para el número máximo de réplicas de Neptune en `max-capacity`. Debe haber al menos un lector en el clúster de base de datos para que funcione el escalado automático.

  Este parámetro equivale al parámetro `MinCapacity` de la API de Application Auto Scaling.

- **`max-capacity`**: el número máximo de instancias de réplica de base de datos de lector en el clúster de base de datos, incluidas las instancias preexistentes y las instancias nuevas administradas por Application Auto Scaling. Este valor debe establecerse en el rango comprendido entre 0 y 15 y debe ser igual o superior al valor especificado para el número mínimo de réplicas de Neptune en `min-capacity`.

  El `max-capacity` AWS CLI parámetro es equivalente al `MaxCapacity` parámetro de la API Application Auto Scaling.

Al registrar el clúster de base de datos, Application Auto Scaling crea un rol vinculado a un servicio `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster`. Para obtener más información, consulte Roles vinculados a servicios para el escalado automático de aplicaciones en la Guía del usuario de Application Auto Scaling.

2. Definición de una política de escalado automático para su uso con un clúster de base de datos

Una política de escalado de seguimiento de destino se define como un objeto de texto JSON que también se puede guardar en un archivo de texto. Para Neptuno, esta política actualmente solo puede usar la métrica de Neptuno como una `CPUUtilization` CloudWatch métrica predefinida denominada. `NeptuneReaderAverageCPUUtilization`

A continuación se muestra un ejemplo de la política de configuración de escalado de seguimiento de destino para Neptune:

```
{
  "PredefinedMetricSpecification": { "PredefinedMetricType":
 "NeptuneReaderAverageCPUUtilization" },
  "TargetValue": 60.0,
  "ScaleOutCooldown" : 600,
  "ScaleInCooldown" : 600
}
```

Este elemento **`TargetValue`** incluye el porcentaje de utilización de la CPU por encima del cual el escalado automático se escala horizontalmente (es decir, añade más réplicas) y por debajo se escala verticalmente (es decir, elimina las réplicas). En este caso, el porcentaje de destino que desencadena el escalado es de `60.0` %.

El elemento **`ScaleInCooldown`** especifica la cantidad de tiempo, en segundos, tras completarse un actividad de reducción horizontal antes de que pueda comenzar otra. El valor predeterminado es de 300 segundos. En este caso, el valor de 600 especifica que deben transcurrir al menos diez minutos entre la finalización de la eliminación de una réplica y el inicio de otra.

El elemento **`ScaleOutCooldown`** especifica la cantidad de tiempo, en segundos, tras completarse un actividad de escalado horizontal antes de que pueda comenzar otra. El valor predeterminado es de 300 segundos. En este caso, el valor de 600 especifica que deben transcurrir al menos diez minutos entre la finalización de la adición de una réplica y el inicio de otra.

El elemento **`DisableScaleIn`** es un valor booleano que, si está presente y establecido en `true` deshabilita la reducción horizontal por completo, lo que significa que el escalado automático puede añadir réplicas, pero nunca eliminará ninguna. De forma predeterminada, el escalado interno está habilitado, y el valor `DisableScaleIn` está establecido en `false`.

Tras registrar el clúster de base de datos de Neptune con Application Auto Scaling y definir una política de escalado JSON en un archivo de texto, puede aplicar la política de escalado al clúster de base de datos registrado. Para ello, puede utilizar el AWS CLI `put-scaling-policy`comando con parámetros como los siguientes:

```
aws application-autoscaling put-scaling-policy \
  --policy-name (name of the scaling policy) \
  --policy-type TargetTrackingScaling \
  --resource-id cluster:(name of your Neptune DB cluster) \
  --service-namespace neptune \
  --scalable-dimension neptune:cluster:ReadReplicaCount \
  --target-tracking-scaling-policy-configuration file://(path to the JSON configuration
 file)
```

Cuando haya aplicado la política de escalado automático, este se habilitará en el clúster de base de datos.

También puede usar el AWS CLI `put-scaling-policy`comando para actualizar una política de autoscalamiento existente.

Consulte también la PutScalingPolítica en la Referencia de la API Application Auto Scaling.

Eliminación del escalado automático de un clúster de base de datos de Neptune

Para eliminar el autoscaling de un clúster de base de datos de Neptune, utilice los comandos AWS CLIdelete-scaling-policy y deregister-scalable-target.

Mantenimiento del clúster de base de datos de Amazon Neptune

Neptune realiza un mantenimiento periódico de todos los recursos que utiliza, incluidos:

- Sustitución del hardware subyacente según sea necesario. Esto ocurre en segundo plano, sin que tenga que realizar ninguna acción y, por lo general, no afecta a sus operaciones.
- Actualización del sistema operativo subyacente. Las actualizaciones del sistema operativo de las instancias del clúster de base de datos se llevan a cabo para mejorar el rendimiento y la seguridad, por lo que, en general, debe completarlas lo antes posible. Normalmente, las actualizaciones tardan unos 10 minutos en completarse. Las actualizaciones del sistema operativo no cambian la versión del motor de la base de datos ni la clase de instancia de la base de datos.

  Normalmente, es mejor actualizar primero las instancias de lector en un clúster de base de datos y, posteriormente, la instancia de escritor. La actualización simultánea de los lectores y del escritor puede provocar un tiempo de inactividad en caso de una conmutación por error. Tenga en cuenta que no se realiza una copia de seguridad de forma automática de las instancias de base de datos antes de una actualización del sistema operativo, así que asegúrese de realizar copias de seguridad manuales antes de aplicar una actualización del sistema operativo.
- Actualización del motor de base de datos de Neptune. Neptune publica periódicamente una serie de actualizaciones del motor para introducir nuevas características y mejoras y corregir errores.

Números de la versión del motor

Numeración de versiones antes de la versión 1.3.0.0 del motor

Antes de noviembre de 2019, Neptune solo admitía una versión de motor a la vez y todos los números de versión de motor tenían el formato `1.0.1.0.200`*`<xxx>`*, donde `xxx` era el número de parche. Todas las nuevas versiones de motor se publicaron como parches de las versiones anteriores.

A partir de noviembre de 2019, Neptune empezó a admitir varias versiones, lo que permite a los clientes un mejor control sobre sus rutas de actualización. Como resultado, la numeración de las versiones de motor ha cambiado.

Desde noviembre de 2019 hasta la versión 1.3.0.0 del motor, los números de versión del motor constaban de 5 partes. Tome como ejemplo el número de versión `1.0.2.0.R2`:

- La primera parte siempre era 1.

- La segunda parte, `0` en `1.0.2.0.R2`), era el número de versión principal de la base de datos.
- La tercera y la cuarta parte, `2.0` en `1.0.2.0.R2`), eran números de versión secundarios.
- La quinta parte (`R2` en `1.0.2.0.R2`) era el número de parche.

La mayoría de las actualizaciones eran actualizaciones de parches, y la distinción entre parches y actualizaciones de versiones secundarias no siempre estaba clara.

Numeración de versiones a partir de la versión 1.3.0.0 del motor

A partir de la versión 1.3.0.0 del motor, Neptune cambió la forma en que se numeran y gestionan las actualizaciones del motor.

Los números de versión del motor ahora tienen cuatro partes, cada una de las cuales corresponde a un tipo de versión, como se indica a continuación:

product-version.*major-version*.*minor-version*.*patch-version*

Los cambios de no separación, que antes se publicaban como parches, ahora se publican como versiones secundarias que se pueden gestionar mediante la configuración de la instancia `AutoMinorVersionUpgrade`.

Esto significa que, si lo desea, puede recibir una notificación cada vez que se publique una nueva versión secundaria suscribiéndose al evento `RDS-EVENT-0156` (consulte Suscripción a la notificación de eventos de Neptune).

Las versiones de los parches ahora están reservadas para correcciones urgentes seleccionadas, y se numeran utilizando la última parte del número de versión (`*.*.*.1`, `*.*.*.2`, etc.).

Diferentes tipos de versiones de motor en Amazon Neptune

Los cuatro tipos de versión del motor que corresponden a las cuatro partes del número de versión del motor son los siguientes:

- Versión del producto: solo cambia si el producto sufre cambios radicales y fundamentales en la funcionalidad o la interfaz. La versión actual del producto Neptune es 1.
- Versión principal: las versiones principales introducen nuevas características y cambios importantes y, por lo general, tienen una vida útil de al menos dos años.
- Versión secundaria: las versiones secundarias pueden contener nuevas características, mejoras y correcciones de errores, pero no contienen cambios importantes. Puede elegir si desea que se

apliquen automáticamente o no durante el siguiente período de mantenimiento, y también puede optar por recibir una notificación cada vez que se publique una.

- Versión de parche: las versiones de parche se publican únicamente para corregir errores urgentes o realizar actualizaciones de seguridad críticas. Rara vez contienen cambios importantes y se aplican automáticamente durante el siguiente período de mantenimiento tras su publicación.

Actualizaciones de las versiones principales de Amazon Neptune

Por lo general, una actualización de una versión principal introduce una o varias características nuevas importantes y, a menudo, contiene cambios bruscos. Por lo general, tiene una vida útil de soporte de alrededor de dos años. Las versiones principales de Neptune se indican en las versiones del motor, junto con la fecha en que se publicaron y su fecha estimada de fin de vida útil.

Las actualizaciones de las versiones principales son totalmente opcionales hasta que la versión principal que esté utilizando llegue al final de su vida útil. Si decide actualizar a una nueva versión principal, debe instalar la nueva versión usted mismo con la AWS CLI o la consola de Neptune, tal y como se describe en Actualizaciones de la versión principal.

No obstante, si la versión principal que está utilizando llega al final de su vida útil, se le notificará que debe actualizar a una versión principal más reciente. A continuación, si no realiza la actualización dentro de un período de gracia tras la notificación, se programará automáticamente una actualización a la versión principal más reciente durante el siguiente período de mantenimiento. Para obtener más información, consulte Vida útil de la versión del motor.

Actualizaciones de versiones secundarias de Amazon Neptune

La mayoría de las actualizaciones del motor de Neptune son actualizaciones de versiones secundarias. Se producen con bastante frecuencia y no contienen cambios importantes.

Si el campo `AutoMinorVersionUpgrade` está establecido en `true` en la instancia de escritura (principal) del clúster de base de datos, las actualizaciones de las versiones secundarias se aplicarán automáticamente a todas las instancias del clúster de base de datos durante el siguiente período de mantenimiento tras su publicación.

Si el campo `AutoMinorVersionUpgrade` está establecido en `false` en la instancia de escritor de su clúster de base de datos, solo se aplicarán si las instala de forma explícita.

> **Note**
>
> Las actualizaciones de las versiones secundarias son independientes (no dependen de las actualizaciones anteriores de la misma versión principal) y acumulativas (contienen todas las características y correcciones introducidas en las actualizaciones de versiones secundarias anteriores). Esto significa que puede instalar cualquier actualización de versión secundaria independientemente de si ha instalado las anteriores o no.

Realizar un seguimiento de las publicaciones de versiones secundarias es sencillo. Para ello, suscríbase al evento `RDS-EVENT-0156` (consulte Suscripción a la notificación de eventos de Neptune). Recibirá una notificación cada vez que se publique una nueva versión secundaria.

Además, tanto si se suscribe a las notificaciones como si no, siempre podrá comprobar qué actualizaciones están pendientes.

Actualizaciones de versiones de parche de Amazon Neptune

En el caso de problemas de seguridad u otros defectos graves que afecten a la fiabilidad de la instancia, Neptune implementa parches obligatorios. Se aplican a todas las instancias del clúster de base de datos durante el siguiente período de mantenimiento sin ninguna intervención por su parte.

Una versión de parche solo se implementa cuando los riesgos de no implementarla superan los riesgos y el tiempo de inactividad asociados a su implementación. Las versiones de parche tiene lugar con poca frecuencia (normalmente, una vez cada varios meses) y suele requerir tan solo una fracción del período de mantenimiento.

Planificación de la vida útil de la versión principal del motor de Amazon Neptune

Las versiones del motor de Neptune casi siempre llegan al final de su vida útil al final de un trimestre natural. Solo se realizan excepciones cuando surgen problemas importantes de seguridad o disponibilidad.

Cuando una versión del motor llegue al final de su vida útil, tendrá que actualizar la base de datos de Neptune a una versión más reciente.

En general, las versiones del motor de Neptune siguen estando disponibles de la siguiente manera:

- Versiones de motor secundarias: las versiones del motor secundarias permanecen disponibles durante al menos 6 meses después de su lanzamiento.
- Versiones de motor principales: las versiones del motor principales permanecen disponibles durante al menos 6 meses después de su lanzamiento.

Al menos 3 meses antes de que la versión del motor llegue al final de su vida útil, AWS enviará una notificación automática por correo electrónico a la dirección de correo electrónico asociada a su cuenta de AWS y publicará el mismo mensaje en su AWS Health Dashboard. Esto le dará tiempo a planificar y prepararse para la actualización.

Cuando una versión del motor llegue al final de su vida útil, ya no podrá crear nuevos clústeres o instancias con esa versión. Tampoco se podrán crear instancias con esa versión mediante el escalado automático.

Una versión del motor que llegue al final de su vida útil se actualiza automáticamente durante un período de mantenimiento. El mensaje que se le envía tres meses antes del final de la vida útil de la versión de motor contiene detalles sobre lo que implica esa actualización automática, incluida la versión a la que se actualizaría automáticamente, el impacto en sus clústeres de bases de datos y las acciones que recomendamos.

Important

Usted es responsable de mantener actualizadas las versiones de su motor de base de datos. AWSinsta a todos los clientes a actualizar sus bases de datos a la última versión del motor para poder beneficiarse de las medidas de seguridad, privacidad y disponibilidad más actuales. Si utiliza su base de datos en un motor o software no compatibles después de la fecha de caducidad (“motor heredado”), es probable que corra riesgos operativos, de seguridad y de privacidad, lo que incluye sufrir períodos de inactividad.
El funcionamiento de su base de datos en cualquier motor está sujeto al Acuerdo que rige su uso de los servicios de AWS. Los motores heredados no están disponibles de forma general. AWS ya no es compatible con el motor heredado y AWS puede limitar el acceso o el uso de cualquier motor heredado en cualquier momento si AWS determina que ese motor heredado representa un riesgo de seguridad o responsabilidad, o un riesgo de daño, para los servicios, AWS, sus filiales o cualquier tercero. Su decisión de seguir publicando su contenido en un motor heredado podría tener como consecuencia que su contenido deje de estar disponible, se dañe o sea irrecuperable. Las bases de datos que se ejecutan en un motor heredado están sujetas a las excepciones del acuerdo de nivel de servicio (SLA).

LAS BASES DE DATOS Y EL SOFTWARE RELACIONADO QUE SE EJECUTAN EN UN MOTOR HEREDADO CONTIENEN ERRORES, DEFECTOS O COMPONENTES DAÑINOS. EN CONSECUENCIA, Y SIN PERJUICIO DE CUALQUIER DISPOSICIÓN AL CONTRARIO DEL ACUERDO O EN LAS CONDICIONES DEL SERVICIO, AWS PROPORCIONA EL MOTOR HEREDADO “TAL CUAL”.

Administración de las actualizaciones del motor de su clúster de base de datos de Neptune

> **Note**
>
> Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos. En raras ocasiones puede ser necesaria una conmutación por error Multi-AZ para que se complete una actualización de mantenimiento en una instancia.
> En el caso de las actualizaciones de las versiones principales que pueden tardar más en aplicarse, puede utilizar una estrategia de implementación azul-verde para minimizar el tiempo de inactividad.

Determinación de la versión del motor que utiliza actualmente

Puede usar el comando AWS CLI `get-engine-status` para comprobar qué versión de motor utiliza actualmente su clúster de base de datos:

```
aws neptunedata get-engine-status
```

La salida de JSON incluye un campo `"dbEngineVersion"` como este:

```
  "dbEngineVersion": "1.3.0.0",
```

Comprobar qué actualizaciones están pendientes y disponibles

Puede comprobar las actualizaciones pendientes de su clúster de base de datos mediante la consola de Neptune. Seleccione Bases de datos en la columna de la izquierda y, a continuación, seleccione

su clúster de base de datos en el panel de bases de datos. Las actualizaciones pendientes se muestran en la columna Mantenimiento. Si selecciona Acciones y, a continuación, Mantenimiento, tiene tres opciones sobre qué hacer:

- Actualizar ahora.
- Actualizar en el siguiente periodo.
- Aplazar la actualización.

Puede enumerar las actualizaciones pendientes del motor con la AWS CLI de la siguiente manera:

```
aws neptune describe-pending-maintenance-actions \
  --resource-identifier (ARN of your DB cluster)
  --region (your region) \
  --engine neptune
```

También puede enumerar las actualizaciones de motor disponibles mediante la AWS CLI de la siguiente manera:

```
aws neptune describe-db-engine-versions \
  --region (your region) \
  --engine neptune
```

La lista de versiones de motor disponibles solo incluye las versiones que tienen un número de versión superior al actual y para las que se define una ruta de actualización.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. En una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código incluso sin ningún cambio brusco.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es utilizar la solución de implementación azul/verde de Neptune. De esta forma, puede ejecutar aplicaciones y consultas en la nueva versión sin que ello afecte a su clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

Periodo de mantenimiento de Neptune

El período de mantenimiento semanal es un período de 30 minutos durante el cual se aplican las actualizaciones programadas del motor y otros cambios del sistema. La mayoría de los eventos de mantenimiento se completan durante el periodo de 30 minutos, aunque, en ocasiones, otros eventos de mantenimiento pueden tardar más en completarse.

Cada clúster de base de datos tiene un período de mantenimiento semanal de 30 minutos. Si no especifica una hora preferida al crear el clúster de base de datos, Neptune elige al azar un día de la semana y, a continuación, asigna al azar un período de 30 minutos dentro de un bloque de 8 horas que varía según la región.

Por ejemplo, aquí se pueden ver los bloques de tiempo de 8 horas para los períodos de mantenimiento que se utilizan en varias regiones de AWS:

Region	Bloque de tiempo
Región del oeste de EE. UU. (Oregón)	06:00 — 14:00 UTC
Región del oeste de EE. UU. (Norte de California)	06:00 — 17:00 UTC
Región del este de EE. UU. (Ohio)	03:00 — 11:00 UTC

Region	Bloque de tiempo
Región de Europa (Irlanda)	22:00 — 06:00 UTC

El período de mantenimiento determina cuándo comienzan las operaciones pendientes. La mayoría de las operaciones de mantenimiento se completan dentro del período, pero las tareas de mantenimiento más grandes pueden continuar más allá de la hora de finalización del período.

Traslado del período de mantenimiento de base de datos

Lo ideal es que el período de mantenimiento coincida con el momento de menor uso del clúster. Si ese no es el caso de su período actual, puede cambiarlo a un momento mejor, de la siguiente manera:

Para cambiar la ventana de mantenimiento del clúster de base de datos

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Bases de datos.
3. Elija el clúster de base de datos cuyo periodo de mantenimiento desea cambiar.
4. Elija Modify.
5. Seleccione Mostrar más en la parte inferior de la página Modificar el clúster.
6. En la sección Período de mantenimiento preferido, defina el día, la hora y la duración del período de mantenimiento como prefiera.
7. Elija Siguiente.

   En la página de confirmación, revise los cambios.
8. Para aplicar los cambios al periodo de mantenimiento de forma inmediata, seleccione Apply immediately (Aplicar inmediatamente).
9. Seleccione Enviar para aplicar los cambios.

   Para editar los cambios elija Atrás o para cancelar sus cambios elija Cancelar.

Uso de **`AutoMinorVersionUpgrade`** para controlar las actualizaciones automáticas de versiones secundarias

> **Important**
>
> `AutoMinorVersionUpgrade` solo es efectivo para las actualizaciones de versiones secundarias superiores a la versión 1.3.0.0 del motor.

Si el campo `AutoMinorVersionUpgrade` está establecido en `true` en la instancia de escritura (principal) del clúster de base de datos, las actualizaciones de las versiones secundarias se aplicarán automáticamente a todas las instancias del clúster de base de datos durante el siguiente período de mantenimiento tras su publicación.

Si el campo `AutoMinorVersionUpgrade` está establecido en `false` en la instancia de escritor de su clúster de base de datos, solo se aplicarán si las instala de forma explícita.

> **Note**
>
> Las versiones de parche (`*.*.*.1`, `*.*.*.2`, etc.) siempre se instalan automáticamente durante el siguiente período de mantenimiento, independientemente de cómo esté configurado el parámetro `AutoMinorVersionUpgrade`.

Puede establecer `AutoMinorVersionUpgrade` utilizando la AWS Management Console del modo siguiente:

Para establecer **`AutoMinorVersionUpgrade`** mediante la consola de Neptune

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija la instancia principal (escritor) del clúster de base de datos para el que desea establecer `AutoMinorVersionUpgrade`.
4. Elija Modificar.
5. Seleccione Mostrar más en la parte inferior de la página Modificar el clúster.

6. En la parte inferior de la página ampliada, seleccione Activar la actualización automática de versiones secundarias o Desactivar la actualización automática de versiones secundarias.
7. Elija Siguiente.

   En la página de confirmación, revise los cambios.
8. Para aplicar los cambios a la actualización automática de la versión secundaria, seleccione Aplicar inmediatamente.
9. Seleccione Enviar para aplicar los cambios.

   Para editar los cambios elija Atrás o para cancelar sus cambios elija Cancelar.

También puede usar la AWS CLI para configurar el campo `AutoMinorVersionUpgrade`. Por ejemplo, para establecerlo en `true`, puede utilizar un comando como este:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --auto-minor-version-upgrade \
  --apply-immediately
```

Del mismo modo, para establecerlo en `false`, utilice un comando como este:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --no-auto-minor-version-upgrade \
  --apply-immediately
```

Instalación manual de las actualizaciones del motor de Neptune

Instalación de una actualización de versión principal del motor

Las versiones principales del motor siempre deben instalarse de forma manual. Para minimizar el tiempo de inactividad y disponer de tiempo suficiente para las pruebas y la validación, la mejor forma de instalar una nueva versión principal suele ser utilizar la solución de implementación azul/verde de Neptune.

En algunos casos, también puede utilizar la plantilla de AWS CloudFormation con la que creó el clúster de base de datos para instalar una actualización de la versión principal (consulte Uso de una AWS CloudFormation plantilla para actualizar la versión del motor de su clúster de base de datos Neptune).

Si desea instalar inmediatamente una actualización de la versión principal, puede utilizar un comando de la CLI como el siguiente:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (identifier for your neptune cluster) \
  --engine neptune \
  --engine-version (the new engine version) \
  --apply-immediately
```

Asegúrese de especificar la versión del motor a la que desea actualizar. Si no lo hace, puede que el motor se actualice a una versión que no sea la más reciente o la que espera.

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de especificarlo con este parámetro:

```
  --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de especificar este parámetro:

```
  ---db-instance-parameter-group-name (name of the custom instance parameter group)
```

Instalación de una actualización del motor de una versión menor mediante la AWS Management Console

Para realizar una actualización de versión menor con la consola de Neptune

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Bases de datos y, a continuación, elija el clúster de base de datos que desee modificar.
3. Elija Modificar.
4. En Especificaciones de la instancia, elija la nueva versión a la que desea actualizar.
5. Elija Siguiente.
6. Si desea aplicar los cambios inmediatamente, active la casilla Aplicar inmediatamente.
7. Seleccione Enviar para actualizar el clúster de base de datos.

Instalación de una actualización del motor de una versión menor mediante la AWS CLI

Puede utilizar un comando como el siguiente para realizar una actualización de una versión secundaria sin tener que esperar al siguiente período de mantenimiento:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your-neptune-cluster) \
  --engine-version (new-engine-version) \
  --apply-immediately
```

Si realiza las actualizaciones de forma manual mediante la AWS CLI, asegúrese de incluir la versión de motor a la que desea actualizarlo. Si no lo hace, puede que el motor se actualice a una versión que no sea la más reciente o la que espera.

Actualización del motor a la versión 1.2.0.0 o posterior desde una versión anterior a la 1.2.0.0

En la versión 1.2.0.0 del motor, se introdujeron varios cambios importantes que podrían complicar más de lo normal la actualización desde una versión anterior:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la depuración de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo

de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Uso de una AWS CloudFormation plantilla para actualizar la versión del motor de su clúster de base de datos Neptune

Puede volver a utilizar la plantilla AWS CloudFormation Neptune que utilizó para crear su clúster de base de datos Neptune para actualizar su versión de motor.

Las actualizaciones de la versión del motor de Neptune pueden ser secundarias o principales. El uso de una AWS CloudFormation plantilla puede ayudarle a realizar actualizaciones importantes de las versiones, que suelen contener cambios importantes. Dado que las actualizaciones principales de la versión pueden incluir cambios realizados en la base de datos que no son compatibles con las versiones anteriores de las aplicaciones, es posible que también necesite realizar cambios realizados en las aplicaciones durante la actualización. Siempre realice una prueba antes de realizar un actualización, y le recomendamos encarecidamente que siempre cree una instantánea manual del clúster de base de datos antes de realizar una actualización.

Tenga en cuenta que debe realizar una actualización del motor independiente para cada versión principal. No puede omitir una versión principal ni actualizarla directamente a la siguiente versión principal siguiente.

Antes del 17 de mayo de 2023, si utilizaba la AWS CloudFormation pila de Neptune para actualizar la versión de su motor, simplemente creaba un nuevo clúster de base de datos vacío en lugar del actual. Sin embargo, a partir del 17 de mayo de 2023, la AWS CloudFormation pila de Neptune ahora admite actualizaciones de motor locales que preservan los datos existentes.

Para una actualización de una versión principal, la plantilla debe establecer las siguientes propiedades en `DBCluster`:

- `DBClusterParameterGroup` (personalizado o predeterminado)
- `DBInstanceParameterGroupName`
- `EngineVersion`

Del mismo modo, para DBInstances asociadas a DBCluster, debe establecer:

- `DBParameterGroup` (personalizado/predeterminado)

Asegúrese de que todos los grupos de parámetros estén definidos en la plantilla, ya sean predeterminados o personalizados.

En el caso de un grupo personalizado de parámetros, asegúrese de que la familia del grupo personalizado de existente sea compatible con la nueva versión del motor. Las versiones del motor anteriores a la versión 1.2.0.0 utilizaban la familia de grupos de parámetros `neptune1`, mientras que las versiones del motor a partir de la versión 1.2.0.0 requieren la familia de grupos de parámetros `neptune1.2`. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

Para las principales actualizaciones de la versión del motor, especifique un grupo de parámetros con la familia adecuada en el campo `DBInstanceParameterGroupName` de `DBCluster`.

El grupo predeterminado de parámetros debe actualizarse a uno que sea compatible con la nueva versión del motor.

Tenga en cuenta que Neptune reinicia automáticamente las instancias de base de datos tras una actualización del motor.

Temas

Ejemplo: actualización secundaria del motor de la versión 1.2.0.1 a la versión 1.2.0.2

Busque el clúster de base de datos que desea actualizar y la plantilla que utilizó para crearlo. Por ejemplo:

```
Description: Base Template to create Neptune Stack with Engine Version 1.2.0.1 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
```

```
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.1
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Actualice la propiedad `EngineVersion` de `1.2.0.1` a `1.2.0.2`:

```
Description: Template to upgrade minor engine version to 1.2.0.2
```

```
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-cluster-parameter-group-description
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: test-cfn-neptune-db-parameter-group-description
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
```

```
      Ref: NeptuneDBCluster
```

Ahora se usa AWS CloudFormation para ejecutar la plantilla revisada.

Ejemplo: actualización principal de la versión 1.1.1.0 a la versión 1.2.0.2 con los grupos de parámetros predeterminados

Busque el `DBCluster` que desea actualizar y la plantilla que utilizó para crearlo. Por ejemplo:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Actualice el `DBClusterParameterGroup` predeterminado al de la familia de grupos de parámetros utilizada en la nueva versión del motor (en este caso, `default.neptune1.2`).
- En cada `DBInstance` asociada al `DBCluster`, actualice el `DBParameterGroup` predeterminado al de la familia utilizada en la nueva versión del motor (en este caso, `default.neptune1.2`).

- Establezca la propiedad `DBInstanceParameterGroupName` en el grupo predeterminado de parámetros de esa familia (en este caso, `default.neptune1.2`).
- Actualice la propiedad `EngineVersion` de `1.1.0.0` a `1.2.0.2`.

La plantilla debe tener el siguiente aspecto:

```
Description: Template to upgrade major engine version to 1.2.0.2 by using upgraded
 default parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName: default.neptune1.2
      DBInstanceParameterGroupName: default.neptune1.2
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
```

Ahora se usa AWS CloudFormation para ejecutar la plantilla revisada.

Ejemplo: actualización principal de la versión 1.1.1.0 a la versión 1.2.0.2 con los grupos de parámetros personalizados

Busque el `DBCluster` que desea actualizar y la plantilla que utilizó para crearlo. Por ejemplo:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpg
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpg
      Family: neptune1
      Description: 'NeptuneDBParameterGroup1 with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
```

```
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Actualice la `DBClusterParameterGroup` familia personalizada a la utilizada en la nueva versión del motor `default.neptune1.2` (aquí).
- Actualice la `DBCluster` `DBParameterGroup` familia personalizada de cada `DBInstance` una de ellas con la que se utilice en la nueva versión del motor (aquí`default.neptune1.2`).
- Establezca la propiedad `DBInstanceParameterGroupName` en el grupo de parámetros de esa familia (en este caso, `default.neptune1.2`).
- Actualice la propiedad `EngineVersion` de `1.1.0.0` a `1.2.0.2`.

La plantilla debe tener el siguiente aspecto:

```
Description: Template to upgrade major engine version to 1.2.0.2 by modifying existing
 custom parameter groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Name: engineupgradetestcpgnew
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Name: engineupgradetestpgnew
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
```

```
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  NeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Ahora usa AWS CloudFormation para ejecutar la plantilla revisada.

Ejemplo: actualización principal de la versión 1.1.1.0 a la versión 1.2.0.2 con una combinación de grupos de parámetros predeterminados y personalizados

Busque el `DBCluster` que desea actualizar y la plantilla que utilizó para crearlo. Por ejemplo:

```
Description: Base Template to create Neptune Stack with Engine Version 1.1.1.0 using
 custom Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
```

```
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBClusterParameterGroup with family neptune1'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1
      Description: 'NeptuneDBParameterGroup with family neptune1'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.1.1.0
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
```

```
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

- Para un grupo personalizado de parámetros de clúster, actualice la familia de `DBClusterParameterGroup` a la que corresponda de la nueva versión del motor, es decir, `neptune1.2`.
- Para un grupo predeterminado de parámetros de clúster, actualice el `DBClusterParameterGroup` a la opción predeterminada que corresponda de la nueva versión del motor, es decir, `default.neptune1.2`.
- Para cada `DBInstance` asociada a `DBCluster`, actualice un `DBParameterGroup` predeterminado al de la familia utilizada en la nueva versión del motor (en este caso, `default.neptune1.2`) y un grupo personalizado de parámetros a uno que utilice la familia compatible con la nueva versión del motor (en este caso, `neptune1.2`).
- Establezca la propiedad `DBInstanceParameterGroupName` en el grupo de parámetros de la familia compatible con la nueva versión del motor.

La plantilla debe tener el siguiente aspecto:

```
Description: Template to update Neptune Stack to Engine Version 1.2.0.1 using custom
 and default Parameter Groups
Parameters:
  DbInstanceType:
    Description: Neptune DB instance type
    Type: String
    Default: db.r5.large
Resources:
  NeptuneDBClusterParameterGroup:
    Type: 'AWS::Neptune::DBClusterParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBClusterParameterGroup with family neptune1.2'
      Parameters:
        neptune_enable_audit_log: 0
  NeptuneDBParameterGroup:
```

```
    Type: 'AWS::Neptune::DBParameterGroup'
    Properties:
      Family: neptune1.2
      Description: 'NeptuneDBParameterGroup1 with family neptune1.2'
      Parameters:
        neptune_query_timeout: 20000
  NeptuneDBCluster:
    Type: 'AWS::Neptune::DBCluster'
    Properties:
      EngineVersion: 1.2.0.2
      DBClusterParameterGroupName:
        Ref: NeptuneDBClusterParameterGroup
      DBInstanceParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBClusterParameterGroup
  CustomNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName:
        Ref: NeptuneDBParameterGroup
    DependsOn:
      - NeptuneDBCluster
      - NeptuneDBParameterGroup
  DefaultNeptuneDBInstance:
    Type: 'AWS::Neptune::DBInstance'
    Properties:
      DBClusterIdentifier:
        Ref: NeptuneDBCluster
      DBInstanceClass:
        Ref: DbInstanceType
      DBParameterGroupName: default.neptune1.2
    DependsOn:
      - NeptuneDBCluster
Outputs:
  DBClusterId:
    Description: Neptune Cluster Identifier
    Value:
      Ref: NeptuneDBCluster
```

Ahora se usa AWS CloudFormation para ejecutar la plantilla revisada.

Clonación de bases de datos en Neptune

Mediante la clonación de bases de datos, puede crear de una forma rápida y rentable clones de todas las bases de datos en Amazon Neptune. Las bases de datos clonadas requieren un espacio adicional mínimo cuando se crean inicialmente. La clonación de bases de datos utiliza un protocolo de copia en escritura. Los datos se copian en el momento en que cambian, tanto en las bases de datos de origen como en las clonadas. Puede crear varios clones partiendo del mismo clúster de base de datos. También puede crear clones adicionales desde otros clones. Para obtener más información acerca del funcionamiento del protocolo de copia en escritura en el contexto del almacenamiento de Neptune, consulte Protocolo de copia en escritura.

Puede usar la clonación de bases de datos en diversos casos de uso, especialmente cuando no desee que su entorno de producción se vea afectado, como por ejemplo en las siguientes situaciones:

- Experimentar con el impacto de los cambios, como por ejemplo los cambios de esquema o los cambios de grupos de parámetros, y valorar dicho impacto.
- Realizar operaciones intensivas, como exportar datos o ejecutar consultas analíticas.
- Crear una copia de un clúster de base de datos de producción en un entorno que no sea de producción para el desarrollo o las pruebas.

Para crear un clon de un clúster de base de datos con la AWS Management Console

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Instances (Instancias). Elija la instancia principal del clúster de base de datos del que desea crear un clon.
3. Elija Instance actions (Acciones de instancias) y, a continuación, elija Create clone (Crear clon).
4. En la página Create Clone (Crear clon), escriba un nombre para la instancia principal del clúster de base de datos clonado como DB instance identifier (Identificador de instancias de bases de datos).

   Si lo desea, configure los demás ajustes del clúster de base de datos clonado. Para obtener información acerca de los distintos ajustes de clúster de base de datos, consulte Lanzamiento mediante la consola.
5. Elija Create Clone (Crear clon) para lanzar el clúster de base de datos clonado.

Para crear un clon de un clúster de base de datos con la AWS CLI

- Llame al comando de la AWS CLI restore-db-cluster-to-point-in-time de Neptune y proporcione los siguientes valores:
  - `--source-db-cluster-identifier`: el nombre del clúster de base de datos origen del que desea crear un clon.
  - `--db-cluster-identifier`: el nombre del clúster de base de datos clonado.
  - `--restore-type copy-on-write`: el valor `copy-on-write` indica que se debe crear un clúster de base de datos clonado.
  - `--use-latest-restorable-time`: indica que se utiliza la hora de la última copia de seguridad restaurable.

> **Note**
>
> El comando restore-db-cluster-to-point-in-time de la AWS CLI solo clona el clúster de la base de datos, no las instancias de base de datos dicho clúster.

El ejemplo siguiente de Linux/UNIX crea un clon a partir del clúster de base de datos `source-db-cluster-id` y le asigna un nombre al clon `db-clone-cluster-id`.

```
aws neptune restore-db-cluster-to-point-in-time \
  --region us-east-1 \
  --source-db-cluster-identifier source-db-cluster-id \
  --db-cluster-identifier db-clone-cluster-id \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

El mismo ejemplo funciona en Windows si el carácter de escape del extremo de la línea \ se sustituye por el ^ equivalente de Windows:

```
aws neptune restore-db-cluster-to-point-in-time ^
  --region us-east-1 ^
  --source-db-cluster-identifier source-db-cluster-id ^
  --db-cluster-identifier db-clone-cluster-id ^
  --restore-type copy-on-write ^
  --use-latest-restorable-time
```

Limitaciones

La clonación de bases de datos en Neptune tiene las siguientes limitaciones:

- No puede crear bases de datos clonadas entre regiones de AWS diferentes. Las bases de datos clonadas se deben crear en la misma región que las bases de datos de origen.
- Una base de datos clonada siempre utiliza el parche más reciente de la versión del motor de Neptune que utiliza la base de datos a partir del que se ha clonado. Esto se aplica incluso si la base de datos de origen aún no se ha actualizado a esa versión de parche. Sin embargo, la versión del motor no cambia.
- Por el momento, existe un límite de no más de 15 clones por copia del clúster de base de datos de Neptune, incluidos los clones basados en otros clones. Una vez alcanzado ese límite, debe hacer otra copia de la base de datos en lugar de clonarla. No obstante, si ace una copia nueva, puede tener también hasta 15 clones.
- La clonación de bases de datos entre varias cuentas no se admite actualmente.
- Puede proporcionar una Virtual Private Cloud (VPC) diferente para su clon. Sin embargo, las subredes de esas VPC deben estar asignadas al mismo conjunto de zonas de disponibilidad.

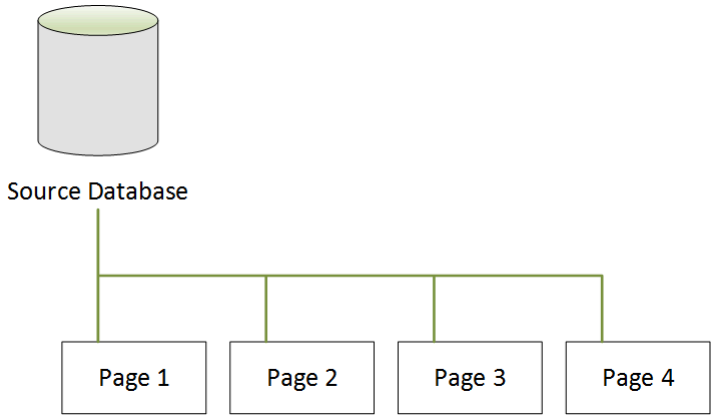
Protocolo de copia en escritura para la clonación de bases de datos

Las siguientes situaciones ilustran el funcionamiento del protocolo de copia en escritura.

- Base de datos de Neptune antes de la clonación
- Base de datos de Neptune después de la clonación
- Cuando se efectúa un cambio en la base de datos de origen
- Cuando se realiza un cambio en la base de datos clonada

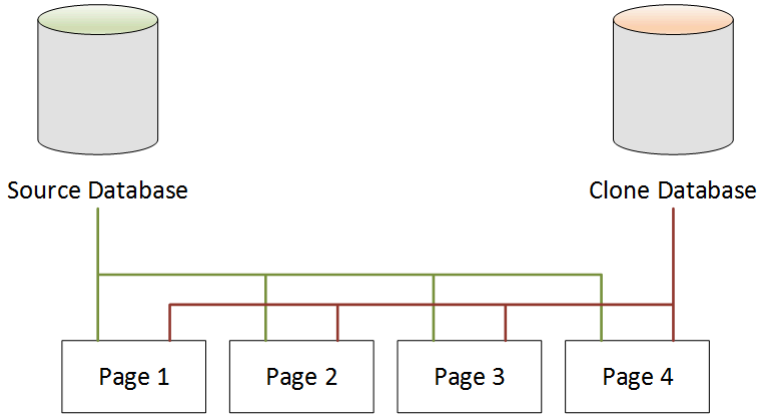
Base de datos de Neptune antes de la clonación

En una base de datos de origen, los datos se almacenan en páginas. En el diagrama siguiente, la base de datos de origen tiene cuatro páginas.



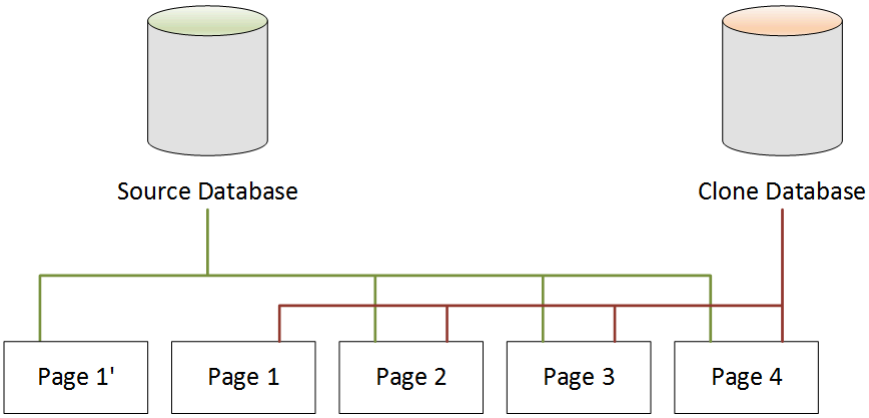
Base de datos de Neptune después de la clonación

Como se muestra en el diagrama siguiente, no se producen cambios en la base de datos de origen después de la clonación. Tanto la base de datos de origen como la base de datos clonada apuntan a las mismas cuatro páginas. Ninguna página se ha copiado físicamente, por lo que no se necesita almacenamiento adicional.



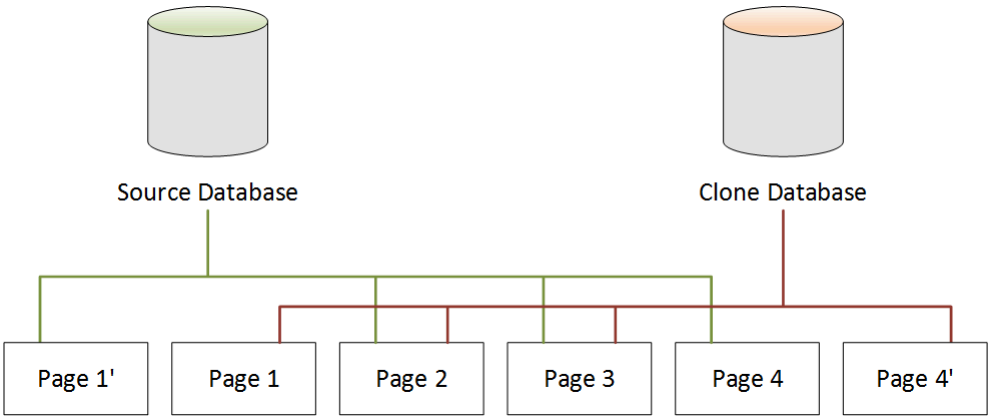
Cuando se efectúa un cambio en la base de datos de origen

En el siguiente ejemplo, la base de datos de origen realiza un cambio en los datos de la `Page 1`. En lugar de escribir en la `Page 1` original, usa almacenamiento adicional para crear una nueva página denominada `Page 1'`. Ahora, la base de datos de origen apunta a la nueva `Page 1'` y también a la `Page 2`, la `Page 3` y la `Page 4`. La base de datos clonada sigue apuntando a la `Page 1` a través de la `Page 4`.



Cuando se realiza un cambio en la base de datos clonada

En el siguiente diagrama, la base de datos clonada también ha cambiado, esta vez en la `Page 4`. En lugar de escribir en la `Page 4` original, se usa almacenamiento adicional para crear una nueva página denominada `Page 4'`. La base de datos de origen sigue apuntando a la `Page 1'` y también a la `Page 2` a través de la `Page 4`, pero ahora la base de datos clonada apunta a la `Page 1` a través de la `Page 3` y también de la `Page 4'`.



Como se muestra en la segunda situación, después de la clonación de la base de datos no se requiere almacenamiento adicional en el momento de la creación del clon. Sin embargo, a medida que se producen cambios en la base de datos de origen y en la base de datos clonada, solo se crean las páginas modificadas, como se muestra en las situaciones tercera y cuarta. A medida que se producen más cambios a lo largo del tiempo tanto en la base de datos de origen como en la base de datos clonada, irá necesitando más almacenamiento para capturar y almacenar los cambios.

Eliminación de una base de datos de origen

La eliminación de una base de datos de origen no afecta a las bases de datos clonadas asociadas a ella. Las bases de datos clonadas siguen apuntando a las páginas que antes pertenecían a la base de datos de origen.

Administración de instancias de Amazon Neptune

Las siguientes secciones contienen información sobre las operaciones de nivel de instancia.

Temas

Clase de instancia ampliable de Neptune T3

Además de las clases de instancias de rendimiento fijo como R5 y R6, Amazon Neptune le da la opción de usar una instancia T3 de rendimiento ampliable. Mientras desarrolla la aplicación de gráficos, desea que la base de datos sea rápida y que tenga capacidad de respuesta, pero no necesita usarla todo el tiempo. La clase de instancia db.t3.medium de Neptune es justo lo que debería usar en esa situación, a un costo bastante más bajo que la clase de instancia de rendimiento fijo menos costosa.

Las instancias de ráfaga se ejecutan en un nivel de referencia de rendimiento de la CPU hasta que una carga de trabajo necesita más, y entonces rompen la referencia durante tanto tiempo como sea necesario para la carga de trabajo. Su precio por hora cubre las ráfagas, siempre que la utilización media de la CPU no supere la base de referencia durante un periodo de 24 horas. En la mayoría de las situaciones de desarrollo y prueba, eso se traduce en un buen rendimiento a un bajo coste.

Si empieza con una clase de instancia T3, más tarde podrá cambiar con facilidad a una clase de instancia de rendimiento fijo cuando esté preparado para entrar en producción mediante la AWS Management Console, la AWS CLI o uno de los SDK de AWS.

Las ráfagas T3 se rigen por créditos de CPU

Un crédito de CPU representa el uso total de un núcleo de CPU virtual (vCPU) durante un minuto. Esto también puede traducirse como el 50 % del uso de una vCPU durante dos minutos, o un 25 % del uso de dos vCPU durante dos minutos, y así sucesivamente.

Una instancia T3 acumula créditos de CPU cuando está inactiva y los usa cuando está activa, y lo mide con una resolución de milisegundos. La clase de instancia db.t3.medium tiene dos vCPU, cada una de las cuales gana 12 créditos de CPU por hora cuando está inactiva. Esto significa que el 20 % del uso de cada vCPU deriva en un balance de crédito de CPU de cero. Los 12 créditos de CPU obtenidos se emplean en un 20 % del uso de la CPU (ya que el 20 % de 60 minutos también es 12). Este uso del 20 % es, por lo tanto, la tasa de uso de referencia que no produce un balance positivo ni negativo de créditos de la CPU.

El tiempo de inactividad (el uso de la CPU inferior al 20 % del total disponible) hace que los créditos de la CPU se almacenen en un depósito de balance de crédito, hasta un límite de 576 (el número máximo de créditos de la CPU que pueden acumularse en 24 horas, es decir, 2 x 12 x 24) para una clase de instancia db.t3.medium. Por encima de ese límite, los créditos de CPU simplemente se descartan.

El uso de la CPU puede aumentar hasta un 100 % cuando sea necesario y durante el tiempo requerido para una carga de trabajo, incluso después de que el balance de créditos de la CPU caiga por debajo de cero. Si la instancia mantiene un balance negativo continuo durante 24 horas, incurre en un coste adicional de 0,05 USD por cada -60 créditos de la CPU acumulados durante ese periodo. Sin embargo, en la mayoría de las cargas de trabajo de desarrollo y prueba, la fragmentación suele estar cubierta por el tiempo de inactividad antes o después de la ráfaga.

> **Note**
>
> La clase de instancia T3 de Neptune está configurada como elmodo ilimitado de Amazon EC2.

Usar la AWS Management Console para crear una instancia de ráfagas T3

En la AWS Management Console, puede crear una instancia de clúster de base de datos principal o una instancia de réplica de lectura que utilice la clase de instancia `db.t3.medium`, o puede modificar una instancia existente para utilizar la clase de instancia `db.t3.medium`.

Por ejemplo, para crear una nueva instancia principal del clúster de base de datos en la consola de Neptune:

- Elija Create Database (Crear base de datos).
- Elija una versión del motor de la base de datos igual o posterior a `1.0.2.2`.
- En Purpose (Finalidad), elija Development and Testing (Desarrollo y pruebas).
- Como DB Instance class (Clase de instancia de base de datos), acepte el valor predeterminado: `db.t3.medium — 2 vCPU, 4 GiB RAM`.

Usar la AWS CLI para crear una instancia de ráfagas T3

También puede usar la AWS CLI para hacer lo mismo:

```
aws neptune create-db-cluster \
    --db-cluster-identifier (name for a new DB cluster) \
    --engine neptune \
    --engine-version "1.0.2.2"

aws neptune create-db-instance \
```

```
    --db-cluster-identifier (name of the new DB cluster) \
    --db-instance-identifier (name for the primary writer instance in the cluster) \
    --engine neptune \
    --db-instance-class db.t3.medium
```

Modificación de una instancia de base de datos de Neptune (y aplicación inmediata)

Puede aplicar la mayoría de cambios a una instancia de base de datos de Amazon Neptune de forma inmediata o retrasarlos hasta el próximo periodo de mantenimiento. Algunas modificaciones, como los cambios de grupo de parámetros, requieren que reinicie manualmente la instancia de base de datos para que el cambio surta efecto.

> **Important**
>
> Las modificaciones provocan una interrupción, ya que Neptune debe reiniciar la instancia de base de datos para que el cambio se aplique. Antes de modificar la configuración de una instancia de base de datos, evalúe los efectos que puede tener en la base de datos y en las aplicaciones.

Configuración común e implicaciones del tiempo de inactividad

La siguiente tabla contiene detalles sobre las opciones que se pueden cambiar, cuándo se pueden aplicar los cambios y si los cambios provocan tiempo de inactividad en la instancia de base de datos.

Configuración de la instancia de base de datos	Notas acerca del tiempo de inactividad	
DB instance class (Clase de instancia de base de datos)	Se produce una interrupc ión durante este cambio, ya sea que se aplique inmediata mente o en el siguiente periodo de mantenimiento.	
DB Instance Identifier (Identifi cador de instancias de bases de datos)	La instancia de base de datos se reinicia y se produce una interrupción durante este cambio, ya sea que se aplique inmediatamente o en el siguiente periodo de mantenimiento.	

Configuración de la instancia de base de datos	Notas acerca del tiempo de inactividad	
Subnet group (Grupo de subredes)	La instancia de base de datos se reinicia y se produce una interrupción durante este cambio, ya sea que se aplique inmediatamente o en el siguiente periodo de mantenimiento.	
Security group (Grupo de seguridad)	El cambio se aplica de forma asíncrona lo antes posible, independientemente de cuándo especifique que se deben realizar los cambios, y no se produce ninguna interrupción.	–
Certificate Authority (Entidad de certificación)	De forma predeterminada, la instancia de base de datos se reinicia cuando se asigna una nueva entidad de certificación.	
Puerto de base de datos	El cambio siempre se produce de forma inmediata, lo que provoca que la instancia de base de datos se reinicie y se produzca una interrupción.	

Configuración de la instancia de base de datos	Notas acerca del tiempo de inactividad	
Grupo de parámetros de base de datos	Cambiar este ajuste no produce una interrupción. El nombre del grupo de parámetros se modifica inmediatamente, pero los cambios del parámetro no se aplican hasta que se reinicia la instancia sin conmutación por error. En este caso, la instancia de base de datos no se reiniciará automátic amente y los cambios en los parámetros no se aplicarán en el siguiente periodo de mantenimiento. Sin embargo, si modifica parámetros dinámicos en el grupo de parámetros de base de datos recién asociado, dichos cambios se aplican inmediata mente sin reiniciar. Para obtener más informaci ón, consulte Reinicio de una instancia de base de datos en Amazon Neptune.	
Grupo de parámetros de clúster de base de datos	El nombre del grupo de parámetros de base de datos se cambia inmediatamente.	

Configuración de la instancia de base de datos	Notas acerca del tiempo de inactividad	
Backup retention period (Periodo de retención de copia de seguridad)	Si especifica que los cambios deben producirse inmediata mente, este cambio se produce de forma inmediata . De lo contrario, si cambia la configuración de un valor distinto de cero a otro valor distinto de cero, el cambio se aplica de forma asíncrona, tan pronto como sea posible. Cualquier otro cambio se produce en el siguiente periodo de mantenimiento. Se produce una interrupción si se cambia el valor de cero a un valor distinto de cero o de un valor distinto de cero a cero.	
Registro de auditoría	Seleccione Registro de auditoría si desea utilizar el registro de auditoría a través de los registros de CloudWatc h. También debe establece r el parámetro `neptune_e nable_audit_log` del grupo de parámetros del clúster de base de datos en `enable` (1) para habilitar el registro de auditoría.	

Configuración de la instancia de base de datos	Notas acerca del tiempo de inactividad	
Actualización automática de versiones secundarias	Seleccione Actualización automática a versiones secundarias si desea habilitar el clúster de base de datos de Neptune para recibir actualizaciones de las versiones secundarias del motor de base de datos de Neptune automáticamente cuando estén disponibles. La opción Actualización automática a versiones secundarias solo es válida para las actualizaciones secundarias de las versiones del motor para el clúster de base de datos de Amazon Neptune. No tiene validez para los parches periódicos que se utilizan para mantener la estabilidad del sistema.	

Cambio del nombre de una instancia de base de datos de Neptune

Puede cambiar el nombre de una instancia de base de datos de Amazon Neptune mediante la AWS Management Console. Cambiar el nombre de una instancia de base de datos puede tener grandes repercusiones. A continuación, podrá encontrar una lista de consecuencias que debería conocer antes de hacerlo.

- Cuando cambia el nombre de una instancia de base de datos, se modifica su punto de enlace, porque la URL incluye el nombre asignado a dicha instancia. Deberá redirigir siempre el tráfico desde la URL antigua a la nueva.
- Si cambia el nombre de una instancia de base de datos, el nombre DNS anterior que utilizaba la instancia se elimina de inmediato, pero puede quedar almacenado en la caché durante varios minutos. El nuevo nombre de DNS de la instancia de base de datos se hace efectivo después de, aproximadamente, 10 minutos. La instancia de base de datos en cuestión no estará disponible hasta que se haga efectivo el nombre nuevo.
- Si cambia el nombre de una instancia, no puede usar el nombre de una instancia de base de datos existente.
- Todas las réplicas de lectura asociadas a una instancia de base de datos quedan asociadas a esa instancia después de cambiarle el nombre. Por ejemplo, suponga que tiene una instancia de base de datos que atiende a su base de datos de producción y que esa instancia tiene varias réplicas de lectura asociadas. Si cambia el nombre de la instancia de base de datos y, luego, lo reemplaza en el entorno de producción con una instantánea de base de datos, la instancia de base de datos cuyo nombre cambió sigue teniendo asociadas esas réplicas de lectura.
- Las métricas y los eventos asociados al nombre de una instancia de base de datos se mantienen si se reutiliza dicho nombre. Por ejemplo, si promociona una réplica de lectura y le asigna el nombre de la instancia de base de datos principal anterior, los eventos y las métricas asociados a esta instancia se asocian a la instancia con el nuevo nombre.
- Las etiquetas de la instancia de base de datos permanecen con dicha instancia, independientemente del cambio de nombre.
- Las instantáneas de base de datos se conservan para una instancia de base de datos a la que se le haya cambiado el nombre.

Para cambiar el nombre de una instancia de base de datos con la consola de Neptune

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.

2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione el botón de opción situado junto a la instancia de base de datos cuyo nombre desea cambiar.
4. En el menú Instance actions (Acciones de instancias), elija Modify (Modificar).
5. Escriba un nombre nuevo en el cuadro de texto DB Instance Identifier (Identificador de instancias de bases de datos). Seleccione Apply immediately (Aplicar inmediatamente) y, a continuación, elija Continue (Continuar).
6. Elija Modify DB instance (Modificar instancia de base de datos) para completar el cambio.

Reinicio de una instancia de base de datos en Amazon Neptune

En algunos casos, si modifica una instancia de base de datos de Amazon Neptune, cambia el grupo de parámetros de base de datos asociado a la instancia o cambia un parámetro estático de base de datos de un grupo de parámetros utilizado por la instancia, debe reiniciar la instancia para aplicar los cambios.

Cuando se reinicia una instancia de base de datos, se reinicia el servicio del motor de base de datos. El reinicio también aplica a la instancia de base de datos cualquier cambio realizado en el grupo de parámetros de base de datos asociado que estuviera pendiente. Al reiniciar una instancia de base de datos, se produce una interrupción momentánea de la instancia, durante la cual su estado se establece en rebooting. Si la instancia de Neptune está configurada para Multi-AZ, el reinicio puede realizarse mediante una conmutación por error. Cuando finaliza el reinicio, se crea un evento de Neptune.

Si la instancia de base de datos es una implementación Multi-AZ, es posible forzar una conmutación por error desde una zona de disponibilidad a otra cuando se elige la opción Reboot (Reiniciar). Cuando se fuerza una conmutación por error de la instancia de base de datos, Neptune cambia automáticamente a una réplica en espera de otra zona de disponibilidad. A continuación, actualiza el registro DNS de la instancia para que apunte a la instancia de base de datos en espera. Como consecuencia, debe eliminar y restablecer las conexiones existentes a la instancia de base de datos.

Reboot with failover (Reinicio con conmutación por error) resulta beneficioso cuando se desea simular un error en una instancia de base de datos para realizar pruebas o restaurar operaciones en la zona de disponibilidad original después de que se produzca una conmutación por error. Para obtener más información, consulte Alta disponibilidad (Multi-AZ) en la Guía del usuario de Amazon RDS. Al reiniciar un clúster de base de datos, este conmuta por error a la réplica en espera. El reinicio de una réplica de Neptune no inicia una conmutación por error.

El tiempo necesario para reiniciar es una función del proceso de recuperación tras bloqueo. Para mejorar el tiempo de reinicio, recomendamos reducir las actividades de la base de datos tanto como sea posible durante el proceso de reinicio para reducir la actividad de restauración para las transacciones en tránsito.

En la consola, la opción Reboot (Reiniciar) puede estar deshabilitada si la instancia de base de datos no se encuentra en el estado Available (Disponible). Esto puede deberse a varias razones, como un backup en curso, una modificación solicitada por el cliente o una acción durante un periodo de mantenimiento.

> **Note**
>
> Antes de Versión: 1.2.0.0 (21/07/2022), todas las réplicas de lectura de un clúster de base de datos se reiniciaban automáticamente cuando se reiniciaba la instancia principal (escritor). A partir de Versión: 1.2.0.0 (21/07/2022), el reinicio de la instancia principal no provoca el reinicio de ninguna de las réplicas. Esto significa que si va a cambiar un parámetro de clúster, debe reiniciar cada instancia por separado para detectar el cambio de parámetro (consulte Grupos de parámetros).

Para reiniciar una instancia de base de datos con la consola de Neptune

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija la instancia de base de datos que desea reiniciar.
4. Elija Instance actions y, a continuación, Reboot.
5. Para forzar una conmutación por error de una zona de disponibilidad a otra, seleccione Reboot with failover? (¿Reiniciar con conmutación por error?) en el cuadro de diálogo Reboot DB Instance (Reiniciar instancia de base de datos).
6. Elija Reboot. Para cancelar el reinicio, elija Cancel (Cancelar) en su lugar.

Eliminación de una instancia de base de datos en Amazon Neptune

Puede eliminar una instancia de base de datos de Amazon Neptune en cualquier estado y en cualquier momento, siempre y cuando la instancia se haya iniciado y la protección contra eliminación esté deshabilitada en la instancia.

No se puede eliminar una instancia de base de datos si la protección de eliminación está habilitada

Solo puede eliminar instancias de bases de datos que tengan deshabilitada la protección contra eliminación. Neptune fuerza la protección contra eliminación al utilizar la consola, la AWS CLI o las API para eliminar una instancia de base de datos.

La protección contra eliminación se habilita de forma predeterminada cuando crea una instancia de base de datos de producción con la AWS Management Console.

La protección contra eliminación está deshabilitada de forma predeterminada si usa comandos de la API o la AWS CLI para crear una instancia de base de datos.

Para eliminar una instancia de base de datos que tenga la protección contra eliminación habilitada, modifique primero la instancia para establecer el campo `DeletionProtection` en `false`.

Habilitar o deshabilitar la protección contra eliminación no provoca una interrupción.

Realización de una instantánea final de la instancia de base de datos antes de eliminarla

Para eliminar una instancia de base de datos, debe especificar el nombre de la instancia e indicar si desea crear una instantánea de base de datos final. Si la instancia de base de datos que está eliminando tiene el estado Creating (Creándose), no podrá tomar una instantánea de base de datos final. Si la instancia de base de datos está en un estado de error y tiene el estado failed, incompatible-restore o incompatible-network, solo puede eliminar la instancia cuando el parámetro `SkipFinalSnapshot` esté establecido en `true`.

Si elimina todas las instancias de bases de datos de Neptune en un clúster de base de datos con la AWS Management Console, todo el clúster de base de datos se eliminará de forma automática. Si utiliza la AWS CLI o el SDK, debe eliminar el clúster de base de datos de forma manual después de eliminar la última instancia.

> **Important**
>
> Si elimina todo un clúster de base de datos, todas sus copias de seguridad automatizadas se eliminarán al mismo tiempo y no se podrán recuperar. Esto significa que, a no ser que decida crear una instantánea de base de datos final, no podrá restaurar posteriormente la instancia de base de datos a su estado final. Las instantáneas manuales de una instancia no se eliminan cuando se elimina el clúster.

Si la instancia de base de datos que desea eliminar tiene una réplica de lectura, debe promocionar la réplica de lectura o eliminarla.

En los siguientes ejemplos, se elimina una instancia de base de datos creando y sin crear una instantánea de base de datos final.

Eliminación de una instancia de base de datos sin crear una instantánea final

Puede omitir la creación de una instantánea de base de datos final si desea eliminar rápidamente una instancia de base de datos. Al eliminar una instancia de base de datos, se eliminan todas las copias de seguridad automatizadas y no se pueden recuperar. Las instantáneas manuales no se eliminan.

Para eliminar una instancia de base de datos sin una instantánea de base de datos final con la consola de Neptune

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. En la lista Instances (Instancias), seleccione el botón de opción situado junto a la instancia de base de datos que desea eliminar.
4. Elija Instance actions y, a continuación, Delete.
5. Elija No en el cuadro de lista desplegable Create final Snapshot? (¿Crear instantánea final?).
6. Elija Eliminar.

Eliminación de una instancia de base de datos creando una instantánea de base de datos final

Si desea poder restaurar más adelante una instancia de base de datos eliminada, puede crear una instantánea de base de datos final. Todas las copias de seguridad automatizadas se eliminan también y no se pueden recuperar. Las instantáneas manuales no se eliminan.

Para eliminar una instancia de base de datos con una instantánea de base de datos final con la consola de Neptune

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. En la lista Instances (Instancias), seleccione el botón de opción situado junto a la instancia de base de datos que desea eliminar.
4. Elija Instance actions y, a continuación, Delete.
5. Elija Yes (Sí) en el cuadro Create final Snapshot? (¿Crear instantánea final?).
6. En el cuadro Final Snapshot name (Nombre de instantánea final), escriba el nombre de la instantánea de base de datos final.
7. Elija Eliminar.

Puede comprobar el estado de una instancia, determinar qué tipo de instancia es, averiguar qué versión del motor tiene instalada actualmente y obtener otra información sobre una instancia utilizando la API instance-status.

Amazon Neptune sin servidor

Amazon Neptune sin servidor es una configuración de escalado automático bajo demanda que se ha diseñado para escalar el clúster de base de datos según sea necesario con el fin de hacer frente incluso a los grandes aumentos de la demanda de procesamiento y, luego, volver a reducir verticalmente cuando la demanda disminuya. Ayuda a automatizar los procesos de monitorización de la carga de trabajo y ajuste de la capacidad de la base de datos de Neptune. Dado que la capacidad se ajusta de forma automática en función de la demanda de la aplicación, solo se le cobrará por los recursos que la aplicación realmente necesite.

Casos de uso de Neptune sin servidor

Neptune sin servidor admite muchos tipos de cargas de trabajo. Es adecuado para cargas de trabajo exigentes y muy variables, y puede ser de gran ayuda si el uso de la base de datos suele ser intensivo durante cortos periodos de tiempo, seguidos de largos periodos de poca actividad o ninguna actividad en absoluto. Neptune sin servidor es especialmente útil para los siguientes casos de uso:

- Cargas de trabajo variables: las cargas de trabajo que tienen un aumento repentino e impredecible de la actividad de la CPU. Con Neptune sin servidor, la base de datos de gráficos escala la capacidad automáticamente para hacer frente a las necesidades de la carga de trabajo y vuelve a reducirla verticalmente cuando termina el aumento de actividad. Ya no tiene que aprovisionar la capacidad para los picos ni para la carga promedio. Puede especificar un límite de capacidad superior para hacer frente a los picos de cargas de trabajo. Esta capacidad no se utiliza a menos que fuera necesario.

  La granularidad de la reducción horizontal de Neptune Serverless le ayuda a adaptar la capacidad a las necesidades de su carga de trabajo. Neptune sin servidor puede añadir o eliminar capacidad en incrementos de registro detallado en función de lo que se necesite. Puede añadir tan solo media unidad de capacidad de Neptune (NCU) cuando solo se necesite un poco más de capacidad.
- Aplicaciones de varios inquilinos: al sacar partido de Neptune sin servidor, puede crear un clúster de base de datos independiente para cada una de las aplicaciones que necesite ejecutar sin tener que administrar cada clúster de forma individual. Cada uno de los clústeres de inquilino puede tener diferentes periodos de actividad e inactividad en función de varios factores, pero Neptune sin servidor puede escalarlos de forma eficaz sin su intervención.

- Aplicaciones nuevas: al implementar una nueva aplicación, a menudo no está seguro de cuánta capacidad de base de datos necesitará. Con Neptune sin servidor, puede configurar un clúster de base de datos que se pueda escalar automáticamente para cumplir con los requisitos de capacidad de la nueva aplicación a medida que se va desarrollando.
- Planificación de capacidad: supongamos que normalmente ajusta la capacidad de la base de datos o verifica la capacidad de la base de datos óptima para la carga de trabajo modificando las clases de instancias de base de datos de todas las instancias de base de datos de un clúster. Con Neptune sin servidor, puede evitar esta sobrecarga administrativa. En su lugar, puede modificar las instancias de base de datos existentes de aprovisionadas a sin servidor o de sin servidor a aprovisionadas sin tener que crear un nuevo clúster o instancia de base de datos.
- Desarrollo y pruebas: Neptune sin servidor también es perfecto para entornos de desarrollo y pruebas. Con Neptune sin servidor, puede crear instancias de base de datos con una capacidad máxima lo suficientemente alta como para probar la aplicación más exigente y una capacidad mínima baja para todas las demás ocasiones en las que el sistema pueda estar inactivo entre pruebas.

Neptune sin servidor solo escala la capacidad de computación. El volumen de almacenamiento sigue siendo el mismo y no se ve afectado por el escalado sin servidor.

> **Note**
>
> También puede usar el escalado automático de Neptune con Neptune sin servidor para gestionar diferentes tipos de variaciones de carga de trabajo.

Restricciones de Amazon Neptune sin servidor

- No está disponible en todas las regiones: Neptune sin servidor solo está disponible en las siguientes regiones:
  - Este de EE. UU. (Norte de Virginia): `us-east-1`
  - Este de EE. UU. (Ohio): `us-east-2`
  - Oeste de EE. UU. (Norte de California): `us-west-1`
  - Oeste de EE. UU. (Oregón): `us-west-2`
  - Canadá (centro): `ca-central-1`
  - Europa (Estocolmo): `eu-north-1`

- Europa (Irlanda): `eu-west-1`
- Europa (Londres): `eu-west-2`
- Europa (Fráncfort): `eu-central-1`
- Asia-Pacífico (Tokio): `ap-northeast-1`
- Asia-Pacífico (Singapur): `ap-southeast-1`
- Asia-Pacífico (Sídney): `ap-southeast-2`

- No disponible en las primeras versiones del motor: Neptune sin servidor solo está disponible en las versiones del motor 1.2.0.1 o posteriores.
- No es compatible con la caché de búsqueda de Neptune: la caché de búsqueda no funciona con instancias de base de datos sin servidor.
- La memoria máxima de una instancia sin servidor es de 256 GB: si se establece `MaxCapacity` en 128 NCU (la configuración más alta admitida), una instancia de Neptune sin servidor se puede escalar hasta una capacidad de 256 GB de memoria, lo que equivale a la de un tipo de instancia aprovisionada `R6g.8XL`.

Escalado de capacidad en un clúster de base de datos de Neptune sin servidor

La configuración de un clúster de base de datos de Neptune sin servidor es similar a la configuración de un clúster normal aprovisionado, con una configuración adicional para las unidades mínimas y máximas de escalado y con el tipo de instancia establecido en `db.serverless`. La configuración de escalado se define en las unidades de capacidad de Neptune (NCU), cada una de las cuales consta de 2 GiB (gibibytes) de memoria (RAM), además de las redes y la capacidad de procesador virtual (vCPU) asociadas. Se configura como parte de un objeto `ServerlessV2ScalingConfiguration` y se representa en JSON, tal y como se muestra a continuación:

```
"ServerlessV2ScalingConfiguration": {
  "MinCapacity": (minimum NCUs, a floating-point number such as   1.0),
  "MaxCapacity": (maximum NCUs, a floating-point number such as 128.0)
}
```

En cualquier momento, cada instancia de escritor o lector de Neptune tiene una capacidad medida por un número en coma flotante que representa el número de NCU que utiliza actualmente dicha instancia. Puede utilizar la métrica de CloudWatch ServerlessDatabasecapacidad a nivel de instancia

para saber cuántas NCUs utiliza actualmente una instancia de base de datos determinada y la métrica de utilización de la NCUUtilización para averiguar qué porcentaje de su capacidad máxima utiliza la instancia. Ambas métricas también están disponibles en el nivel de clúster de base de datos para mostrar la utilización promedio de los recursos del clúster de base de datos en su conjunto.

Al crear un clúster de base de datos de Neptune sin servidor, establece el número mínimo y máximo de unidades de capacidad de Neptune (NCU) para todas las instancias sin servidor.

El valor mínimo de NCU especificado determina el menor tamaño al que puede reducirse una instancia sin servidor del clúster de base de datos y, del mismo modo, el valor máximo de NCU determina el mayor tamaño al que puede aumentar una instancia sin servidor. El valor máximo de NCU más alto que puede establecer es 128,0 NCU y el mínimo más bajo es 1,0 NCU.

Neptune realiza un seguimiento continuo de la carga en cada instancia de Neptune sin servidor mediante la monitorización de la utilización de recursos como la CPU, la memoria y la red. La carga la generan las operaciones de la base de datos de la aplicación, el procesamiento en segundo plano del servidor y otras tareas administrativas.

Cuando la carga de una instancia sin servidor alcanza el límite de la capacidad actual o cuando Neptune detecta cualquier otro problema de rendimiento, la instancia se escala verticalmente de forma automática. Cuando la carga de la instancia disminuye, la capacidad se reduce verticalmente hasta alcanzar las unidades de capacidad mínimas configuradas, y la capacidad de la CPU se libera antes que la memoria. Esta arquitectura permite liberar recursos de forma gradual y controlada y gestiona las fluctuaciones de la demanda de forma eficaz.

Puede hacer que una instancia de lector se escale junto con la instancia de escritor o que se escale de forma independiente mediante la configuración de su nivel de promoción. Las instancias de lector en los niveles de promoción 0 y 1 se escalan al mismo tiempo que las de escritor, lo que permite que tengan el tamaño adecuado para asumir de forma rápida la carga de trabajo del escritor en caso de que se produzca una conmutación por error. Los lectores de los niveles de promoción 2 a 15 se escalan independientemente de la instancia de escritor y por separado.

Si ha creado un clúster de base de datos de Neptune como un clúster de varias zonas de disponibilidad (AZ) para garantizar una alta disponibilidad, Neptune sin servidor escala y reduce verticalmente las instancias de todas las AZ en función de la carga de la base de datos. Puede establecer el nivel de promoción de una instancia de lector en una AZ secundaria en 0 o 1 para que se escale o reduzca verticalmente junto con la capacidad de la instancia de escritor en la AZ principal, para que esté lista para asumir la carga de trabajo actual en cualquier momento.

> **Note**
>
> El almacenamiento de un clúster de bases de datos de Neptune consta de seis copias de todos los datos, repartidas en tres AZ, independientemente de si ha creado el clúster como un clúster de varias AZ o no. La replicación del almacenamiento la gestiona el subsistema de almacenamiento y no se ve afectada por Neptune sin servidor.

Selección de un valor de capacidad mínimo para un clúster de base de datos de Neptune sin servidor

El valor menor que se puede establecer para la capacidad mínima es de `1.0` NCU.

Asegúrese de no establecer el valor mínimo por debajo de lo que la aplicación requiere para funcionar de forma eficaz. Si se establece un valor demasiado bajo, se puede producir una mayor tasa de tiempos de espera en determinadas cargas de trabajo que consumen mucha memoria.

Si se establece el valor mínimo lo más bajo posible, se puede ahorrar dinero, ya que el clúster utilizará el número mínimo de recursos cuando la demanda sea baja. Sin embargo, si la carga de trabajo tiende a fluctuar de forma drástica, de muy baja a muy alta, puede ser más conveniente establecer el valor mínimo más alto, ya que de esta manera las instancias de Neptune sin servidor se escalarán verticalmente con mayor rapidez.

El motivo de esto es que Neptune elige los incrementos de escalado en función de la capacidad actual. Si la capacidad actual es baja, al principio Neptune escalará verticalmente de forma paulatina. Si el valor mínimo es mayor, Neptune comienza con un incremento de escalado mayor y, por lo tanto, puede escalar verticalmente de forma más rápida para hacer frente a un gran aumento repentino de la carga de trabajo.

Selección de un valor de capacidad máxima para un clúster de base de datos de Neptune sin servidor

El valor más alto que puede establecer para la capacidad máxima es `128.0` NCU y el más bajo es `2.5` NCU. Sea cual sea el valor máximo de capacidad que establezca, debe ser al menos igual al valor mínimo de capacidad que establezca.

Por regla general, establezca un valor máximo lo suficientemente alto como para hacer frente a los picos de carga que probablemente tenga la aplicación. Si se establece un valor demasiado bajo,

se puede producir una mayor tasa de tiempos de espera en determinadas cargas de trabajo que consumen mucha memoria.

Establecer el valor máximo lo más alto posible tiene la ventaja de que probablemente la aplicación pueda gestionar incluso las cargas de trabajo más inesperadas. La desventaja es que se pierde parte de la capacidad de predecir y controlar los costos de los recursos. Un aumento inesperado de la demanda puede terminar costando mucho más de lo que su presupuesto había previsto.

La ventaja de un valor máximo específicamente establecido es que le permite hacer frente a los picos de demanda y, al mismo tiempo, limitar los costos de computación de Neptune.

> **Note**
>
> El cambio del rango de capacidad de un clúster de base de datos de Neptune sin servidor provoca cambios en los valores predeterminados de algunos parámetros de configuración. Neptune puede aplicar algunos de estos nuevos valores predeterminados con efecto inmediato, pero algunos de los cambios en los parámetros dinámicos se aplicarán después de reiniciar el sistema. El estado `pending-reboot` indica que es necesario reiniciar para aplicar algunos cambios en los parámetros.

Uso de la configuración actual para estimar los requisitos del sistema sin servidor

Si normalmente modifica la clase de instancia de base de datos de las instancias de base de datos aprovisionadas para hacer frente a cargas de trabajo especialmente altas o bajas, puede sacar partido de esa experiencia para hacer una estimación aproximada del rango de capacidad equivalente de Neptune sin servidor.

Estimación de la mejor configuración de capacidad mínima

Puede aplicar lo que sabe sobre el clúster de base de datos de Neptune existente para estimar la mejor configuración de capacidad mínima sin servidor.

Por ejemplo, si la carga de trabajo aprovisionada tiene requisitos de memoria demasiado altos para clases de instancias de base de datos pequeñas como `T3` o `T4g`, elija una configuración de NCU mínima que proporcione memoria que sea comparable con una clase de instancia de base de datos `R5` o `R6g`.

O bien, supongamos que utiliza la clase de instancia de base de datos `db.r6g.xlarge` cuando el clúster tiene una carga de trabajo baja. La clase de instancia de base de datos tiene 32 GiB de memoria, por lo que puede especificar una configuración mínima de NCU de 16 para crear instancias sin servidor que pueden reducirse verticalmente a aproximadamente la misma capacidad (cada NCU corresponde a unos 2 GiB de memoria). Si, en ocasiones, la instancia `db.r6g.xlarge` se ha infrautilizado, es posible que pueda especificar un valor más bajo.

Si la aplicación funciona de manera óptima cuando las instancias de base de datos pueden contener una determinada cantidad de datos en la memoria o en la caché del búfer, plantéese especificar una configuración mínima de NCU lo suficientemente grande como para proporcionar suficiente memoria. De lo contrario, es posible que los datos se expulsen de la caché del búfer cuando las instancias sin servidor se reduzcan verticalmente y, con el tiempo, deberán volver a leerse en la caché del búfer cuando las instancias se vuelvan a escalar verticalmente. Si la cantidad de E/S para devolver los datos a la caché del búfer es importante, podría merecer la pena elegir un valor de NCU mínimo más alto.

Si descubre que las instancias sin servidor se ejecutan la mayor parte del tiempo con una capacidad determinada, sería buena opción establecer la capacidad mínima un poco más baja. Neptune sin servidor puede estimar de forma eficaz cuánto y con qué rapidez deben escalarse verticalmente cuando la capacidad actual no sea drásticamente inferior a la capacidad necesaria.

En una configuración mixta, con lectores de Neptune sin servidor y un escritor aprovisionado, los lectores no se pueden escalar junto con el escritor. Dado que se escalan de forma independiente, establecer una capacidad mínima baja para ellos puede provocar un retraso excesivo de la replicación. Es posible que no tengan la capacidad suficiente para mantenerse al día con los cambios que realiza el escritor cuando hay una carga de trabajo con uso muy intensivo de escritura. En esta situación, establezca una capacidad mínima que sea comparable con la capacidad del escritor. En concreto, si observa un retraso de réplica en los lectores que se encuentran en los niveles de promoción 2 a 15, aumente la configuración de capacidad mínima del clúster.

Estimación de la mejor configuración de capacidad máxima

También puede aplicar lo que sabe sobre el clúster de base de datos de Neptune existente para estimar la configuración de capacidad máxima sin servidor que funcione mejor.

Por ejemplo, supongamos que utiliza la clase de instancia de base de datos `db.r6g.4xlarge` cuando el clúster tiene una carga de trabajo elevada. Esa clase de instancia de base de datos tiene 128 GiB de memoria, por lo que puede especificar una configuración de NCU máxima de 64 para configurar instancias equivalentes de Neptune sin servidor (cada NCU corresponde a unos 2 GiB

de memoria). Puede especificar un valor más alto para permitir que la instancia de base de datos se escale verticalmente aún más en caso de que la instancia `db.r6g.4xlarge` no pueda gestionar siempre la carga de trabajo.

Si es poco frecuente que se produzcan picos inesperados en la carga de trabajo, se recomienda establecer una capacidad máxima lo suficientemente alta como para mantener el rendimiento de las aplicaciones incluso durante estos picos. Por otro lado, es posible que desee establecer una capacidad máxima más baja que pueda reducir el rendimiento durante picos inusuales, pero que permita a Neptune gestionar las cargas de trabajo previstas sin problemas, y esto limita los costos.

Configuración adicional para instancias y clústeres de bases de datos de Neptune sin servidor

Además de establecer la capacidad mínima y máxima para el clúster de base de datos de Neptune sin servidor, hay otras opciones de configuración que se deben tener en cuenta.

Combinación de instancias aprovisionadas y sin servidor de un clúster de base de datos

Un clúster de base de datos no tiene por qué ser únicamente sin servidor: puede crear una combinación de instancias aprovisionadas y sin servidor (una configuración mixta).

Por ejemplo, supongamos que necesita más capacidad de escritura de la disponible en una instancia sin servidor. En ese caso, puede configurar el clúster con un escritor aprovisionado de gran capacidad y seguir utilizando instancias sin servidor para los lectores.

O bien, supongamos que la carga de trabajo de escritura del clúster varía pero la carga de trabajo de lectura es estable. En este caso, puede configurar el clúster con un escritor sin servidor y uno o varios lectores aprovisionados.

Consulte Uso de Amazon Neptune sin servidor para obtener información sobre cómo crear un clúster de base de datos de configuración mixta.

Configuración de los niveles de promoción para las instancias de Neptune sin servidor

En el caso de clústeres que incluyen varias instancias sin servidor o una combinación de instancias sin servidor y aprovisionadas, preste atención a la configuración del nivel de promoción para cada

instancia sin servidor. Esta configuración controla más comportamiento para instancias sin servidor que para instancias de base de datos aprovisionadas.

En esta configuración AWS Management Console, utilice la prioridad de conmutación por error en Configuración adicional, en las páginas Crear base de datos, Modificar instancia y Añadir lector. Puede ver esta propiedad para las instancias existentes en la columna opcional Nivel prioritario de la página Bases de datos. También puede ver esta propiedad en la página de detalles de una instancia o clúster de base de datos.

En el caso de las instancias aprovisionadas, la elección de nivel 0 a 15 determina únicamente el orden en que Neptune elige qué instancia de lector debe promocionarse al escritor durante una operación de conmutación por error. En el caso de las instancias de lector de Neptune sin servidor, el número de nivel también determina si la instancia se escala verticalmente para adaptarse a la capacidad de la instancia de escritor o si se escala independientemente de ella en función únicamente de su propia carga de trabajo.

Las instancias de lector de Neptune sin servidor de nivel 0 o 1 se mantienen con una capacidad mínima al menos igual a la de la instancia de escritor, para que estén listas para sustituir al escritor en caso de conmutación por error. Si el escritor es una instancia aprovisionada, Neptune estima la capacidad sin servidor equivalente y usa esa estimación como capacidad mínima para la instancia de lector sin servidor.

Las instancias de lector de Neptune sin servidor de los niveles 2 a 15 no tienen la misma restricción en cuanto a la capacidad mínima y se escalan de forma independiente del escritor. Cuando están inactivas, se reducen verticalmente hasta el valor mínimo de NCU especificado en el rango de capacidad del clúster. Sin embargo, esto puede provocar problemas si la carga de trabajo de lectura aumenta rápidamente.

Mantener la capacidad del lector alineada con la capacidad del escritor

Un aspecto importante que debe tener en cuenta es que debe asegurarse de que las instancias de lector puedan seguir el ritmo de la instancia de escritor, con el fin de evitar un retraso excesivo en la replicación. Esto es especialmente preocupante en dos situaciones, en las que las instancias de lector sin servidor no se reducen horizontalmente de forma automática ni sincronizada con la instancia de escritor:

- Cuando el escritor está aprovisionado y los lectores no tienen servidor.
- Cuando el escritor no tiene servidor y los lectores sin servidor tienen niveles de promoción de 2 a 15.

En ambos casos, establezca la capacidad mínima sin servidor para que coincida con la capacidad prevista de escritor, con el fin de garantizar que las operaciones del lector no agoten el tiempo de espera y puedan provocar reinicios. En el caso de una instancia de escritor aprovisionada, establezca la capacidad mínima para que coincida con la de la instancia aprovisionada. En el caso de un escritor sin servidor, la configuración óptima puede ser más difícil de predecir.

Dado que el rango de capacidad de las instancias se establece en el nivel de clúster, todas las instancias sin servidor se controlan con la misma configuración de capacidad mínima y máxima. Las instancias de lector de los niveles 0 y 1 se reducen horizontalmente y de forma sincronizada con la instancia de escritor, pero las instancias de los niveles de promoción 2 a 15 se escalan de forma independiente de las unas de las otras y de la instancia de escritor, en función de su carga de trabajo. Al establecer una capacidad mínima demasiado baja, las instancias inactivas de los niveles 2 a 15 pueden reducirse verticalmente demasiado bajo para volver a escalarse verticalmente con la suficiente rapidez como para hacer frente a un aumento repentino de la actividad del escritor.

Evitar la configuración de un valor de tiempo de espera demasiado amplio

Es posible incurrir en costos inesperados si se establece un valor de tiempo de espera de consulta demasiado alto en una instancia sin servidor.

Si no se establece un tiempo de espera razonable, es posible que emita una consulta sin darse cuenta que requiere un tipo de instancia potente y costosa y que siga ejecutándose durante mucho tiempo, lo que generará costes que no había previsto. Para evitar esta situación, utilice un valor de tiempo de espera de consulta que se adapte a la mayoría de las consultas y que solo provoque que se agoten inesperadamente las que se ejecutan durante mucho tiempo.

Esto es válido tanto para los valores generales de tiempo de espera de consulta establecidos mediante parámetros como para los valores de tiempo de espera por consulta establecidos mediante sugerencias de consulta.

Optimización de la configuración de Neptune sin servidor

Si el clúster de base de datos Neptune sin servidor no está ajustado a la carga de trabajo que está ejecutando, es posible que observe que no se ejecuta de forma óptima. Puede ajustar la configuración de capacidad mínima o máxima para que pueda escalarse sin problemas de memoria.

- Aumente la configuración de capacidad mínima del clúster. Esto puede solucionar la situación en la que una instancia inactiva se vuelve a escalar a una capacidad que tiene menos memoria de la que necesitan la aplicación y las características habilitadas.

- Aumente la configuración de capacidad máxima del clúster. Esto puede solucionar la situación en la que una base de datos ocupada no puede escalar verticalmente a una capacidad con suficiente memoria para gestionar la carga de trabajo y las características de uso intensivo de memoria habilitadas.
- Cambie la carga de trabajo de la instancia en cuestión. Por ejemplo, puede añadir instancias de lector al clúster para distribuir la carga de lectura entre más instancias.
- Ajuste las consultas de la aplicación para que utilicen menos recursos.
- Intente utilizar una instancia aprovisionada que sea mayor que la cantidad máxima de NCU disponibles en Neptune sin servidor para comprobar si se ajusta mejor a los requisitos de memoria y CPU de la carga de trabajo.

Uso de Amazon Neptune sin servidor

Puede crear un nuevo clúster de base de datos de Neptune como uno sin servidor o, en algunos casos, puede convertir un clúster de base de datos existente para usarlo sin servidor. También puede convertir las instancias de base de datos de un clúster de base de datos sin servidor en instancias sin servidor y a partir de ellas. Neptune Serverless solo se puede utilizar en aquellos lugares en los que Regiones de AWS sea compatible, con algunas otras limitaciones (consulte). Restricciones de Amazon Neptune sin servidor

También puede usar la pila de AWS CloudFormation de Neptune para crear un clúster de base de datos de Neptune sin servidor.

Creación de un nuevo clúster de base de datos que utilice la tecnología sin servidor

Para crear un clúster de base de datos de Neptune que utilice la tecnología sin servidor, puede hacerlo mediante la AWS Management Console del mismo modo que lo hace para crear un clúster aprovisionado. La diferencia es que en la opción Tamaño de la instancia de base de datos, debe establecer la clase de instancia de base de datos en sin servidor. Cuando lo haga, tendrá que establecer el rango de capacidad sin servidor para el clúster.

También puede crear un clúster de base de datos sin servidor mediante comandos como AWS CLI los siguientes (en Windows, sustituya '\' por '^'):

```
aws neptune create-db-cluster \
```

```
  --region (an Región de AWS region that supports serverless) \
  --db-cluster-identifier (ID for the new serverless DB cluster) \
  --engine neptune \
  --engine-version (optional: 1.2.0.1 or above) \
  --serverless-v2-scaling-configuration "MinCapacity=1.0, MaxCapacity=128.0"
```

También puede especificar el parámetro `serverless-v2-scaling-configuration`, tal y como se indica a continuación:

```
  --serverless-v2-scaling-configuration '{"MinCapacity":1.0, "MaxCapacity":128.0}'
```

A continuación, puede ejecutar el comando `describe-db-clusters` para el atributo `ServerlessV2ScalingConfiguration`, que debería devolver la configuración del rango de capacidad que ha especificado:

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": (the specified minimum number of NCUs),
    "MaxCapacity": (the specified maximum number of NCUs)
}
```

Conversión de un clúster o instancia de base de datos existente en la tecnología sin servidor

Si tiene un clúster de base de datos de Neptune que utiliza la versión 1.2.0.1 o posterior del motor, puede convertirlo en un clúster sin servidor. Este proceso provoca cierto tiempo de inactividad.

El primer paso es añadir un rango de capacidad al clúster existente. Puede hacerlo mediante el AWS Management Console comando o mediante un AWS CLI comando como este (en Windows, sustituya '\' por '^'):

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your DB cluster ID) \
  --serverless-v2-scaling-configuration \
      MinCapacity=(minimum number of NCUs, such as  2.0), \
      MaxCapacity=(maximum number of NCUs, such as 24.0)
```

El siguiente paso consiste en crear una nueva instancia de base de datos sin servidor para reemplazar la instancia principal existente (el escritor) en el clúster. De nuevo, puede realizar este y

todos los pasos siguientes utilizando el AWS Management Console o el AWS CLI. En cualquier caso, debe especificar la clase de instancias de bases de datos como sin servidor. El AWS CLI comando tendría este aspecto (en Windows, sustituya '\' por '^'):

```
aws neptune create-db-instance \
  --db-instance-identifier (an instance ID for the new writer instance) \
  --db-cluster-identifier (ID of the DB cluster) \
  --db-instance-class db.serverless
  --engine neptune
```

Cuando la nueva instancia de escritor esté disponible, realice una conmutación por error para convertirla en la instancia de escritor del clúster:

```
aws neptune failover-db-cluster \
  --db-cluster-identifier (ID of the DB cluster) \
  --target-db-instance-identifier (instance ID of the new serverless instance)
```

A continuación, elimine la antigua instancia de escritor:

```
aws neptune delete-db-instance \
  --db-instance-identifier (instance ID of the old writer instance) \
  --skip-final-snapshot
```

Por último, haga lo mismo para crear una nueva instancia sin servidor que sustituya a cada instancia de lector aprovisionada existente que desee convertir en una instancia sin servidor, y elimine las instancias aprovisionadas existentes (no es necesaria una conmutación por error para las instancias de lector).

Modificación del rango de capacidad de un clúster de base de datos sin servidor existente

Puede cambiar el rango de capacidad de un clúster de base de datos Neptune sin servidor mediante la AWS CLI de la siguiente manera (en Windows, sustituya “\” por “^”):

```
aws neptune modify-db-cluster \
  --region (an AWS region that supports serverless) \
  --db-cluster-identifier (ID of the serverless DB cluster) \
  --apply-immediately \
```

```
  --serverless-v2-scaling-configuration MinCapacity=4.0, MaxCapacity=32
```

El cambio del rango de capacidad provoca cambios en los valores predeterminados de algunos parámetros de configuración. Neptune puede aplicar algunos de estos nuevos valores predeterminados de inmediato, pero algunos de los cambios en los parámetros dinámicos entrarán en vigor solo después de reiniciar. El estado `pending-reboot` indica que es necesario reiniciar para aplicar algunos cambios en los parámetros.

Cambio de una instancia de base de datos sin servidor a aprovisionada

Todo lo que necesita hacer para convertir una instancia de Neptune sin servidor en una aprovisionada es cambiar su clase de instancia por una de las clases de instancias aprovisionadas. Consulte Modificación de una instancia de base de datos de Neptune (y aplicación inmediata).

Supervisión de la capacidad sin servidor con Amazon CloudWatch

Puede usarlo CloudWatch para monitorear la capacidad y el uso de las instancias sin servidor de Neptune en su clúster de base de datos. Hay dos CloudWatch métricas que le permiten realizar un seguimiento de la capacidad actual sin servidor tanto a nivel de clúster como a nivel de instancia:

- **`ServerlessDatabaseCapacity`**: como métrica en el nivel de instancia, `ServerlessDatabaseCapacity` indica la capacidad actual de la instancia en NCU. Como métrica en el nivel de clúster, representa la media de todos los valores de `ServerlessDatabaseCapacity` de todas las instancias de base de datos del clúster.
- **`NCUUtilization`**: esta métrica indica el porcentaje de capacidad posible que se está utilizando. Se calcula como la `ServerlessDatabaseCapacity` actual (tanto en el nivel de instancia o de clúster) dividida por la configuración de capacidad máxima del clúster de base de datos.

  Si esta métrica se acerca al 100 % en el nivel de clúster, lo que significa que el clúster ha escalado lo más alto posible, plantéese aumentar la configuración de capacidad máxima.

  Si se acerca al 100% para una instancia de lector mientras que la instancia de escritor no está cerca de su capacidad máxima, plantéese añadir más instancias de lector para distribuir la carga de trabajo de lectura.

Tenga en cuenta que las métricas `CPUUtilization` y `FreeableMemory` tienen significados ligeramente diferentes para las instancias sin servidor que para las instancias aprovisionadas. En un contexto sin servidor, `CPUUtilization` es un porcentaje que se calcula como la cantidad de

CPU que se utiliza actualmente dividida por la cantidad de CPU disponible a máxima capacidad. Del mismo modo, `FreeableMemory` indica la cantidad de memoria que se puede liberar que estaría disponible si una instancia estuviera al máximo de su capacidad.

En el siguiente ejemplo, se muestra cómo utilizarla AWS CLI en Linux para recuperar los valores de capacidad mínima, máxima y media de una instancia de base de datos determinada, medidos cada 10 minutos durante una hora. El comando de Linux `date` especifica las horas de inicio y finalización en relación con la fecha y la hora en curso. La función `sort_by` en el parámetro `--query` ordena los resultados cronológicamente en función del campo `Timestamp`:

```
aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" \
  --end-time "$(date -d 'now')" \
  --period 600 \
  --namespace "AWS/Neptune"
  --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=(instance ID) \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
  --output table
```

Captura de cambios de gráficos en tiempo real con flujos de Neptune

Los flujos de Neptune registran cada cambio en el gráfico en el momento en el que se produce y en el orden en que se realiza de forma totalmente administrada. Una vez habilitados los flujos, Neptune se encarga de la disponibilidad, la copia de seguridad, la seguridad y el vencimiento.

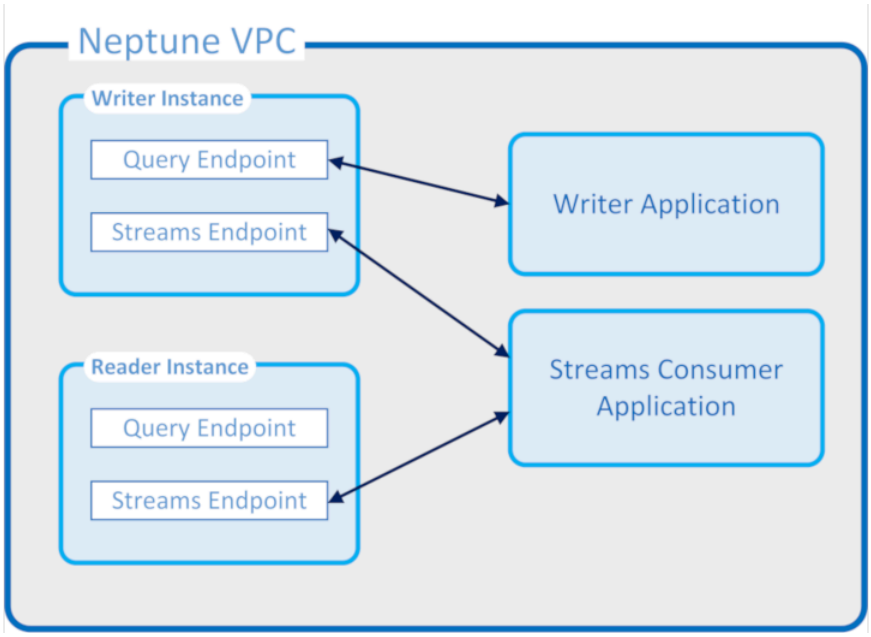
> **Note**
>
> Esta característica estaba disponible en el modo de laboratorio desde Versión 1.0.1.0.200463.0 (15/10/2019) y está disponible para su uso en producción a partir de la versión 1.0.2.2.R2 del motor de Neptune.

A continuación se indican algunos de los muchos casos de uso en los que es posible que desee capturar cambios en un gráfico a medida que se producen:

- Es posible que desee que su aplicación avise automáticamente a las personas cuando se realicen determinados cambios.
- Es posible que también desee mantener una versión actual de los datos de sus gráficos en otro almacén de datos, como Amazon OpenSearch Service ElastiCache, Amazon o Amazon Simple Storage Service (Amazon S3).

Neptune utiliza el mismo almacenamiento nativo para el flujo de registro de cambios que para los datos de gráficos. Escribe las entradas del registro de cambios de forma síncrona junto con la transacción que realiza dichos cambios. Puede recuperar estos registros de cambios del flujo de registros mediante una API de REST HTTP. (Para obtener más información, consulte Llamada a la API de Streams).

En el siguiente diagrama, se muestra cómo se pueden recuperar los datos de registro de cambios de los flujos de Neptune.



Garantías de flujos de Neptune

- Los cambios realizados por una transacción se pueden leer inmediatamente tanto del escritor como de los lectores en cuanto se completa la transacción (aparte de cualquier retraso de replicación normal en los lectores).
- Los registros de cambios aparecen estrictamente de forma secuencial y en el orden en que se produjeron (esto incluye los cambios realizados en una transacción).
- Las secuencias de cambios no contienen duplicados. Cada cambio se registra solo una vez.
- Las secuencias de cambios están completas. No se pierden ni se omiten cambios.
- Las secuencias de cambios contienen toda la información necesaria para determinar el estado completo de la base de datos en cualquier momento siempre que se conozca el estado inicial.
- Las transmisiones se pueden activar o desactivar en cualquier momento.

Propiedades operativas de flujos de Neptune

- El flujo de registros de cambios está completamente administrado.
- Los datos de registro de cambios se escriben de forma síncrona como parte de la misma transacción que realiza un cambio.
- Cuando los flujos de Neptune estén habilitados, se le cobrarán cargos de E/S y almacenamiento asociados con los datos del registro de cambios.

- De manera predeterminada, los registros de cambios se purgan automáticamente una semana después de su creación. A partir de la versión 1.2.0.0 del motor, este período de retención se puede cambiar mediante el parámetro de clúster de base de datos neptune_streams_expiry_days por cualquier número de días comprendido entre 1 y 90.
- El rendimiento de lectura en las secuencias se escala en función de las instancias.
- Puede conseguir un alto nivel de disponibilidad y rendimiento de lectura con réplicas de lectura. No existe ningún límite en cuanto al número de lectores de transmisiones que puede crear y utilizar de forma simultánea.
- Los datos de los registros de cambios se replican en varias zonas de disponibilidad, lo que hace que sean muy duraderos.
- Los datos de registro son tan seguros como los propios datos de gráficos. Se puede cifrar en reposo y en tránsito. El acceso se puede controlar mediante IAM, Amazon VPC AWS Key Management Service y AWS KMS(). Al igual que los datos del gráfico, se puede hacer una copia de seguridad de ellos y restaurarlos posteriormente mediante point-in-time restauraciones (PITR).
- La escritura síncrona de los datos de secuencia como parte de cada transacción provoca una ligera reducción del rendimiento general de escritura.
- Los datos de flujos no se fragmentan, ya que Neptune dispone de un diseño de un solo fragmento.
- La API `GetRecords` de flujo de registro utiliza los mismos recursos que todas las demás operaciones de gráficos de Neptune. Esto significa que los clientes deben equilibrar la carga entre las solicitudes de secuencia y otras solicitudes de base de datos.
- Cuando las secuencias están deshabilitadas, todos los datos de registro pasan a ser inaccesibles inmediatamente. Esto significa que debe leer todos los datos de registro que le interesen antes de deshabilitar el registro.
- Actualmente no existe una integración nativa con AWS Lambda. El flujo de registro no genera un evento que pueda activar una función de Lambda.

Temas

Uso de flujos de Neptune

Con la característica de flujos de Neptune, puede generar una secuencia completa de entradas de registros de cambios que capten cada cambio realizado en los datos de gráficos a medida que se produzca. Para obtener información general sobre esta característica, consulte Captura de cambios de gráficos en tiempo real con flujos de Neptune.

Temas

Habilitación de flujos de Neptune

Puede habilitar o deshabilitar los flujos de Neptune en cualquier momento estableciendo el parámetro del clúster de la base de datos `neptune_streams`. Al establecer el parámetro en `1`, se habilita Streams, y al definirlo en `0`, se deshabilita Streams.

> **Note**
>
> Una vez que haya cambiado el parámetro del clúster de base de datos `neptune_streams`, debe reiniciar todas las instancias de base de datos del clúster para que el cambio se aplique.

Puede configurar el parámetro de clúster de base de datos neptune_streams_expiry_days para controlar cuántos días, de 1 a 90, permanecerán los registros de flujos en el servidor antes de eliminarlos. El valor predeterminado es 7.

Los flujos de Neptune se introdujeron inicialmente como una característica experimental que se habilitaba o deshabilitaba en el modo de laboratorio mediante el parámetro `neptune_lab_mode` del clúster de base de datos (consulte Modo de laboratorio de Neptune). El uso del modo lab para habilitar Streams ahora está obsoleto y se deshabilitará en el futuro.

Deshabilitación de flujos de Neptune

Puede deshabilitar los flujos de Neptune en cualquier momento en que se esté ejecutando.

Para desactivar Streams, actualice el grupo de parámetros de clúster de base de datos para que el valor del parámetro `neptune_streams` se establezca en 0.

> **Important**
>
> En cuanto se desactive Streams, no podrá obtener acceso a los datos del registro de cambios. Asegúrese de leer lo que le interesa antes de desactivar Streams.

Llamada a la API de REST de flujos de Neptune

Puede acceder a los flujos de Neptune mediante una API de REST que envía una solicitud HTTP GET a uno de los siguientes puntos de conexión locales:

- Para una base de datos de gráficos SPARQL: `https://`*`Neptune-DNS`*`:8182/sparql/stream`.
- Para una base de datos de gráficos de Gremlin u openCypher: `https://`*`Neptune-DNS`*`:8182/propertygraph/stream` o `https://`*`Neptune-DNS`*`:8182/pg/stream`.

> **Note**
>
> A partir de la versión 1.1.0.0 del motor, el punto de conexión de flujo de Gremlin (`https://`*`Neptune-DNS`*`:8182/gremlin/stream`) está en desuso, junto con su formato de salida asociado (`GREMLIN_JSON`). Sigue siendo compatible con versiones anteriores, pero es posible que se elimine en futuras versiones.

Solo se permite una operación `GET` HTTP.

Neptune admite la compresión `gzip` de la respuesta, siempre que la solicitud HTTP incluya un encabezado `Accept-Encoding` que especifique `gzip` como formato de compresión aceptado (es decir, `"Accept-Encoding: gzip"`).

Parámetros

- `limit`: largo, opcional. Rango: 1–100 000. Predeterminado: 10.

  Especifica el número máximo de registros que se van a devolver. También existe un límite de tamaño de 10 MB en la respuesta que no se puede modificar y que prevalece sobre el número de registros especificado en el parámetro `limit`. La respuesta incluye un registro de incumplimiento de umbral si se alcanzó el límite de 10 MB.
- `iteratorType`: cadena, opcional.

  Este parámetro puede tener uno de los siguientes valores:

  - `AT_SEQUENCE_NUMBER` (predeterminado): indica que la lectura debe comenzar con el número de secuencia de eventos especificado conjuntamente por los parámetros `commitNum` y `opNum`.
  - `AFTER_SEQUENCE_NUMBER`: indica que la lectura debe comenzar justo después del número de secuencia de eventos especificado conjuntamente por los parámetros `commitNum` y `opNum`.
  - `TRIM_HORIZON`: indica que la lectura debe comenzar en el último registro no recortado del sistema, que es el registro más antiguo que no ha caducado (aún no se ha eliminado) del flujo de registro de cambios. Este modo es útil durante el inicio de la aplicación, cuando no tiene un número secuencial de evento inicial específico.
  - `LATEST`: indica que la lectura debe comenzar en el registro más reciente del sistema, que es el último registro que no ha caducado (aún no se ha eliminado) del flujo de registro de cambios. Esto resulta útil cuando es necesario leer los registros de la parte superior actual de los flujos para no procesar registros antiguos, por ejemplo, durante una recuperación de desastres o una actualización sin tiempo de inactividad. Tenga en cuenta que, en este modo, solo se devuelve como máximo un registro.
- `commitNum`: largo, obligatorio cuando iteratorType es `AT_SEQUENCE_NUMBER` o `AFTER_SEQUENCE_NUMBER`.

  El número de confirmación del registro de inicio que se va a leer del flujo de registro de cambios.

  Este parámetro se omite cuando `iteratorType` es `TRIM_HORIZON` o `LATEST`.
- `opNum`: largo, opcional (el valor predeterminado es 1).

  El número secuencial de la operación dentro de la confirmación especificada desde la que empezar a leer en los datos del flujo de registro de cambios.

Por lo general, las operaciones que cambian los datos de gráficos SPARQL solo generan un único registro de cambios por operación. Sin embargo, las operaciones que cambian los datos de gráficos de Gremlin pueden generar varios registros de cambio por operación, como en los siguientes ejemplos:

- `INSERT`: un vértice de Gremlin puede tener varias etiquetas y un elemento de Gremlin puede tener varias propiedades. Se genera un registro de cambio independiente para cada etiqueta y propiedad cuando se inserta un elemento.
- `UPDATE`: cuando se modifica una propiedad de un elemento de Gremlin, se generan dos registros de cambios: el primero para eliminar el valor anterior y el segundo para insertar el nuevo valor.
- `DELETE`: se genera un registro de cambios independiente para cada propiedad del elemento que se elimina. Por ejemplo, cuando se elimina un borde Gremlin con propiedades, se genera un registro de cambio para cada una de las propiedades y, a continuación, se genera uno para la eliminación de la etiqueta de borde.

  Cuando se elimina un vértice Gremlin, se eliminan primero todas las propiedades de borde de entrada y salida, después las etiquetas de borde, a continuación las propiedades de vértice y, por último, las etiquetas de vértice. Cada una de estas eliminaciones genera un registro de cambios.

Formato de respuesta de la API de flujos de Neptune

Una respuesta a una solicitud de la API de REST de los flujos de Neptune tiene los siguientes campos:

- `lastEventId`: identificador de secuencia del último cambio en la respuesta del flujo. Un ID de evento se compone de dos campos: `commitNum` identifica una transacción que ha cambiado el gráfico y `opNum` identifica una operación específica dentro de dicha transacción. Esto se muestra en el siguiente ejemplo.

  ```
  "eventId": {
    "commitNum": 12,
    "opNum": 1
  }
  ```

- `lastTrxTimestamp`: la hora a la que se solicitó la confirmación de la transacción, en milisegundos a partir de la fecha de inicio de Unix.

- `format`: formato de serialización de los registros de cambios que se devuelven. Los valores posibles son `PG_JSON` para registros de cambios de Gremlin u openCypher y `NQUADS` para registros de cambios de SPARQL.
- `records`: una matriz de registros de flujos de registro de cambios serializados incluidos en la respuesta. Cada registro de la matriz de `records` contiene los siguientes campos:
  - `commitTimestamp`: la hora a la que se solicitó la confirmación de la transacción, en milisegundos a partir de la fecha de inicio de Unix.
  - `eventId`: el identificador de secuencia del registro de cambios del flujo.
  - `data`— El registro serializado de Gremlin, SPARQL o change. OpenCypher Los formatos de serialización de cada registro se describen con más detalle en la siguiente sección, Formatos de serialización en flujos de Neptune.
  - op: la operación que creó el cambio.
  - `isLastOp`: solo está presente si esta operación es la última de su transacción. Cuando está presente, se establece en `true`. Es útil para garantizar que se realice una transacción completa.
- `totalRecords`: el número total de registros de la respuesta.

Por ejemplo, la siguiente respuesta devuelve los datos de cambios de Gremlin para una transacción que contiene más de una operación:

```
{
  "lastEventId": {
    "commitNum": 12,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
```

```
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD"
    }
  ],
  "totalRecords": 1
}
```

La siguiente respuesta devuelve los datos de cambios de SPARQL de la última operación de una transacción (la operación identificada por `EventId(97, 1)` con el número de transacción 97).

```
{
  "lastEventId": {
    "commitNum": 97,
    "opNum": 1
  },
  "lastTrxTimestamp": 1561489355102,
  "format": "NQUADS",
  "records": [
    {
      "commitTimestamp": 1561489355102,
      "eventId": {
        "commitNum": 97,
        "opNum": 1
      },
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Excepciones de la API de flujos de Neptune

En la siguiente tabla, se describen las excepciones de los flujos de Neptune.

Código de error	Código de HTTP	¿Reintentar?	Mensaje
`InvalidParameterException`	400	No	Se ha proporcionado un out-of-range valor o no válido como parámetro de entrada.
`ExpiredStreamException`	400	No	Todos los registros solicitados superan la edad máxima permitida y han caducado.
`ThrottlingException`	500	Sí	Rate of requests exceeds the maximum throughput (La tasa de solicitudes supera el desempeño máximo).
`StreamRecordsNotFoundException`	404	No	No se ha encontrado el recurso solicitado. Es posible que la secuencia no se especifique correctamente.
`MemoryLimitExceededException`	500	Sí	The request processing did not succeed due to lack of memory, but can be retried when the server is less busy (El procesamiento de la solicitud no se ha realizado correctamente debido a la falta de memoria, pero

Código de error	Código de HTTP	¿Reintentar?	Mensaje
			se puede volver a intentar cuando el servidor no esté tan saturado).

Formatos de serialización en flujos de Neptune

Amazon Neptune utiliza dos formatos diferentes para serializar los datos de cambios de gráficos en flujos de registro, en función de si el gráfico se creó con Gremlin o SPARQL.

Ambos formatos comparten un formato de serialización de registros común, como se describe en Formato de respuesta de la API de flujos de Neptune, que contiene los siguientes campos:

- `commitTimestamp`: la hora a la que se solicitó la confirmación de la transacción, en milisegundos a partir de la fecha de inicio de Unix.
- `eventId`: el identificador de secuencia del registro de cambios del flujo.
- `data`— El registro serializado de Gremlin, SPARQL o cambios. OpenCypher Los formatos de serialización de cada registro se describen con más detalle en las siguientes secciones.
- op: la operación que creó el cambio.

Temas

- Formato de serialización de cambios PG_JSON
- Formato de serialización de cambios SPARQL NQUADS

Formato de serialización de cambios PG_JSON

> **Note**
>
> A partir de la versión 1.1.0.0 del motor, el formato de salida de flujos de Gremlin (`GREMLIN_JSON`) emitido por el punto de conexión de flujos de Gremlin (`https://Neptune-DNS:8182/gremlin/stream`) está en desuso. Se sustituye por PG_JSON, que actualmente es idéntico a `GREMLIN_JSON`.

Un registro de cambios de Gremlin u openCypher, contenido en el campo `data` de una respuesta de flujo de registro, tiene los siguientes campos:

- `id`: cadena, obligatorio.

  El ID del elemento de Gremlin u openCypher.

- `type`: cadena, obligatorio.

  El tipo de este elemento de Gremlin u openCypher. Debe ser una de las siguientes:

  - `vl`: etiqueta de vértice para Gremlin; etiqueta de nodo para openCypher.
  - `vp`: propiedades de vértice para Gremlin; propiedades de nodo para openCypher.
  - `e`: borde y etiqueta de borde para Gremlin; relación y tipo de relación para openCypher.
  - `ep`: propiedades de borde para Gremlin; propiedades de relación para openCypher.

- `key`: cadena, obligatorio.

  Nombre de la propiedad. Para las etiquetas de elementos, se trata de "etiqueta".

- `value`: objeto `value`, obligatorio.

  Se trata de un objeto JSON que contiene un campo `value` para el propio valor y un campo `datatype` para el tipo de datos JSON de ese valor.

  ```
  "value": {
    "value": "the new value",
    "dataType": "the JSON datatype of the new value"
  }
  ```

- `from`: cadena, opcional.

  Si se trata de un borde (tipo = e), el identificador del vértice from o nodo de origen correspondientes.

- `to`: cadena, opcional.

  Si se trata de un borde (tipo="e"), el identificador del vértice to o nodo de destino correspondientes.

Ejemplos de Gremlin

- A continuación se muestra un ejemplo de una etiqueta de vértice de Gremlin.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the vertex label",
    "dataType": "String"
  }
}
```

- A continuación se muestra un ejemplo de una propiedad de vértice de Gremlin.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the vertex property",
    "dataType": "the datatype of the vertex property"
  }
}
```

- A continuación se muestra un ejemplo de un borde de Gremlin.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the edge",
    "dataType": "String"
  },
  "from": "the ID of the corresponding "from" vertex",
  "to": "the ID of the corresponding "to" vertex"
}
```

Ejemplos de openCypher

- A continuación, se muestra un ejemplo de una etiqueta de nodo de openCypher.

```
{
```

```
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the node label",
    "dataType": "String"
  }
}
```

- A continuación, se muestra un ejemplo de una propiedad de nodo de openCypher.

```
{
  "id": "an ID string",
  "type": "vp",
  "key": "the property name",
  "value": {
    "value": "the new value of the node property",
    "dataType": "the datatype of the node property"
  }
}
```

- A continuación, se muestra un ejemplo de una relación de openCypher.

```
{
  "id": "an ID string",
  "type": "e",
  "key": "label",
  "value": {
    "value": "the new value of the relationship",
    "dataType": "String"
  },
  "from": "the ID of the corresponding source node",
  "to": "the ID of the corresponding target node"
}
```

Formato de serialización de cambios SPARQL NQUADS

Neptune registra cambios en los cuádruples de SPARQL en el gráfico utilizando el lenguaje N-QUADS del marco de descripción de recursos (RDF) definido en la especificación W3C RDF 1.1 N-Quads.

El campo `data` del registro de cambios simplemente contiene un campo `stmt` que contiene una instrucción N-QUADS que expresa el cuadrante modificado, como en el siguiente ejemplo.

```
  "stmt" : "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
```

Ejemplos de flujos de Neptune

Los siguientes ejemplos muestran cómo obtener acceso a los datos del flujo de registro de cambios en Amazon Neptune.

Temas

Registro de cambios de **AT_SEQUENCE_NUMBER**

En el siguiente ejemplo, se muestra un registro de cambios `AT_SEQUENCE_NUMBER` de Gremlin u openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1560011610678,
```

```
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Aquí se muestra un ejemplo de SPARQL de un registro de cambios de AT_SEQUENCE_NUMBER.

```
curl -s "https://localhost:8182/sparql/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AT_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1571252030566,
  "format": "NQUADS",
  "records": [
    {
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "commitTimestamp": 1571252030566,
      "data": {
        "stmt": "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Registro de cambios de **AFTER_SEQUENCE_NUMBER**

En el siguiente ejemplo, se muestra un registro de cambios `AFTER_SEQUENCE_NUMBER` de Gremlin u openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&commitNum=1&opNum=1&iteratorType=AFTER_SEQUENCE_NUMBER" |jq
{
  "lastEventId": {
    "commitNum": 2,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011633768,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011633768,
      "eventId": {
        "commitNum": 2,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Registro de cambios de **TRIM_HORIZON**

En el siguiente ejemplo, se muestra un registro de cambios `TRIM_HORIZON` de Gremlin u openCypher.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?
limit=1&iteratorType=TRIM_HORIZON" |jq
{
  "lastEventId": {
    "commitNum": 1,
    "opNum": 1
  },
  "lastTrxTimestamp": 1560011610678,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1560011610678,
      "eventId": {
        "commitNum": 1,
        "opNum": 1
      },
      "data": {
        "id": "d2b59bf8-0d0f-218b-f68b-2aa7b0b1904a",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "vertex",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Registro de cambios de **LATEST**

En el siguiente ejemplo, se muestra un registro de cambios LATEST de Gremlin u openCypher. Tenga en cuenta que los parámetros de la API `limit`, `commitNum` y opNum son completamente opcionales.

```
curl -s "https://Neptune-DNS:8182/propertygraph/stream?iteratorType=LATEST" | jq
{
  "lastEventId": {
    "commitNum": 21,
```

```
    "opNum": 4
  },
  "lastTrxTimestamp": 1634710497743,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1634710497743,
      "eventId": {
        "commitNum": 21,
        "opNum": 4
      },
      "data": {
        "id": "24be4e2b-53b9-b195-56ba-3f48fa2b60ac",
        "type": "e",
        "key": "label",
        "value": {
          "value": "created",
          "dataType": "String"
        },
        "from": "4",
        "to": "5"
      },
      "op": "REMOVE",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Registro de cambios de compresión

En el siguiente ejemplo, se muestra un registro de cambios de compresión de Gremlin u openCypher.

```
curl -sH \
  "Accept-Encoding: gzip" \
  "https://Neptune-DNS:8182/propertygraph/stream?limit=1&commitNum=1" \
  -H "Accept-Encoding: gzip" \
  -v |gunzip -|jq
> GET /propertygraph/stream?limit=1 HTTP/1.1
> Host: localhost:8182
> User-Agent: curl/7.64.0
> Accept: /
```

```
> Accept-Encoding: gzip
*> Accept-Encoding: gzip*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Connection: keep-alive
*< content-encoding: gzip*
< content-length: 191
<
{ [191 bytes data]
Connection #0 to host localhost left intact
{
  "lastEventId": "1:1",
  "lastTrxTimestamp": 1558942160603,
  "format": "PG_JSON",
  "records": [
    {
      "commitTimestamp": 1558942160603,
      "eventId": "1:1",
      "data": {
        "id": "v1",
        "type": "vl",
        "key": "label",
        "value": {
          "value": "person",
          "dataType": "String"
        }
      },
      "op": "ADD",
      "isLastOp": true
    }
  ],
  "totalRecords": 1
}
```

Uso AWS CloudFormation para configurar la replicación de Neptuno a Neptuno con la aplicación Streams Consumer

Puede usar una AWS CloudFormation plantilla para configurar la aplicación de consumo Neptune Streams para admitir la replicación de Neptuno a Neptuno.

Temas

- Elija una plantilla para su región AWS CloudFormation
- Añada detalles acerca de la pila del consumidor de flujos de Neptune que crea
- Ejecute la plantilla AWS CloudFormation
- Para actualizar el sondeador de flujos con los artefactos de Lambda más recientes

Elija una plantilla para su región AWS CloudFormation

Para lanzar la AWS CloudFormation pila adecuada en la AWS CloudFormation consola, selecciona uno de los botones de lanzamiento de la siguiente tabla, en función de la AWS región que quieras usar.

Región	Visualización	Ver en Designer	iniciar
Este de EE. UU. (Norte de Virginia)	Ver	Ver en Designer	
Este de EE. UU. (Ohio)	Ver	Ver en Designer	
Oeste de EE. UU. (Norte de California)	Ver	Ver en Designer	
Oeste de EE. UU. (Oregón)	Ver	Ver en Designer	
Canadá (centro)	Ver	Ver en Designer	
América del Sur (São Paulo)	Ver	Ver en Designer	
Europa (Estocolmo)	Ver	Ver en Designer	
Europa (Irlanda)	Ver	Ver en Designer	
Europa (Londres)	Ver	Ver en Designer	

Región	Visualización	Ver en Designer	iniciar
Europa (París)	Ver	Ver en Designer	
Europa (Fráncfort)	Ver	Ver en Designer	
Medio Oriente (Baréin)	Ver	Ver en Designer	
Medio Oriente (EAU)	Ver	Ver en Designer	
Israel (Tel Aviv)	Ver	Ver en Designer	
África (Ciudad del Cabo)	Ver	Ver en Designer	
Asia-Pacífico (Tokio)	Ver	Ver en Designer	
Asia-Pacífico (Hong Kong)	Ver	Ver en Designer	
Asia-Pacífico (Seúl)	Ver	Ver en Designer	
Asia-Pacífico (Singapur)	Ver	Ver en Designer	
Asia-Pacífico (Sídney)	Ver	Ver en Designer	
Asia-Pacífico (Bombay)	Ver	Ver en Designer	
China (Pekín)	Ver	Ver en Designer	
China (Ningxia)	Ver	Ver en Designer	

Región	Visualización	Ver en Designer	iniciar
AWS GovCloud (EE. UU.-Oeste)	Ver	Ver en Designer	
AWS GovCloud (EE. UU.-Este)	Ver	Ver en Designer	

En la página Create Stack (Crear pila), elija Next (Siguiente).

Añada detalles acerca de la pila del consumidor de flujos de Neptune que crea

La página Specify Stack Details (Especificar los detalles de la pila) proporciona las propiedades y los parámetros que puede utilizar para controlar la configuración de la aplicación:

Nombre de la pila: el nombre de la nueva AWS CloudFormation pila que está creando. Por lo general puede utilizar el valor predeterminado, `NeptuneStreamPoller`.

En Parámetros, proporcione lo siguiente:

Configuración de red para la VPC en la que se ejecuta el consumidor de flujos

- **`VPC`**: proporcione el nombre de la VPC en la que se ejecutará la función de Lambda de sondeo.
- **`SubnetIDs`**: las subredes en las que se establece una interfaz de red. Añada las subredes correspondientes a su clúster de Neptune.
- **`SecurityGroupIds`**: proporcione los identificadores de los grupos de seguridad que otorgan acceso de escritura entrante al clúster de base de datos de Neptune de origen.
- **`RouteTableIds`**: esto es necesario para crear un punto de conexión de Amazon DynamoDB en su VPC de Neptune, si todavía no dispone de uno. Debe proporcionar una lista separada por comas de ID de tabla de ruta asociados a las subredes.
- **`CreateDDBVPCEndPoint`**: valor booleano que es, de manera predeterminada, `true`, que indica si es necesario o no crear un punto de conexión de VPC de Dynamo DB. Solo necesita cambiarlo por `false` si ya ha creado un punto de conexión de DynamoDB en su VPC.
- **`CreateMonitoringEndPoint`**: valor booleano que es, de manera predeterminada, `true`, que indica si es necesario o no crear un punto de conexión de VPC de monitorización. Solo necesita cambiarlo por `false` si ya ha creado un punto de enlace de monitorización en su VPC.

Sondeador de flujo

- **`ApplicationName`**: por lo general, puede dejar esta opción con el valor predeterminado (`NeptuneStream`). Si utiliza un nombre diferente, debe ser único.
- **`LambdaMemorySize`**: se utiliza para establecer el tamaño de memoria disponible para la función de sondeo de Lambda. El valor predeterminado es 2048 megabytes.
- **`LambdaRuntime`**: el lenguaje utilizado en la función de Lambda que recupera elementos del flujo de Neptune. Puede configurarlo en `python3.9` o en `java8`.
- **`LambdaS3Bucket`**: el bucket de Amazon S3 que contiene artefactos de código Lambda. Deje este campo en blanco a menos que utilice una función de sondeo de Lambda personalizada que se cargue desde un bucket de Amazon S3 diferente.
- **`LambdaS3Key`**: la clave de Amazon S3 que corresponde a sus artefactos de código Lambda. Deje este campo en blanco a menos que utilice una función de sondeo de Lambda personalizada.
- **`LambdaLoggingLevel`**: en general, deje esta opción con el valor predeterminado, que es `INFO`.
- **`ManagedPolicies`**: muestra las políticas administradas que se van a utilizar para la ejecución de la función de Lambda. En general, deje este campo en blanco a menos que utilice una función de sondeo de Lambda personalizada.
- **`StreamRecordsHandler`**: en general, deje este campo en blanco a menos que utilice un controlador personalizado para los registros en los flujos de Neptune.
- **`StreamRecordsBatchSize`**: el número máximo de registros que se recuperará del flujo. Puede utilizar este parámetro para ajustar el rendimiento. El valor predeterminado (`5000`) es un buen lugar para empezar. El valor máximo permitido es 10 000. Cuanto mayor sea el número, menos llamadas de red se necesitan para leer registros del flujo, pero más memoria se precisa para procesar los registros. Los valores más bajos de este parámetro dan como resultado un rendimiento inferior.
- **`MaxPollingWaitTime`**: el tiempo máximo de espera entre dos sondeos (en segundos). Determina la frecuencia con la que se invoca el sondeador Lambda para sondear los flujos de Neptune. Establezca este valor en 0 para un sondeo continuo. El valor máximo es de 3600 segundos (1 hora). El valor predeterminado (60 segundos) es un buen lugar para empezar según la velocidad con la que cambien los datos del gráfico.
- **`MaxPollingInterval`**: el período máximo de sondeo continuo (en segundos). Se utiliza para establecer un tiempo de espera para la función de sondeo de Lambda. El valor debe estar en el intervalo entre 5 y 900 segundos. El valor predeterminado (600 segundos) es un buen lugar para empezar.

- **`StepFunctionFallbackPeriod`**— El número de unidades que deben esperar step-function-fallback-period al sondeador, tras lo cual se invoca la función step a través de Amazon CloudWatch Events para recuperarse de un fallo. El valor predeterminado (5 minutos) es un buen lugar para empezar.
- **`StepFunctionFallbackPeriodUnit`**: las unidades de tiempo utilizadas para medir el `StepFunctionFallbackPeriodUnit` anterior (`minutes`, `hours` o `days`). El valor predeterminado (`minutes`) generalmente es suficiente.

Flujo de Neptune

- **`NeptuneStreamEndpoint`**: (obligatorio) el punto de conexión del flujo de origen de Neptune. Adopta una de estas dos formas:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (o su alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - **`https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`**.
- **`Neptune Query Engine`**: elija Gremlin, openCypher o SPARQL.
- **`IAMAuthEnabledOnSourceStream`**: si su clúster de base de datos de Neptune usa la autenticación de IAM, establezca este parámetro como `true`.
- **`StreamDBClusterResourceId`**: si su clúster de base de datos de Neptune usa la autenticación de IAM, establezca este parámetro como el identificador de recurso de clúster. El ID del recurso no es el mismo que el ID del clúster. En su lugar, adopta el formato: `cluster-` seguido de 28 caracteres alfanuméricos. Se puede encontrar en Detalles del clúster en la consola de Neptune.

Clúster de base de datos Neptune de destino

- **`TargetNeptuneClusterEndpoint`**: el punto de conexión del clúster (solo nombre de host) del clúster de copia de seguridad de destino.

  Tenga en cuenta que si especifica `TargetNeptuneClusterEndpoint`, no puede especificar también `TargetSPARQLUpdateEndpoint`.
- **`TargetNeptuneClusterPort`**: el número de puerto del clúster de destino.

  Tenga en cuenta que si especifica `TargetSPARQLUpdateEndpoint`, se ignorará la configuración de `TargetNeptuneClusterPort`.

- **`IAMAuthEnabledOnTargetCluster`**: establézcalo en true si se va a habilitar la autenticación de IAM en el clúster de destino.
- **`TargetAWSRegion`**— La AWS región del clúster de backup de destino, por ejemplo`us-east-1`). Debe proporcionar este parámetro solo cuando la AWS región del clúster de respaldo de destino sea diferente de la región del clúster de origen de Neptune, como en el caso de la replicación entre regiones. Si las regiones de origen y destino son las mismas, este parámetro es opcional.

  Tenga en cuenta que si el `TargetAWSRegion` valor no es una AWS región válida compatible con Neptune, el proceso fallará.
- **`TargetNeptuneDBClusterResourceId`**: opcional: esto solo es necesario cuando la autenticación de IAM está habilitada en el clúster de base de datos de destino. Se establece en el identificador de recurso del clúster de destino.
- **`SPARQLTripleOnlyMode`**: indicador booleano que determina si el modo solo triple está habilitado. En el modo de solo triple, no hay replicación de gráficos con nombre. El valor predeterminado es `false`.
- **`TargetSPARQLUpdateEndpoint`**: URL del punto de conexión de destino para la actualización de SPARQL, como `https://abc.com/xyz`. Este punto de conexión puede ser cualquier almacén de SPARQL que admita cuádruples o triples.

  Tenga en cuenta que si especifica `TargetSPARQLUpdateEndpoint`, no podrá especificar también `TargetNeptuneClusterEndpoint` y se omitirá la configuración de `TargetNeptuneClusterPort`.
- **`BlockSparqlReplicationOnBlankNode`** — Indicador booleano que, si se establece en`true`, detiene la replicación BlankNode en los datos de SPARQL (RDF). El valor predeterminado es `false`.

Alarma

- **`Required to create Cloud watch Alarm`**— Configúrelo `true` si quiere crear una CloudWatch alarma para la nueva pila.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— El ARN del tema SNS al que CloudWatch se deben enviar las notificaciones de alarma (solo es necesario si las alarmas están habilitadas).
- **`Email for Alarm Notifications`**: la dirección de correo electrónico a la que se deben enviar las notificaciones de alarma (solo es necesaria si las alarmas están activadas).

Como destino de la notificación de alarma, puede añadir solo SNS, solo correo electrónico o tanto SNS como correo electrónico.

Ejecute la plantilla AWS CloudFormation

Ahora puede completar el proceso de aprovisionamiento de una instancia de aplicación de consumidor de flujos de Neptune de la siguiente manera:

1. En AWS CloudFormation, en la página Especificar los detalles de la pila, elija Siguiente.
2. En la página Opciones, seleccione Siguiente.
3. En la página Revisar, seleccione la primera casilla para confirmar que AWS CloudFormation debe crear los recursos de IAM. Seleccione la segunda casilla para confirmar `CAPABILITY_AUTO_EXPAND` para la nueva pila.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` confirma explícitamente que las macros se expandirán al crear la pila, sin revisión previa. Los usuarios suelen crear un conjunto de cambios a partir de una plantilla procesada para que los cambios realizados por las macros puedan revisarse antes de crear la pila. Para obtener más información, consulte la AWS CloudFormation CreateStackAPI en la referencia de la AWS CloudFormation API.

   A continuación, seleccione Crear.

Para actualizar el sondeador de flujos con los artefactos de Lambda más recientes

Puede actualizar el sondeador de flujos con los artefactos de Lambda más recientes de la siguiente manera:

1. En AWS Management Console, navegue hasta la AWS CloudFormation pila principal principal AWS CloudFormation y selecciónela.
2. Seleccione la opción Actualizar para la pila.
3. Seleccione Reemplazar la plantilla actual.
4. Para la fuente de la plantilla, elija la URL de Amazon S3 e introduzca la siguiente URL de S3:

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_neptune.json
```

5. Seleccione Siguiente sin cambiar ningún AWS CloudFormation parámetro.
6. Elija Update Stack (Actualizar pilar).

La pila actualizará ahora los artefactos de Lambda con los más recientes.

Uso de la replicación entre regiones de los flujos de Neptune para la recuperación de desastres

Neptune ofrece dos formas de implementar capacidades de conmutación por error entre regiones:

- Copia y restauración de instantáneas entre regiones
- Uso de los flujos de Neptune para replicar datos entre dos clústeres en dos regiones diferentes.

La copia y restauración de instantáneas entre regiones tiene la sobrecarga operativa más baja para recuperar un clúster de Neptune en una región diferente. Sin embargo, copiar una instantánea entre regiones puede requerir un tiempo de transferencia de datos considerable, ya que una instantánea es una copia de seguridad completa del clúster de Neptune. Como resultado, la copia y restauración de instantáneas entre regiones se pueden utilizar en escenarios que solo requieren un objetivo de punto de recuperación (RPO) de horas y un objetivo de tiempo de recuperación (RTO) de horas.

Un objetivo de punto de recuperación (RPO) se mide por el tiempo transcurrido entre copias de seguridad. Define la cantidad de datos que se pueden perder entre el momento en que se realizó la última copia de seguridad y el momento en que se recuperó la base de datos.

Un objetivo de tiempo de recuperación (RTO) se mide por el tiempo que se tarda en realizar una operación de recuperación. Es el tiempo que tarda el clúster de base de datos en realizar la conmutación por error a una base de datos recuperada tras producirse un error.

Los flujos de Neptune proporcionan una forma de mantener un clúster de Neptune de copia de seguridad sincronizado con el clúster de producción principal en todo momento. Si se produce un error, la base de datos se conmuta por error al clúster de copia de seguridad. Esto reduce el RPO y el RTO a minutos, ya que los datos se copian constantemente al clúster de copia de seguridad, que está disponible de forma inmediata como destino de conmutación por error en cualquier momento.

El inconveniente de usar los flujos de Neptune de esta manera es que tanto la sobrecarga operativa requerida para mantener los componentes de replicación como el costo de tener un segundo clúster de base de datos de Neptune en línea todo el tiempo pueden ser significativos.

Configuración de la replicación de Neptune a Neptune

El clúster de base de datos de producción principal reside en una VPC en una región de origen determinada. Hay tres elementos principales que debe replicar o emular en una región de recuperación diferente para poder realizar una recuperación de desastres:

- Los datos almacenados en el clúster.
- La configuración del clúster principal. Esto incluiría si utiliza o no la autenticación de IAM, si está cifrada o no, los parámetros del clúster de base de datos, los parámetros de las instancias, los tamaños de las instancias, etc.
- La topología de red que utiliza, incluida la VPC de destino, sus grupos de seguridad, etc.

Puede utilizar las API de administración de Neptune, como las siguientes, para recopilar esa información:

- `DescribeDBClusters`
- `DescribeDBInstances`
- `DescribeDBClusterParameters`
- `DescribeDBParameters`
- `DescribeVpcs`

Con la información recopilada, puede usar el siguiente procedimiento para configurar un clúster de copia de seguridad en una región diferente, a la que su clúster de producción pueda conmutar por error en caso de que se produzca un error.

1: Habilite los flujos de Neptune

Puede usar ModifyDBClusterParameterGroup para establecer el parámetro `neptune_streams` en 1. A continuación, reinicie todas las instancias del clúster de base de datos para que el cambio se aplique.

Se recomienda realizar al menos una operación de adición o actualización en el clúster de base de datos de origen después de habilitar los flujos de Neptune. Esto rellena el flujo de cambios con

puntos de datos a los que se puede hacer referencia más adelante al volver a sincronizar el clúster de producción con el clúster de copia de seguridad.

2: Cree una nueva VPC en la región en la que desee configurar el clúster de copia de seguridad

Antes de crear un nuevo clúster de base de datos de Neptune en una región diferente a la del clúster principal, debe establecer una nueva VPC en la región de destino para alojar el clúster. La conectividad entre los clústeres principal y de copia de seguridad se establece mediante la interconexión con VPC, que utiliza el tráfico entre subredes privadas de diferentes VPC. Sin embargo, para establecer la interconexión con VPC entre dos VPC, no deben tener bloques CIDR ni espacios de direcciones IP superpuestos. Esto significa que no puede usar la VPC predeterminada en ambas regiones, ya que el bloque CIDR de una VPC predeterminada es siempre el mismo (`172.31.0.0/16`).

Puede utilizar una VPC existente en la región de destino siempre que cumpla las siguientes condiciones:

- No tiene ningún bloque de CIDR que se superponga con el bloque de CIDR de la VPC en la que está ubicado el clúster principal.
- Aún no está interconectada con otra VPC que tenga el mismo bloque de CIDR que la VPC en la que se encuentra el clúster principal.

Si no hay ninguna VPC adecuada disponible en la región de destino, cree una con la API de `CreateVpc` de Amazon EC2.

3: Cree una instantánea de su clúster principal y restáurela en la región de copia de seguridad de destino

Ahora cree un nuevo clúster de Neptune en una VPC adecuada en la región de copia de seguridad de destino que es una copia de su clúster de producción:

Haga una copia de su clúster de producción en la región de copia de seguridad

1. En la región de copia de seguridad de destino, vuelva a crear los parámetros y los grupos de parámetros que utiliza el clúster de base de datos de producción. Puede hacerlo mediante `CreateDBClusterParameterGroup`, `CreateDBParameterGroup`, `ModifyDBClusterParameterGroup` y `ModifyDBParameterGroup`.

Tenga en cuenta que las API `CopyDBClusterParameterGroup` y `CopyDBParameterGroup` no admiten actualmente la copia entre regiones.

2. Utilice `CreateDBClusterSnapshot` para crear una instantánea del clúster de producción en la VPC de su región de producción.
3. Utilice `CopyDBClusterSnapshot` para copiar la instantánea a la VPC de la región de copia de seguridad de destino.
4. Utilice `RestoreDBClusterFromSnapshot` para crear un nuevo clúster de base de datos en la VPC de la región de copia de seguridad de destino mediante la instantánea copiada. Utilice los parámetros y ajustes de configuración que copió del clúster de producción principal.
5. El nuevo clúster de Neptune ya existe, pero no contiene ninguna instancia. Se usa `CreateDBInstance` para crear una nueva instancia principal/de escritura que tenga el mismo tipo y tamaño que la instancia de escritura del clúster de producción. No es necesario crear réplicas de lectura adicionales en este momento, a menos que la instancia de copia de seguridad se utilice para dar servicio a las operaciones de E/S de lectura en la región de destino antes de una conmutación por error.

4: Establezca la interconexión con VPC entre la VPC de su clúster principal y la VPC de su nuevo clúster de copia de seguridad

Al configurar la interconexión con VPC, permite que la VPC del clúster principal se comunique con la VPC del clúster de copia de seguridad como si se tratara de una sola red privada. Para ello, siga estos pasos:

1. Desde la VPC de su clúster de producción, llame a la API de `CreateVpcPeeringConnection` para establecer la conexión de interconexión.
2. Desde la VPC de su clúster de copia de seguridad de destino, llame a la API de `AcceptVpcPeeringConnection` para establecer la conexión de interconexión.
3. Desde la VPC de su clúster de producción, utilice la API de `CreateRoute` para añadir una ruta a la tabla de enrutamiento de la VPC que redirija todo el tráfico al bloque CIDR de la VPC de destino para que utilice la lista de prefijos de interconexión con VPC.
4. Del mismo modo, desde la VPC del clúster de copia de seguridad de destino, use la API de `CreateRoute` para añadir una ruta a la tabla de enrutamiento de la VPC que dirija el tráfico a la VPC del clúster principal.

5: Configure la infraestructura de replicación de los flujos de Neptune

Ahora que ambos clústeres están desplegados y se ha establecido la comunicación de red entre ambas regiones, utilice la AWS CloudFormation plantilla Neptune-Neptuno para implementar la función Lambda de consumo de flujos de Neptuno con la infraestructura adicional que admite la replicación de datos. Hágalo en la VPC de su clúster de producción principal.

Los parámetros que necesitará proporcionar para esta pila son: AWS CloudFormation

- **`NeptuneStreamEndpoint`**: el punto de conexión del flujo del clúster principal, en formato URL. Por ejemplo: `https://(cluster name):8182/pg/stream`.
- **`QueryEngine`**: debe ser `gremlin`, `sparql` o `openCypher`.
- **`RouteTableIds`**: le permite añadir rutas tanto para un punto de conexión de VPC de DynamoDB como para un punto de conexión de VPC de monitorización.

  Dos parámetros adicionales, `CreateMonitoringEndpoint` y `CreateDynamoDBEndpoint`, también se deben establecer en true si aún no existen en la VPC del clúster principal. Si ya existen, asegúrese de que estén configurados como falsos o se producirá un error en la AWS CloudFormation creación.
- **`SecurityGroupIds`**: especifica el grupo de seguridad que utiliza el consumidor de Lambda para comunicarse con el punto de conexión del flujo de Neptune del clúster principal.

  En el clúster de copia de seguridad de destino, asocie un grupo de seguridad que permita que el tráfico se origine desde este grupo de seguridad.
- **`SubnetIds`**: una lista de los identificadores de subred de la VPC del clúster principal que el consumidor de Lambda puede utilizar para comunicarse con el clúster principal.
- **`TargetNeptuneClusterEndpoint`**: el punto de conexión del clúster (solo nombre de host) del clúster de copia de seguridad de destino.
- **`TargetAWSRegion`**— La AWS región del clúster de respaldo de destino, por ejemplo`us-east-1`). Debe proporcionar este parámetro solo cuando la AWS región del clúster de respaldo de destino sea diferente de la región del clúster de origen de Neptune, como en el caso de la replicación entre regiones. Si las regiones de origen y destino son las mismas, este parámetro es opcional.

  Tenga en cuenta que si el `TargetAWSRegion` valor no es una AWS región válida compatible con Neptune, el proceso fallará.
- **`VPC`**: el identificador de la VPC del clúster principal.

Todos los demás parámetros se pueden dejar con sus valores predeterminados.

Una vez implementada la AWS CloudFormation plantilla, Neptune empezará a replicar cualquier cambio del clúster principal al clúster de respaldo. Puede supervisar esta replicación en los CloudWatch registros generados por la función de consumo de Lambda.

Otras consideraciones

- Si necesita usar la autenticación de IAM entre el clúster principal y el de respaldo, también puede configurarla al invocar la plantilla. AWS CloudFormation
- Si el cifrado en reposo está habilitado en su clúster principal, considere cómo administrar las claves de KMS asociadas al copiar la instantánea a la región de destino y asociar una nueva clave de KMS a la región de destino.
- Una práctica recomendada es utilizar los CNAME de DNS delante de los puntos de conexión de Neptune que se utilizan en las aplicaciones. Luego, si necesita realizar una conmutación por error manual al clúster de copia de seguridad de destino, estos CNAME se pueden cambiar para que apunten al clúster de destino o a los puntos de conexión de la instancia.

Búsqueda de texto completo en Amazon Neptune mediante Amazon Service OpenSearch

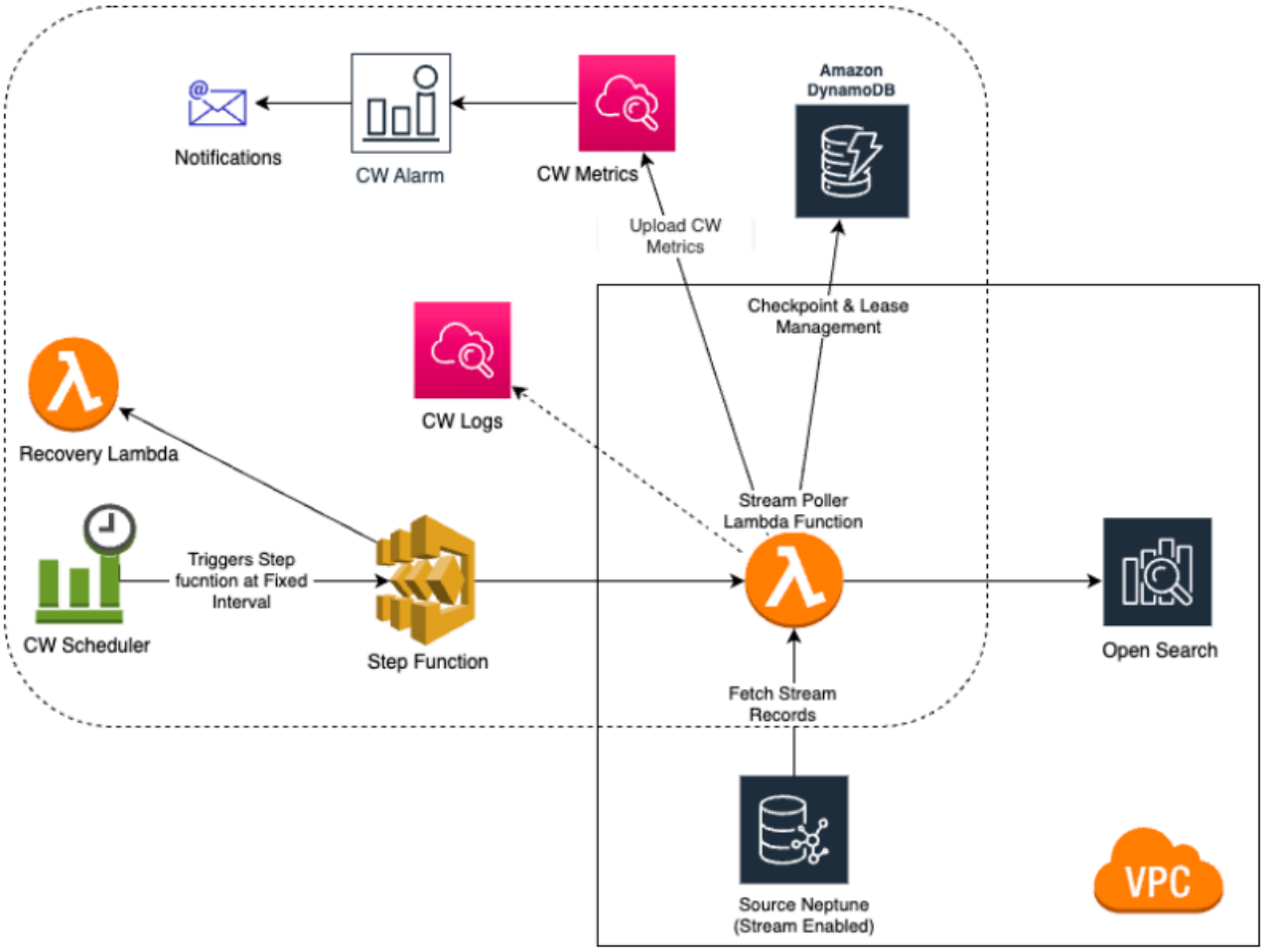
Neptune se integra con Amazon OpenSearch Service (OpenSearch Servicio) para admitir la búsqueda de texto completo en las consultas de Gremlin y SPARQL. Esta característica está disponible a partir de la versión 1.0.2.1 del motor de Neptune, aunque recomendamos usarla con la versión del motor `1.0.4.2` o posterior para sacar partido de las últimas correcciones.

A partir de la versión 1.3.0.0 del motor, Amazon Neptune admite el uso de OpenSearch Amazon Service Serverless para la búsqueda de texto completo en consultas de Gremlin y SPARQL.

> **Note**
>
> Al integrarse con Amazon OpenSearch Service, Neptune requiere la versión 7.1 o superior de Elasticsearch y funciona con las versiones OpenSearch 2.3, 2.5 y superiores. Neptune también funciona con OpenSearch Serverless.

Puede usar Neptune con un clúster de OpenSearch servicios existente que se haya rellenado de acuerdo con. Modelo de datos de Neptune para datos de OpenSearch O bien, puede crear un dominio OpenSearch de servicio vinculado a Neptune mediante una AWS CloudFormation pila.

> **Important**
>
> El proceso de Neptune a OpenSearch replicación descrito aquí no replica los nodos vacíos. Es importante tener en cuenta esta limitación.
> Además, si habilita un control de acceso detallado en su OpenSearch clúster, también debe habilitar la autenticación de IAM en su base de datos de Neptune.



Temas

- Amazon Neptune a replicación OpenSearch
- Replicación a OpenSearch Serverless
- Consultas desde un clúster de OpenSearch con el control de acceso detallado (FGAC) habilitado
- Uso de la sintaxis de consulta de Apache Lucene en las consultas de búsqueda de texto completo de Neptune
- Modelo de datos de Neptune para datos de OpenSearch
- Parámetros de búsqueda de texto completo de Neptune
- Indexación de OpenSearch sin cadenas en Amazon Neptune
- Ejecución de consultas de búsqueda de texto completo en Amazon Neptune
- Consultas de SPARQL de ejemplo mediante la búsqueda de texto completo en Neptune
- Uso de la búsqueda de texto completo de Neptune en las consultas de Gremlin
- Solución de problemas de búsqueda de texto completo de Neptune

Amazon Neptune a replicación OpenSearch

Amazon Neptune admite la búsqueda de texto completo en las consultas de Gremlin y SPARQL mediante Amazon Service (Servicio). OpenSearch OpenSearch Puede usar una AWS CloudFormation pila para vincular un dominio OpenSearch de servicio a Neptune. La AWS CloudFormation plantilla crea una instancia de aplicación consumidora de flujos que proporciona una replicación de Neptuno a otra. OpenSearch

Antes de empezar, necesita un clúster de base de datos de Neptune existente con transmisiones habilitadas para que sirva como origen y un dominio de OpenSearch servicio que sirva como destino de la replicación.

Si ya tiene un dominio de OpenSearch servicio de destino al que Lambda pueda acceder en la VPC en la que se encuentra el clúster de base de datos de Neptune, la plantilla puede usarlo. De lo contrario, es necesario crear una nueva.

> **Note**
>
> El OpenSearch clúster y la función Lambda que cree deben estar ubicados en la misma VPC que el clúster de base de datos de Neptune, y el clúster OpenSearch debe estar configurado en modo VPC (no en modo Internet).

Le recomendamos que utilice una instancia de Neptune recién creada para usarla con OpenSearch Service. Si utilizas una instancia existente que ya contiene datos, debes realizar una sincronización de datos del OpenSearch Servicio antes de realizar consultas o podría haber inconsistencias en los datos. Este GitHub proyecto proporciona un ejemplo de cómo realizar la sincronización: Export Neptune to OpenSearch (https://github.com/awslabs/ amazon-neptune-tools export-neptune-to-elasticsearch /tree/master/).

> **Important**
>
> Al integrarse con Amazon OpenSearch Service, Neptune requiere la versión 7.1 o superior de Elasticsearch y funciona con las versiones OpenSearch 2.3, 2.5 y futuras de Opensearch compatibles.

> **Note**
>
> A partir de la versión 1.3.0.0 del motor, Amazon Neptune admite el uso de OpenSearch Amazon Service Serverless para la búsqueda de texto completo en consultas de Gremlin y SPARQL.

Temas

Uso de una AWS CloudFormation plantilla para iniciar la replicación de Neptuno a OpenSearch

Lanza una AWS CloudFormation pila específica para tu región

Cada una de las AWS CloudFormation plantillas siguientes crea una instancia de aplicación streams-consumer en una región específica AWS . Para lanzar la pila correspondiente mediante la AWS CloudFormation consola, selecciona uno de los botones de lanzar la pila de la siguiente tabla, en función de la AWS región que quieras usar.

Región	Visualización	Ver en Designer	Lanzar
Este de EE. UU. (Norte de Virginia)	Ver	Ver en Designer	Launch Stack
Este de EE. UU. (Ohio)	Ver	Ver en Designer	Launch Stack
Oeste de EE. UU. (Norte de California)	Ver	Ver en Designer	Launch Stack
Oeste de EE. UU. (Oregón)	Ver	Ver en Designer	Launch Stack

Región	Visualización	Ver en Designer	Lanzar
Canadá (centro)	Ver	Ver en Designer	
América del Sur (São Paulo)	Ver	Ver en Designer	
Europa (Estocolmo)	Ver	Ver en Designer	
Europa (Irlanda)	Ver	Ver en Designer	
Europa (Londres)	Ver	Ver en Designer	
Europa (París)	Ver	Ver en Designer	
Europa (Fráncfort)	Ver	Ver en Designer	
Medio Oriente (Baréin)	Ver	Ver en Designer	
Medio Oriente (EAU)	Ver	Ver en Designer	
Israel (Tel Aviv)	Ver	Ver en Designer	
África (Ciudad del Cabo)	Ver	Ver en Designer	
Asia-Pacífico (Hong Kong)	Ver	Ver en Designer	
Asia-Pacífico (Tokio)	Ver	Ver en Designer	
Asia-Pacífico (Seúl)	Ver	Ver en Designer	

Región	Visualización	Ver en Designer	Lanzar
Asia-Pacífico (Singapur)	Ver	Ver en Designer	
Asia-Pacífico (Bombay)	Ver	Ver en Designer	
China (Pekín)	Ver	Ver en Designer	
China (Ningxia)	Ver	Ver en Designer	
AWS GovCloud (EE. UU.-Oeste)	Ver	Ver en Designer	
AWS GovCloud (EE. UU.-Este)	Ver	Ver en Designer	

En la página Create Stack (Crear pila), elija Next (Siguiente).

Agrega detalles sobre la nueva OpenSearch pila que estás creando

En la página Especificar los detalles de la pila, se proporcionan las propiedades y los parámetros que puede utilizar para controlar la configuración de la búsqueda de texto completo:

Nombre de la pila: el nombre de la nueva AWS CloudFormation pila que está creando. Por lo general puede utilizar el valor predeterminado, `NeptuneStreamPoller`.

En Parámetros, proporcione lo siguiente:

Configuración de red para la VPC en la que se ejecuta el consumidor de flujos

- **`VPC`**: proporcione el nombre de la VPC en la que se ejecutará la función de Lambda de sondeo.
- **`List of Subnet IDs`**: las subredes en las que se establece una interfaz de red. Añada las subredes correspondientes a su clúster de Neptune.
- **`List of Security Group Ids`**: proporcione los identificadores de los grupos de seguridad que otorgan acceso de escritura entrante al clúster de base de datos de Neptune de origen.

- **`List of Route Table Ids`**: esto es necesario para crear un punto de conexión de Amazon DynamoDB en su VPC de Neptune, si todavía no dispone de uno. Debe proporcionar una lista separada por comas de ID de tabla de ruta asociados a las subredes.
- **`Require to create Dynamo DB VPC Endpoint`**: un valor booleano que, de forma predeterminada, es `true`. Solo necesita cambiarlo por `false` si ya ha creado un punto de conexión de DynamoDB en su VPC.
- **`Require to create Monitoring VPC Endpoint`**: un valor booleano que, de forma predeterminada, es `true`. Solo necesita cambiarlo por `false` si ya ha creado un punto de enlace de monitorización en su VPC.

Sondeador de flujo

- **`Application Name`**: por lo general, puede dejar esta opción con el valor predeterminado (`NeptuneStream`). Si utiliza un nombre diferente, debe ser único.
- **`Memory size for Lambda Poller`**: se utiliza para establecer el tamaño de memoria disponible para la función de sondeo de Lambda. El valor predeterminado es 2048 megabytes.
- **`Lambda Runtime`**: el lenguaje utilizado en la función de Lambda que recupera elementos del flujo de Neptune. Puede configurarlo en `python3.9` o en `java8`.
- **`S3 Bucket having Lambda code artifacts`**: deje este campo en blanco a menos que utilice una función de sondeo de Lambda personalizada que se cargue desde un bucket de S3 diferente.
- **`S3 Key corresponding to Lambda Code artifacts`**: deje este campo en blanco a menos que utilice una función de sondeo de Lambda personalizada.
- **`StartingCheckpoint`**: el punto de control inicial del sondeador de flujos. El valor predeterminado es `0:0`, lo que significa comenzar desde el principio del flujo de Neptune.
- **`StreamPollerInitialState`**: el estado inicial del sondeador. El valor predeterminado es `ENABLED`, lo que significa que la replicación del flujo se iniciará en cuanto se complete la creación de toda la pila.
- **`Logging level for Lambda`**: en general, deje esta opción con el valor predeterminado, `INFO`.
- **`Managed Policies for Lambda Execution`**: en general, deje este campo en blanco a menos que utilice una función de sondeo de Lambda personalizada.
- **`Stream Records Handler`**: en general, deje este campo en blanco a menos que utilice un controlador personalizado para los registros en los flujos de Neptune.

- **`Maximum records Fetched from Stream`**: puede utilizar este parámetro para ajustar el rendimiento. El valor predeterminado (`100`) es un buen lugar para empezar. El valor máximo permitido es 10 000. Cuanto mayor sea el número, menos llamadas de red se necesitan para leer registros del flujo, pero más memoria se precisa para procesar los registros.
- **`Max wait time between two Polls (in Seconds)`**: determina la frecuencia con la que se invoca el sondeador de Lambda para sondear los flujos de Neptune. Establezca este valor en 0 para un sondeo continuo. El valor máximo es de 3600 segundos (1 hora). El valor predeterminado (60 segundos) es un buen lugar para empezar según la velocidad con la que cambien los datos del gráfico.
- **`Maximum Continuous polling period (in Seconds)`**: se utiliza para establecer un tiempo de espera para la función de sondeo de Lambda. Debería estar entre 5 segundos y 900 segundos. El valor predeterminado (600 segundos) es un buen lugar para empezar.
- **`Step Function Fallback Period`**— El número de step-function-fallback-period unidades que esperarán al sondeador, tras lo cual se invoca la función step a través de Amazon CloudWatch Events para recuperarse de un fallo. El valor predeterminado (5 minutos) es un buen lugar para empezar.
- **`Step Function Fallback Period Unit`**: las unidades de tiempo utilizadas para medir el `Step Function Fallback Period` anterior (minutos, horas, días). El valor predeterminado generalmente es suficiente (minutos).
- **`Data replication scope`**— Determina si se debe replicar tanto en los nodos como en los bordes, o solo en los nodos OpenSearch (esto se aplica únicamente a los datos del motor Gremlin). El valor predeterminado (All) es un buen lugar para empezar por lo general.
- **`Ignore OpenSearch missing document error`**— Marca para determinar si se OpenSearch puede ignorar un error en un documento faltante. Los errores de falta de documentos se producen en raras ocasiones, pero necesitan intervención manual si no se ignoran. El valor predeterminado (`True`) suele ser un buen lugar para empezar.
- **`Enable Non-String Indexing`**: indicador para habilitar o deshabilitar la indexación de campos que no tienen contenido de cadena. Si este indicador está establecido en `true` OpenSearch, los campos que no son cadenas se indexan o si `false` solo se indexan los campos de cadena. El valor predeterminado es `true`.
- **`Properties to exclude from being inserted into OpenSearch`**— Una lista delimitada por comas de claves de propiedades o predicados para excluirlas de la indexación. OpenSearch Si el valor de este parámetro de CFN se deja en blanco, se indexan todas las claves de propiedad.

- **`Datatypes to exclude from being inserted into OpenSearch`**— Una lista delimitada por comas de tipos de datos de propiedades o predicados para excluirlos de la indexación. OpenSearch Si el valor de este parámetro CFN se deja en blanco, se indexan todos los valores de propiedad que se puedan convertir de forma segura en tipos de datos. OpenSearch

Flujo de Neptune

- **`Endpoint of source Neptune Stream`**: (obligatorio) adopta una de estas dos formas:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (o su alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - `https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`
- **`Neptune Query Engine`**: elija Gremlin o SPARQL.
- **`Is IAM Auth Enabled?`**: si su clúster de base de datos de Neptune usa la autenticación de IAM, establezca este parámetro como `true`.
- **`Neptune Cluster Resource Id`**: si su clúster de base de datos de Neptune usa la autenticación de IAM, establezca este parámetro como el identificador de recurso de clúster. El ID del recurso no es el mismo que el ID del clúster. En su lugar, adopta el formato: `cluster-` seguido de 28 caracteres alfanuméricos. Se puede encontrar en Detalles del clúster en la consola de Neptune.

Clúster de destino OpenSearch

- **`Endpoint for OpenSearch service`**— (Obligatorio) Proporcione el punto final del OpenSearch servicio en su VPC.
- **`Number of Shards for OpenSearch Index`**: el valor predeterminado (5) suele ser un buen lugar para empezar.
- **`Number of Replicas for OpenSearch Index`**: el valor predeterminado (1) suele ser un buen lugar para empezar.
- **`Geo Location Fields for Mapping`**: si utiliza campos de geolocalización, enumere las claves de propiedad aquí.

Alarma

- **`Require to create Cloud watch Alarm`**— Configúrelo en `true` si quiere crear una CloudWatch alarma para la nueva pila.

- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— El ARN del tema SNS al que CloudWatch se deben enviar las notificaciones de alarma (solo es necesario si las alarmas están habilitadas).
- **`Email for Alarm Notifications`**: la dirección de correo electrónico a la que se deben enviar las notificaciones de alarma (solo es necesaria si las alarmas están activadas).

Como destino de la notificación de alarma, puede añadir solo SNS, solo correo electrónico o tanto SNS como correo electrónico.

Ejecute la plantilla AWS CloudFormation

Ahora puede completar el proceso de aprovisionamiento de una instancia de aplicación de consumidor de flujos de Neptune de la siguiente manera:

1. En AWS CloudFormation, en la página Especificar los detalles de la pila, seleccione Siguiente.
2. En la página Opciones, seleccione Siguiente.
3. En la página Revisar, seleccione la primera casilla para confirmar que AWS CloudFormation debe crear los recursos de IAM. Seleccione la segunda casilla para confirmar `CAPABILITY_AUTO_EXPAND` para la nueva pila.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` confirma explícitamente que las macros se expandirán al crear la pila, sin revisión previa. Los usuarios suelen crear un conjunto de cambios a partir de una plantilla procesada para que los cambios realizados por las macros puedan revisarse antes de crear la pila. Para obtener más información, consulte el funcionamiento de la AWS CloudFormation CreateStackAPI en la referencia de la AWS CloudFormation API.

   A continuación, seleccione Crear.

Habilitar la búsqueda de texto completo en las bases de datos de Neptune existentes

Si puede pausar sus cargas de trabajo de escritura

La mejor forma de habilitar la búsqueda de texto completo en una base de datos de Neptune existente suele ser la siguiente, siempre que pueda pausar las cargas de trabajo de escritura. Para ello, es necesario crear un clon, habilitar los flujos mediante un parámetro de clúster y reiniciar todas las instancias. La creación de un clon es una operación relativamente rápida, por lo que el tiempo de inactividad necesario es limitado.

Estos son los pasos necesarios:

1. Detenga todas las cargas de trabajo de escritura en la base de datos.
2. Habilite los flujos en la base de datos (consulte Enabling Neptune Streams).
3. Cree un clon de la base de datos (consulte Database Cloning in Neptune).
4. Reanude las cargas de trabajo de escritura.
5. Usa la export-neptune-to-elasticsearchherramienta en GitHub para realizar una sincronización única entre la base de datos clonada y el OpenSearch dominio.
6. Use la plantilla de AWS CloudFormation de su región para iniciar la sincronización desde su base de datos original con una actualización continua (no es necesario cambiar la configuración de la plantilla).
7. Elimina la base de datos clonada y la AWS CloudFormation pila creada para la `export-neptune-to-elasticsearch` herramienta.

Si no puede pausar sus cargas de trabajo de escritura

Si no puede permitirse suspender las cargas de trabajo de escritura en su base de datos, aquí tiene un enfoque que requiere incluso menos tiempo de inactividad que el enfoque recomendado anteriormente, pero debe hacerse con cuidado:

1. Habilite los flujos en la base de datos (consulte Enabling Neptune Streams).
2. Cree un clon de la base de datos (consulte Database Cloning in Neptune).
3. Obtenga el último `eventID` de los flujos de la base de datos clonada mediante la ejecución de un comando de este tipo en el punto de conexión de la API de flujos (consulte Calling the Neptune Streams REST API para obtener más información):

```
curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
iteratorType=LATEST"
```

Anote los valores de los campos `commitNum` y `opNum` del objeto `lastEventId` en la respuesta.

4. Usa la export-neptune-to-elasticsearchherramienta en GitHub para realizar una sincronización única de la base de datos clonada al OpenSearch dominio.
5. Use la plantilla de AWS CloudFormation de su región para iniciar la sincronización desde su base de datos original con una actualización continua.

   Realice el siguiente cambio al crear la pila: en la página de detalles de la pila, en la sección Parámetros, establezca el valor del campo `StartingCheckpoint` en *`commitNum`*:*`opnum`* utilizando los valores `commitNum` y `opNum` que registró anteriormente.
6. Elimina la base de datos clonada y la AWS CloudFormation pila creada para la `export-neptune-to-elasticsearch` herramienta.

Actualización del sondeador de flujos

Para actualizar el sondeador de flujos con los artefactos de Lambda más recientes

Puede actualizar el sondeador de flujos con los artefactos de Lambda más recientes de la siguiente manera:

1. En AWS Management Console, navegue hasta la AWS CloudFormation pila principal principal AWS CloudFormation y selecciónela.
2. Seleccione la opción Actualizar para la pila.
3. Seleccione Reemplazar la plantilla actual.
4. Para la fuente de la plantilla, elija la URL de Amazon S3 e introduzca la siguiente URL de S3:

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_elastic_search.json
   ```
5. Seleccione Siguiente sin cambiar ningún AWS CloudFormation parámetro.
6. Elija Update Stack (Actualizar pilar).

La pila actualizará ahora los artefactos de Lambda con los más recientes.

Ampliación del sondeador de flujos para admitir campos personalizados

El sondeador de flujos actual se puede ampliar fácilmente para escribir código personalizado para gestionar campos personalizados, como se explica en detalle en esta entrada del blog: Capture graph changes using Neptune Streams.

> **Note**
>
> Al añadir un campo personalizado OpenSearch, asegúrese de añadir el nuevo campo como objeto interno de un predicado (consulteModelo de datos de búsqueda de texto completo de Neptune).

Habilitación y nueva habilitación del proceso del sondeador de flujos

> **Warning**
>
> Tenga cuidado al deshabilitar el proceso de sondeador de flujos. Se podrían perder datos si el proceso se detiene durante más tiempo que el período de caducidad del flujo. El período predeterminado es de 7 días, pero a partir de la versión 1.2.0.0 del motor, puede configurar un período de caducidad del flujo personalizado hasta un máximo de 90 días.

Deshabilitación (pausa) del proceso del sondeador de flujos

1. Inicia sesión en la EventBridge consola de Amazon AWS Management Console y ábrela en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Reglas.
3. Selecciona la regla cuyo nombre contenga el nombre que proporcionaste como nombre de la aplicación en la AWS CloudFormation plantilla que utilizaste para configurar el sondeador de transmisiones.
4. Elija Deshabilitar.
5. Abra la consola de Step Functions en https://console.aws.amazon.com/states/.
6. Seleccione la función de paso en ejecución que corresponda al proceso del sondeador de flujos. De nuevo, el nombre de esa función de paso contiene el nombre que proporcionó como nombre de la aplicación en la AWS CloudFormation plantilla que utilizó para configurar el sondeador

de flujos. Puede filtrar por estado de ejecución de la función para ver solo las funciones En ejecución.

7. Elija Detener.

Nueva habilitación del proceso del sondeador de flujos

1. Inicia sesión en la EventBridge consola de Amazon AWS Management Console y ábrela en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Reglas.
3. Selecciona la regla cuyo nombre contenga el nombre que proporcionaste como nombre de la aplicación en la AWS CloudFormation plantilla que utilizaste para configurar el sondeador de transmisiones.
4. Elija Deshabilitar. La regla de eventos basada en el intervalo programado especificado activará ahora una nueva ejecución de la función de paso.

Replicación a OpenSearch Serverless

A partir de la versión 1.3.0.0 del motor, Amazon Neptune admite el uso de Amazon OpenSearch Service Serverless para la búsqueda de texto completo en consultas de Gremlin y SPARQL.

Si va a replicar en OpenSearch Serverless, añada el rol de ejecución del sondeador de flujo de Lambda a la política de acceso a datos de la colección de OpenSearch Serverless. El ARN de la rol de ejecución del sondeador de flujo de Lambda tiene este formato:

```
arn:aws:iam::(account ID):role/stack-name-NeptuneOSReplication-
NeptuneStreamPollerExecu-(uuid)
```

Para obtener más información, consulte Control de acceso a datos de Amazon OpenSearch sin servidor.

Si ha habilitado el control de acceso detallado en el clúster de OpenSearch, también debe habilitar la autenticación de IAM en la base de datos de Neptune.

La entidad de IAM (usuario o rol) utilizada para conectarse a la base de datos de Neptune debe tener permisos tanto para Neptune como para la recopilación de OpenSearch Serverless . Esto significa que su usuario o rol deben tener asociada una política de OpenSearch Serverless como la siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::(account ID):root"
      },
      "Action": "aoss:APIAccessAll",
      "Resource": "arn:aws:aoss:(region):(account ID):collection/(collection ID)"
    }
  ]
}
```

Para obtener más información, consulte Declaraciones de políticas de acceso a datos de IAM personalizadas para Amazon Neptune.

Consultas desde un clúster de OpenSearch con el control de acceso detallado (FGAC) habilitado

Si ha habilitado el control de acceso detallado en el clúster de OpenSearch, también debe habilitar la autenticación de IAM en la base de datos de Neptune.

La entidad de IAM (usuario o rol) utilizada para conectarse a la base de datos de Neptune debe tener permisos tanto para Neptune como para el clúster de OpenSearch. Esto significa que su usuario o rol deben tener asociada una política de OpenSearch Service como la siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-id:root"
      },
      "Action": "es:*",
      "Resource": "arn:aws:es:region:account-id:es-resource-id/*"
    }
  ]
}
```

Para obtener más información, consulte Declaraciones de políticas de acceso a datos de IAM personalizadas para Amazon Neptune.

Uso de la sintaxis de consulta de Apache Lucene en las consultas de búsqueda de texto completo de Neptune

OpenSearch admite el uso de la sintaxis de Apache Lucene para las consultas query_string. Esto resulta especialmente útil para pasar varios filtros en una consulta.

Neptune utiliza una estructura anidada para almacenar propiedades en un documento de OpenSearch (consulte Modelo de datos de búsqueda de texto completo de Neptune). Al utilizar la sintaxis de Lucene, es necesario utilizar rutas completas a las propiedades de este modelo anidado.

Aquí tiene un ejemplo de Gremlin:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V()
 .has("*", "Neptune#fts predicates.name.value:\"Jane Austin\" AND entity_type:Book")
```

Aquí tiene un ejemplo de SPARQL:

```
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
  neptune-fts:config neptune-fts:endpoint 'http://localhost:9200 (http://
localhost:9200/)' .
  neptune-fts:config neptune-fts:queryType 'query_string' .
  neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*name.value:Ronak AND
 predicates.\\*foaf\\*surname.value:Sh*" .
  neptune-fts:config neptune-fts:field '*' .
  neptune-fts:config neptune-fts:return ?res .
}
```

Modelo de datos de Neptune para datos de OpenSearch

Amazon Neptune utiliza una estructura de documento JSON unificada para almacenar datos de SPARQL y Gremlin en OpenSearch Service. Cada documento de OpenSearch corresponde a una entidad y almacena toda la información relevante de esa entidad. En Gremlin, los vértices y los

bordes se consideran entidades, por lo que los documentos correspondientes de OpenSearch tienen información sobre vértices, etiquetas y propiedades. En SPARQL, los sujetos pueden considerarse entidades, por lo que los documentos correspondientes de OpenSearch tienen información sobre todos los pares de predicado-objeto que hay en un documento.

> **Note**
>
> La implementación de la replicación de Neptune a OpenSearch solo almacena datos de cadena. Sin embargo, puede modificarla para almacenar otros tipos de datos.

La estructura de documento JSON unificada tiene el siguiente aspecto.

```
{
  "entity_id": "Vertex Id/Edge Id/Subject URI",
  "entity_type": [List of Labels/rdf:type object value],
  "document_type": "vertex/edge/rdf-resource"
  "predicates": {
    "Property name or predicate URI": [
      {
        "value": "Property Value or Object Value",
        "graph": "(Only for Sparql) Named Graph Quad is present"
        "language": "(Only for Sparql) rdf:langString"
      },
      {
        "value": "Property Value 2/ Object Value 2",
      }
    ]
  }
}
```

- `entity_id`: identificador único de la entidad que representa el documento.
  - Para SPARQL, se trata del URI de asunto.
  - Para Gremlin, es el `Vertex_ID` o `Edge_ID`.
- `entity_type`: representa una o más etiquetas de un vértice o un borde, o cero o más valores de predicado `rdf:type` para un sujeto.
- `document_type`: se utiliza para especificar si el documento actual representa un vértice, un borde o un recurso RDF.

- `predicates`: en el caso de Gremlin, almacena las propiedades y los valores de un vértice o un borde. Para SPARQL, almacena pares de predicado-objeto.

  El nombre de la propiedad tiene el formato `properties.name.value` de OpenSearch. Para consultarlo, debe nombrarlo de esa forma.
- `value` : un valor de propiedad para Gremlin o un valor de objeto para SPARQL.
- `graph`: un gráfico con nombre para SPARQL.
- `language`: una etiqueta de idioma para un literal `rdf:langString` en SPARQL.

Ejemplo de documento de OpenSearch de SPARQL

Datos

```
@prefix dt:   <http://example.org/datatype#> .
@prefix ex:   <http://example.org/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:simone   rdf:type     ex:Person                    ex:g1
ex:michael  rdf:type     ex:Person                    ex:g1
ex:simone   ex:likes     "spaghetti"                  ex:g1

ex:simone   ex:knows     ex:michael                   ex:g2   # Not stored in ES
ex:simone   ex:likes     "spaghetti"                  ex:g2
ex:simone   ex:status    "La vita è un sogno"@it      ex:g2

ex:simone   ex:age       "40"^^xsd:int                DG      # Not stored in ES
ex:simone   ex:dummy     "testData"^^dt:newDataType   DG
ex:simone   ex:hates     _:bnode                              # Not stored in ES
_:bnode     ex:means     "coding"                     DG      # Not stored in ES
```

Documentos

```
{
  "entity_id": "http://example.org/simone",
  "entity_type": ["http://example.org/Person"],
  "document_type": "rdf-resource"
  "predicates": {
    "http://example.org/likes": [
      {
```

```
        "value": "spaghetti",
        "graph": "http://example.org/g1"
      },
      {
        "value": "spaghetti",
        "graph": "http://example.org/g2"
      }
    ]
    "http://example.org/status": [
      {
        "value": "La vita è un sogno",
        "language": "it"       // Only present for rdf:langString
      }
    ]
  }
}
```

```
{
  "entity_id" : "http://example.org/michael",
  "entity_type" : ["http://example.org/Person"],
  "document_type": "rdf-resource"
}
```

Ejemplo de documento de OpenSearch de Gremlin

Datos

```
# Vertex 1
simone   label    Person       <== Label
simone   likes    "spaghetti"  <== Property
simone   likes    "rice"       <== Property
simone   age      40           <== Property

# Vertex 2
michael  label    Person       <== Label

# Edge 1
simone  knows     michael      <== Edge
e1      updated  "2019-07-03"  <== Edge Property
e1      through  "company"     <== Edge Property
e1      since     10           <== Edge Property
```

Documentos

```
{
  "entity_id": "simone",
  "entity_type": ["Person"],
  "document_type": "vertex",
  "predicates": {
    "likes": [
      {
        "value": "spaghetti"
      },
      {
        "value": "rice"
      }
    ]
  }
}
```

```
{
  "entity_id" : "michael",
  "entity_type" : ["Person"],
  "document_type": "vertex"
}
```

```
{
  "entity_id": "e1",
  "entity_type": ["knows"],
  "document_type": "edge"
  "predicates": {
    "through": [
      {
        "value": "company"
      }
    ]
  }
}
```

Parámetros de búsqueda de texto completo de Neptune

Amazon Neptune utiliza los siguientes parámetros para especificar las consultas de OpenSearch de texto completo tanto en Gremlin como en SPARQL:

- **`queryType`**: (obligatorio) el tipo de consulta de OpenSearch. (Para obtener una lista de los tipos de consultas, consulte la documentación de OpenSearch). Neptune admite los siguientes tipos de consultas de OpenSearch:
  - simple_query_string: devuelve los documentos en función de la cadena de consulta proporcionada, mediante un analizador con una sintaxis de Lucene limitada pero tolerante a errores. Este es el tipo de consulta predeterminado.

    Esta consulta utiliza una sintaxis simple para analizar y dividir la cadena de consulta proporcionada en términos basados en operadores especiales. A continuación, la consulta analiza cada término de forma independiente antes de devolver documentos coincidentes.

    Aunque su sintaxis está más limitada que la consulta `query_string`, la consulta `simple_query_string` no devuelve errores para la sintaxis no válida. En su lugar, no tiene en cuenta cualquier parte no válida de la cadena de consulta.
  - match: la consulta `match` es la consulta estándar para realizar una búsqueda de texto completo, e incluye opciones de coincidencia parcial.
  - prefix: devuelve los documentos que contienen un prefijo específico en un campo proporcionado.
  - fuzzy: devuelve documentos que contienen términos similares al término de búsqueda, medidos por medio de una distancia de edición de Levenshtein.

    Una distancia de edición es el número de cambios de un carácter necesarios para convertir un término en otro. Estos cambios pueden incluir:
    - Cambio de un carácter (box a fox).
    - Eliminación de un carácter (black a lack).
    - Inserción de un carácter (sic a sick).
    - Transposición de dos caracteres adyacentes (act a cat).

    Para encontrar términos similares, la consulta aproximada crea un conjunto de todas las variaciones y expansiones posibles del término de búsqueda dentro de una distancia de edición especificada y, a continuación, devuelve coincidencias exactas para cada una de esas variantes.
  - term: devuelve documentos que contienen una coincidencia exacta de un término especificado en uno de los campos especificados.

    Puede utilizar la consulta `term` para buscar documentos basados en un valor preciso, como un precio, el ID de un producto o un nombre de usuario.

> **Warning**
>
> Evite usar el término consulta para los campos de texto. De forma predeterminada, OpenSearch cambia los valores de los campos de texto como parte de su análisis, lo que puede dificultar la búsqueda de coincidencias exactas para los valores de los campos de texto.
> Para buscar valores de campo de texto, utilice la consulta de coincidencia en su lugar.

- query_string: devuelve los documentos en función de la cadena de consulta proporcionada, mediante un analizador con una sintaxis estricta (sintaxis de Lucene).

  Esta consulta utiliza una sintaxis para analizar y dividir la cadena de consulta proporcionada basadas en operadores, como Y y NO. A continuación, la consulta analiza cada texto dividido de forma independiente antes de devolver documentos coincidentes.

  Puede utilizar la consulta `query_string` para crear una búsqueda compleja que incluya caracteres comodín, búsquedas en varios campos y mucho más. Aunque versátil, la consulta es estricta y devuelve un error si la cadena de consulta incluye cualquier sintaxis no válida.

  > **Warning**
  >
  > Dado que devuelve un error para cualquier sintaxis no válida, no recomendamos usar la consulta `query_string` para los cuadros de búsqueda.
  > Si no tiene que admitir una sintaxis de consulta, considere el uso de la consulta `match`. Si necesita las características de una sintaxis de consulta, utilice la consulta `simple_query_string`, que es menos estricta.

- **`field`**: el campo de OpenSearch en el que se va a ejecutar la búsqueda. Solo se puede omitir si `queryType` lo permite (como hacen `simple_query_string` y `query_string`), en cuyo caso la búsqueda se efectúa en todos los campos. En Gremlin, está implícito.

  Se pueden especificar varios campos si la consulta lo permite, como hacen `simple_query_string` y `query_string`.

- **`query`**: (obligatorio) la consulta que se va a ejecutar en OpenSearch. El contenido de este campo puede variar según el valor de queryType. Los distintos valores de queryType aceptan diferentes sintaxis, como, por ejemplo, hace `Regexp`. En Gremlin, `query` está implícito.

- **`maxResults`**: el número máximo de resultados que se deben devolver. El valor predeterminado es la configuración `index.max_result_window` de OpenSearch, que de manera predeterminada es 10 000. El parámetro `maxResults` puede especificar cualquier número menor que eso.

  > **Important**
  >
  > Si establece `maxResults` en un valor mayor que el valor de `index.max_result_window` de OpenSearch e intenta recuperar más de `index.max_result_window` resultados, OpenSearch produce un error `Result window is too large`. Sin embargo, Neptune maneja esto con fluidez sin propagar el error. Tenga esto en cuenta si trata de obtener más resultados de `index.max_result_window`.

- **`minScore`**: la puntuación mínima que debe tener un resultado de búsqueda para ser devuelto. Consulte OpenSearch relevance documentation para obtener una explicación de la puntuación de los resultados.
- **`batchSize`**: Neptune siempre obtiene los datos por lotes (el tamaño de lote predeterminado es 100). Puede utilizar este parámetro para ajustar el rendimiento. El tamaño del lote no puede superar la configuración de `index.max_result_window` de OpenSearch, que de manera predeterminada es 10 000.
- **`sortBy`**: un parámetro opcional que permite ordenar los resultados que devuelve OpenSearch según una de las siguientes opciones:
  - Un campo de cadena concreto del documento:

    Por ejemplo, en una consulta SPARQL, puede especificar:

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name .
    ```

    En una consulta Gremlin similar, puede especificar:

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name')
    ```

  - Un campo concreto que no es una cadena (*`long`*, *`double`*, etc.) del documento:

    Tenga en cuenta que, al ordenar por un campo que no sea una cadena, debe agregar `.value` al nombre del campo para diferenciarlo de un campo de cadena.

    Por ejemplo, en una consulta SPARQL, puede especificar:

```
    neptune-fts:config neptune-fts:sortBy foaf:name.value .
```

En una consulta Gremlin similar, puede especificar:

```
    .withSideEffect('Neptune#fts.sortBy', 'name.value')
```

- `score`: ordena por puntuación de coincidencia (valor predeterminado).

  Si el parámetro `sortOrder` está presente pero `sortBy` no está presente, los resultados se ordenan por `score` en el orden especificado por `sortOrder`.

- `id`: ordena por identificador, es decir, el URI del sujeto de SPARQL o el identificador de vértice o borde de Gremlin.

  Por ejemplo, en una consulta SPARQL, puede especificar:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
  ```

  En una consulta Gremlin similar, puede especificar:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
  ```

- `label`: ordena por etiqueta.

  Por ejemplo, en una consulta SPARQL, puede especificar:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
  ```

  En una consulta Gremlin similar, puede especificar:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
  ```

- `doc_type`: ordena por tipo de documento (es decir, SPARQL o Gremlin).

  Por ejemplo, en una consulta SPARQL, puede especificar:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
  ```

  En una consulta Gremlin similar, puede especificar:

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
```

De forma predeterminada, los resultados de OpenSearch no se ordenan y su orden no es determinista, lo que significa que la misma consulta puede devolver elementos con un orden diferente cada vez que se ejecuta. Por esta razón, si el conjunto de resultados es mayor que `max_result_window`, se podría devolver un subconjunto bastante diferente de los resultados totales cada vez que se ejecuta una consulta. Al ordenar, sin embargo, puede hacer que los resultados de diferentes ejecuciones sean más equivalentes de forma directa.

Si ningún parámetro `sortOrder` acompaña a `sortBy`, se utiliza el orden descendente (`DESC`) de mayor a menor.

- **`sortOrder`**: un parámetro opcional que permite especificar si los resultados de OpenSearch se ordenan de menor a mayor o de mayor a menor (opción predeterminada):

  - `ASC`: orden ascendente, de menor a mayor.
  - `DESC`: orden descendente, de mayor a menor.

    Este es el valor predeterminado, que se utiliza cuando el parámetro `sortBy` está presente pero no se especifica `sortOrder`.

  Si no están presentes `sortBy` ni `sortOrder`, los resultados de OpenSearch no se ordenan de forma predeterminada.

Indexación de OpenSearch sin cadenas en Amazon Neptune

La indexación de OpenSearch sin cadenas en Amazon Neptune permite replicar en OpenSearch valores que no son cadenas para predicados mediante el sondeador de flujos. Todos los valores de predicados que se puedan convertir de forma segura en un mapeo o tipo de datos de OpenSearch correspondiente se replican en OpenSearch.

Para habilitar la indexación sin cadenas en una pila nueva, el indicador `Enable Non-String Indexing` de la plantilla AWS CloudFormation debe estar establecido en `true`. Este es el valor predeterminado. Para actualizar una pila existente para que sea compatible con la indexación sin cadenas, consulte Actualización de una pila existente a continuación.

> **Note**
>
> - Es mejor no habilitar la indexación sin cadenas en las versiones del motor anteriores a **`1.0.4.2`**.
> - Las consultas de OpenSearch que utilizan expresiones regulares para nombres de campo que coinciden con varios campos, algunos de los cuales contienen valores de cadena y otros contienen valores que sin cadenas, fallan y se produce un error. Lo mismo ocurre si las consultas de búsqueda de texto completo en Neptune son de ese tipo.
> - Al ordenar por un campo sin cadenas, añada ".value" al nombre del campo para diferenciarlo de un campo de cadena.

Contenido

Actualización de una pila de búsqueda de texto completo de Neptune existente para admitir la indexación sin cadenas

Si ya utiliza la búsqueda de texto completo en Neptune, estos son los pasos que debe seguir para admitir la indexación sin cadenas:

1. Detenga la función de Lambda del sondeador de flujo. Esto garantiza que no se copien nuevas actualizaciones durante la exportación. Para ello, deshabilite la regla de eventos de nube que invoca la función de Lambda:

   - En la AWS Management Console, vaya a CloudWatch.
   - Seleccione Reglas.
   - Elija la regla con el nombre del sondeador de flujo de Lambda.
   - Seleccione Deshabilitar para deshabilitar temporalmente la regla.

2. Elimine el índice de Neptune actual en OpenSearch. Utilice la siguiente consulta `curl` para eliminar el índice `amazon_neptune` del clúster de OpenSearch:

   ```
   curl -X DELETE "your OpenSearch endpoint/amazon_neptune"
   ```

3. Inicie una exportación puntual de Neptune a OpenSearch. Lo mejor es configurar una nueva pila de OpenSearch en este momento, de forma que el buscador que realice la exportación pueda obtener los nuevos artefactos.

   Siga los pasos que se indican aquí en GitHub para iniciar la exportación puntual de sus datos de Neptune a OpenSearch.

4. Actualice los artefactos de Lambda para el sondeador de flujos existente. Una vez que la exportación de los datos de Neptune a OpenSearch se haya realizado correctamente, siga estos pasos:

   - En AWS Management Console, vaya a AWS CloudFormation.
   - Elija la pila AWS CloudFormation principal.
   - Seleccione la opción Actualizar para esa pila.
   - Seleccione Reemplazar la plantilla actual desde opciones.
   - En el origen de la plantilla, seleccione URL de Amazon S3.
   - Para la URL de Amazon S3, introduzca:

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_elastic_search.json
```

- Seleccione Siguiente sin cambiar ningún parámetro de AWS CloudFormation.
- Seleccione Actualizar pila. AWS CloudFormation sustituirá a los artefactos del código de Lambda del sondeador de flujos por los artefactos más recientes.

5. Vuelva a iniciar el sondeador de flujo. Para ello, habilite la regla de CloudWatch adecuada:

- En la AWS Management Console, vaya a CloudWatch.
- Seleccione Reglas.
- Elija la regla con el nombre del sondeador de flujo de Lambda.
- Seleccione Habilitar.

Filtrado de los campos que se indexan en la búsqueda de texto completo de Neptune

Hay dos campos en los detalles de la plantilla de AWS CloudFormation que permiten especificar las claves de propiedades o predicados o los tipos de datos que se van a excluir de la indexación de OpenSearch:

Filtre por propiedad o nombre de predicado

Puede usar el parámetro de plantilla de AWS CloudFormation opcional denominado `Properties to exclude from being inserted into Elastic Search Index` para proporcionar una lista delimitada por comas de claves de predicados o propiedades que desee excluir de la indexación de OpenSearch.

Por ejemplo, supongamos que establece este parámetro en bob.

```
"Properties to exclude from being inserted into Elastic Search Index" : bob
```

En ese caso, el registro de flujo de la siguiente consulta de actualización de Gremlin se eliminaría en lugar de pasar al índice:

```
g.V("1").property("bob", "test")
```

Del mismo modo, puede establecer el parámetro en `http://my/example#bob`:

```
"Properties to exclude from being inserted into Elastic Search Index" : http://my/
example#bob
```

En ese caso, el registro de flujo de la siguiente consulta de actualización de SPARQL se eliminaría en lugar de pasar al índice:

```
PREFIX ex: <http://my/example#>
INSERT DATA { ex:s1 ex:bob "test"}.
```

Si no introduce nada en este parámetro de plantilla de AWS CloudFormation, se indexarán todas las claves de propiedad que no se excluyan de algún modo.

Filtre por propiedad o tipo de valor de predicado

Puede usar el parámetro de plantilla de AWS CloudFormation opcional denominado `Datatypes to exclude from being inserted into Elastic Search Index` para proporcionar una lista delimitada por comas de tipos de datos de valores de predicados o propiedades que desee excluir de la indexación de OpenSearch.

En el caso de SPARQL, no es necesario incluir el URI de tipo XSD completo, sino simplemente el token del tipo de datos. Los tokens de tipos de datos válidos que puede enumerar son:

- `string`
- `boolean`
- `float`
- `double`
- `dateTime`
- `date`
- `time`
- `byte`
- `short`
- `int`
- `long`
- `decimal`

- integer
- nonNegativeInteger
- nonPositiveInteger
- negativeInteger
- unsignedByte
- unsignedShort
- unsignedInt
- unsignedLong

En el caso de Gremlin, los tipos de datos válidos que se pueden enumerar son:

- string
- date
- bool
- byte
- short
- int
- long
- float
- double

Por ejemplo, supongamos que establece este parámetro en `string`.

```
"Datatypes to exclude from being inserted into Elastic Search Index" : string
```

En ese caso, el registro de flujo de la siguiente consulta de actualización de Gremlin se eliminaría en lugar de pasar al índice:

```
g.V("1").property("myStringval", "testvalue")
```

Del mismo modo, puede establecer el parámetro en `int`:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : int
```

En ese caso, el registro de flujo de la siguiente consulta de actualización de SPARQL se eliminaría en lugar de pasar al índice:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
INSERT DATA { ex:s1 ex:bob "11"^^xsd:int }.
```

Si no introduce nada en este parámetro de plantilla de AWS CloudFormation, se indexarán todas las propiedades cuyos valores se puedan convertir de forma segura en equivalentes de OpenSearch. Se omiten los tipos enumerados que no son compatibles con el lenguaje de consulta.

Mapeo de tipos de datos SPARQL y Gremlin a OpenSearch

Los nuevos mapeos de tipos de datos en OpenSearch se crean en función del tipo de datos que se utiliza en la propiedad u objeto. Dado que algunos campos contienen valores de distintos tipos, es posible que en el mapeo inicial se excluyan algunos valores del campo.

Los tipos de datos de Neptune se mapean a los tipos de datos de OpenSearch de la siguiente manera:

Tipos de SPARQL	Tipos de Gremlin	Tipos de OpenSearch
`XSD:int`	`byte`	`long`
`XSD:unsignedInt`	`short`	
`XSD:integer`	`int`	
`XSD:byte`	`long`	
`XSD:unsignedByte`		
`XSD:short`		
`XSD:unsignedShort`		
`XSD:long`		
`XSD:unsignedLong`		
`XSD:float`	`float`	`double`

Tipos de SPARQL	Tipos de Gremlin	Tipos de OpenSearch

`XSD:double`
 	
`XSD:decimal`
	`double`	

`XSD:boolean`	`bool`	`boolean`

`XSD:datetime`
 	
`XSD:date`
	`date`	`date`

`XSD:string`
 	
`XSD:time`
	`string`	`text`

Tipo de datos personalizado	N/D	`text`
Cualquier otro tipo de datos	N/D	`text`

Por ejemplo, la siguiente consulta de actualización de Gremlin hace que se añada un nuevo mapeo de "newField" a OpenSearch, es decir, `{ "type" : "double" }`:

```
g.V("1").property("newField" 10.5)
```

Del mismo modo, la siguiente consulta de actualización de SPARQL hace que se añada un nuevo mapeo de "ex:byte" a OpenSearch, es decir, `{ "type" : "long" }`:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

INSERT DATA { ex:test ex:byte "123"^^xsd:byte }.
```

> **Note**
>
> Como puede ver, un elemento mapeado de Neptune a OpenSearch puede terminar con un tipo de datos diferente en OpenSearch al que tiene en Neptune. Sin embargo, hay un campo de texto explícito en OpenSearch, "datatype", que registra el tipo de datos que tiene el elemento en Neptune.

Validación de mapeos de datos

Los datos se replican a OpenSearch desde Neptune mediante este proceso:

- Si ya hay un mapeo del campo en cuestión en OpenSearch:
  - Si los datos se pueden convertir de forma segura al mapeo existente mediante reglas de validación de datos, guarde el campo en OpenSearch.
  - Si no es así, borre el registro de actualización del flujo correspondiente.
- Si no existe ningún mapeo para el campo en cuestión, busque un tipo de datos de OpenSearch que corresponda al tipo de dato del campo en Neptune.
  - Si los datos del campo se pueden convertir de forma segura al tipo de dato de OpenSearch mediante reglas de validación de datos, almacene los nuevos datos de mapeo y campo en OpenSearch.
  - Si no es así, borre el registro de actualización del flujo correspondiente.

Los valores se validan con los tipos de OpenSearch equivalentes o mapeos de OpenSearch existentes en lugar de con los tipos de Neptune. Por ejemplo, la validación del valor `"123"` en `"123"^^xsd:int` se realiza en función del tipo `long` y no del tipo `int`.

Aunque Neptune intenta replicar todos los datos en OpenSearch, hay casos en los que los tipos de datos de OpenSearch son totalmente diferentes de los de Neptune y, en esos casos, los registros se omiten en lugar de indexarse en OpenSearch.

Por ejemplo, en Neptune, una propiedad puede tener varios valores de distintos tipos, mientras que en OpenSearch un campo debe tener el mismo tipo en todo el índice.

Al habilitar los registros de depuración, puede ver qué registros se han eliminado durante la exportación de Neptune a OpenSearch. Este es un ejemplo de entrada en el registro de depuración:

```
Dropping Record : Data type not a valid Gremlin type
<Record>
```

Los tipos de datos se validan de la siguiente manera:

- **`text`**: todos los valores de Neptune se pueden mapear de forma segura a texto en OpenSearch.

- **`long`**: las siguientes reglas para los tipos de datos de Neptune se aplican cuando el tipo de mapeo de OpenSearch es largo (en los ejemplos siguientes, se supone que `"testLong"` tiene un tipo de mapeo `long`):
  - `boolean`: no es válido, no se puede convertir y se elimina el registro de actualización del flujo correspondiente.

    Estos son algunos ejemplos de Gremlin no válidos:

    ```
    "testLong" : true.
    "testLong" : false.
    ```

    Estos son algunos ejemplos de SPARQL no válidos:

    ```
    ":testLong" : "true"^^xsd:boolean
    ":testLong" : "false"^^xsd:boolean
    ```

  - `datetime`: no es válido, no se puede convertir y se elimina el registro de actualización del flujo correspondiente.

    Este es un ejemplo de un valor de Gremlin no válido:

    ```
    ":testLong" :  datetime('2018-11-04T00:00:00').
    ```

    Este es un ejemplo de un valor de SPARQL no válido:

    ```
    ":testLong" : "2016-01-01"^^xsd:date
    ```

  - `float`,`double` o `decimal`: si el valor de Neptune es un número entero que puede caber en 64 bits, es válido y se almacena en OpenSearch como long, pero si tiene una parte de fracción, es un `NaN` o un `INF`, o es mayor que 9.223.372.036.854.775.807 o menor que -9.223.372.036.854.775.808, entonces no es válido y se elimina el registro de actualización de flujo correspondiente.

    Estos son algunos ejemplos de valores de Gremlin válidos:

    ```
    "testLong" :  145.0.
    ":testLong" :  123
    ":testLong" :  -9223372036854775807
    ```

Estos son algunos ejemplos de SPARQL válidos:

```
":testLong" : "145.0"^^xsd:float
":testLong" :  145.0
":testLong" : "145.0"^^xsd:double
":testLong" : "145.0"^^xsd:decimal
":testLong" : "-92233720368547758O7"
```

Estos son algunos ejemplos de Gremlin no válidos:

```
"testLong" :  123.45
":testLong" :  9223372036854775900
```

Estos son algunos ejemplos de SPARQL no válidos:

```
":testLong" :  123.45
":testLong" :  9223372036854775900
":testLong" : "123.45"^^xsd:float
":testLong" : "123.45"^^xsd:double
":testLong" : "123.45"^^xsd:decimal
```

- `string`: si el valor de Neptune es una representación en cadena de un entero que puede estar contenido en un entero de 64 bits, entonces es válido y se convierte en `long` en OpenSearch. Cualquier otro valor de cadena no es válido para un mapeo `long` de Elasticseearch y se elimina el registro de actualización del flujo correspondiente.

  Estos son algunos ejemplos de valores de Gremlin válidos:

  ```
  "testLong" :  "123".
  ":testLong" :  "145.0"
  ":testLong" :  "-9223372036854775807"
  ```

  Estos son algunos ejemplos de SPARQL válidos:

  ```
  ":testLong" : "145.0"^^xsd:string
  ":testLong" : "-92233720368547758O7"^^xsd:string
  ```

  Estos son algunos ejemplos de Gremlin no válidos:

```
"testLong" :  "123.45"
":testLong" :  "9223372036854775900"
":testLong" :  "abc"
```

Estos son algunos ejemplos de SPARQL no válidos:

```
":testLong" : "123.45"^^xsd:string
":testLong" : "abc"
":testLong" : "9223372036854775900"^^xsd:string
```

- **`double`**: si el tipo de mapeo de OpenSearch es `double`, se aplican las siguientes reglas (en este caso, se supone que el campo "testDouble" tiene un mapeo `double` en OpenSearch):
  - `boolean`: no es válido, no se puede convertir y se elimina el registro de actualización del flujo correspondiente.

    Estos son algunos ejemplos de Gremlin no válidos:

    ```
    "testDouble" : true.
    "testDouble" : false.
    ```

    Estos son algunos ejemplos de SPARQL no válidos:

    ```
    ":testDouble" : "true"^^xsd:boolean
    ":testDouble" : "false"^^xsd:boolean
    ```

  - `datetime`: no es válido, no se puede convertir y se elimina el registro de actualización del flujo correspondiente.

    Este es un ejemplo de un valor de Gremlin no válido:

    ```
    ":testDouble" :  datetime('2018-11-04T00:00:00').
    ```

    Este es un ejemplo de un valor de SPARQL no válido:

    ```
    ":testDouble" : "2016-01-01"^^xsd:date
    ```

  - `NaN` o `INF` de punto flotante: si el valor de SPARQL es un `NaN` o `INF` de punto flotante, entonces no es válido y se elimina el registro de actualización del flujo correspondiente.

Estos son algunos ejemplos de SPARQL no válidos:

```
"  :testDouble" : "NaN"^^xsd:float
  ":testDouble" : "NaN"^^double
  ":testDouble" : "INF"^^double
  ":testDouble" : "-INF"^^double
```

- number o cadena numérica: si el valor de Neptune es cualquier otro número o cadena numérica que represente un número que se pueda expresar de forma segura como `double`, entonces es válido y se convierte en `double` en OpenSearch. Cualquier otro valor de cadena no es válido para un mapeo `double` de OpenSearch y se elimina el registro de actualización del flujo correspondiente.

  Estos son algunos ejemplos de valores de Gremlin válidos:

  ```
    "testDouble" :  123
    ":testDouble" :  "123"
    ":testDouble" :  145.67
    ":testDouble" :  "145.67"
  ```

  Estos son algunos ejemplos de SPARQL válidos:

  ```
    ":testDouble" :  123.45
    ":testDouble" :  145.0
    ":testDouble" : "123.45"^^xsd:float
    ":testDouble" : "123.45"^^xsd:double
    ":testDouble" : "123.45"^^xsd:decimal
    ":testDouble" : "123.45"^^xsd:string
  ```

  Este es un ejemplo de un valor de Gremlin no válido:

  ```
    ":testDouble" :  "abc"
  ```

  Estos son algunos ejemplos de SPARQL no válidos:

  ```
    ":testDouble" : "abc"
  ```

- **`date`**: si el tipo de mapeo de OpenSearch es `date`, los valores de `date` y `dateTime` de Neptune son válidos, al igual que cualquier valor de cadena que se pueda analizar correctamente en un formato `dateTime`.

  Estos son algunos ejemplos de valores válidos en Gremlin o SPARQL:

```
Date(2016-01-01)
"2016-01-01" "
2003-09-25T10:49:41"
"2003-09-25T10:49"
"2003-09-25T10"
"20030925T104941-0300"
"20030925T104941"
"2003-Sep-25" "
Sep-25-2003"
"2003.Sep.25"
"2003/09/25"
"2003 Sep 25" "
Wed, July 10, '96"
"Tuesday, April 12, 1952 AD 3:30:42pm PST"
"123"
"-123"
"0"
"-0"
"123.00"
"-123.00"
```

  Estos son ejemplos de valores no válidos:

```
123.45
True
"abc"
```

Ejemplos de consultas de OpenSearch sin cadenas en Neptune

Actualmente, Neptune no admite directamente las consultas de rango de OpenSearch. Sin embargo, puede lograr el mismo efecto con la sintaxis de Lucene y query-type="query_string", como puede ver en las siguientes consultas de ejemplo.

1. Obtenga todos los vértices con una antigüedad superior a 30 años y un nombre que comience por "Si"

En Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:>30 && predicates.name.value:Si*');
```

En SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:>30 AND
 predicates.\\*foaf\\*name.value:Si*" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Aquí, se usa "`\\*foaf\\*age` en lugar del URI completo por motivos de concisión. Esta expresión regular recupera todos los campos que contengan tanto `foaf` como `age` en el URI.

2. Obtenga todos los nodos con una antigüedad comprendida entre 10 y 50 años y un nombre con una coincidencia parcial con "Ronka"

En Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.age.value:[10 TO 50] AND
 predicates.name.value:Ronka~');
```

En SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
```

```
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\\*age.value:[10 TO 50] AND
 predicates.\\*foaf\\*name.value:Ronka~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

3. Obtenga todos los nodos con una marca temporal que esté comprendida dentro de los últimos 25 días

En Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>now-25d');
```

En SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://localhost:9200' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query "predicates.\\*foaf\
\*timestamp.value:>now-25d~" .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

4. Obtenga todos los nodos con una marca temporal que esté dentro de un año y un mes determinados

En Gremlin, usando expresiones matemáticas de fecha en la sintaxis de Lucene, para diciembre de 2020:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>2020-12');
```

Una alternativa a Gremlin:

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:[2020-12 TO 2021-01]');
```

Ejecución de consultas de búsqueda de texto completo en Amazon Neptune

En una consulta que incluye búsqueda de texto completo, Neptune intenta poner primero las llamadas de búsqueda de texto completo, antes que otras partes de la consulta. Esto reduce el número de llamadas a OpenSearch y, en la mayoría de los casos, mejora significativamente el rendimiento. Sin embargo, esta no es de ninguna manera una regla estricta. Hay situaciones, por ejemplo, en las que `PatternNode` o `UnionNode` pueden preceder a una llamada de búsqueda de texto completo.

Analicemos la siguiente consulta de Gremlin a una base de datos que contiene 100 000 instancias de `Person`:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Si esta consulta se ejecutó en el orden en que aparecen los pasos, fluirán 100 000 soluciones a OpenSearch, lo que causará cientos de llamadas de OpenSearch. De hecho, Neptune llama primero a OpenSearch y, a continuación, une los resultados con los resultados de Neptune. En la mayoría de los casos, esto es mucho más rápido que ejecutar la consulta en el orden original.

Puede evitar este reordenamiento de la ejecución de pasos de la consulta utilizando la sugerencia de consulta noReordering:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .withSideEffect('Neptune#noReordering', true)
```

```
  .hasLabel('Person')
  .has('name', 'Neptune#fts marcello~');
```

En este segundo caso, se ejecuta primero el paso `.hasLabel` y después el paso `.has('name', 'Neptune#fts marcello~')`.

Para ver otro ejemplo, considere una consulta SPARQL con el mismo tipo de datos:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT ?person WHERE {
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

De nuevo, aquí Neptune ejecuta primero la parte `SERVICE` de la consulta y, a continuación, une los resultados con los datos de `Person`. Puede suprimir este comportamiento utilizando la sugerencia de consulta joinOrder:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ?person WHERE {
  hint:Query hint:joinOrder "Ordered" .
  ?person rdf:type foaf:Person .
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mike' .
    neptune-fts:config neptune-fts:return ?person .
  }
}
```

Nuevamente, en la segunda consulta las partes se ejecutan en el orden en que aparecen en la consulta.

Consultas de SPARQL de ejemplo mediante la búsqueda de texto completo en Neptune

A continuación, se presentan algunos ejemplos de consultas de SPARQL que utilizan la búsqueda de texto completo en Amazon Neptune.

Ejemplo de consulta match de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'match' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'michael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta prefix de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'prefix' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mich' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta fuzzy de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
```

```
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'fuzzy' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'mikael' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta term de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'term' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:query 'Dr. Kunal' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta query_string de SPARQL

Esta consulta especifica varios campos.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ OR rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:field foaf:surname .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta simple_query_string de SPARQL

La siguiente consulta especifica los campos con el carácter comodín ("*").

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint.com' .
    neptune-fts:config neptune-fts:queryType 'simple_query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field '*' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta sort by string field de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy foaf:name .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta sort by non-string field de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name.value .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy dc:date.value .
    neptune-fts:config neptune-fts:return ?res .
  }
```

```
}
```

Ejemplo de consulta sort by ID de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta sort by label de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo de consulta sort by doc_type de SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
```

```
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'mikael~ | rondelli' .
    neptune-fts:config neptune-fts:field foaf:name .
    neptune-fts:config neptune-fts:sortOrder 'asc' .
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Ejemplo del uso de la sintaxis de Lucene en SPARQL

La sintaxis de Lucene solo es compatible con las consultas `query_string` de OpenSearch.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neptune-fts: <http://aws.amazon.com/neptune/vocab/v01/services/fts#>
SELECT * WHERE {
  SERVICE neptune-fts:search {
    neptune-fts:config neptune-fts:endpoint 'http://your-es-endpoint' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:queryType 'query_string' .
    neptune-fts:config neptune-fts:query 'predicates.\\foaf\\name.value:micheal AND
 predicates.\\foaf\\surname.value:sh' .
    neptune-fts:config neptune-fts:field ‘’ .
    neptune-fts:config neptune-fts:return ?res .
  }
}
```

Uso de la búsqueda de texto completo de Neptune en las consultas de Gremlin

`NeptuneSearchStep` permite las consultas de búsqueda de texto completo para la parte de un recorrido de Gremlin que no se convierte en pasos de Neptune. Por ejemplo, plantéese una consulta como la siguiente:

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")
  .V()
      .tail(100)
      .has("name", "Neptune#fts mark*")            <== # Limit the search on name
```

Esta consulta se convierte en el siguiente recorrido optimizado en Neptune.

```
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .],
 {estimatedCardinality=INFINITY}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=4}
    },
    NeptuneTraverserConverterStep
]
+ not converted into Neptune steps: [NeptuneTailGlobalStep(100),
 NeptuneTinkerpopTraverserConverterStep, NeptuneSearchStep {
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
    JoinGroupNode {
        SearchNode[(idVar=?3, query=mark*, field=name) . project ask .],
 {endpoint=your-OpenSearch-endpoint-URL}
    }
}]
```

Los siguientes ejemplos son de consultas de Gremlin de datos de rutas aéreas:

Consulta **`match`** básica que no distingue entre mayúsculas y minúsculas de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts dallas")

==>v[186]
==>v[8]
```

Consulta **`match`** de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts southampton")
```

```
      .local(values('code','city').fold())
      .limit(5)

==>[SOU, Southampton]
```

Consulta **fuzzy** de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
==>Walla Walla
==>Velas
==>Altai
```

Consulta **query_string** aproximada de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts allas~").values('city').limit(5)

==>Dallas
==>Dallas
```

Consulta de expresión regular **query_string** de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has("city","Neptune#fts /[dp]allas/").values('city').limit(5)

==>Dallas
==>Dallas
```

Consulta híbrida de Gremlin

Esta consulta utiliza un índice interno de Neptune y el índice de OpenSearch en la misma consulta.

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has("region","GB-ENG")
      .has('city','Neptune#fts L*')
      .values('city')
      .dedup()
      .limit(10)

==>London
==>Leeds
==>Liverpool
==>Land's End
```

Ejemplo de búsqueda de texto completo sencilla de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts regional municipal')
      .local(values('code','desc').fold())
      .limit(100)

==>[HYA, Barnstable Municipal Boardman Polando Field]
==>[SPS, Sheppard Air Force Base-Wichita Falls Municipal Airport]
==>[ABR, Aberdeen Regional Airport]
==>[SLK, Adirondack Regional Airport]
==>[BFD, Bradford Regional Airport]
==>[EAR, Kearney Regional Airport]
==>[ROT, Rotorua Regional Airport]
==>[YHD, Dryden Regional Airport]
==>[TEX, Telluride Regional Airport]
==>[WOL, Illawarra Regional Airport]
==>[TUP, Tupelo Regional Airport]
==>[COU, Columbia Regional Airport]
==>[MHK, Manhattan Regional Airport]
==>[BJI, Bemidji Regional Airport]
==>[HAS, Hail Regional Airport]
==>[ALO, Waterloo Regional Airport]
==>[SHV, Shreveport Regional Airport]
==>[ABI, Abilene Regional Airport]
==>[GIZ, Jizan Regional Airport]
==>[USA, Concord Regional Airport]
==>[JMS, Jamestown Regional Airport]
```

```
==>[COS, City of Colorado Springs Municipal Airport]
==>[PKB, Mid Ohio Valley Regional Airport]
```

Consulta de Gremlin con **`query_string`** y los operadores “+” y “-”

Aunque el tipo de consulta `query_string` es mucho menos tolerante que el tipo predeterminado `simple_query_string`, permite consultas más precisas. La primera consulta a continuación utiliza `query_string`, mientras que la segunda utiliza el valor predeterminado `simple_query_string`:

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
 . V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
```

Observe cómo `simple_query_string` en los ejemplos siguientes no tiene en cuenta los operadores “+” y “-” progresivamente:

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .V().has('desc','Neptune#fts +London -(Stansted|Gatwick)')
      .local(values('code','desc').fold())
      .limit(10)

==>[LHR, London Heathrow]
==>[YXU, London Airport]
==>[LGW, London Gatwick]
==>[STN, London Stansted Airport]
==>[LTN, London Luton Airport]
==>[SEN, London Southend Airport]
==>[LCY, London City Airport]
==>[SKG, Thessaloniki Macedonia International Airport]
==>[ADB, Adnan Menderes International Airport]
==>[BTV, Burlington International Airport]
```

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts +(regional|municipal) -(international|bradford)')
      .local(values('code','desc').fold())
      .limit(10)

==>[CZH, Corozal Municipal Airport]
==>[MMU, Morristown Municipal Airport]
==>[YBR, Brandon Municipal Airport]
==>[RDD, Redding Municipal Airport]
==>[VIS, Visalia Municipal Airport]
==>[AIA, Alliance Municipal Airport]
==>[CDR, Chadron Municipal Airport]
==>[CVN, Clovis Municipal Airport]
==>[SDY, Sidney Richland Municipal Airport]
==>[SGU, St George Municipal Airport]
```

Consulta **`query_string`** de Gremlin con los operadores **AND** y **OR**

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'query_string')
  .V().has('desc','Neptune#fts (St AND George) OR (St AND Augustin)')
      .local(values('code','desc').fold())
      .limit(10)

==>[YIF, St Augustin Airport]
==>[STG, St George Airport]
==>[SGO, St George Airport]
==>[SGU, St George Municipal Airport]
```

Consulta **`term`** de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'term')
  .V().has("SKU","Neptune#fts ABC123DEF9")
      .local(values('code','city').fold())
      .limit(5)
```

```
==>[AUS, Austin]
```

Consulta **`prefix`** de Gremlin

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'prefix')
  .V().has("icao","Neptune#fts ka")
      .local(values('code','icao','city').fold())
      .limit(5)

==>[AZO, KAZO, Kalamazoo]
==>[APN, KAPN, Alpena]
==>[ACK, KACK, Nantucket]
==>[ALO, KALO, Waterloo]
==>[ABI, KABI, Abilene]
```

Uso de la sintaxis de Lucene en Neptune Gremlin

En Neptune Gremlin, también puede escribir consultas muy eficaces con la sintaxis de consulta de Lucene. Tenga en cuenta que la sintaxis de Lucene solo es compatible con las consultas `query_string` de OpenSearch.

Suponga los siguientes datos:

```
g.addV("person")
        .property(T.id, "p1")
        .property("name", "simone")
        .property("surname", "rondelli")

g.addV("person")
        .property(T.id, "p2")
        .property("name", "simone")
        .property("surname", "sengupta")

g.addV("developer")
        .property(T.id, "p3")
        .property("name", "simone")
        .property("surname", "rondelli")
```

Usando la sintaxis de Lucene, que se invoca cuando `queryType` es `query_string`, puede buscar estos datos por nombre y apellido de la siguiente manera:

```
g.withSideEffect("Neptune#fts.endpoint", "es_endpoint")
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli")

==> v[p1], v[p3]
```

Tenga en cuenta que en el paso `has()` anterior, el campo se reemplaza por `"*"`. En realidad, cualquier valor colocado allí se sustituye por los campos a los que tiene acceso desde la consulta. Puede acceder al campo de nombre utilizando `predicates.name.value,` porque así es como está estructurado el modelo de datos.

Puede buscar por nombre, apellido y etiqueta, de la siguiente manera:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person")

==> v[p1]
```

Se obtiene acceso a la etiqueta utilizando `entity_type`, de nuevo porque así es como está estructurado el modelo de datos.

También puede incluir condiciones de anidamiento:

```
g.withSideEffect("Neptune#fts.endpoint", getEsEndpoint())
    .withSideEffect("Neptune#fts.queryType", "query_string")
    .V()
    .has("*", "Neptune#fts (predicates.name.value:simone AND
 predicates.surname.value:rondelli AND entity_type:person) OR
 predicates.surname.value:sengupta")

==> v[p1], v[p2]
```

Inserción de un gráfico moderno de TinkerPop

```
g.addV('person').property(T.id, '1').property('name', 'marko').property('age', 29)
 .addV('personr').property(T.id, '2').property('name', 'vadas').property('age', 27)
 .addV('software').property(T.id, '3').property('name', 'lop').property('lang', 'java')
 .addV('person').property(T.id, '4').property('name', 'josh').property('age', 32)
 .addV('software').property(T.id, '5').property('name', 'ripple').property('lang',
 'java')
 .addV('person').property(T.id, '6').property('name', 'peter').property('age', 35)

g.V('1').as('a').V('2').as('b').addE('knows').from('a').to('b').property('weight',
 0.5f).property(T.id, '7')
 .V('1').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '9')
 .V('4').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.4f).property(T.id, '11')
 .V('4').as('a').V('5').as('b').addE('created').from('a').to('b').property('weight',
 1.0f).property(T.id, '10')
 .V('6').as('a').V('3').as('b').addE('created').from('a').to('b').property('weight',
 0.2f).property(T.id, '12')
 .V('1').as('a').V('4').as('b').addE('knows').from('a').to('b').property('weight',
 1.0f).property(T.id, '8')
```

Ejemplo de valor del campo Ordenar por cadena

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'name')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Ejemplo de valor del campo Ordenar por no cadena

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'age.value')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Ejemplo de valor del campo Ordenar por ID

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
.withSideEffect('Neptune#fts.queryType', 'query_string')
.withSideEffect('Neptune#fts.sortOrder', 'asc')
.withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
.V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Ejemplo de valor del campo Ordenar por etiqueta

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Ejemplo de valor del campo Ordenar por **`document_type`**

```
g.withSideEffect("Neptune#fts.endpoint", "your-OpenSearch-endpoint-URL")
 .withSideEffect('Neptune#fts.queryType', 'query_string')
 .withSideEffect('Neptune#fts.sortOrder', 'asc')
 .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
 .V().has('name', 'Neptune#fts marko OR vadas OR ripple')
```

Solución de problemas de búsqueda de texto completo de Neptune

> **Note**
>
> Si ha habilitado el control de acceso detallado en el clúster de OpenSearch, también debe habilitar la autenticación de IAM en la base de datos de Neptune.

Para diagnosticar problemas de replicación de Neptune a OpenSearch, consulte los registros de CloudWatch de la función de Lambda del sondeador. Estos registros proporcionan detalles sobre el número de registros leídos de la transmisión y el número de registros replicados correctamente en OpenSearch.

También puede cambiar el nivel de REGISTRO para la función de Lambda al cambiar la variable del entorno `LoggingLevel`.

> **Note**
>
> Si `LoggingLevel` está establecido en `DEBUG`, puede ver detalles adicionales, como los registros de transmisiones eliminados y el motivo por el que se han eliminado, mientras replica los datos mediante StreamPoller desde Neptune a OpenSearch. Esto puede resultar útil si descubre que le faltan registros.

La aplicación de consumidor de transmisiones de Neptune publica dos métricas en CloudWatch que también pueden ayudarle a diagnosticar problemas:

- `StreamRecordsProcessed` – El número de registros procesados por la aplicación por unidad de tiempo. Útil en el seguimiento de la tasa de ejecución de la aplicación.
- `StreamLagTime` – El tiempo que transcurre en milisegundos entre la hora actual y la hora de confirmación de un registro de transmisión que se está procesando. Esta métrica muestra hasta qué punto la aplicación del consumidor se queda rezagada.

Además, todas las métricas relacionadas con el proceso de replicación se muestran en un panel en CloudWatch con el mismo nombre que el `ApplicationName` proporcionado al crear una instancia de la aplicación mediante la plantilla de CloudWatch.

También puede optar por crear una alarma de CloudWatch que se active cuando el sondeo falle más de dos veces seguidas. Para ello, establezca el campo `CreateCloudWatchAlarm` en `true` cuando cree una instancia para la aplicación. A continuación, especifique las direcciones de correo electrónico a las que desea que se le notifique cuando se active la alarma.

Solución de problemas de un proceso que falla al leer los registros de la secuencia

Si un proceso falla al leer los registros del flujo, asegúrese de que tenga lo siguiente:

- El flujo está habilitado en el clúster.
- El punto de conexión de la transmisión de Neptune está en el formato correcto:
  - Para Gremlin u openCypher: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/propertygraph/stream` o su alias, `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/pg/stream`.

  - Para SPARQL: `https://your cluster endpoint:your cluster port/sparql/stream`

- El punto de conexión de DynamoDB está configurado para la VPC.
- El punto de conexión de monitorización está configurado para las subredes de VPC.

Solución de problemas de un proceso que falla al escribir datos en OpenSearch

Si un se produce un error al escribir los registros en OpenSearch, asegúrese de lo siguiente:

- Que cuente con la versión de Elasticsearch 7.1 o posterior; Opensearch 2.3 y versiones posteriores.
- Que se pueda acceder a OpenSearch desde la función de Lambda del sondeador en la VPC.
- Que la política de seguridad adjunta a OpenSearch permita solicitudes HTTP/HTTPS entrantes.

Solución de problemas de desincronización entre Neptune y OpenSearch en una configuración de replicación existente

Puede seguir los pasos que se indican a continuación para volver a sincronizar una base de datos de Neptune y un dominio de OpenSearch con los datos más recientes en caso de que se produzcan problemas de desincronización entre ellos como resultado de una `ExpiredStreamException` o daños en los datos.

Tenga en cuenta que este enfoque elimina todos los datos del dominio OpenSearch y los vuelve a sincronizar con el estado actual de la base de datos de Neptune, por lo que no es necesario volver a cargar ningún dato en la base de datos de Neptune.

1. Deshabilite el proceso de replicación, tal y como se describe en Deshabilitar (pausar) el proceso del sondeador de transmisiones.
2. Elimine el índice de Neptune en el dominio OpenSearch mediante el siguiente comando:

```
curl -X DELETE "(your OpenSearch endpoint)/amazon_neptune"
```

3. Cree un clon de la base de datos (consulte Database Cloning in Neptune).

4. Obtenga el último `eventID` de los flujos de la base de datos clonada mediante la ejecución de un comando de este tipo en el punto de conexión de la API de flujos (consulte Calling the Neptune Streams REST API para obtener más información):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Anote los valores de los campos `commitNum` y `opNum` del objeto `lastEventId` en la respuesta.
5. Use la herramienta export-neptune-to-elasticsearch en github para realizar una sincronización única desde la base de datos clonada al dominio de OpenSearch.
6. Vaya a la tabla de DynamoDB de la pila de replicación. El nombre de la tabla será el nombre de la aplicación que haya especificado en la plantilla AWS CloudFormation (el nombre predeterminado es `NeptuneStream`) con un sufijo `-LeaseTable`. Es decir, el nombre de la tabla predeterminado es `NeptuneStream-LeaseTable`.

   Realice un escaneo para explorar las filas de la tabla, ya que solo debe haber una fila en la tabla. Realice los siguientes cambios con los valores `commitNum` y `opNum` que ha registrado anteriormente:

   - Cambie el valor del campo `checkpoint` de la tabla por el valor que ha anotado para `commitNum`.
   - Cambie el valor del campo `checkpointSubSequenceNumber` de la tabla por el valor que ha anotado para `opNum`.
7. Vuelva a habilitar el proceso de replicación, tal y como se describe en Volver a habilitar el proceso del sondeador de transmisiones.
8. Elimine la base de datos clonada y la pila de AWS CloudFormation creada para la herramienta `export-neptune-to-elasticsearch`.

Uso de funciones de AWS Lambda en Amazon Neptune

Las funciones de AWS Lambda tienen muchos usos en las aplicaciones de Amazon Neptune. A continuación, ofrecemos instrucciones generales sobre cómo utilizar las funciones de Lambda con cualquiera de los conocidos controladores y variantes de lenguaje de Gremlin, así como ejemplos específicos de las funciones de Lambda escritas en Java, JavaScript y Python.

> **Note**
>
> La mejor forma de utilizar las funciones de Lambda con Neptune ha cambiado con las versiones recientes del motor. Neptune solía dejar abiertas las conexiones inactivas mucho después de reciclar un contexto de ejecución de Lambda, lo que podía provocar una pérdida de recursos en el servidor. Para mitigar esta problema, solíamos recomendar abrir y cerrar una conexión con cada invocación a Lambda. Sin embargo, a partir de la versión 1.0.3.0 del motor, se ha reducido el tiempo de espera de las conexiones inactivas para que dejen de tener pérdidas tras reciclar un contexto de ejecución de Lambda inactivo, por lo que ahora recomendamos utilizar una sola conexión durante el contexto de ejecución. Esto debería incluir la gestión de errores y el uso de código reutilizable de retroceso y reintento para evitar que las conexiones se cierren de forma inesperada.

Administración de las conexiones WebSocket de Gremlin en las funciones de AWS Lambda

Si utiliza una variante del lenguaje Gremlin para consultar Neptune, el controlador se conectará a la base de datos mediante una conexión WebSocket. Los WebSockets se han diseñado para admitir situaciones de conexión cliente-servidor de larga duración. Por otro lado, AWS Lambda se ha diseñado para admitir ejecuciones relativamente efímeras y sin estado. Esta discordancia en la filosofía de diseño puede provocar algunos problemas inesperados al utilizar Lambda para consultar Neptune.

Una función de AWS Lambda se ejecuta en un contexto de ejecución que la aísla de otras funciones. El contexto de ejecución se crea la primera vez que se invoca la función y se puede volver a utilizar para invocaciones posteriores de la misma función.

Sin embargo, nunca se utiliza un contexto de ejecución para gestionar varias invocaciones simultáneas de la función. Si varios clientes invocan la función de forma simultánea, Lambda genera

un contexto de ejecución adicional para cada instancia de la función. Todos estos nuevos contextos de ejecución pueden, a su vez, volver a utilizarse en invocaciones posteriores de la función.

En algún momento, Lambda recicla los contextos de ejecución, especialmente si han estado inactivos durante algún tiempo. AWS Lambda expone el ciclo de vida del contexto de ejecución, incluidas las fases `Init`, `Invoke` y `Shutdown`, mediante extensiones de Lambda. Con estas extensiones, puede escribir código que limpie los recursos externos, como las conexiones a bases de datos, cuando se recicle el contexto de ejecución.

Una práctica recomendada habitual es abrir la conexión a la base de datos fuera de la función de controlador de Lambda para que se pueda volver a utilizar con cada llamada al controlador. Si la conexión a la base de datos se interrumpe en algún momento, puede volver a conectarse desde el interior del controlador. Sin embargo, existe el peligro de que se produzcan pérdidas de conexión con este enfoque. Si una conexión inactiva permanece abierta durante mucho tiempo después de que se destruya un contexto de ejecución, las situaciones de invocación de Lambda intermitentes o en ráfagas pueden perder conexiones gradualmente y agotar los recursos de la base de datos.

Los límites y los tiempos de espera de conexión de Neptune han cambiado con las versiones más recientes del motor. Anteriormente, cada instancia admitía hasta 60 000 conexiones de WebSocket. Ahora, el número máximo de conexiones de WebSocket simultáneas por instancia de Neptune varía según el tipo de instancia.

Además, a partir de la versión 1.0.3.0 del motor, Neptune redujo el tiempo de inactividad de las conexiones de una hora a aproximadamente 20 minutos. Si un cliente no cierra una conexión, esta se cerrará automáticamente tras un tiempo de inactividad de 20 a 25 minutos. AWS Lambda no documenta la duración del contexto de ejecución, pero los experimentos muestran que el nuevo tiempo de espera de la conexión de Neptune se ajusta bien con los tiempos de espera inactivos del contexto de ejecución de Lambda. Cuando se recicla un contexto de ejecución inactivo, es muy probable que Neptune ya haya cerrado su conexión o que se cierre poco después.

Recomendaciones para utilizar AWS Lambda con Amazon Neptune Gremlin

Ahora recomendamos utilizar un único origen de recorrido de gráficos y conexión durante toda la vida útil de un contexto de ejecución de Lambda, en lugar de utilizar uno para cada invocación de función (cada invocación de función gestiona solo una solicitud de cliente). Dado que las solicitudes de clientes simultáneas las gestionan distintas instancias de funciones que se ejecutan en contextos de ejecución distintos, no es necesario mantener un grupo de conexiones para gestionar estas

solicitudes dentro de una instancia de función. Si el controlador de Gremlin que utiliza tiene un grupo de conexiones, configúrelo para que utilice solo una conexión.

Para gestionar los errores de conexión, utilice la lógica de reintento en cada consulta. Aunque el objetivo sea mantener una conexión única durante todo el tiempo que dure un contexto de ejecución, los eventos de red inesperados pueden provocar que la conexión se interrumpa de forma abrupta. Estos errores de conexión se manifiestan como errores diferentes en función del controlador que se utilice. Debe codificar la función de Lambda para gestionar estos problemas de conexión e intentar volver a conectarse si es necesario.

Algunos controladores de Gremlin gestionan automáticamente las reconexiones. El controlador de Java, por ejemplo, intenta restablecer automáticamente la conectividad con Neptune en nombre del código del cliente. Con este controlador, el código de función solo necesita retroceder y volver a intentar la consulta. En cambio, los controladores de JavaScript y Python no implementan ninguna lógica de reconexión automática, por lo que con estos controladores el código de la función debe intentar volver a conectarse después de retroceder y solo volver a intentar la consulta una vez que se haya restablecido la conexión.

Los ejemplos de código que se indican a continuación incluyen la lógica de reconexión en lugar de suponer que el cliente se está encargando de ello.

Recomendaciones para utilizar las solicitudes de escritura de Gremlin en Lambda

Si la función de Lambda modifica los datos del gráfico, plantéese adoptar una estrategia de retroceso y reintento para gestionar las siguientes excepciones:

- **`ConcurrentModificationException`**: la semántica de las transacciones de Neptune significa que, en ocasiones, se produce un error en las solicitudes de escritura con una `ConcurrentModificationException`. En estas situaciones, pruebe un mecanismo de reintento exponencial basado en el retroceso.
- **`ReadOnlyViolationException`**: dado que la topología del clúster puede cambiar en cualquier momento como resultado de eventos planificados o imprevistos, las responsabilidades de escritura pueden migrar de una instancia del clúster a otra. Si el código de función intenta enviar una solicitud de escritura a una instancia que ya no es la instancia principal (escritor), se produce un error en la solicitud con una `ReadOnlyViolationException`. Cuando esto suceda, cierre la conexión existente, vuelva a conectarse al punto de conexión del clúster y, a continuación, vuelva a intentar la solicitud.

Además, si utiliza una estrategia de retroceso y reintento para solucionar los problemas de las solicitudes de escritura, plantéese implementar consultas idempotentes para las solicitudes de creación y actualización. Por ejemplo, mediante fold().coalesce().unfold().

Recomendaciones para utilizar las solicitudes de lectura de Gremlin en Lambda

Si tiene una o varias réplicas de lectura en el clúster, es buena idea equilibrar las solicitudes de lectura entre estas réplicas. Una opción es utilizar el punto de conexión del lector. El punto de conexión del lector equilibra las conexiones entre las réplicas, incluso si la topología del clúster cambia al añadir o eliminar réplicas, o al promover una réplica para que se convierta en la nueva instancia principal.

Sin embargo, el uso del punto de conexión del lector puede provocar un uso irregular de los recursos del clúster en algunas circunstancias. El punto de conexión del lector funciona mediante el cambio periódico del host al que apunta la entrada de DNS. Si un cliente abre muchas conexiones antes de que cambie la entrada de DNS, todas las solicitudes de conexión se envían a una única instancia de Neptune. Este puede ser el caso de un caso de Lambda de alto rendimiento en el que un gran número de solicitudes simultáneas a la función de Lambda provoca la creación de varios contextos de ejecución, cada uno con su propia conexión. Si todas estas conexiones se crean casi simultáneamente, es probable que todas apunten a la misma réplica del clúster y que sigan apuntando a esa réplica hasta que se reciclen los contextos de ejecución.

Una forma de distribuir las solicitudes entre las instancias es configurar la función de Lambda para que se conecte a un punto de conexión de la instancia, elegido al azar en una lista de puntos de conexión de la instancia de réplica, en lugar del punto de conexión del lector. El inconveniente de este enfoque es que requiere que el código de Lambda gestione los cambios en la topología del clúster mediante la supervisión del clúster y la actualización de la lista de puntos de conexión cada vez que cambia la pertenencia al clúster.

Si está escribiendo una función de Lambda de Java que necesita equilibrar las solicitudes de lectura entre las instancias del clúster, puede utilizar el cliente Gremlin para Amazon Neptune, un cliente Java Gremlin que conoce la topología del clúster y que distribuye de forma equitativa las conexiones y solicitudes entre un conjunto de instancias de un clúster de Neptune. Esta publicación de blog incluye un ejemplo de función de Lambda de Java que usa el cliente Gremlin para Amazon Neptune.

Factores que pueden ralentizar los arranques en frío de las funciones de Lambda en Neptune Gremlin

La primera vez que se invoca una función de AWS Lambda se denomina arranque en frío. Existen varios factores que pueden aumentar la latencia de un arranque en frío:

- Asegúrese de asignar memoria suficiente a la función de Lambda. La compilación durante un arranque en frío puede ser considerablemente más lenta en una función de Lambda que en EC2, ya que AWS Lambda asigna los ciclos de CPU de forma lineal en proporción a la memoria que se asigna a la función. Con 1792 MB de memoria, la función recibe el equivalente de una vCPU completa (un segundo de créditos de vCPU por segundo). El impacto de no asignar suficiente memoria para recibir ciclos de CPU adecuados es especialmente marcado en el caso de las funciones de Lambda de gran tamaño escritas en Java.
- Tenga en cuenta que al habilitar la autenticación de bases de datos de IAM se puede ralentizar un arranque en frío. La autenticación de base de datos de AWS Identity and Access Management (IAM) también puede ralentizar los arranques en frío, especialmente si la función de Lambda tiene que generar una nueva clave de firma. Esta latencia solo afecta al arranque en frío y no a las solicitudes posteriores, ya que una vez que la autenticación de la base de datos de IAM haya establecido las credenciales de conexión, Neptune solo validará periódicamente que siguen siendo válidas.

Uso de AWS CloudFormation para crear una función de Lambda para usarla en Neptune

Puede utilizar una plantilla de AWS CloudFormation para crear una función de AWS Lambda que pueda acceder a Neptune.

1. Para lanzar la pila de la función de Lambda en la consola de AWS CloudFormation, elija uno de los botones Lanzar pila de la siguiente tabla.

Region	Visualización	Ver en Designer	Lanzar
Este de EE. UU. (Norte de Virginia)	Ver	Ver en Designer	Launch Stack

Region	Visualización	Ver en Designer	Lanzar
Este de EE. UU. (Ohio)	Ver	Ver en Designer	Launch Stack
Oeste de EE. UU. (Norte de California)	Ver	Ver en Designer	Launch Stack
Oeste de EE. UU. (Oregón)	Ver	Ver en Designer	Launch Stack
Canadá (Centro)	Ver	Ver en Designer	Launch Stack
América del Sur (São Paulo)	Ver	Ver en Designer	Launch Stack
Europa (Estocolmo)	Ver	Ver en Designer	Launch Stack
Europa (Irlanda)	Ver	Ver en Designer	Launch Stack
Europa (Londres)	Ver	Ver en Designer	Launch Stack
Europa (París)	Ver	Ver en Designer	Launch Stack
Europa (Fráncfort)	Ver	Ver en Designer	Launch Stack
Medio Oriente (Baréin)	Ver	Ver en Designer	Launch Stack
Medio Oriente (EAU)	Ver	Ver en Designer	Launch Stack
Israel (Tel Aviv)	Ver	Ver en Designer	Launch Stack
África (Ciudad del Cabo)	Ver	Ver en Designer	Launch Stack

Region	Visualización	Ver en Designer	Lanzar
Asia-Pacífico (Hong Kong)	Ver	Ver en Designer	Launch Stack
Asia-Pacífico (Tokio)	Ver	Ver en Designer	Launch Stack
Asia-Pacífico (Seúl)	Ver	Ver en Designer	Launch Stack
Asia-Pacífico (Singapur)	Ver	Ver en Designer	Launch Stack
Asia-Pacífico (Sídney)	Ver	Ver en Designer	Launch Stack
Asia-Pacífico (Bombay)	Ver	Ver en Designer	Launch Stack
China (Pekín)	Ver	Ver en Designer	Launch Stack
China (Ningxia)	Ver	Ver en Designer	Launch Stack
AWS GovCloud (Oeste de EE. UU.)	Ver	Ver en Designer	Launch Stack
AWS GovCloud (Este de EE. UU.)	Ver	Ver en Designer	Launch Stack

2. En la página Select Template, elija Next.
3. En la página Specify Details (Especificar detalles), configure las siguientes opciones:

   a. Elija el tiempo de ejecución de Lambda, en función del lenguaje que quiera utilizar en su función de Lambda. Estas plantillas de AWS CloudFormation actualmente admiten los siguientes lenguajes:

      - Python 3.9 (se mapea a `python39` en la URL de Amazon S3)
      - NodeJS 18 (se mapea a `nodejs18x` en la URL de Amazon S3)

- Ruby 2.5 (se mapea a `ruby25` en la URL de Amazon S3)

b. Proporcione el punto de conexión adecuado del clúster de Neptune y el número de puerto.

c. Proporcione el grupo de seguridad adecuado de Neptune.

d. Proporcione los parámetros adecuados de subred de Neptune.

4. Elija Siguiente.
5. En la página Opciones, seleccione Siguiente.
6. En la página Revisar, seleccione la primera casilla para confirmar que AWS CloudFormation debe crear los recursos de IAM.

   A continuación, elija Crear.

Si necesita realizar sus propios cambios en el tiempo de ejecución de Lambda, puede descargar una genérica desde una ubicación de Amazon S3 en su región:

```
https://s3.Amazon region.amazonaws.com/aws-neptune-customer-samples-Amazon region/
lambda/runtime-language/lambda_function.zip.
```

Por ejemplo:

```
https://s3.us-west-2.amazonaws.com/aws-neptune-customer-samples-us-west-2/lambda/
python36/lambda_function.zip
```

Ejemplos de funciones de AWS Lambda para Amazon Neptune

Las siguientes funciones de AWS Lambda de ejemplo, escritas en Java, JavaScript y Python, muestran cómo llevar a cabo actualizaciones o inserciones en un único vértice con un identificador generado de forma aleatoria mediante la expresión `fold().coalesce().unfold()`.

Gran parte del código de cada función es código reutilizable, responsable de administrar las conexiones y reintentar las conexiones y consultas si se produce un error. La lógica de la aplicación real y la consulta de Gremlin se implementan en los métodos `doQuery()` y `query()`, respectivamente. Si utiliza estos ejemplos como base para sus propias funciones de Lambda, puede centrarse en modificar `doQuery()` y `query()`.

Las funciones se han configurado para reintentar las consultas erróneas cinco veces, esperando un segundo entre cada reintento.

Las funciones requieren que los valores estén presentes en las siguientes variables del entorno de Lambda:

- **`NEPTUNE_ENDPOINT`**: el punto de conexión del clúster de base de datos de Neptune. Para Python, debería ser `neptuneEndpoint`.
- **`NEPTUNE_PORT`**: el puerto de Neptune. Para Python, debería ser `neptunePort`.
- **`USE_IAM`** : (`true` o `false`) si la base de datos tiene habilitada la autenticación de bases de datos AWS Identity and Access Management (IAM), establezca la variable del entorno `USE_IAM` en `true`. Esto hace que la función de Lambda firme las solicitudes de conexión a Neptune mediante Sigv4. Para dichas solicitudes de autenticación de base de datos de IAM, asegúrese de que el rol de ejecución de la función de Lambda tenga adjunta una política de IAM adecuada que permita a la función conectarse al clúster de base de datos de Neptune (consulte Tipos de políticas de IAM).

Ejemplo de función de Lambda de Java para Amazon Neptune

Estos son algunos aspectos que se deben tener en cuenta sobre las funciones de AWS Lambda de Java:

- El controlador de Java mantiene su propio conjunto de conexiones, que no es necesario, por lo que debe configurar el objeto `Cluster` con `minConnectionPoolSize(1)` y `maxConnectionPoolSize(1)`.
- El objeto `Cluster` puede tardar en crearse porque crea uno o varios serializadores (Gyro, de forma predeterminada, y otro si lo ha configurado para formatos de salida adicionales, como, por ejemplo, `binary`). Estos pueden tardar un tiempo en crear una instancia.
- El conjunto de conexiones se inicializa con la primera solicitud. En este momento, el controlador configura la pila `Netty`, asigna búferes de bytes y crea una clave de firma si se utiliza la autenticación de base de datos de IAM. Todos estos factores pueden aumentar la latencia de arranque en frío.
- El conjunto de conexiones del controlador Java supervisa la disponibilidad de los hosts del servidor e intenta volver a conectarse automáticamente si se produce un error en la conexión. Inicia una tarea en segundo plano para intentar restablecer la conexión. Utilice `reconnectInterval( )` para configurar el intervalo entre los intentos de reconexión. Mientras el controlador intenta volver a conectarse, la función de Lambda simplemente puede volver a intentar la consulta.

Si el intervalo entre los reintentos es menor que el intervalo entre los intentos de reconexión, los reintentos de conexión vuelven a fallar porque se considera que el host no está disponible. Esto no se aplica a los reintentos de una `ConcurrentModificationException`.

- Utilice Java 8 en lugar de Java 11. Las optimizaciones de `Netty` no están habilitadas de forma predeterminada en Java 11.
- En este ejemplo se utiliza Retry4j para los reintentos.
- Para usar el controlador de firma `Sigv4` en la función de Lambda de Java, consulte los requisitos de dependencia de Conexión con Neptune mediante Java y Gremlin con firma de Signature Version 4.

> **Warning**
>
> Es posible que el `CallExecutor` de Retry4j no sea seguro para subprocesos. Plantéese la posibilidad de que cada subproceso use su propia instancia de `CallExecutor`.

```
package com.amazonaws.examples.social;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestStreamHandler;
import com.evanlennick.retry4j.CallExecutor;
import com.evanlennick.retry4j.CallExecutorBuilder;
import com.evanlennick.retry4j.Status;
import com.evanlennick.retry4j.config.RetryConfig;
import com.evanlennick.retry4j.config.RetryConfigBuilder;
import org.apache.tinkerpop.gremlin.driver.Cluster;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer;
import org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteConnection;
import org.apache.tinkerpop.gremlin.driver.ser.Serializers;
import org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource;
import org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.GraphTraversalSource;
import org.apache.tinkerpop.gremlin.structure.T;

import java.io.*;
import java.time.temporal.ChronoUnit;
import java.util.HashMap;
import java.util.Map;
```

```
import java.util.Random;
import java.util.concurrent.Callable;
import java.util.function.Function;

import static java.nio.charset.StandardCharsets.UTF_8;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.addV;
import static org.apache.tinkerpop.gremlin.process.traversal.dsl.graph.__.unfold;

public class MyHandler implements RequestStreamHandler {

  private final GraphTraversalSource g;
  private final CallExecutor<Object> executor;
  private final Random idGenerator = new Random();

  public MyHandler() {

    this.g = AnonymousTraversalSource
            .traversal()
            .withRemote(DriverRemoteConnection.using(createCluster()));


    this.executor = new CallExecutorBuilder<Object>()
            .config(createRetryConfig())
            .build();

  }

  @Override
  public void handleRequest(InputStream input,
                            OutputStream output,
                            Context context) throws IOException {

    doQuery(input, output);
  }

  private void doQuery(InputStream input, OutputStream output) throws IOException {
    try {

      Map<String, Object> args = new HashMap<>();
      args.put("id", idGenerator.nextInt());

      String result = query(args);

      try (Writer writer = new BufferedWriter(new OutputStreamWriter(output, UTF_8))) {
```

```
          writer.write(result);
      }

    } finally {
        input.close();
        output.close();
    }
  }

  private String query(Map<String, Object> args) {
    int id = (int) args.get("id");

    @SuppressWarnings("unchecked")
    Callable<Object> query = () -> g.V(id)
      .fold()
      .coalesce(
        unfold(),
        addV("Person").property(T.id, id))
      .id().next();

    Status<Object> status = executor.execute(query);

    return status.getResult().toString();
  }

  private Cluster createCluster() {
    Cluster.Builder builder = Cluster.build()

.addContactPoint(System.getenv("NEPTUNE_ENDPOINT"))

.port(Integer.parseInt(System.getenv("NEPTUNE_PORT")))
                                    .enableSsl(true)
                                    .minConnectionPoolSize(1)
                                    .maxConnectionPoolSize(1)
                                    .serializer(Serializers.GRAPHBINARY_V1D0)
                                    .reconnectInterval(2000);

  if (Boolean.parseBoolean(getOptionalEnv("USE_IAM", "true"))) {
    // For versions of TinkerPop 3.4.11 or higher:
    builder.handshakeInterceptor( r ->
      {
        NeptuneNettyHttpSigV4Signer sigV4Signer = new
NeptuneNettyHttpSigV4Signer(region, new DefaultAWSCredentialsProviderChain());
        sigV4Signer.signRequest(r);
```

```
        return r;
      }
    )

    // Versions of TinkerPop prior to 3.4.11 should use the following approach.
    // Be sure to adjust the  imports to include:
    //   import org.apache.tinkerpop.gremlin.driver.SigV4WebSocketChannelizer;
    //   builder = builder.channelizer(SigV4WebSocketChannelizer.class);

    return builder.create();
  }

  private RetryConfig createRetryConfig() {
    return new RetryConfigBuilder().retryOnCustomExceptionLogic(retryLogic())
                                   .withDelayBetweenTries(1000, ChronoUnit.MILLIS)
                                   .withMaxNumberOfTries(5)
                                   .withFixedBackoff()
                                   .build();
  }

  private Function<Exception, Boolean> retryLogic() {
    return e -> {
      StringWriter stringWriter = new StringWriter();
      e.printStackTrace(new PrintWriter(stringWriter));
      String message = stringWriter.toString();

      // Check for connection issues
      if ( message.contains("Timed out while waiting for an available host") ||
           message.contains("Timed-out waiting for connection on Host") ||
           message.contains("Connection to server is no longer active") ||
           message.contains("Connection reset by peer") ||
           message.contains("SSLEngine closed already") ||
           message.contains("Pool is shutdown") ||
           message.contains("ExtendedClosedChannelException") ||
           message.contains("Broken pipe")) {
        return true;
      }

      // Concurrent writes can sometimes trigger a ConcurrentModificationException.
      // In these circumstances you may want to backoff and retry.
      if (message.contains("ConcurrentModificationException")) {
          return true;
      }
```

```
      // If the primary fails over to a new instance, existing connections to the old
 primary will
      // throw a ReadOnlyViolationException. You may want to back and retry.
      if (message.contains("ReadOnlyViolationException")) {
          return true;
      }

      return false;
    };
  }

  private String getOptionalEnv(String name, String defaultValue) {
    String value = System.getenv(name);
    if (value != null && value.length() > 0) {
      return value;
    } else {
      return defaultValue;
    }
  }
}
```

Si desea incluir la lógica de reconexión en la función, consulte Ejemplo de reconexión de Java.

Ejemplo de función de Lambda de JavaScript para Amazon Neptune

Notas sobre este ejemplo

- El controlador de JavaScript no mantiene un grupo de conexiones. Siempre abre una única conexión.
- La función de ejemplo utiliza las utilidades de firma Sigv4 de gremlin-aws-sigv4 para firmar las solicitudes a una base de datos habilitada para la autenticación de IAM.
- Utiliza la función retry() del módulo de utilidad asíncrona de código abierto para gestionar los intentos de retroceso y reintento.
- Los pasos del terminal de Gremlin devuelven una `promise` de JavaScript (consulte la documentación de TinkerPop). En el caso de `next()`, se trata de una tupla de `{value, done}`.
- Los errores de conexión aparecen en el controlador y se solucionan mediante una lógica de retroceso y reintento de acuerdo con las recomendaciones que aquí se describen, con una excepción. Hay un tipo de problema de conexión que el controlador no considera una excepción y que, por lo tanto, no puede solucionarse con esta lógica.

El problema es que si una conexión se cierra después de que un controlador envíe una solicitud, pero antes de que el controlador reciba una respuesta, la consulta parece completarse, aunque devuelve un valor nulo. En lo que respecta al cliente de la función de lambda, la función parece completarse correctamente, pero con una respuesta vacía.

El impacto de este problema depende del tratamiento que dé la aplicación a una respuesta vacía. Algunas aplicaciones pueden tratar una respuesta vacía de una solicitud de lectura como un error, pero otras pueden tratarla erróneamente como un resultado vacío.

Las solicitudes de escritura que tengan este problema de conexión también devolverán una respuesta vacía. ¿Una invocación correcta con una respuesta vacía indica éxito o error? Si el cliente que invoca una función de escritura simplemente considera que la invocación correcta de la función significa que se ha realizado la escritura en la base de datos, en lugar de inspeccionar el cuerpo de la respuesta, puede parecer que el sistema ha perdido datos.

Este problema se debe a la forma en que el controlador trata los eventos emitidos por el socket subyacente. Cuando el socket de red subyacente se cierra con un error `ECONNRESET`, el WebSocket utilizado por el controlador se cierra y emite un evento `'ws close'`. Sin embargo, no hay nada en el controlador que permita gestionar ese evento de forma que pueda utilizarse para generar una excepción. Como resultado, la consulta simplemente desaparece.

Para solucionar este problema, la función de lambda de ejemplo que se muestra aquí añade un controlador de eventos `'ws close'` que lanza una excepción al controlador al crear una conexión remota. Sin embargo, esta excepción no se genera a lo largo de la ruta de solicitud-respuesta de la consulta de Gremlin y, por lo tanto, no se puede utilizar para desencadenar ninguna lógica de retroceso y reintento dentro de la propia función de Lambda. En cambio, la excepción lanzada por el controlador de eventos `'ws close'` da como resultado una excepción no administrada que provoca un error en la invocación de Lambda. Esto permite al cliente que invoca la función gestionar el error y volver a intentar la invocación de Lambda si fuera necesario.

Le recomendamos que implemente una lógica de retroceso y reintento en la propia función de Lambda para proteger a los clientes de problemas de conexión intermitentes. Sin embargo, la solución provisional al problema anterior requiere que el cliente también implemente una lógica de reintento para gestionar los errores que se producen como resultado de este problema de conexión en concreto.

Código Javascript

```
const gremlin = require('gremlin');
const async = require('async');
const {getUrlAndHeaders} = require('gremlin-aws-sigv4/lib/utils');

const traversal = gremlin.process.AnonymousTraversalSource.traversal;
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;
const t = gremlin.process.t;
const __ = gremlin.process.statics;

let conn = null;
let g = null;

async function query(context) {

  const id = context.id;

  return g.V(id)
    .fold()
    .coalesce(
      __.unfold(),
      __.addV('User').property(t.id, id)
    )
    .id().next();
}

async function doQuery() {
  const id = Math.floor(Math.random() * 10000).toString();

  let result = await query({id: id});
  return result['value'];
}

exports.handler = async (event, context) => {

  const getConnectionDetails = () => {
    if (process.env['USE_IAM'] == 'true'){
       return getUrlAndHeaders(
         process.env['NEPTUNE_ENDPOINT'],
         process.env['NEPTUNE_PORT'],
         {},
         '/gremlin',
```

```
        'wss');
    } else {
      const database_url = 'wss://' + process.env['NEPTUNE_ENDPOINT'] + ':' +
process.env['NEPTUNE_PORT'] + '/gremlin';
      return { url: database_url, headers: {}};
    }
 };


 const createRemoteConnection = () => {
   const { url, headers } = getConnectionDetails();

   const c = new DriverRemoteConnection(
     url,
     {
       mimeType: 'application/vnd.gremlin-v2.0+json',
       headers: headers
     });

    c._client._connection.on('close', (code, message) => {
        console.info(`close - ${code} ${message}`);
        if (code == 1006){
          console.error('Connection closed prematurely');
          throw new Error('Connection closed prematurely');
        }
      });

    return c;
 };

 const createGraphTraversalSource = (conn) => {
   return traversal().withRemote(conn);
 };

 if (conn == null){
   console.info("Initializing connection")
   conn = createRemoteConnection();
   g = createGraphTraversalSource(conn);
 }

 return async.retry(
   {
     times: 5,
     interval: 1000,
```

```
      errorFilter: function (err) {

        // Add filters here to determine whether error can be retried
        console.warn('Determining whether retriable error: ' + err.message);

        // Check for connection issues
        if (err.message.startsWith('WebSocket is not open')){
          console.warn('Reopening connection');
          conn.close();
          conn = createRemoteConnection();
          g = createGraphTraversalSource(conn);
          return true;
        }

        // Check for ConcurrentModificationException
        if (err.message.includes('ConcurrentModificationException')){
          console.warn('Retrying query because of ConcurrentModificationException');
          return true;
        }

        // Check for ReadOnlyViolationException
        if (err.message.includes('ReadOnlyViolationException')){
          console.warn('Retrying query because of ReadOnlyViolationException');
          return true;
        }

        return false;
      }

    },
    doQuery);
};
```

Ejemplo de función de Lambda de Python para Amazon Neptune

Estos son algunos aspectos que se deben tener en cuenta sobre la siguiente función de AWS Lambda de ejemplo de Python:

- Utiliza el módulo de retroceso.
- Establece `pool_size=1` para evitar la creación de un grupo de conexiones innecesario.
- Establece `message_serializer=serializer.GraphSONSerializersV2d0()`.

```
import os, sys, backoff, math
from random import randint
from gremlin_python import statics
from gremlin_python.driver.driver_remote_connection import DriverRemoteConnection
from gremlin_python.driver.protocol import GremlinServerError
from gremlin_python.driver import serializer
from gremlin_python.process.anonymous_traversal import traversal
from gremlin_python.process.graph_traversal import __
from gremlin_python.process.strategies import *
from gremlin_python.process.traversal import T
from aiohttp.client_exceptions import ClientConnectorError
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.credentials import ReadOnlyCredentials
from types import SimpleNamespace

import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)


reconnectable_err_msgs = [
    'ReadOnlyViolationException',
    'Server disconnected',
    'Connection refused',
    'Connection was already closed',
    'Connection was closed by server',
    'Failed to connect to server: HTTP Error code 403 - Forbidden'
]

retriable_err_msgs = ['ConcurrentModificationException'] + reconnectable_err_msgs

network_errors = [OSError, ClientConnectorError]

retriable_errors = [GremlinServerError, RuntimeError, Exception] + network_errors

def prepare_iamdb_request(database_url):

    service = 'neptune-db'
    method = 'GET'

    access_key = os.environ['AWS_ACCESS_KEY_ID']
    secret_key = os.environ['AWS_SECRET_ACCESS_KEY']
```

```
    region = os.environ['AWS_REGION']
    session_token = os.environ['AWS_SESSION_TOKEN']

    creds = SimpleNamespace(
        access_key=access_key, secret_key=secret_key, token=session_token,
 region=region,
    )

    request = AWSRequest(method=method, url=database_url, data=None)
    SigV4Auth(creds, service, region).add_auth(request)

    return (database_url, request.headers.items())

def is_retriable_error(e):

    is_retriable = False
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_retriable = True
    else:
        is_retriable = any(retriable_err_msg in err_msg for retriable_err_msg in
 retriable_err_msgs)

    logger.error('error: [{}] {}'.format(type(e), err_msg))
    logger.info('is_retriable: {}'.format(is_retriable))

    return is_retriable

def is_non_retriable_error(e):
    return not is_retriable_error(e)

def reset_connection_if_connection_issue(params):

    is_reconnectable = False

    e = sys.exc_info()[1]
    err_msg = str(e)

    if isinstance(e, tuple(network_errors)):
        is_reconnectable = True
    else:
        is_reconnectable = any(reconnectable_err_msg in err_msg for
 reconnectable_err_msg in reconnectable_err_msgs)
```

```
    logger.info('is_reconnectable: {}'.format(is_reconnectable))

    if is_reconnectable:
        global conn
        global g
        conn.close()
        conn = create_remote_connection()
        g = create_graph_traversal_source(conn)

@backoff.on_exception(backoff.constant,
    tuple(retriable_errors),
    max_tries=5,
    jitter=None,
    giveup=is_non_retriable_error,
    on_backoff=reset_connection_if_connection_issue,
    interval=1)
def query(**kwargs):

    id = kwargs['id']

    return (g.V(id)
        .fold()
        .coalesce(
            __.unfold(),
            __.addV('User').property(T.id, id)
        )
        .id().next())

def doQuery(event):
    return query(id=str(randint(0, 10000)))

def lambda_handler(event, context):
    result = doQuery(event)
    logger.info('result - {}'.format(result))
    return result

def create_graph_traversal_source(conn):
    return traversal().withRemote(conn)

def create_remote_connection():
    logger.info('Creating remote connection')

    (database_url, headers) = connection_info()
```

```
    return DriverRemoteConnection(
        database_url,
        'g',
        pool_size=1,
        message_serializer=serializer.GraphSONSerializersV2d0(),
        headers=headers)

def connection_info():

    database_url = 'wss://{}:{}/gremlin'.format(os.environ['neptuneEndpoint'],
 os.environ['neptunePort'])

    if 'USE_IAM' in os.environ and os.environ['USE_IAM'] == 'true':
        return prepare_iamdb_request(database_url)
    else:
        return (database_url, {})

conn = create_remote_connection()
g = create_graph_traversal_source(conn)
```

Estos son algunos resultados de ejemplo, que muestran periodos alternos de carga elevada y ligera:


GremlinWebSocketOpenConnections
Count
261
131
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
GremlinRequestsPreSecond
Count/Second
861
430
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
neptunedbinstance-l22qsahgvgvl
replica-python-lambdas-neptunebasestack-5rzhq9jvp2ri
Lambda-ConcurrentExecutions
Count
86
43.5
1
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship social-b6501310_add-follows-relationship
social-b6501310_create-user social-b6501310_create-user
social-b6501310_get-recent-follows social-b6501310_get-recent-follows
social-b6501310_get-user-details social-b6501310_get-user-details
Lambda-P99-Latency
Milliseconds
2.10k
1.05k
5.46
06:05 06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows
Workload
Various units
Count
2.17k
1.09k
9e-3
147k
73.7k
2
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
P99Latency
AverageThroughput
Succeeded
Errors
Count
1
0.5
0
06:10 06:15 06:20 06:25 06:30 06:35 06:40 06:45 06:50 06:55 07:00 07:05
social-b6501310_add-follows-relationship
social-b6501310_create-user
social-b6501310_get-recent-follows
social-b6501310_get-user-details
social-b6501310_recommend-new-follows

Amazon Neptune ML para el machine learning en gráficos

A menudo hay información valiosa en grandes conjuntos de datos conectados que puede resultar difícil de extraer mediante consultas basadas únicamente en la intuición humana. Las técnicas de machine learning (ML) pueden ayudar a encontrar correlaciones ocultas en gráficos con miles de millones de relaciones. Estas correlaciones pueden ser útiles para recomendar productos, predecir la solvencia crediticia, identificar el fraude y muchas otras cosas.

La característica Neptune ML permite crear y entrenar modelos útiles de machine learning en gráficos de gran tamaño en cuestión de horas en lugar de semanas. Para ello, Neptune ML utiliza la tecnología de redes neuronales de gráficos (GNN) con tecnología de Amazon SageMaker y la biblioteca Deep Graph Library (DGL) (que es de código abierto). Las redes neuronales de gráficos son un campo emergente de la inteligencia artificial (consulte, por ejemplo, A Comprehensive Survey on Graph Neural Networks [Una encuesta exhaustiva sobre las redes neuronales de gráficos]). Para ver un tutorial práctico sobre el uso de las GNN con la DGL, consulte Learning graph neural networks with Deep Graph Library (Obtener información sobre las redes neuronales de gráficos con Deep Graph Library).

> **Note**
>
> Los vértices de los gráficos se identifican en los modelos de Neptune ML como “nodos”. Por ejemplo, la clasificación de vértices utiliza un modelo de machine learning de clasificación de nodos y la regresión de vértices utiliza un modelo de regresión de nodos.

Qué puede hacer Neptune ML

Neptune admite tanto la inferencia transductiva, que devuelve predicciones que se precalcularon en el momento del entrenamiento, en función de los datos del gráfico en ese momento, como la inferencia inductiva, que devuelve el procesamiento de datos aplicado y la evaluación del modelo en tiempo real, en función de los datos actuales. Consulte La diferencia entre la inferencia inductiva y la transductiva.

Neptune ML puede entrenar modelos de machine learning para que admitan cinco categorías diferentes de inferencia:

Tipos de tareas de inferencia compatibles actualmente con Neptune ML

- Clasificación de nodos: predice la característica categórica de una propiedad de vértice.

  Por ejemplo, con la película Cadena perpetua Neptune ML puede predecir su propiedad `genre` como `story` a partir de un conjunto candidato de `[story, crime, action, fantasy, drama, family, ...]`.

  Existen dos tipos de tareas de clasificación de nodos:

  - Clasificación de clase única: en este tipo de tarea, cada nodo tiene solo una característica de destino. Por ejemplo, la propiedad `Place_of_birth` de `Alan Turing` tiene el valor UK.
  - Clasificación de varias clases: en este tipo de tarea, cada nodo puede tener más de una característica de destino. Por ejemplo, la propiedad `genre` de la película El padrino tiene los valores `crime` y `story`.
- Regresión de nodos: predice una propiedad numérica de un vértice.

  Por ejemplo, con la película Vengadores: Endgame, Neptune ML puede predecir que su propiedad `popularity` tiene un valor de `5.0`.
- Clasificación de bordes: predice la característica categórica de una propiedad de borde.

  Existen dos tipos de tareas de clasificación de bordes:

  - Clasificación de clase única: en este tipo de tarea, cada borde tiene solo una característica de destino. Por ejemplo, un borde de valoración entre un usuario y una película pueden tener la propiedad `liked` con un valor “Sí” o “No”.
  - Clasificación de varias clases: en este tipo de tarea, cada borde puede tener más de una característica de destino. Por ejemplo, las valoraciones entre un usuario y una película pueden tener varios valores en la etiqueta de propiedad, como, por ejemplo, “Divertido”, “Conmovedor”, “Escalofriante”, etc.
- Regresión de bordes: predice una propiedad numérica de un borde.

  Por ejemplo, un borde de valoración entre un usuario y una película puede tener la propiedad numérica `score` para la que Neptune ML puede predecir un valor con un usuario y una película.
- Predicción de enlaces: predice los nodos de destino más probables par un nodo de origen y un borde de salida en concreto, o los nodos de origen más probables para un nodo de destino y un borde de entrada en concreto.

Por ejemplo, con un gráfico de conocimientos sobre enfermedades relacionadas con las drogas, con `Aspirin` como nodo de origen y `treats` como borde de salida, Neptune ML puede predecir los nodos de destino más relevantes como `heart disease`, `fever`, etc.

O bien, con el gráfico de conocimientos de Wikimedia, con `President-of` como borde o relación y `United-States` como nodo de destino, Neptune ML puede predecir los dirigentes más relevantes como `George Washington`, `Abraham Lincoln`, `Franklin D. Roosevelt`, etc.

> **Note**
>
> La clasificación de nodos y la clasificación de bordes solo admiten valores de cadena. Esto significa que no se admiten valores de propiedades numéricas como `0` o `1`, aunque sí se admiten los equivalentes de cadena `"0"` y `"1"`. Del mismo modo, los valores de la propiedad Boolean `true` y `false` no funcionan, pero sí lo hacen `"true"` y `"false"`.

Con Neptune ML, puede utilizar modelos de machine learning que se dividen en dos categorías generales:

Tipos de modelos de machine learning compatibles actualmente con Neptune ML

- Modelos de redes neuronales de gráficos (GNN): incluyen redes convolucionales de gráficos relacionales (R-GCN). Los modelos GNN funcionan para los tres tipos de tareas anteriores.
- Modelos de incrustación de gráficos de conocimientos (KGE): incluyen los modelos `TransE`, `DistMult` y `RotatE`. Solo funcionan para la predicción de enlaces.

Modelos definidos por el usuario: Neptune ML también le permite proporcionar su propia implementación de modelo personalizado para todos los tipos de tareas indicados anteriormente. Puede usar el kit de herramientas de Neptune ML para desarrollar y probar la implementación de un modelo personalizado basado en Python antes de usar la API de entrenamiento de Neptune ML con su modelo. Consulte Modelos personalizados de Neptune ML para obtener información sobre cómo estructurar y organizar su implementación para que sea compatible con la infraestructura de entrenamiento de Neptune ML.

Configuración de Neptune ML

La forma más sencilla de comenzar a usar Neptune ML es utilizar la plantilla de inicio rápido de AWS CloudFormation. Esta plantilla instala todos los componentes necesarios, incluido un nuevo clúster de base de datos de Neptune, todos los roles de IAM necesarios y un nuevo cuaderno de gráficos de Neptune para facilitar el trabajo con Neptune ML.

También puede instalar manualmente Neptune ML, tal y como se explica en Configuración de Neptune ML sin utilizar la plantilla de AWS CloudFormation de inicio rápido.

Uso de la plantilla de AWS CloudFormation de Neptune ML para empezar a usar rápidamente un nuevo clúster de base de datos

La forma más sencilla de comenzar a usar Neptune ML es utilizar la plantilla de inicio rápido de AWS CloudFormation. Esta plantilla instala todos los componentes necesarios, incluido un nuevo clúster de base de datos de Neptune, todos los roles de IAM necesarios y un nuevo cuaderno de gráficos de Neptune para facilitar el trabajo con Neptune ML.

Para crear la pila de inicio rápido de Neptune ML

1. Para lanzar la pila de AWS CloudFormation en la consola de AWS CloudFormation, elija uno de los botones Lanzar pila de la tabla siguiente:

Region	Visualización	Ver en Designer	Lanzar
Este de EE. UU. (Norte de Virginia)	Ver	Ver en Designer	
Este de EE. UU. (Ohio)	Ver	Ver en Designer	
Oeste de EE. UU. (Norte de California)	Ver	Ver en Designer	
Oeste de EE. UU. (Oregón)	Ver	Ver en Designer	
Canadá (Centro)	Ver	Ver en Designer	
América del Sur (São Paulo)	Ver	Ver en Designer	
Europa (Estocolmo)	Ver	Ver en Designer	
Europa (Irlanda)	Ver	Ver en Designer	

Region	Visualización	Ver en Designer	Lanzar
Europa (Londres)	Ver	Ver en Designer	
Europa (París)	Ver	Ver en Designer	
Europa (Fráncfort)	Ver	Ver en Designer	
Medio Oriente (Baréin)	Ver	Ver en Designer	
Medio Oriente (EAU)	Ver	Ver en Designer	
Israel (Tel Aviv)	Ver	Ver en Designer	
África (Ciudad del Cabo)	Ver	Ver en Designer	
Asia-Pacífico (Hong Kong)	Ver	Ver en Designer	
Asia-Pacífico (Tokio)	Ver	Ver en Designer	
Asia-Pacífico (Seúl)	Ver	Ver en Designer	
Asia-Pacífico (Singapur)	Ver	Ver en Designer	
Asia-Pacífico (Sídney)	Ver	Ver en Designer	
Asia-Pacífico (Bombay)	Ver	Ver en Designer	
China (Pekín)	Ver	Ver en Designer	

Region	Visualización	Ver en Designer	Lanzar
China (Ningxia)	Ver	Ver en Designer	Launch Stack
AWS GovCloud (Oeste de EE. UU.)	Ver	Ver en Designer	Launch Stack

2. En la página Select Template, elija Next.
3. En la página Specify Details (Especificar detalles), elija Next (Siguiente).
4. En la página Opciones, seleccione Siguiente.
5. En la página Revisar, hay dos casillas de verificación que debe marcar:

   - La primera reconoce que AWS CloudFormation podría crear recursos de IAM con nombres personalizados.
   - La segunda reconoce que AWS CloudFormation podría necesitar la capacidad de `CAPABILITY_AUTO_EXPAND` para la nueva pila. `CAPABILITY_AUTO_EXPAND` permite de forma explícita que AWS CloudFormation amplíe las macros automáticamente al crear la pila, sin revisión previa.

     Los clientes suelen crear un conjunto de cambios a partir de una plantilla procesada para que los cambios realizados por las macros puedan revisarse antes de crear la pila. Para obtener más información, consulte la API CreateStack de AWS CloudFormation.

   A continuación, elija Crear.

La plantilla de inicio rápido crea y configura las siguientes opciones:

- Un clúster de base de datos de Neptune.
- Los roles de IAM necesarios (y los adjunta).
- El grupo de seguridad de Amazon EC2 necesario.
- Los puntos de conexión de VPC de SageMaker necesarios.
- Un grupo de parámetros de clúster de base de datos para Neptune ML.
- Los parámetros necesarios para este grupo.
- Un cuaderno de SageMaker con ejemplos de cuadernos rellenados previamente para Neptune ML. Tenga en cuenta que no todos los tamaños de instancia están disponibles en todas las

regiones, por lo que debe asegurarse de que el tamaño de instancia del cuaderno seleccionado sea compatible con su región.

- El servicio Neptune-Export.

Cuando la pila de inicio rápido esté lista, vaya al cuaderno de SageMaker que creó la plantilla y consulte los ejemplos rellenados previamente. Le ayudarán a descargar conjuntos de datos de ejemplo que se pueden utilizar para experimentar con las capacidades de Neptune ML.

También pueden ahorrarle mucho tiempo al utilizar Neptune ML. Por ejemplo, consulte el comando mágico de línea %neptune_ml y el comando mágico de celda %%neptune_ml que son compatibles con estos cuadernos.

También puede usar el siguiente comando AWS CLI para ejecutar la plantilla de AWS CloudFormation de inicio rápido:

```
aws cloudformation create-stack \
  --stack-name neptune-ml-fullstack-$(date '+%Y-%m-%d-%H-%M') \
  --template-url https://aws-neptune-customer-samples.s3.amazonaws.com/v2/
cloudformation-templates/neptune-ml-nested-stack.json \
  --parameters ParameterKey=EnableIAMAuthOnExportAPI,ParameterValue=(true if you have
 IAM auth enabled, or false otherwise) \
               ParameterKey=Env,ParameterValue=test$(date '+%H%M')\
  --capabilities CAPABILITY_IAM \
  --region (the AWS region, like us-east-1) \
  --disable-rollback \
  --profile (optionally, a named CLI profile of yours)
```

Configuración de Neptune ML sin utilizar la plantilla de AWS CloudFormation de inicio rápido

1. Comenzar con un clúster de base de datos de Neptune que funcione

Si no utiliza la plantilla de inicio rápido de AWS CloudFormation para configurar Neptune ML, necesitará un clúster de base de datos de Neptune existente con el que trabajar. Si lo desea, puede usar uno que ya tenga, clonar uno que ya esté utilizando o crear uno nuevo (consulte Creación de un clúster de base de datos).

2. Instalación del servicio Neptune-Export

Si aún no lo ha hecho, instale el servicio Neptune-Export, tal y como se explica en Uso del servicio Neptune-Export para exportar datos de Neptune.

Añada una regla de entrada al grupo de seguridad de `NeptuneExportSecurityGroup` que cree la instalación, con la siguiente configuración:

- Tipo: `Custom TCP`
- Protocolo: `TCP`
- Intervalo de puertos: `80 - 443`
- Origen: *`(ID del grupo de seguridad del clúster de base de datos de Neptune)`*

3. Creación de un rol de IAM de **`NeptuneLoadFromS3`** personalizado

Si aún no lo ha hecho, cree un rol de IAM de `NeptuneLoadFromS3` personalizado, tal como se explica en Creación de un rol de IAM para acceder a Amazon S3.

Para crear un rol de **`NeptuneSageMakerIAMRole`** personalizado

Utilice la consola de IAM para crear un `NeptuneSageMakerIAMRole` personalizado mediante la siguiente política:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:CreateNetworkInterface",
```

```
        "ec2:CreateNetworkInterfacePermission",
        "ec2:CreateVpcEndpoint",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "sagemaker.amazonaws.com"
          ]
        }
      },
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:CreateGrant",
```

```
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:*:*:key/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:AddTags",
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateHyperParameterTuningJob",
        "sagemaker:CreateModel",
        "sagemaker:CreateProcessingJob",
        "sagemaker:CreateTrainingJob",
        "sagemaker:CreateTransformJob",
        "sagemaker:DeleteEndpoint",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteModel",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeHyperParameterTuningJob",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeProcessingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:InvokeEndpoint",
        "sagemaker:ListTags",
        "sagemaker:ListTrainingJobsForHyperParameterTuningJob",
        "sagemaker:StopHyperParameterTuningJob",
```

```
        "sagemaker:StopProcessingJob",
        "sagemaker:StopTrainingJob",
        "sagemaker:StopTransformJob",
        "sagemaker:UpdateEndpoint",
        "sagemaker:UpdateEndpointWeightsAndCapacities"
      ],
      "Resource": [
        "arn:aws:sagemaker:*:*:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:ListEndpointConfigs",
        "sagemaker:ListEndpoints",
        "sagemaker:ListHyperParameterTuningJobs",
        "sagemaker:ListModels",
        "sagemaker:ListProcessingJobs",
        "sagemaker:ListTrainingJobs",
        "sagemaker:ListTransformJobs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:AbortMultipartUpload",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Al crear este rol, edite la relación de confianza para que quede como se indica a continuación:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "rds.amazonaws.com",
          "sagemaker.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Por último, copie el ARN asignado a este nuevo rol de `NeptuneSageMakerIAMRole`.

> **Important**
>
> - Asegúrese de que los permisos de Amazon S3 del `NeptuneSageMakerIAMRole` coincidan con los anteriores.
> - El ARN universal, `arn:aws:s3:::*`, se utiliza para el recurso de Amazon S3 en la política anterior. Si por algún motivo no se puede utilizar el ARN universal, se deben añadir a la sección de recursos `arn:aws:s3:::graphlytics*` y el ARN de cualquier otro recurso de Amazon S3 del cliente que utilicen los comandos de Neptune ML.

Configuración de un clúster de base de datos para habilitar Neptune ML

Para configurar un clúster de base de datos para Neptune ML

1. En la consola de Neptune, vaya a Grupos de parámetros y, a continuación, al grupo de parámetros del clúster de base de datos asociado al clúster de base de datos que utilizará. Establezca el parámetro `neptune_ml_iam_role` en el ARN asignado al rol de `NeptuneSageMakerIAMRole` que acaba de crear.
2. Vaya a Bases de datos y, a continuación, seleccione el clúster de base de datos que utilizará para Neptune ML. Seleccione Acciones y, a continuación, Administrar roles de IAM.

3. En la página Administrar roles de IAM, seleccione Añadir rol y añada `NeptuneSageMakerIAMRole`. A continuación, añada el rol de `NeptuneLoadFromS3`.
4. Reinicie la instancia de escritor del clúster de base de datos.

Creación de dos puntos de conexión de SageMaker en la VPC de Neptune

Por último, para que el motor Neptune acceda a las API de administración de SageMaker necesarias, debe crear dos puntos de conexión de SageMaker en la VPC de Neptune, tal como se explica en Creación de dos puntos de conexión para SageMaker en la VPC de Neptune.

Configuración manual de un cuaderno de Neptune para Neptune ML

Los cuadernos de Neptune SageMaker vienen precargados con una variedad de cuadernos de ejemplo para Neptune ML. Puede obtener una vista previa de estos ejemplos en el repositorio de GitHub de cuadernos gráficos de código abierto.

Puede utilizar uno de los cuadernos de Neptune existentes o, si lo desea, puede crear uno propio siguiendo las instrucciones que se indican en Uso del entorno de trabajo de Neptune para alojar los cuadernos de Neptune.

También puede configurar un cuaderno de Neptune predeterminado para usarlo con Neptune ML. Para ello, siga estos pasos:

Modificación de un cuaderno para Neptune ML

1. Abra la consola de Amazon SageMaker en https://console.aws.amazon.com/sagemaker/.
2. En el panel de navegación de la izquierda, seleccione Cuaderno y, a continuación, Instancias de cuaderno. Busque el nombre del cuaderno de Neptune que desee utilizar para Neptune ML y selecciónelo para ir a su página de detalles.
3. Si la instancia del cuaderno se está ejecutando, seleccione el botón Detener, situado en la parte superior derecha de la página de detalles del cuaderno.
4. En Configuración de instancias de cuaderno, en Configuración del ciclo de vida, seleccione el enlace para abrir la página del ciclo de vida del cuaderno.
5. Seleccione Editar en la parte superior derecha y, a continuación, Continuar.
6. En la pestaña Iniciar cuaderno, modifique el script para incluir comandos de exportación adicionales y para rellenar los campos del rol de IAM de Neptune ML y del URI del servicio de exportación, algo parecido a lo siguiente en función del intérprete de comandos:

```
echo "export NEPTUNE_ML_ROLE_ARN=(your Neptune ML IAM role ARN)" >> ~/.bashrc
echo "export NEPTUNE_EXPORT_API_URI=(your export service URI)" >> ~/.bashrc
```

7. Seleccione Update (Actualizar).
8. Vuelva a la página de instancias del cuaderno. En Permisos y cifrado hay un campo para el ARN del rol de IAM. Seleccione el enlace de este campo para ir al rol de IAM con el que se ejecuta esta instancia de cuaderno.
9. Cree una nueva política insertada como esta:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "arn:aws:cloudwatch:[AWS_REGION]:[AWS_ACCOUNT_ID]:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:[AWS_REGION]:[AWS_ACCOUNT_ID]:*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::*",
      "Effect": "Allow"
    },
    {
      "Action": "execute-api:Invoke",
```

```
      "Resource": "arn:aws:execute-api:[AWS_REGION]:[AWS_ACCOUNT_ID]:*/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "sagemaker:CreateModel",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateEndpoint",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DeleteModel",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteEndpoint"
      ],
      "Resource": "arn:aws:sagemaker:[AWS_REGION]:[AWS_ACCOUNT_ID]:*/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "[YOUR_NEPTUNE_ML_IAM_ROLE_ARN]"",
      "Effect": "Allow"
    }
  ]
}
```

10. Guarde esta nueva política y asóciela al rol de IAM del paso 8.
11. Seleccione Iniciar en la parte superior derecha de la página de detalles de la instancia del cuaderno de SageMaker para iniciar la instancia del cuaderno.

Uso de AWS CLI para configurar Neptune ML en un clúster de base de datos

Además de la plantilla de inicio rápido de AWS CloudFormation y la AWS Management Console, también puede configurar Neptune ML mediante la AWS CLI.

1. Creación de un grupo de parámetros de clúster de base de datos para el nuevo clúster de Neptune ML

Los siguientes comandos de la AWS CLI crean un nuevo grupo de parámetros de clúster de base de datos y lo configuran para que funcione con Neptune ML:

Para crear y configurar un grupo de parámetros de clúster de base de datos de Neptune ML

1. Cree un nuevo grupo de parámetros del clúster de base de datos:

```
aws neptune create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --db-parameter-group-family neptune1
  --description "(description of your machine learning project)" \
  --region (AWS region, such as us-east-1)
```

2. Cree un conjunto de parámetros de clúster de base de datos de `neptune_ml_iam_role` con el ARN de `SageMakerExcecutionIAMRole` para el clúster de base de datos que utilizará cuando llame a SageMaker para crear trabajos y obtener predicciones a partir de modelos de ML alojados:

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --parameters "ParameterName=neptune_ml_iam_role, \
                ParameterValue=ARN of the SageMakerExcecutionIAMRole, \
                Description=NeptuneMLRole, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

   Si se establece este parámetro, Neptune podrá acceder a SageMaker sin que tenga que transferir el rol en cada llamada.

   Para obtener más información sobre cómo crear el `SageMakerExcecutionIAMRole`, consulte Para crear un rol de `NeptuneSageMakerIAMRole` personalizado.

3. Por último, utilice `describe-db-cluster-parameters` para comprobar que todos los parámetros del nuevo grupo de parámetros del clúster de bases de datos estén configurados tal y como desee:

```
aws neptune describe-db-cluster-parameters \
  --db-cluster-parameter-group-name (name of the new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Adjuntar el nuevo grupo de parámetros de clúster de base de datos al clúster de base de datos que usará con Neptune ML

Ahora puede adjuntar el nuevo grupo de parámetros del clúster de base de datos que acaba de crear a un clúster de base de datos existente mediante el siguiente comando:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (the name of your existing DB cluster) \
  --apply-immediately
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --region (AWS region, such as us-east-1)
```

Para que todos los parámetros entren en vigor, puede reiniciar el clúster de base de datos:

```
aws neptune reboot-db-instance
  --db-instance-identifier (name of the primary instance of your DB cluster) \
  --profile (name of your AWS profile to use) \
  --region (AWS region, such as us-east-1)
```

O bien, si va a crear un nuevo clúster de base de datos para usarlo con Neptune ML, puede usar el siguiente comando para crear el clúster con el nuevo grupo de parámetros adjunto y, a continuación, crear una nueva instancia principal (escritor):

```
cluster-name=(the name of the new DB cluster)
aws neptune create-db-cluster
  --db-cluster-identifier ${cluster-name}
  --engine graphdb \
  --engine-version 1.0.4.1 \
  --db-cluster-parameter-group-name (name of your new DB cluster parameter group) \
  --db-subnet-group-name (name of the subnet to use) \
  --region (AWS region, such as us-east-1)
```

```
aws neptune create-db-instance
  --db-cluster-identifier ${cluster-name}
  --db-instance-identifier ${cluster-name}-i \
  --db-instance-class (the instance class to use, such as db.r5.xlarge)
  --engine graphdb \
  --region (AWS region, such as us-east-1)
```

Adjuntar el **NeptuneSageMakerIAMRole** al clúster de base de datos para que pueda acceder a los recursos de SageMaker y Amazon S3

Por último, siga las instrucciones indicadas en Para crear un rol de `NeptuneSageMakerIAMRole` personalizado para crear un rol de IAM que permita que el clúster de base de datos se comunique con SageMaker y Amazon S3. A continuación, utilice el siguiente comando para adjuntar el rol de `NeptuneSageMakerIAMRole` que creó al clúster de base de datos:

```
aws neptune add-role-to-db-cluster
  --db-cluster-identifier ${cluster-name}
  --role-arn arn:aws:iam::(the ARN number of the role's ARN):role/NeptuneMLRole \
  --region (AWS region, such as us-east-1)
```

Creación de dos puntos de conexión para SageMaker en la VPC de Neptune

Neptune ML necesita dos puntos de conexión de SageMaker en la VPC del clúster de base de datos de Neptune:

- `com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime`
- `com.amazonaws.(AWS region, like us-east-1).sagemaker.api`

Si no ha utilizado la plantilla de AWS CloudFormation de inicio rápido, que los crea automáticamente, puede utilizar los siguientes comandos de la AWS CLI para crearlos:

Este crea el punto de conexión de `sagemaker.runtime`:

```
create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
```

```
  --private-dns-enabled
```

Este otro crea el punto de conexión de `sagemaker.api`:

```
aws create-vpc-endpoint
  --vpc-id (the ID of your Neptune DB cluster's VPC)
  --service-name com.amazonaws.(AWS region, like us-east-1).sagemaker.api
  --subnet-ids (the subnet ID or IDs that you want to use)
  --security-group-ids (the security group for the endpoint network interface, or omit
 to use the default)
  --private-dns-enabled
```

También puede usar la consola de la VPC para crear estos puntos de conexión. Consulte Protección de llamadas de predicciones en Amazon SageMaker con AWS PrivateLink y Protección de todas las llamadas a la API de Amazon SageMaker con AWS PrivateLink.

Creación de un parámetro de punto de conexión de inferencia de SageMaker en el grupo de parámetros de clúster de base de datos

Para evitar tener que especificar el punto de conexión de inferencia de SageMaker del modelo que está utilizando en cada consulta que le realice, cree un parámetro de clúster de base de datos denominado `neptune_ml_endpoint` en el grupo de parámetros del clúster de base de datos de Neptune ML. Establezca el parámetro en el `id` del punto de conexión de la instancia en cuestión.

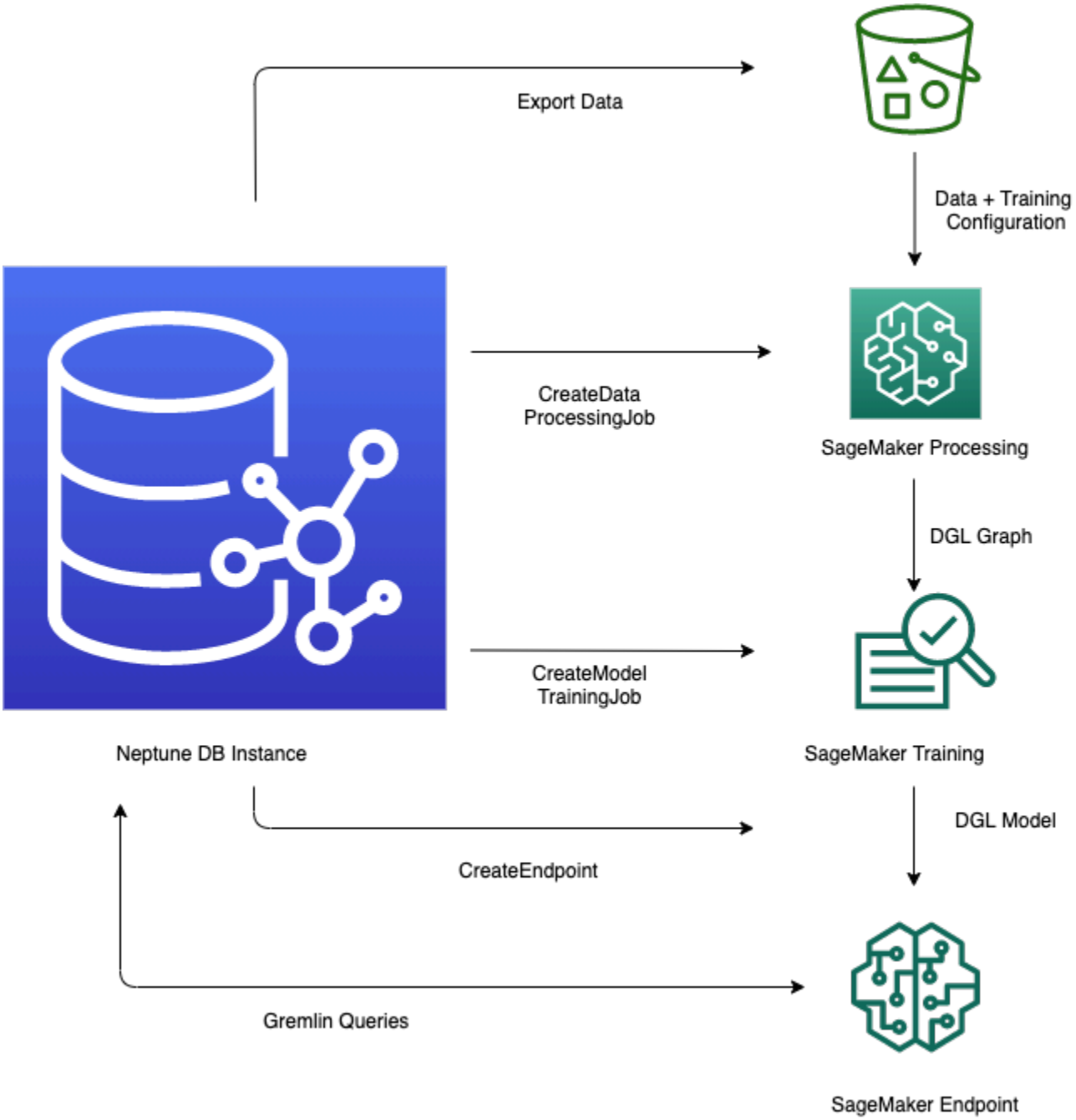
Puede usar el siguiente comando de la AWS CLI para hacerlo:

```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name neptune-ml-demo \
  --parameters "ParameterName=neptune_ml_endpoint, \
                ParameterValue=(the name of the SageMaker inference endpoint you want
 to query), \
                Description=NeptuneMLEndpoint, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

Información general sobre cómo utilizar la característica Neptune ML

Inicio del flujo de trabajo para usar Neptune ML

Para empezar, el uso de la característica Neptune ML en Amazon Neptune suele implicar los cinco pasos siguientes:



1. Configuración y exportación de datos: el paso de exportación de datos utiliza el servicio Neptune-Export o la herramienta de línea de comandos de `neptune-export` para exportar datos de Neptune a Amazon Simple Storage Service (Amazon S3) en formato CSV. Al mismo tiempo, se genera automáticamente un archivo de configuración denominado `training-data-configuration.json`, que especifica cómo se pueden cargar los datos exportados en un gráfico entrenable.
2. Preprocesamiento de datos: en este paso, el conjunto de datos exportado se procesa previamente mediante técnicas estándar para prepararlo para el entrenamiento de modelos. La normalización de características se puede realizar para datos numéricos, y las características de texto se pueden codificar mediante `word2vec`. Al final de este paso, se genera un gráfico DGL a partir del conjunto de datos exportado para utilizarlo en el paso de entrenamiento de modelos.

   Este paso se implementa mediante un trabajo de procesamiento de SageMaker en su cuenta, y los datos resultantes se almacenan en la ubicación de Amazon S3 que haya especificado.
3. Entrenamiento de modelos: el paso de entrenamiento de modelos entrena el modelo de machine learning que se utilizará para las predicciones.

   El entrenamiento de modelos se realiza en dos etapas:

   - La primera etapa utiliza un trabajo de procesamiento de SageMaker para generar un conjunto de configuraciones de estrategias de entrenamiento de modelos que especifica qué tipo de modelo y qué rangos de hiperparámetros del modelo se utilizarán para el entrenamiento.
   - La segunda etapa utiliza un trabajo de ajuste del modelo de SageMaker para probar diferentes configuraciones de hiperparámetros y seleccionar el trabajo de entrenamiento que produjo el modelo con el mejor rendimiento. El trabajo de ajuste ejecuta un número previamente especificado de pruebas de trabajo de entrenamiento de modelos con los datos procesados. Al final de esta etapa, se utilizan los parámetros del modelo entrenado del mejor trabajo de entrenamiento para generar artefactos del modelo para su inferencia.
4. Creación de un punto de conexión de inferencia en Amazon SageMaker: el punto de conexión de inferencia es una instancia de punto de conexión de SageMaker que se lanza con los artefactos del modelo producidos por el mejor trabajo de entrenamiento. Cada modelo está vinculado a un único punto de conexión. El punto de conexión puede aceptar las solicitudes entrantes de la base de datos de gráficos y devolver las predicciones de modelos para las entradas de las solicitudes. Una vez creado el punto de conexión, permanecerá activo hasta que lo elimine.
5. Consulta del modelo de machine learning mediante Gremlin: puede utilizar extensiones del lenguaje de consultas de Gremlin para consultar las predicciones desde el punto de conexión de inferencia.

> **Note**
>
> El entorno de trabajo de Neptune incluye un comando mágico de línea y un comando mágico de celda que pueden ahorrarle mucho tiempo en la administración de estos pasos, es decir:
>
> - %neptune_ml
> - %%neptune_ml

Hacer predicciones basadas en los datos de los gráficos en constante evolución

Con un gráfico que cambia continuamente, es posible que desee crear de forma periódica nuevas predicciones por lotes con datos nuevos. La consulta de predicciones calculadas previamente (inferencia transductiva) puede ser considerablemente más rápida que generar nuevas predicciones sobre la marcha a partir de los datos más recientes (inferencia inductiva). Ambos enfoques son válidos en función de la rapidez con la que cambien los datos y de los requisitos de rendimiento.

La diferencia entre la inferencia inductiva y la transductiva

Al realizar una inferencia transductiva, Neptune busca y devuelve las predicciones que se han calculado previamente en el momento del entrenamiento.

Al realizar una inferencia inductiva, Neptune crea el subgráfico correspondiente y obtiene sus propiedades. A continuación, el modelo DGL GNN aplica el procesamiento de datos y la evaluación del modelo en tiempo real.

Por lo tanto, la inferencia inductiva puede generar predicciones con nodos y bordes que no estaban presentes en el momento del entrenamiento y que reflejan el estado actual del gráfico. Sin embargo, esto se produce a costa de una mayor latencia.

Si el gráfico es dinámico, es posible que desee utilizar la inferencia inductiva para asegurarse de tener en cuenta los datos más recientes, pero si el gráfico es estático, la inferencia transductiva es más rápida y eficaz.

La inferencia inductiva está deshabilitada de forma predeterminada. Si desea habilitarla para una consulta, utilice de la siguiente manera el predicado Neptune#ml.inductiveInference de Gremlin en la consulta:

```
.with( "Neptune#ml.inductiveInference")
```

Flujos de trabajo de inferencias transductivas incrementales

Mientras actualiza los artefactos de modelos (para ello, vuelve a realizar los pasos del uno al tres [desde Configuración y exportación de datos hasta Transformación de modelos]), Neptune ML va admitiendo formas más sencillas de actualizar sus predicciones de ML por lotes con nuevos datos. Una opción es utilizar un flujo de trabajo de modelos incrementales, y otra consiste en utilizar el reentrenamiento de modelos con un inicio en caliente.

Flujo de trabajo de modelos incrementales

En este flujo de trabajo, se actualizan las predicciones de ML sin volver a entrenar el modelo de ML.

> **Note**
>
> Solo puede hacerlo cuando los datos del gráfico se hayan actualizado con nuevos nodos o bordes. Actualmente, no funcionará cuando se eliminen los nodos.

1. Configuración y exportación de datos: este paso es el mismo que en el flujo de trabajo principal.
2. Preprocesamiento incremental de datos: este paso es similar al paso de preprocesamiento de datos del flujo de trabajo principal, pero utiliza la misma configuración de procesamiento utilizada anteriormente, que corresponde a un determinado modelo entrenado.
3. Transformación de modelos: en lugar de un paso de entrenamiento del modelo, este paso de transformación del modelo extrae el modelo entrenado del flujo de trabajo principal y los resultados del paso de preprocesamiento incremental de datos, y genera nuevos artefactos de modelos para utilizarlos en la inferencia. El paso de transformación de modelos inicia un trabajo de procesamiento de SageMaker para realizar el cálculo que genera los artefactos de modelos actualizados.
4. Actualización del punto de conexión de inferencia de Amazon SageMaker: si lo prefiere, si cuenta con un punto de conexión de inferencia existente, este paso actualiza el punto de conexión con los nuevos artefactos de modelos generados por el paso de transformación de modelos. Además, puede crear un nuevo punto de conexión de inferencia con los nuevos artefactos de modelos.

Reentrenamiento de modelos con un inicio en caliente

Con este flujo de trabajo, puede entrenar e implementar un nuevo modelo de ML para hacer predicciones con los datos de los gráficos incrementales, pero basándose en un modelo existente generado mediante el flujo de trabajo principal:

1. Configuración y exportación de datos: este paso es el mismo que en el flujo de trabajo principal.
2. Preprocesamiento incremental de datos: este paso es el mismo que en el flujo de trabajo de inferencia del modelo incremental. Los nuevos datos del gráfico deben procesarse con el mismo método de procesamiento que se utilizó anteriormente para el entrenamiento del modelo.
3. Entrenamiento de modelos con un inicio en caliente: el entrenamiento de modelos es similar a lo que ocurre en el flujo de trabajo principal, pero puede acelerar la búsqueda de hiperparámetros del modelo con el fin de sacar partido de la información de la tarea anterior de entrenamiento con modelos.
4. Actualización del punto de conexión de inferencia de Amazon SageMaker: este paso es el mismo que en el flujo de trabajo de inferencia de modelos incrementales.

Flujos de trabajo de modelos personalizados en Neptune ML

Neptune ML le permite implementar y entrenar sus propios modelos personalizados para cualquiera de las tareas compatibles con Neptune ML. El flujo de trabajo para desarrollar e implementar un modelo personalizado es básicamente el mismo que el de los modelos integrados, con algunas diferencias, tal y como se explica en Flujo de trabajo de modelos personalizados.

Selección de instancias para las etapas de Neptune ML

Las diferentes etapas del procesamiento de Neptune ML utilizan distintas instancias de SageMaker. A continuación, explicaremos cómo elegir el tipo de instancia adecuado para cada etapa. Puede encontrar información sobre los tipos de instancia de SageMaker y sus precios en Precios de Amazon SageMaker.

Selección de una instancia para el procesamiento de datos

El paso de procesamiento de datos de SageMaker requiere una instancia de procesamiento que tenga suficiente memoria y almacenamiento en disco para los datos de entrada, intermedios y de salida. La cantidad específica de memoria y almacenamiento en disco que se necesita depende de las características del gráfico de Neptune ML y sus características exportadas.

De forma predeterminada, Neptune ML elige la instancia `ml.r5` de menor tamaño cuya memoria sea diez veces mayor que el tamaño de los datos de gráficos exportados en el disco.

Selección de una instancia para el entrenamiento y la transformación de modelos

La selección del tipo de instancia correcto para el entrenamiento de modelos o la transformación de modelos depende del tipo de tarea, el tamaño del gráfico y los requisitos de entrega. Las instancias de GPU proporcionan el mejor rendimiento. Por lo general, recomendamos las instancias de serie `p3` y `g4dn`. También puede usar instancias `p2` o `p4d`.

De forma predeterminada, Neptune ML elige la instancia de GPU de menor tamaño y con más memoria que la que necesitan el entrenamiento y la transformación de modelos. Puede encontrar esta selección en el archivo `train_instance_recommendation.json`, en la ubicación de salida del procesamiento de datos de Amazon S3. A continuación, se muestra un ejemplo del contenido de un archivo `train_instance_recommendation.json`:

```
{
  "instance":     "(the recommended instance type for model training and transform)",
  "cpu_instance": "(the recommended instance type for base processing instance)",
  "disk_size":    "(the estimated disk space required)",
  "mem_size":     "(the estimated memory required)"
}
```

Selección de una instancia para un punto de conexión de inferencia

La selección del tipo de instancia correcto para un punto de conexión de inferencia depende del tipo de tarea, del tamaño del gráfico y del presupuesto. De forma predeterminada, Neptune ML elige la instancia `ml.m5d` de menor tamaño y con más memoria que la que necesita el punto de conexión de inferencia.

> **Note**
>
> Si se necesitan más de 384 GB de memoria, Neptune ML usa una instancia `ml.r5d.24xlarge`.

Puede ver el tipo de instancia que recomienda Neptune ML en el archivo `infer_instance_recommendation.json`, que se encuentra en la ubicación de Amazon S3 que está utilizando para el entrenamiento de modelos. A continuación, se muestra un ejemplo del contenido de ese archivo:

```
{
  "instance" :   "(the recommended instance type for an inference endpoint)",
  "disk_size" :  "(the estimated disk space required)",
  "mem_size" :   "(the estimated memory required)"
}
```

Uso de la herramienta **`neptune-export`** o el servicio Neptune-Export para exportar datos de Neptune para Neptune ML

Neptune ML requiere que proporcione datos de entrenamiento para que la biblioteca Deep Graph Library (DGL) cree y evalúe modelos.

Puede exportar datos de Neptune mediante el servicio Neptune-Export o la utilidad `neptune-export`. Tanto el servicio como la herramienta de línea de comandos publican datos en Amazon Simple Storage Service (Amazon S3) en formato CSV, que se cifran mediante cifrado del lado del servidor de Amazon S3 (`SSE-S3`). Consulte Archivos exportados por Neptune-Export y `neptune-export`.

Además, al configurar una exportación de datos de entrenamiento para Neptune ML, el trabajo de exportación crea y publica un archivo de configuración de entrenamiento de modelos cifrado junto con los datos exportados. De forma predeterminada, este archivo recibe el nombre de `training-data-configuration.json`.

Ejemplos del uso del servicio Neptune-Export para exportar datos de entrenamiento para Neptune ML

Esta solicitud exporta datos de entrenamiento de gráficos de propiedades para una tarea de clasificación de nodos:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-pg",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "Movie",
```

```
                "property": "genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

Esta solicitud exporta datos de entrenamiento de RDF para una tarea de clasificación de nodos:

```
curl \
  (your NeptuneExportApiUri) \
  -X POST \
  -H 'Content-Type: application/json' \
  -d '{
        "command": "export-rdf",
        "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
        "params": {
          "endpoint": "(your Neptune endpoint DNS name)",
          "profile": "neptune_ml"
        },
        "additionalParams": {
          "neptune_ml": {
            "version": "v2.0",
            "targets": [
              {
                "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
                "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/
genre",
                "type": "classification"
              }
            ]
          }
        }
      }'
```

Campos que hay que configurar en el objeto **`params`** al exportar los datos de entrenamiento

El objeto `params` de una solicitud de exportación puede incluir varios campos, tal y como se describe en la documentación de `params`. Los siguientes son los más relevantes a la hora de exportar datos de entrenamiento de machine learning:

- **`endpoint`**: utilice `endpoint` para especificar un punto de conexión de una instancia de Neptune en su clúster de base de datos que el proceso de exportación pueda consultar para extraer datos.
- **`profile`**: el campo `profile` del objeto `params` debe establecerse en **`neptune-ml`**.

  Esto provoca que el proceso de exportación formatee los datos exportados de forma adecuada para el entrenamiento de modelos de Neptune ML, en formato CSV para datos de gráficos de propiedades o como N-Triples para datos de RDF. También origina que se cree y escriba un archivo `training-data-configuration.json` en la misma ubicación de Amazon S3 que los datos de entrenamiento exportados.
- **`cloneCluster`**: si se establece en `true`, el proceso de exportación clona el clúster de base de datos, lo exporta desde el clon y, al finalizar, elimina el clon.
- **`useIamAuth`**: si el clúster de base de datos tiene habilitada la autenticación de IAM, debe incluir este campo establecido en `true`.

El proceso de exportación también proporciona varias formas de filtrar los datos que exporta (consulte estos ejemplos).

Uso del objeto **`additionalParams`** para ajustar la exportación de la información de entrenamiento de modelos

El objeto `additionalParams` incluye campos que puede utilizar para especificar las etiquetas y características de las clases de machine learning con fines de entrenamiento y para dar instrucciones en la creación de un archivo de configuración de datos de entrenamiento.

El proceso de exportación no puede inferior automáticamente las propiedades de nodo y borde que deben ser las etiquetas de las clases de machine learning para que sirvan de ejemplo con fines de entrenamiento. Tampoco puede inferir automáticamente la mejor codificación de características para las propiedades numéricas, categóricas y de texto, por lo que es necesario proporcionar sugerencias mediante los campos del objeto `additionalParams` para especificar estas opciones o anular la codificación predeterminada.

En el caso de los datos de gráficos de propiedades, la estructura de nivel superior de `additionalParams` de una solicitud de exportación podría tener el siguiente aspecto:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
```

```
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ],
        "features": [ (an array of node feature hints) ]
    }
  }
}
```

En el caso de los datos de RDF, su estructura de nivel superior podría tener este aspecto:

```
{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams": {
      "neptune_ml": {
        "version": "v2.0",
        "targets": [ (an array of node and edge class label targets) ]
    }
  }
}
```

También puede establecer varias configuraciones de exportación mediante el campo `jobs`:

```
{
  "command": "export-pg",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "profile": "neptune_ml"
  },
  "additionalParams" : {
    "neptune_ml" : {
      "version": "v2.0",
      "jobs": [
```

```
        {
          "name" : "(training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        },
        {
          "name" : "(another training data configuration name)",
          "targets": [ (an array of node and edge class label targets) ],
          "features": [ (an array of node feature hints) ]
        }
      ]
    }
  }
}
```

Elementos de nivel superior del campo **neptune_ml** de **additionalParams**

El elemento **version** de **neptune_ml**

Especifica la versión de la configuración de datos de entrenamiento que se va a generar.

(Opcional), Tipo: cadena. Valor predeterminado: “v2.0”.

Si incluye `version`, establézcala en `v2.0`.

El campo **jobs** de **neptune_ml**

Incluye una matriz de objetos de configuración de datos de entrenamiento, cada uno de los cuales define un trabajo de procesamiento de datos e incluye:

- **name**: el nombre de la configuración de datos de entrenamiento que se va a crear.

  Por ejemplo, una configuración de datos de entrenamiento con el nombre “job-number-1” da como resultado un archivo de configuración de datos de entrenamiento denominado `job-number-1.json`.
- **targets**: una matriz de JSON de destinos de etiquetas de clases de nodos y bordes que representan las etiquetas de clases de machine learning con fines de entrenamiento. Consulte El campo `targets` de un objeto `neptune_ml`.
- **features**: una matriz de JSON de características de propiedades de nodos. Consulte El campo `features` de `neptune_ml`.

El campo **`targets`** de un objeto **`neptune_ml`**

El campo `targets` de una configuración de exportación de datos de entrenamiento de JSON incluye una matriz de objetos de destino que especifica una tarea de entrenamiento y las etiquetas de las clases de machine learning para entrenar esta tarea. El contenido de los objetos de destino varía en función de si se está entrenando con datos de gráficos de propiedades o con datos RDF.

En el caso de las tareas de regresión y clasificación de nodos de gráficos de propiedades, los objetos de destino de la matriz pueden tener el siguiente aspecto:

```
{
  "node": "(node property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

En el caso de las tareas de predicción de enlaces, regresión o clasificación de bordes de gráficos de propiedades, pueden tener el siguiente aspecto:

```
{
  "edge": "(edge property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification, regression or link_prediction)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

En el caso de las tareas de regresión y clasificación de nodos de RDF, los objetos de destino de la matriz pueden tener el siguiente aspecto:

```
{
  "node": "(node type of an RDF node)",
  "predicate": "(predicate IRI)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0]
}
```

En el caso de las tareas de predicción de enlaces de RDF, los objetos de destino de la matriz pueden tener el siguiente aspecto:

```
{
  "subject": "(source node type of an edge)",
  "predicate": "(relation type of an edge)",
  "object": "(destination node type of an edge)",
  "type" : "link_prediction",
  "split_rate": [0.8,0.2,0.0]
}
```

Los objetos de destino pueden incluir los siguientes campos:

Contenido

Campos de un objeto de destino de gráfico de propiedades

El campo **node** (vértice) de un objeto de destino

La etiqueta de gráfico de propiedades de un nodo de destino (vértice). Un objeto de destino debe incluir un elemento `node` o un elemento `edge`, pero no ambos.

Un `node` puede tomar un único valor, como este:

```
"node": "Movie"
```

O bien, en el caso de un vértice con varias etiquetas, puede tomar una matriz de valores, como, por ejemplo:

```
"node": ["Content", "Movie"]
```

El campo **`edge`** de un objeto de destino de gráfico de propiedades

Especifica un bordes de destino mediante sus etiquetas de nodo inicial, su propia etiqueta y sus etiquetas de nodo final. Un objeto de destino debe incluir un elemento `edge` o un elemento `node`, pero no ambos.

El valor de un campo `edge` es una matriz JSON de tres cadenas que representan las etiquetas del gráfico de propiedades del nodo inicial, la etiqueta del gráfico de propiedades del propio borde y las etiquetas del gráfico de propiedades del nodo final, de la siguiente manera:

```
"edge": ["Person_A", "knows", "Person_B"]
```

Si el nodo inicial o el nodo final tienen varias etiquetas, inclúyalas en una matriz, como, por ejemplo:

```
"edge": [ ["Admin", Person_A"], "knows", ["Admin", "Person_B"] ]
```

El campo **`property`** de un objeto de destino de gráfico de propiedades

Especifica una propiedad del vértice o borde de destino, como, por ejemplo:

```
"property" : "rating"
```

Este campo es obligatorio, excepto cuando la tarea de destino es la predicción de enlaces.

El campo **`type`** de un objeto de destino de gráfico de propiedades

Indica el tipo de tarea de destino que se va a realizar en el `node` o la `edge`, como, por ejemplo:

```
"type" : "regression"
```

Los tipos de tareas compatibles con los nodos son:

- `classification`

- `regression`

Los tipos de tareas compatibles con los bordes son:

- `classification`
- `regression`
- `link_prediction`

Este campo es obligatorio.

El campo **`split_rate`** de un objeto de destino de gráfico de propiedades

(Opcional) Una estimación de las proporciones de los nodos o los bordes que utilizarán las etapas de entrenamiento, validación y prueba, respectivamente. Estas proporciones se representan mediante una matriz de JSON de tres números entre cero y uno que suman uno:

```
"split_rate": [0.7, 0.1, 0.2]
```

Si no proporciona el campo opcional `split_rate`, el valor estimado predeterminado es `[0.9, 0.1, 0.0]`.

El campo **`separator`** de un objeto de destino de gráfico de propiedades

(Opcional) Se utiliza con una tarea de clasificación.

El campo `separator` especifica un carácter que se utiliza para dividir el valor de una propiedad de destino en varios valores categóricos cuando se utiliza para almacenar varios valores de categoría en una cadena. Por ejemplo:

```
"separator": "|"
```

La presencia de un campo `separator` indica que la tarea es una tarea de clasificación con varios destinos.

Campos de un objeto de destino de RDF

El campo **`node`** de un objeto de destino de RDF

Define el tipo de nodo de los nodos de destino. Se utiliza con tareas de clasificación de nodos o tareas de regresión de nodos. El tipo de nodo de un nodo de RDF se define mediante:

```
node_id, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, node_type
```

Un node de RDF puede tomar un único valor, como este:

```
"node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

El campo **`subject`** de un objeto de destino de RDF

En el caso de las tareas de predicción de enlaces, `subject` define el tipo de nodo de origen de los bordes de destino.

```
"subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director"
```

> **Note**
>
> En el caso de las tareas de predicción de enlaces, `subject` debe usarse junto con `predicate` y `object`. Si no se proporciona alguna de estas tres opciones, todos los bordes se consideran el destino de entrenamiento.

El campo **`predicate`** de un objeto de destino de RDF

En el caso de las tareas de regresión y clasificación de nodos, `predicate` define los datos literales que se utilizan como la característica del nodo de destino de un nodo de destino.

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre"
```

> **Note**
>
> Si los nodos de destino solo tienen un predicado que define la característica del nodo de destino, se puede omitir el campo `predicate`.

En el caso de las tareas de predicción de enlaces, `predicate` define el tipo de relación de los bordes de destino:

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/direct"
```

> **Note**
>
> En el caso de las tareas de predicción de enlaces, `predicate` debe usarse junto con `subject` y `object`. Si no se proporciona alguna de estas tres opciones, todos los bordes se consideran el destino de entrenamiento.

El campo **`object`** de un objeto de destino de RDF

En el caso de las tareas de predicción de enlaces, `object` define el tipo de nodo de destino de los bordes de destino:

```
"object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

> **Note**
>
> En el caso de las tareas de predicción de enlaces, `object` debe usarse junto con `subject` y `predicate`. Si no se proporciona alguna de estas tres opciones, todos los bordes se consideran el destino de entrenamiento.

El campo **`type`** de un objeto de destino de RDF

Indica el tipo de tarea de destino que se va a realizar, como, por ejemplo:

```
"type" : "regression"
```

Los tipos de tareas compatibles con los datos de RDF son:

- `link_prediction`
- `classification`
- `regression`

Este campo es obligatorio.

El campo **`split_rate`** de un objeto de destino de gráfico de propiedades

(Opcional) Una estimación de las proporciones de los nodos o los bordes que utilizarán las etapas de entrenamiento, validación y prueba, respectivamente. Estas proporciones se representan mediante una matriz de JSON de tres números entre cero y uno que suman uno:

```
"split_rate": [0.7, 0.1, 0.2]
```

Si no proporciona el campo opcional `split_rate`, el valor estimado predeterminado es `[0.9, 0.1, 0.0]`.

El campo **`features`** de **`neptune_ml`**

Los valores de las propiedades y los literales de RDF están disponibles en diferentes formatos y tipos de datos. Para lograr un buen rendimiento en el machine learning, es fundamental convertir esos valores en codificaciones numéricas conocidas como características.

Neptune ML realiza la extracción y codificación de características como parte de los pasos de exportación y procesamiento de datos, tal y como se describe en Codificación de características en Neptune ML.

En el caso de los conjuntos de datos de gráficos de propiedades, el proceso de exportación infiere automáticamente las características `auto` de las propiedades de cadenas y de las propiedades numéricas que incluyen varios valores. En el caso de las propiedades numéricas que incluyen valores únicos, infiere las características `numerical`. En el caso de las propiedades de fecha, infiere las características `datetime`.

Si desea anular una especificación de característica inferida automáticamente o añadir una especificación numérica de bucket, TF-IDF, FastText o SBERT para una propiedad, puede controlar la codificación de la características mediante el campo de características.

> **Note**
>
> Solo puede usar el campo `features` para controlar las especificaciones de las características de los datos del gráfico de propiedades, no de los datos de RDF.

En el caso de textos de formato libre, Neptune ML puede utilizar varios modelos diferentes para convertir la secuencia de tokens de un valor de propiedad de cadena en un vector de valor real de tamaño fijo:

- `text_fasttext`: utiliza la codificación fastText. Esta es la codificación recomendada para las características que utilizan solo uno de los cinco idiomas que admite FastText.
- `text_sbert`: utiliza los modelos de codificación Sentence BERT (SBERT). Esta es la codificación recomendada para el texto que no admite `text_fasttext`.
- `text_word2vec`: utiliza los algoritmos Word2Vec publicados originalmente por Google para codificar texto. Word2Vec solo admite inglés.

- `text_tfidf`: utiliza un vectorizador de frecuencia de términos - frecuencia inversa de documentos (TF-IDF) para codificar texto. La codificación TF-IDF admite características estadísticas que las demás codificaciones no admiten.

El campo `features` incluye una matriz de JSON de características de propiedades de nodos. Los objetos de la matriz pueden incluir los siguientes campos:

Contenido

El campo **node** de **features**

El campo node especifica una etiqueta de gráfico de propiedades de un vértice de característica. Por ejemplo:

```
"node": "Person"
```

Si un vértice tiene varias etiquetas, utilice una matriz para incluir todas ellas. Por ejemplo:

```
"node": ["Admin", "Person"]
```

El campo **edge** de **features**

El campo edge especifica el tipo de borde del borde de una característica. Un tipo de borde consiste en una matriz que incluye las etiquetas del gráfico de propiedades del vértice de origen, la etiqueta del gráfico de propiedades del borde y las etiquetas del gráfico de propiedades del vértice de destino. Debe proporcionar los tres valores al especificar una característica de borde. Por ejemplo:

```
"edge": ["User", "reviewed", "Movie"]
```

Si un vértice de origen o destino de un tipo de borde tiene varias etiquetas, utilice otra matriz para incluir todas ellas. Por ejemplo:

```
"edge": [["Admin", "Person"]. "edited", "Post"]
```

El campo **property** de **features**

Utilice el parámetro de propiedad para especificar una propiedad del vértice identificado por el parámetro node. Por ejemplo:

```
"property" : "age"
```

Valores posibles del campo **type** para las características

El parámetro type especifica el tipo de característica que se está definiendo. Por ejemplo:

```
"type": "bucket_numerical"
```

Valores posibles del parámetro **type**

- **"auto"**: especifica que Neptune ML debe detectar automáticamente el tipo de propiedad y aplicar una codificación de característica adecuada. Una característica auto también puede tener un campo opcional separator.

  Consulte Codificación automática de características en Neptune ML.
- **"category"**: esta codificación de característica representa el valor de una propiedad como una de varias categorías. Dicho de otro modo, la característica puede tomar uno o varios valores discretos. Una característica category también puede tener un campo opcional separator.

Consulte Características categóricas de Neptune ML.

- **`"numerical"`**: esta codificación de característica representa los valores de propiedades numéricas como números en un intervalo continuo en el que los valores “mayor que” y “menor que” tienen relevancia.

  Una característica `numerical` también puede tener los campos opcionales `norm`, `imputer` y `separator`.

  Consulte Características numéricas de Neptune ML.

- **`"bucket_numerical"`**: esta codificación de característica divide los valores de propiedades numéricas en un conjunto de buckets o categorías.

  Por ejemplo, podrías codificar las edades de las personas en cuatro grupos: niños (0-20), adultos jóvenes (20-40), personas de mediana edad (40-60) y personas mayores (más de 60 años).

  Una característica `bucket_numerical` requiere un campo `range` y un campo `bucket_cnt` y, de forma opcional, también puede incluir un campo `imputer` o `slide_window_size`.

  Consulte Características numéricas por bucket de Neptune ML.

- **`"datetime"`**: esta codificación de característica representa el valor de una propiedad de fecha y hora como una matriz de estas características categóricas: año, mes, día de la semana y hora.

  Se pueden eliminar una o varias de estas cuatro categorías mediante el parámetro `datetime_parts`.

  Consulte Características de fecha y hora de Neptune ML.

- **`"text_fasttext"`**: esta codificación de característica convierte los valores de propiedad que constan de frases o texto de formato libre en vectores numéricos mediante modelos fastText. Es compatible con cinco idiomas: inglés (`en`), chino (`zh`), hindi (`hi`), español (`es`) y francés (`fr`). En el caso de los valores de propiedades de texto en cualquiera de esos cinco idiomas, se recomienda la codificación `text_fasttext`. Sin embargo, no admite casos en los que la misma frase incluya palabras en más de un idioma.

  Para otros idiomas distintos de los que admite fastText, utilice la codificación `text_sbert`.

  Si tiene muchas cadenas de texto de valor de propiedad de más de, por ejemplo, 120 caracteres, utilice el campo `max_length` para limitar el número de tokens en cada cadena que `"text_fasttext"` codifique.

Consulte Codificación fastText de valores de propiedades de texto en Neptune ML.

- **`"text_sbert"`**: esta codificación convierte los valores de las propiedades de texto en vectores numéricos mediante los modelos Sentence BERT (SBERT). Neptune admite dos métodos SBERT, es decir `text_sbert128`, que es el predeterminado si solo se especifica `text_sbert` y. `text_sbert512`. La diferencia entre ellos es el número máximo de tokens de una propiedad de texto que se codifica. La codificación `text_sbert128` solo codifica los primeros 128 tokens, mientras que `text_sbert512` codifica hasta 512 tokens. Como resultado, el uso de `text_sbert512` puede requerir más tiempo de procesamiento que `text_sbert128`. Ambos métodos son más lentos que `text_fasttext`.

  Los métodos `text_sbert*` admiten muchos idiomas y pueden codificar una frase que incluya más de un idioma.

  Consulte Codificación de frases BERT (SBERT) de características de texto en Neptune ML.

- **`"text_word2vec"`**: esta codificación convierte los valores de las propiedades de texto en vectores numéricos mediante los algoritmos Word2Vec. Solo es compatible con el inglés.

  Consulte Codificación Word2Vec de características de texto en Neptune ML.

- **`"text_tfidf"`**: esta codificación convierte los valores de las propiedades de texto en vectores numéricos mediante un vectorizador de frecuencia de términos - frecuencia inversa de documentos (TF-IDF).

  Los parámetros de un codificación de característica `text_tfidf` se definen mediante el campo `ngram_range`, el campo `min_df` y el campo `max_features`.

  Consulte Codificación TF-IDF de características de texto en Neptune ML.

- **`"none"`**: el uso del tipo `none` provoca que no se produzca ninguna codificación de características. En cambio, los valores de las propiedades sin procesar se analizan y se guardan.

  Utilice `none` solo si tiene previsto realizar su propia codificación personalizada de características como parte del entrenamiento con modelos personalizados.

El campo **`norm`**

Este campo es obligatorio para las características numéricas. Especifica un método de normalización para usar en valores numéricos:

```
"norm": "min-max"
```

Se admiten los siguientes métodos de normalización:

- “min-max”: normaliza cada valor restándole el valor mínimo y dividiéndolo por la diferencia entre el valor máximo y el mínimo.
- “standard”: normaliza cada valor dividiéndolo entre la suma de todos los valores.
- “none”: no normaliza los valores numéricos durante la codificación.

Consulte Características numéricas de Neptune ML.

El campo **`language`**

El campo de idioma especifica el idioma utilizado en los valores de las propiedades de texto. Su uso depende del método de codificación del texto:

- En el caso de la codificación `text_fasttext`, este campo es obligatorio y debe especificar uno de los siguientes idiomas:
  - `en` (inglés)
  - `zh` (chino)
  - `hi` (hindi)
  - `es` (español)
  - `fr` (francés)
- En el caso de la codificación `text_sbert`, este campo no se utiliza, ya que la codificación SBERT es multilingüe.
- En el caso de la codificación `text_word2vec`, este campo es opcional, ya que `text_word2vec` solo admite el inglés. Si está presente, debe especificar el nombre del modelo en inglés:

  ```
  "language" : "en_core_web_lg"
  ```
- En el caso de la codificación `text_tfidf`, este campo no se utiliza.

El campo **`max_length`**

El campo `max_length` es opcional para las características `text_fasttext`, ya que especifica el número máximo de tokens que se codificarán en una característica de texto de entrada. Se truncará

el texto de entrada que supere `max_length`. Por ejemplo, si se establece max_length en 128, se ignorará cualquier token situado después del 128 en una secuencia de texto:

```
"max_length": 128
```

El campo **`separator`**

Este campo se usa de forma opcional con las características `category`, `numerical` y `auto`. Especifica un carácter que se puede utilizar para dividir el valor de una propiedad en varios valores categóricos o numéricos:

```
"separator": ";"
```

Utilice el campo `separator` únicamente cuando la propiedad almacene varios valores delimitados en una única cadena, como, por ejemplo, `"Actor;Director"` o `"0.1;0.2"`.

Consulte Características categóricas, Características numéricas y Codificación automática.

El campo **`range`**

Este campo es obligatorio para las características `bucket_numerical`. Especifica el rango de valores numéricos que se van a dividir en buckets, en el formato`[`*`lower-bound`*`, `*`upper-bound`*`]`:

```
"range" : [20, 100]
```

Si el valor de una propiedad es menor que el límite inferior, se asigna al primer bucket, o si es mayor que el límite superior, se asigna al último bucket.

Consulte Características numéricas por bucket de Neptune ML.

El campo **`bucket_cnt`**

Este campo es obligatorio para las características `bucket_numerical`. Especifica el número de buckets en los que debe dividirse el rango numérico definido por el parámetro `range`:

```
"bucket_cnt": 10
```

Consulte Características numéricas por bucket de Neptune ML.

El campo **slide_window_size**

Este campo se usa de forma opcional con características `bucket_numerical` para asignar valores a más de un bucket:

```
"slide_window_size": 5
```

El funcionamiento de una ventana deslizante consiste en que Neptune ML toma el tamaño de la ventana **s** y transforma cada valor numérico **v** de una propiedad en un rango de `v - s/2` a `v + s/2` . A continuación, el valor se asigna a cada bucket en el que se superpone el rango.

Consulte Características numéricas por bucket de Neptune ML.

El campo **imputer**

Este campo se utiliza de forma opcional con las características `numerical` y `bucket_numerical` para proporcionar una técnica de imputación y rellenar los valores que faltan:

```
"imputer": "mean"
```

Las técnicas de imputación admitidas son:

- `"mean"`
- `"median"`
- `"most-frequent"`

Si no incluye el parámetro de imputación, el preprocesamiento de datos se detiene y finaliza cuando se encuentra con un valor que falta.

Consulte Características numéricas de Neptune ML y Características numéricas por bucket de Neptune ML.

El campo **max_features**

Las características `text_tfidf` utilizan este campo de forma opcional para especificar el número máximo de términos que se van a codificar:

```
"max_features": 100
```

Un valor de 100 hace que el vectorizador de TF-IDF codifique solo los 100 términos más comunes. El valor predeterminado, si no se incluye `max_features`, es 5000.

Consulte Codificación TF-IDF de características de texto en Neptune ML.

El campo **`min_df`**

Las características `text_tfidf` utilizan este campo de forma opcional para especificar la frecuencia mínima de documentos de los términos que se van a codificar:

```
"min_df": 5
```

Un valor de 5 indica que un término debe aparecer en al menos 5 valores de propiedad diferentes para codificarse.

El valor predeterminado, si no se incluye el parámetro `min_df`, es 2.

Consulte Codificación TF-IDF de características de texto en Neptune ML.

El campo **`ngram_range`**

Las características `text_tfidf` utilizan este campo de forma opcional para especificar qué tamaño de las secuencias de palabras o tokens debe considerarse como posibles términos individuales para codificar:

```
"ngram_range": [2, 4]
```

El valor `[2, 4]` especifica que las secuencias de 2, 3 y 4 palabras deben considerarse posibles términos individuales.

El valor predeterminado, si no establece de forma explícita `ngram_range`, es `[1, 1]`, lo que significa que solo las palabras o tokens únicos se consideran términos que se van a codificar.

Consulte Codificación TF-IDF de características de texto en Neptune ML.

El campo **`datetime_parts`**

Las características `datetime` utilizan este campo de forma opcional para especificar qué partes del valor de fecha y hora deben codificarse categóricamente:

```
"datetime_parts": ["weekday", "hour"]
```

Si no incluye `datetime_parts`, Neptune ML codifica de forma predeterminada las partes de año, mes, día de la semana y hora del valor de fecha y hora. El valor `["weekday", "hour"]` indica que solo los valores de fecha y hora del día de la semana y hora deben codificarse categóricamente en la característica.

Si una de las partes no tiene más de un valor único en el conjunto de entrenamiento, no se ha codificado.

Consulte Características de fecha y hora de Neptune ML.

Ejemplos del uso de parámetros en **`additionalParams`** para ajustar la configuración del entrenamiento de modelos

Contenido

- Especificación de una tarea de clasificación de nodos para la configuración del entrenamiento de modelos
- Especificación de una tarea de regresión de nodos para la configuración del entrenamiento de modelos
- Especificación de una tarea de predicción de enlaces para determinados bordes
- Especificación de una tarea de predicción de enlaces para todos los bordes

Ejemplos de gráficos de propiedades que utilizan **`additionalParams`**

Especificación de una tasa de división predeterminada para la configuración del entrenamiento de modelos

En el siguiente ejemplo el parámetro `split_rate` establece la tasa de división predeterminada para el entrenamiento de modelos. Si no se especifica ninguna tasa de división predeterminada, el entrenamiento usa un valor de [0,9, 0,1, 0,0]. Para anular el valor predeterminado según el destino, especifique una `split_rate` para cada destino.

En el siguiente ejemplo el campo `default split_rate` indica que se debe utilizar una tasa de división de `[0.7,0.1,0.2]`, a no ser que se anule según el destino:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificación de una tarea de clasificación de nodos para la configuración del entrenamiento de modelos

Para indicar la propiedad de nodo que incluye ejemplos etiquetados con fines de entrenamiento, añada un elemento de clasificación de nodos a la matriz de `targets` mediante `"type" :`

`"classification"`. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

En el siguiente ejemplo el destino `node` indica que la propiedad `genre` de cada nodo `Movie` debe tratarse como una etiqueta de clase de nodo. El valor `split_rate` anula la tasa de división predeterminada:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificación de una tarea de clasificación de nodos de varias clases para la configuración del entrenamiento de modelos

Para indicar la propiedad de nodo que incluye varios ejemplos etiquetados con fines de entrenamiento, añada un elemento de clasificación de nodos a la matriz de destino mediante `"type" : "classification"` y `separator` para especificar un carácter que pueda usarse para dividir el valor de una propiedad de destino en varios valores categóricos. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

En el siguiente ejemplo el destino `node` indica que la propiedad `genre` de cada nodo `Movie` debe tratarse como una etiqueta de clase de nodo. El campo `separator` indica que cada propiedad de género incluye varios valores separados por punto y coma:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
```

```
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "separator": ";"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificación de una tarea de regresión de nodos para la configuración del entrenamiento de modelos

Para indicar la propiedad de nodo que incluye regresiones etiquetadas con fines de entrenamiento, añada un elemento de regresión de nodos a la matriz de destino mediante `"type" : "regression"`. Si desea anular la tasa de división predeterminada, añada el campo split_rate.

El siguiente destino `node` indica que la propiedad `rating` de cada nodo `Movie` debe tratarse como una etiqueta de regresión de nodo.

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "rating",
        "type" : "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      ...
    ]
  }
}
```

Especificación de una tarea de clasificación de bordes para la configuración del entrenamiento de modelos

Para indicar la propiedad de borde que incluye ejemplos etiquetados con fines de entrenamiento, añada un elemento de borde a la matriz de `targets` mediante `"type" : "regression"`. Si desea anular la tasa de división predeterminada, añada el campo split_rate.

El siguiente destino `edge` indica que la propiedad `metAtLocation` de cada borde `knows` debe tratarse como una etiqueta de clase de borde:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "knows", "Person"],
        "property": "metAtLocation",
        "type": "classification"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificación de una tarea de borde de varias clases para la configuración del entrenamiento de modelos

Para indicar la propiedad de borde que incluye varios ejemplos etiquetados con fines de entrenamiento, añada un elemento de borde a la matriz de `targets` mediante `"type" : "classification"` y un campo `separator` para especificar un carácter utilizado para dividir el valor de una propiedad de destino en varios valores categóricos. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

El siguiente destino `edge` indica que la propiedad `sentiment` de cada borde `repliedTo` debe tratarse como una etiqueta de clase de borde. El campo del separador indica que cada propiedad de opinión incluye varios valores separados por comas:

```
  "additionalParams": {
```

```
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "repliedTo", "Message"],
        "property": "sentiment",
        "type": "classification",
        "separator": ","
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificación de una regresión de bordes para la configuración del entrenamiento de modelos

Para indicar la propiedad de borde que incluye ejemplos de regresión etiquetados con fines de entrenamiento, añada un elemento `edge` a la matriz de `targets` mediante `"type" : "regression"`. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

El siguiente destino `edge` indica que la propiedad `rating` de cada borde `reviewed` debe tratarse como una regresión de borde:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "reviewed", "Movie"],
        "property": "rating",
        "type" : "regression"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificación de una tarea de predicción de enlaces para la configuración del entrenamiento de modelos

Para indicar los bordes que deben usarse con fines de entrenamiento de predicción de enlaces, añada un elemento de borde a la matriz de destino mediante `"type" : "link_prediction"`. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

El siguiente destino `edge` indica que los bordes `cites` deben usarse para la predicción de enlaces:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Article", "cites", "Article"],
        "type" : "link_prediction"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Especificación de una característica de bucket numérico

Para especificar una característica de datos numéricos para una propiedad de nodo, añada `"type": "bucket_numerical"` a la matriz de `features`.

La siguiente característica `node` indica que la propiedad `age` de cada nodo `Person` debe tratarse como una característica de bucket numérico:

```
  "additionalParams": {
  "neptune_ml": {
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Person",
        "property": "age",
        "type": "bucket_numerical",
```

```
        "range": [1, 100],
        "bucket_cnt": 5,
        "slide_window_size": 3,
        "imputer": "median"
      }
    ]
  }
}
```

Especificación de una característica **`Word2Vec`**

Para especificar una característica `Word2Vec` para una propiedad de nodo, añada `"type": "text_word2vec"` a la matriz de `features`.

La siguiente característica node indica que la propiedad `description` de cada nodo `Movie` debe tratarse como una característica `Word2Vec`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_word2vec",
        "language": "en_core_web_lg"
      }
    ]
  }
}
```

Especificación de una característica **`FastText`**

Para especificar una característica `FastText` para una propiedad de nodo, añada `"type": "text_fasttext"` a la matriz de `features`. El campo `language` es obligatorio y debe especificar uno de los siguientes códigos de idiomas:

- en (inglés)
- zh (chino)

- `hi` (hindi)
- `es` (español)
- `fr` (francés)

Tenga en cuenta que la codificación `text_fasttext` no puede admitir más de un idioma a la vez en una característica.

La siguiente característica node indica que la propiedad `description` (francés) de cada nodo `Movie` debe tratarse como una característica `FastText`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_fasttext",
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
}
```

Especificación de una característica **`Sentence BERT`**

Para especificar una característica `Sentence BERT` para una propiedad de nodo, añada `"type": "text_sbert"` a la matriz de `features`. No es necesario especificar el idioma, ya que el método codifica automáticamente las características de texto mediante un modelo de idioma multilingüe.

La siguiente característica node indica que la propiedad `description` de cada nodo `Movie` debe tratarse como una característica `Sentence BERT`:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
```

```
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_sbert128",
      }
    ]
  }
}
```

Especificación de una característica **`TF-IDF`**

Para especificar una característica `TF-IDF` para una propiedad de nodo, añada `"type": "text_tfidf"` a la matriz de `features`.

La siguiente característica `node` indica que la propiedad `bio` de cada nodo `Person` debe tratarse como una característica `TF-IDF`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "bio",
        "type": "text_tfidf",
        "ngram_range": [1, 2],
        "min_df": 5,
        "max_features": 1000
      }
    ]
  }
}
```

Especificación de una característica **`datetime`**

El proceso de exportación infiere automáticamente las características `datetime` de las propiedades de fecha. Sin embargo, si desea limitar las `datetime_parts` usadas para una característica

`datetime` o anular una especificación de característica para que una propiedad que normalmente se trataría como una característica `auto` se trate de forma explícita como una característica `datetime`, añada una `"type": "datetime"` a la matriz de característica.

La siguiente característica node indica que la propiedad `createdAt` de cada nodo `Post` debe tratarse como una característica `datetime`:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "createdAt",
        "type": "datetime",
        "datetime_parts": ["month", "weekday", "hour"]
      }
    ]
  }
}
```

Especificación de una característica **`category`**

El proceso de exportación infiere automáticamente las características `auto` de las propiedades de cadenas y de las propiedades numéricas que incluyen varios valores. En el caso de las propiedades numéricas que incluyen valores únicos, infiere las características `numerical`. En el caso de las propiedades de fecha, infiere las características `datetime`.

Si desea anular una especificación de característica para que una propiedad se trate como una característica categórica, añada `"type": "category"` a la matriz de característica. Si la propiedad incluye varios valores, incluya un campo `separator`. Por ejemplo:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
```

```
      {
        "node": "Post",
        "property": "tag",
        "type": "category",
        "separator": "|"
      }
    ]
  }
}
```

Especificación de una característica **`numerical`**

El proceso de exportación infiere automáticamente las características `auto` de las propiedades de cadenas y de las propiedades numéricas que incluyen varios valores. En el caso de las propiedades numéricas que incluyen valores únicos, infiere las características `numerical`. En el caso de las propiedades de fecha, infiere las características `datetime`.

Si desea anular una especificación de característica para que una propiedad se trate como una característica `numerical`, añada `"type": "numerical"` a la matriz de característica. Si la propiedad incluye varios valores, incluya un campo `separator`. Por ejemplo:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Recording",
        "property": "duration",
        "type": "numerical",
        "separator": ","
      }
    ]
  }
}
```

Especificación de una característica **`auto`**

El proceso de exportación infiere automáticamente las características `auto` de las propiedades de cadenas y de las propiedades numéricas que incluyen varios valores. En el caso de las propiedades

numéricas que incluyen valores únicos, infiere las características `numerical`. En el caso de las propiedades de fecha, infiere las características `datetime`.

Si desea anular una especificación de característica para que una propiedad se trate como una característica `auto`, añada `"type": "auto"` a la matriz de característica. Si la propiedad incluye varios valores, incluya un campo `separator`. Por ejemplo:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "User",
        "property": "role",
        "type": "auto",
        "separator": ","
      }
    ]
  }
}
```

Ejemplos de RDF con **`additionalParams`**

Especificación de una tasa de división predeterminada para la configuración del entrenamiento de modelos

En el siguiente ejemplo el parámetro `split_rate` establece la tasa de división predeterminada para el entrenamiento de modelos. Si no se especifica ninguna tasa de división predeterminada, el entrenamiento usa un valor de [0,9, 0,1, 0,0]. Para anular el valor predeterminado según el destino, especifique una `split_rate` para cada destino.

En el siguiente ejemplo el campo `default split_rate` indica que se debe utilizar una tasa de división de `[0.7,0.1,0.2]`, a no ser que se anule según el destino:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
```

```
    "targets": [
      (...)
    ]
  }
}
```

Especificación de una tarea de clasificación de nodos para la configuración del entrenamiento de modelos

Para indicar la propiedad de nodo que incluye ejemplos etiquetados con fines de entrenamiento, añada un elemento de clasificación de nodos a la matriz de `targets` mediante `"type" : "classification"`. Añada un campo node para indicar el tipo de nodo de los nodos de destino. Añada un campo `predicate` para definir los datos literales que se utilizarán como la característica de nodo de destino del nodo de destino. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

En el siguiente ejemplo el destino `node` indica que la propiedad `genre` de cada nodo `Movie` debe tratarse como una etiqueta de clase de nodo. El valor `split_rate` anula la tasa de división predeterminada:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Especificación de una tarea de regresión de nodos para la configuración del entrenamiento de modelos

Para indicar la propiedad de nodo que incluye regresiones etiquetadas con fines de entrenamiento, añada un elemento de regresión de nodos a la matriz de destino mediante `"type" : "regression"`. Añada un campo `node` para indicar el tipo de nodo de los nodos de destino. Añada un campo `predicate` para definir los datos literales que se utilizarán como la característica de nodo

de destino del nodo de destino. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

El siguiente destino `node` indica que la propiedad `rating` de cada nodo `Movie` debe tratarse como una etiqueta de regresión de nodo.

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/rating",
        "type": "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ]
  }
}
```

Especificación de una tarea de predicción de enlaces para determinados bordes

Para indicar los bordes que deben usarse con fines de entrenamiento de predicción de enlaces, añada un elemento de borde a la matriz de destino mediante `"type" : "link_prediction"`. Añada los campos `subject`, `predicate` y `object` para especificar el tipo de borde. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

El siguiente destino `edge` indica que los bordes `directed` que conectan `Directors` a `Movies` deben usarse para la predicción de enlaces:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director",
        "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/directed",
        "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie",
        "type" : "link_prediction"
      }
    ]
  }
```

```
}
```

Especificación de una tarea de predicción de enlaces para todos los bordes

Para indicar que todos los bordes que deben usarse con fines de entrenamiento de predicción de enlaces, añada un elemento `edge` a la matriz de destino mediante `"type" : "link_prediction"`. No añada los campos `subject`, `predicate` ni `object`. Si desea anular la tasa de división predeterminada, añada el campo `split_rate`.

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "type" : "link_prediction"
      }
    ]
  }
}
```

Procesamiento de los datos de gráficos exportados de Neptune para el entrenamiento

El paso de procesamiento de datos toma los datos de gráficos de Neptune creados por el proceso de exportación y crea la información que utiliza la biblioteca Deep Graph Library (DGL) durante el entrenamiento. Esto incluye realizar varios mapeos y transformaciones de datos:

- Analizar nodos y bordes para crear los archivos de mapeo de gráficos e identificador que requiere DGL.
- Convertir las propiedades de nodos y bordes en las características de nodos y bordes que requiere DGL.
- Dividir los datos en conjuntos de entrenamiento, validación y prueba.

Administración del paso de procesamiento de datos para Neptune ML

Una vez que haya exportado los datos de Neptune que desee utilizar para el entrenamiento de modelos, puede iniciar un trabajo de procesamiento de datos mediante un comando `curl` (o `awscurl`) como el siguiente:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)",
        "configFileName" : "training-job-configuration.json"
      }'
```

Los detalles sobre cómo usar este comando se explican en El comando de procesamiento de datos, junto con información sobre cómo obtener el estado de un trabajo en ejecución, cómo detener un trabajo en ejecución y cómo enumerar todos los trabajos en ejecución.

Procesamiento de datos de gráficos actualizados para Neptune ML

También puede proporcionar un `previousDataProcessingJobId` a la API para garantizar que el nuevo trabajo de procesamiento de datos utilice el mismo método de procesamiento que el trabajo anterior. Esto es necesario si desea obtener predicciones para datos de gráficos actualizados en Neptune, ya sea reentrenando el modelo antiguo con los nuevos datos o volviendo a calcular los artefactos de modelos en los nuevos datos.

Para ello, utilice un comando `curl` (o `awscurl`) como este:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{ "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your output
 folder)",
        "previousDataProcessingJobId", "(the job ID of the previous data-processing
 job)"}'
```

Establezca el valor del parámetro `previousDataProcessingJobId` en el ID de trabajo del trabajo de procesamiento de datos anterior que corresponda al modelo entrenado.

> **Note**
>
> Actualmente, no se admiten las eliminaciones de nodos en el gráfico actualizado. Si se han eliminado los nodos de un gráfico actualizado, debe iniciar un trabajo de procesamiento de datos completamente nuevo en lugar de usar `previousDataProcessingJobId`.

Codificación de características en Neptune ML

Los valores de las propiedades están disponibles en diferentes formatos y tipos de datos. Para lograr un buen rendimiento en el machine learning, es fundamental convertir esos valores en codificaciones numéricas conocidas como características.

Neptune ML realiza la extracción y codificación de características como parte de los pasos de procesamiento y exportación de datos mediante técnicas de codificación de características que se describen aquí.

> **Note**
>
> Si tiene previsto implementar su propia codificación de características en una implementación de modelo personalizado, puede deshabilitar la codificación automática de características en la etapa de preprocesamiento de datos. Para ello, seleccione `none` como tipo de codificación de características. No se produce ninguna codificación de características en esa propiedad de nodo o borde y, en su lugar, los valores sin procesar de las propiedades se analizan y guardan en un diccionario. El preprocesamiento de datos sigue creando el gráfico DGL a partir del conjunto de datos exportado, pero este gráfico creado no cuanta con las características preprocesadas para el entrenamiento.
> Debe utilizar esta opción únicamente si tiene previsto realizar su propia codificación personalizada de características como parte del entrenamiento con modelos personalizados. Para obtener más información, consulte Modelos personalizados de Neptune ML.

Características categóricas de Neptune ML

Una propiedad que puede tomar uno o varios valores distintos de una lista fija de valores posibles es una característica categórica. En Neptune ML, las características categóricas se codifican mediante la codificación one-hot. En el siguiente ejemplo se muestra cómo el nombre de propiedad de distintos alimentos se codifica mediante la codificación one-hot según su categoría:

```
 Food        Veg.   Meat   Fruit    Encoding
---------    ----   ----   -----    --------
 Apple         0      0      1         001
 Chicken       0      1      0         010
 Broccoli      1      0      0         100
```

> **Note**
>
> El número máximo de categorías de cualquier característica categórica es de 100. Si una propiedad tiene más de 100 categorías de valor, solo las 99 más comunes se ubican en categorías distintas y el resto se coloca en una categoría especial denominada `OTHER`.

Características numéricas de Neptune ML

Cualquier propiedad cuyos valores sean números reales se puede codificar como una característica numérica en Neptune ML. Las características numéricas se codifican mediante números de coma flotante.

Puede especificar un método de normalización de datos para usarlo al codificar características numéricas, como, por ejemplo: `"norm": "`*`normalization technique`*`"`. Se admiten las siguientes técnicas de normalización:

- “none”: no normaliza los valores numéricos durante la codificación.
- “min-max”: normaliza cada valor restándole el valor mínimo y dividiéndolo por la diferencia entre el valor máximo y el mínimo.
- “standard”: normaliza cada valor dividiéndolo entre la suma de todos los valores.

Características numéricas por bucket de Neptune ML

En lugar de representar una propiedad numérica con números sin procesar, puede condensar los valores numéricos en categorías. Por ejemplo, puede dividir las edades de las personas en categorías, como niños (0-20), adultos jóvenes (20-40), personas de mediana edad (40-60) y personas mayores (a partir de los 60 años). Al usar estos buckets numéricos, estaría transformando una propiedad numérica en una especie de característica categórica.

En Neptune ML, para que una propiedad numérica se codifique como una característica numérica por bucket, debe proporcionar dos cosas:

- Un rango numérico con el formato `"range": [`*`a`*`, `*`b`*`]` , donde a y b son números enteros.
- Un recuento de buckets con el formato `"bucket_cnt": `*`c`* , donde c es el número de buckets, también un número entero.

Neptune ML calcula el tamaño de cada bucket como `( b - a ) / c` y codifica cada valor numérico como el número del bucket en el que se encuentre. Cualquier valor inferior a a se considera que pertenece al primer bucket y cualquier valor superior a b se considera que pertenece al último bucket.

Si lo desea, también puede hacer que los valores numéricos se clasifiquen en más de un bucket. Para ello, debe especificar un tamaño de ventana deslizante, como, por ejemplo: `"slide_window_size": s` , donde s es un número. A continuación, Neptune ML transforma cada valor numérico v de la propiedad en un rango de `v - s/2` a `v + s/2` y asigna el valor v a cada bucket que cubre el rango.

Por último, si lo desea, también puede proporcionar una forma de rellenar los valores que faltan para las características numéricas y las características numéricas por buckets. Para ello, utilice `"imputer": "imputation technique "`, donde la técnica de imputación es una de las siguientes: `"mean"`, `"median"` o `"most-frequent"`. Si no se especifica un parámetro de imputación, esto puede provocar que el procesamiento se detenga.

Codificación de características de texto en Neptune ML

En el caso de textos de formato libre, Neptune ML puede utilizar varios modelos diferentes para convertir la secuencia de tokens de una cadena de valor de propiedad en un vector de valor real de tamaño fijo:

- `text_fasttext`: utiliza la codificación fastText. Esta es la codificación recomendada para las características que utilizan solo uno de los cinco idiomas que admite FastText.
- `text_sbert`: utiliza los modelos de codificación Sentence BERT (SBERT). Esta es la codificación recomendada para el texto que no admite `text_fasttext`.
- `text_word2vec`: utiliza los algoritmos Word2Vec publicados originalmente por Google para codificar texto. Word2Vec solo admite inglés.
- `text_tfidf`: utiliza un vectorizador de frecuencia de términos - frecuencia inversa de documentos (TF-IDF) para codificar texto. La codificación TF-IDF admite características estadísticas que las demás codificaciones no admiten.

Codificación fastText de valores de propiedades de texto en Neptune ML

Neptune ML puede utilizar los modelos de fastText para convertir valores de propiedades de texto en vectores de valores reales de tamaño fijo. Este es el método de codificación recomendado para los valores de propiedades de texto en cualquiera de los cinco idiomas que admite FastText:

- `en` (inglés)
- `zh` (chino)
- `hi` (hindi)
- `es` (español)
- `fr` (francés)

Tenga en cuenta que fastText no puede procesar frases que incluyan palabras en más de un idioma.

Si lo desea, el método `text_fasttext` puede utilizar un campo `max_length` que especifique el número máximo de tokens del valor de una propiedad de texto que se codificará, tras el cual se truncará la cadena. Esto puede mejorar el rendimiento cuando los valores de las propiedades del texto incluyen cadenas largas, ya que si no se especifica `max_length`, fastText codifica todos los tokens independientemente de la longitud de la cadena.

En este ejemplo se especifica que los títulos de películas en francés se codifican mediante fastText:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_fasttext"],
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
```

Codificación de frases BERT (SBERT) de características de texto en Neptune ML

Neptune ML puede convertir la secuencia de tokens de un valor de propiedad de cadena en un vector de valor real de tamaño fijo mediante los modelos Sentence BERT (SBERT). Neptune admite dos métodos SBERT: `text_sbert128`, que es el predeterminado si solo se especifica `text_sbert` y `text_sbert512`. La diferencia entre ambos es la longitud máxima de una cadena de valores de propiedad de texto que se codifica. La codificación `text_sbert128` trunca las cadenas de texto después de codificar 128 tokens, mientras que `text_sbert512` trunca las cadenas de texto después de codificar 512 tokens. Como resultado, `text_sbert512` requiere

más tiempo de procesamiento que `text_sbert128`. Ambos métodos son más lentos que `text_fasttext`.

La codificación SBERT es multilingüe, por lo que no es necesario especificar un idioma para el texto del valor de propiedad que se va a codificar. SBERT admite muchos idiomas y puede codificar una frase que incluya más de un idioma. Si va a codificar valores de propiedades que incluyan texto en uno o varios idiomas que fastText no admite, el método de codificación recomendado es SBERT.

En el siguiente ejemplo se especifica que los títulos de las películas se codifican como SBERT hasta un máximo de 128 tokens:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      { "feature": ["title", "title", "text_sbert128"] }
    ]
  }
```

Codificación Word2Vec de características de texto en Neptune ML

Neptune ML puede codificar valores de propiedades de cadenas como una característica de Word2Vec (Google publicó originalmente los algoritmos de Word2Vec). El método `text_word2vec` codifica los tokens de una cadena como un vector denso mediante uno de los modelos entrenados de spaCy. Esto solo es compatible con el idioma inglés (mediante el modelo en_core_web_lg).

En el siguiente ejemplo se especifica que los títulos de las películas se codifican con Word2Vec:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_word2vec"],
        "language": "en_core_web_lg"
      }
    ]
  }
```

Tenga en cuenta que el campo de idioma es opcional, ya que el modelo `en_core_web_lg` en inglés es el único que admite Neptune.

Codificación TF-IDF de características de texto en Neptune ML

Neptune ML puede codificar valores de propiedades de texto como características `text_tfidf`. Esta codificación convierte la secuencia de palabras del texto en un vector numérico mediante un vectorizador de frecuencia de términos - frecuencia inversa de documentos (TF-IDF), seguido de una operación de reducción de la dimensionalidad.

TF-IDF (frecuencia de términos - frecuencia inversa de documentos) es un valor numérico destinado a medir la importancia de una palabra en un conjunto de documentos. Se calcula dividiendo el número de veces que aparece una palabra en un determinado valor de propiedad entre el número total de dichos valores de propiedad en los que aparece.

Por ejemplo, si la palabra “kiss” aparece dos veces en el título de una película (por ejemplo, “kiss kiss bang bang”) y “kiss” aparece en el título de cuatro películas en total, el valor TF-IDF de “kiss” en el título “kiss kiss bang bang” sería `2 / 4` .

El vector que se crea inicialmente tiene dimensiones d, donde d es el número de términos únicos en todos los valores de propiedad de ese tipo. La operación de reducción de dimensionalidad utiliza una proyección dispersa aleatoria para reducir ese número a un máximo de 100. A continuación, se genera el vocabulario de un gráfico mediante la combinación de todas sus características `text_tfidf`.

El vectorizador de TF-IDF se puede controlar de varias maneras:

- **`max_features`**: con el parámetro `max_features`, puede limitar el número de términos de las características `text_tfidf` a los más comunes. Por ejemplo, si se establece `max_features` en 100, solo se incluyen los 100 términos más utilizados. El valor predeterminado de `max_features` si no se establece de forma explícita es 5000.
- **`min_df`**: con el parámetro `min_df`, puede limitar el número de términos de las características `text_tfidf` a aquellos que tengan al menos una determinada frecuencia de documentos. Por ejemplo, si se establece `min_df` en 5, solo se utilizarán los términos que aparezcan en al menos 5 valores de propiedad diferentes. El valor predeterminado de `min_df` si no se establece de forma explícita es 2.
- **`ngram_range`**: el parámetro `ngram_range` determina qué combinaciones de palabras se consideran términos. Por ejemplo, si `ngram_range` se establece en `[2, 4]`, los seis términos siguientes aparecerían en el título “kiss kiss bang bang”:

- Términos de dos palabras: “kiss kiss”, “kiss bang” y “bang bang”.
- Términos de tres palabras: “kiss kiss bang” y “kiss bang bang”.
- Términos de cuatro palabras: “kiss kiss bang bang”.

El ajuste predeterminado de `ngram_range` es `[1, 1]`.

Características de fecha y hora de Neptune ML

Neptune ML puede convertir partes de los valores de las propiedades `datetime` en características categóricas mediante la codificación de matrices one-hot. Utilice el parámetro `datetime_parts` para especificar una o varias de las siguientes partes para codificar: `["year", "month", "weekday", "hour"]`. Si no se configura `datetime_parts`, las cuatro partes se codificarán de forma predeterminada.

Por ejemplo, si el rango de valores de fecha y hora abarca los años 2010 a 2012, las cuatro partes de la entrada de fecha y hora `2011-04-22 01:16:34` son las siguientes:

- year: `[0, 1, 0]`.

  Como solo hay tres años en el intervalo (2010, 2011 y 2012), la matriz one-hot tiene tres entradas, una para cada año.
- month: `[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]`.

  En este caso, la matriz one-hot tiene una entrada para cada mes del año.
- weekday: `[0, 0, 0, 0, 1, 0, 0]`.

  La norma ISO 8601 establece que el lunes es el primer día de la semana y, dado que el 22 de abril de 2011 era viernes, la correspondiente matriz one-hot de día de la semana ocupa la quinta posición.
- hour: `[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`.

  La hora 1 de la madrugada se establece en una matriz one-hot de 24 miembros.

El día del mes, el minuto y el segundo no están codificados de forma categórica.

Si el rango total de `datetime` en cuestión solo incluye fechas de un año, no se codifica ninguna matriz de `year`.

Puede especificar una estrategia de imputación para rellenar los valores de `datetime` que faltan, mediante el parámetro `imputer` y una de las estrategias disponibles para las características numéricas.

Codificación automática de características en Neptune ML

En lugar de especificar manualmente los métodos de codificación de características que se van a utilizar para las propiedades del gráfico, puede configurar `auto` como un método de codificación de características. A continuación, Neptune ML intenta inferir la mejor codificación de características para cada propiedad en función del tipo de datos subyacente.

Estas son algunas de las heurísticas que Neptune ML utiliza para seleccionar las codificaciones de características adecuadas:

- Si la propiedad solo tiene valores numéricos y se puede convertir en tipos de datos numéricos, Neptune ML suele codificarla como un valor numérico. Sin embargo, si el número de valores únicos de la propiedad es inferior al 10 % del número total de valores y la cardinalidad de esos valores únicos es inferior a 100, Neptune ML utiliza una codificación categórica.
- Si los valores de las propiedades se pueden convertir en un tipo `datetime`, Neptune ML los codificará como una característica `datetime`.
- Si los valores de las propiedades se pueden convertir en valores booleanos (1/0 o True/False), Neptune ML utilizará la codificación por categorías.
- Si la propiedad es una cadena con más del 10 % de sus valores únicos y el número medio de tokens por valor es mayor o igual a 3, Neptune ML inferirá que el tipo de propiedad sea texto y detectará automáticamente el idioma que se está utilizando. Si el idioma detectado es uno de los que admite fastText, es decir, inglés, chino, hindi, español y francés, Neptune ML utilizará `text_fasttext` para codificar el texto. De lo contrario, Neptune ML utilizará `text_sbert`.
- Si la propiedad es una cadena no clasificada como característica de texto, Neptune ML presupone que se trata de una característica categórica y utiliza la codificación por categorías.
- Si cada nodo tiene su propio valor único para una propiedad que se infiere que es una característica de categoría, Neptune ML eliminará la propiedad del gráfico de entrenamiento porque probablemente sea un ID que no sería informativo para el entrenamiento.
- Si se sabe que la propiedad incluye separadores válidos de Neptune, como puntos y comas (";"), Neptune ML solo puede tratar la propiedad como `MultiNumerical` o `MultiCategorical`.
  - Neptune ML primero intenta codificar los valores como características numéricas. Si esto se ejecuta correctamente, Neptune ML utilizará la codificación numérica para crear características vectoriales numéricas.

  - De lo contrario, Neptune ML codificará los valores como multicategóricos.
- Si Neptune ML no puede inferir el tipo de datos de los valores de una propiedad, Neptune ML eliminará la propiedad del gráfico de entrenamiento.

Edición de un archivo de configuración de datos de entrenamiento

El proceso de exportación de Neptune exporta los datos de Neptune ML de un clúster de base de datos de Neptune a un bucket de S3. Exporta los nodos y los bordes por separado a una carpeta `nodes/` y a una carpeta `edges/`. También crea un archivo de configuración de datos de entrenamiento JSON, con el nombre predeterminado `training-data-configuration.json`. Este archivo incluye información sobre el esquema del gráfico, los tipos de sus características, las operaciones de transformación y normalización de características y la característica de destino para una tarea de clasificación o regresión.

Puede haber casos en los que desee modificar directamente el archivo de configuración. Uno de estos casos es cuando desea cambiar la forma en que se procesan las características o la forma en que se crea el gráfico, sin necesidad de volver a ejecutar la exportación cada vez que desee modificar la especificación de la tarea de machine learning que está resolviendo.

Para editar el archivo de configuración de datos de entrenamiento

1. Descargue el archivo en el equipo local.

   A menos que haya especificado uno o más trabajos con nombre en el parámetro `additionalParams/neptune_ml` transferido al proceso de exportación, el archivo tendrá un nombre predeterminado, que es `training-data-configuration.json`. Puede usar un comando de AWS CLI como este para descargar el archivo:

   ```
   aws s3 cp \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json \
     ./
   ```

2. Edite el archivo con un editor de texto.
3. Cargue el archivo modificado. Vuelva a cargar el archivo modificado en la misma ubicación de Amazon S3 desde la que lo descargó mediante un comando de AWS CLI como este:

   ```
   aws s3 cp \
     training-data-configuration.json \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json
   ```

Ejemplo de un archivo de configuración de datos de entrenamiento de JSON

Este es un ejemplo de un archivo de configuración de datos de entrenamiento que describe un gráfico para una tarea de clasificación de nodos:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [
    {
      "edges" : [
        {
          "file_name" : "edges/(movie)-included_in-(genre).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["", "included_in"],
          "dest" : [ "~to", "genre" ]
        },
        {
          "file_name" : "edges/(user)-rated-(movie).csv",
          "separator" : ",",
          "source" : ["~from", "movie"],
          "relation" : ["rating", "prefixname"], # [prefixname#value]
          "dest" : ["~to", "genre"],
          "features" : [
            {
              "feature" : ["rating", "rating", "numerical"],
              "norm" : "min-max"
            }
          ]
        }
      ],
      "nodes" : [
        {
          "file_name" : "nodes/genre.csv",
          "separator" : ",",
          "node" : ["~id", "genre"],
          "features" : [
            {
              "feature": ["name", "genre", "category"],
              "separator": ";"
            }
          ]
```

```
        },
        {
          "file_name" : "nodes/movie.csv",
          "separator" : ",",
          "node" : ["~id", "movie"],
          "features" : [
            {
              "feature": ["title", "title", "word2vec"],
              "language": ["en_core_web_lg"]
            }
          ]
        },
        {
          "file_name" : "nodes/user.csv",
          "separator" : ",",
          "node" : ["~id", "user"],
          "features" : [
            {
              "feature": ["age", "age", "numerical"],
              "norm" : "min-max",
              "imputation": "median",
            },
            {
              "feature": ["occupation", "occupation", "category"],
            }
          ],
          "labels" : [
            {
              "label": ["gender", "classification"],
              "split_rate" : [0.8, 0.2, 0.0]
            }
          ]
        }
      ]
    },
    "warnings" : [ ]
  ]
}
```

La estructura de los archivos de configuración de datos de entrenamiento de JSON

El archivo de configuración de entrenamiento hace referencia a los archivos CSV guardados durante el proceso de exportación en las carpetas `nodes/` y `edges/`.

Cada archivo en `nodes/` almacena información sobre los nodos que tienen la misma etiqueta de nodo del gráfico de propiedades. Cada columna de un archivo de nodo almacena el ID o la propiedad del nodo. La primera línea del archivo incluye un encabezado que especifica el `~id` o el nombre de la propiedad de cada columna.

Cada archivo en `edges/` almacena información sobre los nodos que tienen la misma etiqueta de borde del gráfico de propiedades. Cada columna de un archivo de nodo almacena el identificador del nodo de origen, el identificador del nodo de destino o la propiedad de borde. La primera línea del archivo incluye un encabezado que especifica `~from`, `~to` o el nombre de la propiedad de cada columna.

El archivo de configuración de los datos de entrenamiento tiene tres elementos de nivel superior:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [ ... ]
}
```

- `version`: (Cadena) la versión del archivo de configuración que se está utilizando.
- `query_engine`: (Cadena) el lenguaje de consulta utilizado para exportar los datos del gráfico. Actualmente, solo es válido “gremlin”.
- `graph`: (Matriz de JSON) muestra uno o varios objetos de configuración que incluyen los parámetros del modelo para cada uno de los nodos y bordes que se utilizarán.

  Los objetos de configuración de la matriz de gráficos tienen la estructura que se describe en la siguiente sección.

Contenido de un objeto de configuración que aparece en la matriz de **`graph`**

Un objeto de configuración de la matriz de `graph` puede incluir tres nodos de nivel superior:

```
  {
    "edges"    : [ ... ],
    "nodes"    : [ ... ],
    "warnings" : [ ... ],
  }
```

- `edges`: (matriz de objetos de JSON) cada objeto de JSON especifica un conjunto de parámetros para definir cómo se tratará un borde del gráfico durante el procesamiento y el entrenamiento del modelo. Esto solo se usa con el motor de Gremlin.
- `nodes`: (matriz de objetos de JSON) cada objeto de JSON especifica un conjunto de parámetros para definir cómo se tratará un nodo del gráfico durante el procesamiento y el entrenamiento del modelo. Esto solo se usa con el motor de Gremlin.
- `warnings`: (matriz de objetos de JSON) cada objeto incluye una advertencia generada durante el proceso de exportación de datos.

Contenido de un objeto de configuración de borde que aparece en una matriz de **`edges`**

Un objeto de configuración de borde incluido en una matriz de `edges` puede incluir los siguientes campos de nivel superior:

```
    {
      "file_name" : "(path to a CSV file)",
      "separator" : "(separator character)",
      "source"    : ["(column label for starting node ID)", "(starting node type)"],
      "relation"  : ["(column label for the relationship name)", "(the prefix name
for the relationship name)"],
      "dest"      : ["(column label for ending node ID)", "(ending node type)"],
      "features"  : [(array of feature objects)],
      "labels"    : [(array of label objects)]
    }
```

- **`file_name`**: una cadena que especifica la ruta a un archivo CSV que almacena información sobre los bordes que tienen la misma etiqueta de gráfico de propiedades.

  La primera línea de ese archivo incluye una línea de encabezado de las etiquetas de las columnas.

  Las dos primeras etiquetas de columna son `~from` y `~to`. La primera columna (la columna `~from`) almacena el identificador del nodo inicial del borde y la segunda (la columna `~to`) almacena el identificador del nodo final del borde.

  Las etiquetas de columna restantes de la línea de encabezado especifican, para cada columna restante, el nombre de la propiedad de borde cuyos valores se han exportado a esa columna.
- **`separator`**: una cadena que incluye el delimitador que separa las columnas de ese archivo CSV.

- **`source`**: una matriz de JSON que incluye dos cadenas que especifican el nodo inicial de la periferia. La primera cadena incluye el nombre del encabezado de la columna en la que se almacena el ID del nodo inicial. La segunda cadena especifica el tipo de nodo.
- **`relation`**: una matriz de JSON que incluye dos cadenas que especifican el tipo de relación del borde. La primera cadena incluye el nombre del encabezado de la columna en la que se almacena el nombre de la relación (`relname`). La segunda cadena incluye el prefijo del nombre de la relación (`prefixname`).

  El tipo de relación completa consta de las dos cadenas combinadas, con un guion entre ellas, como, por ejemplo: *`prefixname`*`-`*`relname`*.

  Si la primera cadena está vacía, todos los bordes tienen el mismo tipo de relación, es decir, la cadena `prefixname`.
- **`dest`**: una matriz de JSON que incluye dos cadenas que especifican el nodo final del borde. La primera cadena incluye el nombre del encabezado de la columna en la que se almacena el ID del nodo. La segunda cadena especifica el tipo de nodo.
- **`features`**: una matriz de JSON de objetos de características de valores de propiedad. Cada objeto de característica de valor de propiedad incluye los siguientes campos:
  - feature: una matriz de JSON de tres cadenas. La primera cadena incluye el nombre del encabezado de la columna que incluye el valor de propiedad. La segunda cadena incluye el nombre de la característica. La tercera cadena incluye el tipo de característica.
  - norm: (opcional) especifica un método de normalización para aplicarlo a los valores de las propiedades.

- **`labels`**: una matriz de JSON de objetos. Cada uno de los objetos define una característica de destino de los bordes y especifica las proporciones de los bordes que deben tener las etapas de entrenamiento y validación. Cada objeto incluye los siguientes campos:
  - label: una matriz de JSON de dos cadenas. La primera cadena incluye el nombre del encabezado de la columna que incluye el valor de propiedad de característica de destino. La segunda cadena especifica uno de los siguientes tipos de tareas de destino:
    - `"classification"`: una tarea de clasificación de bordes. Los valores de propiedad proporcionados en la columna identificada por la primera cadena de la matriz de `label` se tratan como valores categóricos. Para una tarea de clasificación de bordes, la primera cadena de la matriz de `label` no puede estar vacía.
    - `"regression"`: una tarea de regresión de bordes. Los valores de propiedad proporcionados en la columna identificada por la primera cadena de la matriz de `label` se tratan como

valores numéricos. Para una tarea de regresión de bordes, la primera cadena de la matriz de `label` no puede estar vacía.

- `"link_prediction"`: una tarea de predicción de enlaces. No es necesario introducir ningún valor de propiedad. Para una tarea de predicción de enlaces, se ignora la primera cadena de la matriz de `label`.

- **`split_rate`**: una matriz de JSON que incluye tres números entre cero y uno que suman uno y que representan una estimación de las proporciones de nodos que se utilizarán en las etapas de entrenamiento, validación y prueba, respectivamente. Se pueden definir este campo o la opción `custom_split_filenames`, pero no ambos. Consulte split_rate.
- **`custom_split_filenames`**: un objeto de JSON que especifica los nombres de los archivos que definen las poblaciones de entrenamiento, validación y prueba. Se pueden definir este campo o la opción `split_rate`, pero no ambos. Para obtener más información, consulte Proporciones personalizadas de pruebas de validación de entrenamientos.

Contenido de un objeto de configuración de nodo que aparece en una matriz de **`nodes`**

Un objeto de configuración de nodo incluido en una matriz de `nodes` puede incluir los siguientes campos:

```
{
  "file_name" : "(path to a CSV file)",
  "separator" : "(separator character)",
  "node"      : ["(column label for the node ID)", "(node type)"],
  "features"  : [(feature array)],
  "labels"    : [(label array)],
}
```

- **`file_name`**: una cadena que especifica la ruta a un archivo CSV que almacena información sobre los nodos que tienen la misma etiqueta de gráfico de propiedades.

  La primera línea de ese archivo incluye una línea de encabezado de las etiquetas de las columnas.

  La etiqueta de la primera columna es `~id`, y la primera columna (la columna `~id`) almacena el ID del nodo.

  Las etiquetas de columna restantes de la línea de encabezado especifican, para cada columna restante, el nombre de la propiedad de nodo cuyos valores se han exportado a esa columna.
- **`separator`**: una cadena que incluye el delimitador que separa las columnas de ese archivo CSV.

- **`node`**: una matriz de JSON que incluye dos cadenas. La primera cadena incluye el nombre del encabezado de la columna en la que se almacena los ID del nodo. La segunda cadena especifica el tipo de nodo del gráfico, que corresponde a una etiqueta de gráfico de propiedades del nodo.
- **`features`**: una matriz de JSON de objetos de características de nodos. Consulte Contenido de un objeto de característica incluido en una matriz de `features` para un nodo o un borde.
- **`labels`**: una matriz de JSON de objetos de etiquetas de nodos. Consulte Contenido de un objeto de etiqueta de nodo que aparece en una matriz de `labels` de nodo.

Contenido de un objeto de característica incluido en una matriz de **`features`** para un nodo o un borde

Un objeto de característica de nodo incluido en una matriz de `features` de nodo puede incluir los siguientes campos de nivel superior:

- **`feature`**: una matriz de JSON de tres cadenas. La primera cadena incluye el nombre del encabezado de la columna que incluye el valor de propiedad de la característica. La segunda cadena incluye el nombre de la característica.

  La tercera cadena incluye el tipo de característica. Los tipos de características válidos se muestran en Valores posibles del campo `type` para las características.
- **`norm`**: este campo es obligatorio para las características numéricas. Especifica un método de normalización para usar en valores numéricos. Los valores válidos son `"none"`, `"min-max"`, y “estándar”. Para obtener más información, consulte El campo `norm`.
- **`language`**: el campo de idioma especifica el idioma que se utiliza en los valores de las propiedades de texto. Su uso depende del método de codificación del texto:
  - En el caso de la codificación `text_fasttext`, este campo es obligatorio y debe especificar uno de los siguientes idiomas:
    - `en` (inglés)
    - `zh` (chino)
    - `hi` (hindi)
    - `es` (español)
    - `fr` (francés)

    Sin embargo, `text_fasttext` no puede admitir más de un idioma a la vez.

  - En el caso de la codificación `text_sbert`, este campo no se utiliza, ya que la codificación SBERT es multilingüe.
  - En el caso de la codificación `text_word2vec`, este campo es opcional, ya que `text_word2vec` solo admite el inglés. Si está presente, debe especificar el nombre del modelo en inglés:

    ```
    "language" : "en_core_web_lg"
    ```

  - En el caso de la codificación `tfidf`, este campo no se utiliza.

- **`max_length`**: este campo es opcional para las características `text_fasttext`, ya que especifica el número máximo de tokens que se codificarán en una característica de texto de entrada. Se ignorará el texto introducido una vez alcanzado `max_length`. Por ejemplo, si se establece max_length en 128, se ignorará cualquier token situado después del 128 en una secuencia de texto.
- **`separator`**: este campo se usa de forma opcional con las características `category`, `numerical` y `auto`. Especifica un carácter que se puede utilizar para dividir el valor de una propiedad en varios valores categóricos o numéricos.

  Consulte El campo `separator`.
- **`range`**: este campo es obligatorio para las características `bucket_numerical`. Especifica el rango de valores numéricos que se van a dividir en buckets.

  Consulte El campo `range`.
- **`bucket_cnt`**: este campo es obligatorio para las características `bucket_numerical`. Especifica el número de buckets en los que debe dividirse el rango numérico definido por el parámetro `range`.

  Consulte Características numéricas por bucket de Neptune ML.
- **`slide_window_size`**: este campo se usa de forma opcional con características `bucket_numerical` para asignar valores a más de un bucket.

  Consulte El campo `slide_window_size`.
- **`imputer`**: este campo se utiliza de forma opcional con las características `numerical`, `bucket_numerical` y `datetime` para proporcionar una técnica de imputación y rellenar los valores que faltan. Las técnicas de imputación admitidas son `"mean"`, `"median"` y `"most_frequent"`.

Consulte El campo `imputer`.

- **`max_features`**: las características `text_tfidf` utilizan este campo de forma opcional para especificar el número máximo de términos que se van a codificar.

  Consulte El campo `max_features`.

- **`min_df`**: las características `text_tfidf` utilizan este campo de forma opcional para especificar la frecuencia mínima de documentos de los términos que se van a codificar.

  Consulte El campo `min_df`.

- **`ngram_range`**: las características `text_tfidf` utilizan este campo de forma opcional para especificar el rango de números de palabras o tokens que deben considerarse como posibles términos individuales que se van codificar.

  Consulte El campo `ngram_range`.

- **`datetime_parts`**: las características `datetime` utilizan este campo de forma opcional para especificar qué partes del valor de fecha y hora deben codificarse categóricamente.

  Consulte El campo `datetime_parts`.

Contenido de un objeto de etiqueta de nodo que aparece en una matriz de **`labels`** de nodo

Un objeto de etiqueta incluido en una matriz de `labels` de nodo define una característica de destino del nodo y especifica las proporciones de los nodos que se utilizarán en las etapas de entrenamiento, validación y prueba. Cada objeto puede incluir los siguientes campos:

```
    {
      "label"      : ["(column label for the target feature property value)", "(task
type)"],
      "split_rate" : [(training proportion), (validation proportion), (test
proportion)],
      "custom_split_filenames" : {"train": "(training file name)", "valid":
"(validation file name)", "test": "(test file name)"},
      "separator"  : "(separator character for node-classification category values)",
    }
```

- **`label`**: una matriz de JSON que incluye dos cadenas. La primera cadena incluye el nombre del encabezado de la columna que almacena el valor de propiedad de la característica. La segunda cadena especifica el tipo de tarea de destino, que puede ser:

  - `"classification"`: una tarea de clasificación de nodos. Los valores de las propiedades de la columna especificada se utilizan para crear una característica categórica.
  - `"regression"`: una tarea de regresión de nodos. Los valores de las propiedades de la columna especificada se utilizan para crear una característica numérica.
- **`split_rate`**: una matriz de JSON que incluye tres números entre cero y uno que suman uno y que representan una estimación de las proporciones de nodos que se utilizarán en las etapas de entrenamiento, validación y prueba, respectivamente. Consulte split_rate.
- **`custom_split_filenames`**: un objeto de JSON que especifica los nombres de los archivos que definen las poblaciones de entrenamiento, validación y prueba. Se pueden definir este campo o la opción `split_rate`, pero no ambos. Para obtener más información, consulte Proporciones personalizadas de pruebas de validación de entrenamientos.
- **`separator`**: una cadena que incluye el delimitador que separa los valores de las características categóricas para una tarea de clasificación.

> **Note**
>
> Si no se proporciona ningún objeto de etiqueta tanto para los bordes como para los nodos, se asume automáticamente que la tarea consiste en una predicción de enlaces y los bordes se dividen de forma aleatoria en un 90 % para el entrenamiento y un 10 % para la validación.

Proporciones personalizadas de pruebas de validación de entrenamientos

De forma predeterminada, Neptune ML utiliza el parámetro `split_rate` para dividir el gráfico de forma aleatoria en poblaciones de entrenamiento, validación y prueba mediante las proporciones definidas en este parámetro. Para tener un control más preciso sobre qué entidades se utilizan en estas distintas poblaciones, se pueden crear archivos que las definan de forma explícita y, a continuación, se puede editar el archivo de configuración de los datos de entrenamiento para asignar estos archivos de indexación a las poblaciones. Este mapeo se especifica mediante un objeto de JSON para la clave `custom_split_filesnames` del archivo de configuración de entrenamiento. Si se utiliza esta opción, se deben proporcionar los nombres de archivo para las claves `train` y `validation`, y es opcional para la clave `test`.

El formato de estos archivos debe coincidir con el formato de datos de Gremlin. En concreto, para las tareas de nivel de nodo, cada archivo debe incluir una columna con el encabezado `~id` que muestre los identificadores de los nodos, y para las tareas de nivel de borde, los archivos deben

especificar `~from` y `~to` para indicar los nodos de origen y destino de los bordes, respectivamente. Estos archivos deben colocarse en la misma ubicación de Amazon S3 que los datos exportados que se utilizan para el procesamiento de datos (consulte: `outputS3Path`).

Para las tareas de clasificación o regresión de propiedades, estos archivos también pueden definir las etiquetas para la tarea de machine learning. En ese caso, los archivos deben tener una columna de propiedades con el mismo nombre de encabezado que el definido en el archivo de configuración de los datos de entrenamiento. Si las etiquetas de propiedades están definidas tanto en los archivos de nodos y de bordes exportados como en los archivos de división personalizada, se da prioridad a los archivos de división personalizada.

Entrenamiento de un modelo con Neptune ML

Una vez que haya procesado los datos que ha exportado de Neptune para el entrenamiento de modelos, puede iniciar un trabajo de entrenamiento de modelos mediante un comando `curl` (o `awscurl`), tal y como se indica a continuación:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Los detalles sobre cómo usar este comando se explican en El comando modeltraining, junto con información sobre cómo obtener el estado de un trabajo en ejecución, cómo detener un trabajo en ejecución y cómo enumerar todos los trabajos en ejecución.

También puede proporcionar un `previousModelTrainingJobId` para utilizar la información de un trabajo de entrenamiento de modelos de Neptune ML que se haya completado para acelerar la búsqueda de hiperparámetros en un nuevo trabajo de entrenamiento. Esto es útil durante el reentrenamiento de modelos con nuevos datos de gráficos, así como durante el entrenamiento incremental de los mismos datos de gráficos. Utilice un comando como este:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the model-training job-id of a completed job)"
      }'
```

Puede entrenar la implementación de su propio modelo en la infraestructura de entrenamiento de Neptune ML proporcionando un objeto `customModelTrainingParameters`, tal y como se indica a continuación:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Consulte El comando modeltraining para obtener más información, por ejemplo, sobre cómo obtener el estado de un trabajo en ejecución, cómo detener un trabajo en ejecución y cómo enumerar todos los trabajos en ejecución. Consulte Modelos personalizados de Neptune ML para obtener información sobre cómo implementar y utilizar un modelo personalizado.

Temas

- Modelos y entrenamiento de modelos en Amazon Neptune ML
- Personalización de las configuraciones de hiperparámetros de modelos en Neptune ML
- Prácticas recomendadas de entrenamiento de modelos

Modelos y entrenamiento de modelos en Amazon Neptune ML

Neptune ML utiliza redes neuronales de gráficos (GNN) para crear modelos para las distintas tareas de machine learning. Se ha demostrado que las redes neuronales de gráficos obtienen resultados de última generación para las tareas de machine learning de gráficos y son excelentes para extraer patrones informativos a partir de datos estructurados mediante gráficos.

Redes neuronales de gráficos (GNN) en Neptune ML

Las redes neuronales de gráficos (GNN) pertenecen a una familia de redes neuronales que calculan las representaciones de los nodos teniendo en cuenta la estructura y las características de los nodos cercanos. Las GNN complementan otros métodos tradicionales de machine learning y redes neuronales que no son adecuados para los datos de gráficos.

Las GNN se utilizan para resolver tareas de machine learning, como la clasificación y regresión de nodos (predicción de propiedades de nodos) y la clasificación y regresión de bordes (predicción de propiedades de los bordes) o la predicción de enlaces (predicción de si dos nodos del gráfico deben estar conectados o no).

En general, el uso de una GNN para una tarea de machine learning consta de dos etapas:

- Una etapa de codificación, en la que la GNN calcula un vector de dimensión d para cada nodo del gráfico. Estos vectores también se denominan representaciones o incrustaciones.
- Una etapa de decodificación, que hace predicciones basadas en las representaciones codificadas.

Para la clasificación y regresión de nodos, las representaciones de los nodos se utilizan directamente para las tareas de clasificación y regresión. Para la clasificación y regresión de bordes, las representaciones de nodos de incidentes en un borde se utilizan como entrada para la clasificación o la regresión. Para la predicción de enlaces, se calcula una puntuación de probabilidad de borde mediante un par de representaciones de nodos y una representación de tipo de borde.

La biblioteca Deep Graph Library (DGL) facilita la definición y el entrenamiento eficaces de las GNN para estas tareas.

Los diferentes modelos de GNN se unifican bajo la formulación de transferencia de mensajes. En esta vista, la representación de un nodo en un gráfico se calcula mediante las representaciones de los vecinos del nodo (los mensajes), junto con la representación inicial del nodo. En Neptune ML, la representación inicial de un nodo se deriva de las características extraídas de sus propiedades, o se puede aprender y depende de la identidad del nodo.

Neptune ML también ofrece la opción de concatenar características de nodos y representaciones de nodos que se pueden aprender para que sirvan como representación del nodo original.

Para las diversas tareas de Neptune ML que implican gráficos con propiedades de nodos, utilizamos la red convolucional de gráficos relacional (R-GCN) para realizar la etapa de codificación. La R-GCN es una arquitectura de GNN adecuada para gráficos con varios tipos de nodos y bordes (estos se conocen como gráficos heterogéneos).

La red R-GCN consta de un número fijo de capas, apiladas una tras otra. Cada capa de la R-GCN utiliza los parámetros de su modelo que se pueden aprender para agregar información de la vecindad inmediata de un salto de un nodo. Dado que las capas posteriores utilizan las representaciones de salida de la capa anterior como entrada, el radio de la vecindad del gráfico que influye en la incrustación final de un nodo depende del número de capas (`num-layer`) de la red R-GCN.

Por ejemplo, esto significa que una red de dos capas usa información de nodos que están a dos saltos de distancia.

Para obtener más información sobre las GNN, consulte A Comprehensive Survey on Graph Neural Networks (Una encuesta exhaustiva sobre las redes neuronales de gráficos). Para obtener más información sobre la biblioteca Deep Graph Library (DGL), visite la página web de la DGL. Para ver un tutorial práctico sobre el uso de la DGL con las GNN, consulte Learning graph neural networks with Deep Graph Library (Obtener información sobre las redes neuronales de gráficos con Deep Graph Library).

Entrenamiento de las redes neuronales de gráficos

En el machine learning, el proceso de hacer que un modelo aprenda a hacer buenas predicciones para una tarea se denomina entrenamiento de modelos. Por lo general, esto se realiza especificando un determinado objetivo que se debe optimizar, así como un algoritmo que se debe utilizar para realizar esta optimización.

Este proceso se emplea para entrenar a una GNN con el fin de que aprenda también buenas representaciones para la tarea posterior. Creamos una función objetivo para esa tarea que se minimiza durante el entrenamiento de modelos. Por ejemplo, para la clasificación de nodos, utilizamos CrossEntropyLoss como objetivo, lo que penaliza las clasificaciones erróneas, y para la regresión de nodos, minimizamos MeanSquareError.

El objetivo suele ser una función de pérdida que toma las predicciones del modelo para un determinado punto de datos y las compara con el valor real de ese punto de datos. Devuelve el valor

de pérdida, que muestra qué tan lejos están las predicciones del modelo. El objetivo del proceso de entrenamiento es minimizar la pérdida y garantizar que las predicciones del modelo se acerquen al valor real.

El algoritmo de optimización utilizado en el aprendizaje profundo para el proceso de entrenamiento suele ser una variante del gradiente descendente. En Neptune ML, utilizamos Adam, que es un algoritmo para la optimización basada en gradientes de primer orden de funciones objetivo estocásticas basadas en estimaciones adaptativas de momentos de orden inferior.

Si bien el proceso de entrenamiento del modelo intenta garantizar que los parámetros del modelo aprendidos estén cerca de los mínimos de la función objetivo, el rendimiento general de un modelo también depende de los hiperparámetros del modelo, que son ajustes del modelo que el algoritmo de entrenamiento no aprende. Por ejemplo, la dimensionalidad de la representación del nodo aprendida, `num-hidden`, es un hiperparámetro que afecta al rendimiento del modelo. Por lo tanto, en el machine learning es habitual realizar una optimización de hiperparámetros (HPO) para elegir los hiperparámetros adecuados.

Neptune ML utiliza un trabajo de ajuste de hiperparámetros de SageMaker para lanzar varias instancias de entrenamiento de modelos con diferentes configuraciones de hiperparámetros con el fin de encontrar el mejor modelo para una serie de ajustes de hiperparámetros. Consulte Personalización de las configuraciones de hiperparámetros de modelos en Neptune ML.

Modelos de incrustación de gráficos de conocimientos de Neptune ML

Los gráficos de conocimientos (KG) son gráficos que codifican información sobre diferentes entidades (nodos) y sus relaciones (bordes). En Neptune ML, los modelos de incrustación de gráficos de conocimientos se aplican de forma predeterminada para realizar la predicción de enlaces cuando el gráfico no incluye propiedades de nodo, solo relaciones con otros nodos. Aunque los modelos de R-GCN con incrustaciones que se pueden aprender también se pueden usar para estos gráficos (si indicamos `"rgcn"` como tipo de modelo), los modelos de incrustación de gráficos de conocimientos son más sencillos y se han diseñado para ser eficaces a la hora de aprender representaciones de gráficos de conocimiento a gran escala.

Los modelos de incrustación de gráficos de conocimientos se utilizan en una tarea de predicción de enlaces para predecir los nodos o relaciones que completan un triple `(h, r, t)`, donde **`h`** es el nodo de origen, **`r`** es el tipo de relación y **`t`** es el nodo de destino.

Los modelos de incrustación de gráficos de conocimientos implementados en Neptune ML son `distmult`, `transE` y `rotatE`. Para obtener más información sobre los modelos de incrustación de gráficos de conocimientos, consulte DGL-KE.

Entrenamiento de modelos personalizados en Neptune ML

Neptune ML le permite definir e implementar sus propios modelos personalizados para situaciones específicas. Consulte Modelos personalizados de Neptune ML para obtener información sobre cómo implementar un modelo personalizado y cómo usar la infraestructura de Neptune ML para entrenarlo.

Personalización de las configuraciones de hiperparámetros de modelos en Neptune ML

Al iniciar un trabajo de entrenamiento de modelos, Neptune ML utiliza automáticamente la información inferida del trabajo de procesamiento de datos anterior. Utiliza la información para generar rangos de configuración de hiperparámetros que se utilizan para crear un trabajo de ajuste de hiperparámetros de SageMaker con el fin de entrenar varios modelos para su tarea. De esta forma, no es necesario especificar una larga lista de valores de hiperparámetros para los modelos con los que se va a entrenar. En cambio, los rangos de hiperparámetros y los valores predeterminados del modelo se seleccionan en función del tipo de tarea, el tipo de gráfico y la configuración del trabajo de ajuste.

Sin embargo, también puede anular la configuración predeterminada de los hiperparámetros y proporcionar hiperparámetros personalizados. Para ello, modifique un archivo de configuración de JSON que genera el trabajo de procesamiento de datos.

Con la API de modelTraining de Neptune ML, puede controlar varias configuraciones de trabajos de ajuste de hiperparámetros de alto nivel, como, por ejemplo, `maxHPONumberOfTrainingJobs`, `maxHPOParallelTrainingJobs` y `trainingInstanceType`. Para un control más detallado de los hiperparámetros de los modelos, puede personalizar el archivo `model-HPO-configuration.json` que genera el trabajo de procesamiento de datos. El archivo se guarda en la ubicación de Amazon S3 que especificó para la salida del trabajo de procesamiento.

Puede descargar el archivo, editarlo para anular las configuraciones de hiperparámetros predeterminadas y volver a cargarlo en la misma ubicación de Amazon S3. No cambie el nombre del archivo y asegúrese de seguir estas instrucciones al editarlo.

Para descargar el archivo de Amazon S3:

```
aws s3 cp \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json \
  ./
```

Cuando haya terminado de editarlo, vuelva a cargar el archivo donde estaba:

```
aws s3 cp \
  model-HPO-configuration.json \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json
```

Estructura del archivo **model-HPO-configuration.json**

El archivo `model-HPO-configuration.json` especifica el modelo que se va a entrenar, la opción `task_type` del machine learning y los hiperparámetros que deben modificarse o fijarse para las distintas ejecuciones de entrenamiento del modelo.

Los hiperparámetros se clasifican como pertenecientes a varios niveles, lo que indica la prioridad que se da a los hiperparámetros cuando se invoca el trabajo de ajuste de hiperparámetros:

- Los hiperparámetros de nivel 1 tienen la prioridad más alta. Si se establece `maxHPONumberOfTrainingJobs` en un valor inferior a 10, solo se ajustan los hiperparámetros de nivel 1 y el resto toma sus valores predeterminados.
- Los hiperparámetros de nivel 2 tienen una prioridad inferior, por lo que si tiene más de 10 pero menos de 50 trabajos de entrenamiento en total para un trabajo de ajuste, se ajustarán los hiperparámetros de nivel 1 y 2.
- Los hiperparámetros de nivel 3 se ajustan junto con los de nivel 1 y 2 solo si tiene más de 50 trabajos de formación en total.
- Por último, los hiperparámetros fijos no se ajustan en absoluto y siempre toman sus valores predeterminados.

Ejemplo de un archivo **model-HPO-configuration.json**

A continuación se muestra un archivo `model-HPO-configuration.json` de ejemplo:

```
{
  "models": [
    {
      "model": "rgcn",
      "task_type": "node_class",
      "eval_metric": {
        "metric": "acc"
      },
      "eval_frequency": {
          "type":  "evaluate_every_epoch",
          "value":  1
      },
      "1-tier-param": [
        {
            "param": "num-hidden",
            "range": [16, 128],
```

```
            "type": "int",
            "inc_strategy": "power2"
        },
        {
          "param": "num-epochs",
          "range": [3,30],
          "inc_strategy": "linear",
          "inc_val": 1,
          "type": "int",
          "node_strategy": "perM"
        },
        {
          "param": "lr",
          "range": [0.001,0.01],
          "type": "float",
          "inc_strategy": "log"
        }
      ],
      "2-tier-param": [
        {
          "param": "dropout",
          "range": [0.0,0.5],
          "inc_strategy": "linear",
          "type": "float",
          "default": 0.3
        },
        {
          "param": "layer-norm",
          "type": "bool",
          "default": true
        }
      ],
      "3-tier-param": [
        {
          "param": "batch-size",
          "range": [128, 4096],
          "inc_strategy": "power2",
          "type": "int",
          "default": 1024
        },
        {
          "param": "fanout",
          "type": "int",
          "options": [[10, 30],[15, 30], [15, 30]],
```

```
        "default": [10, 15, 15]
      },
      {
        "param": "num-layer",
        "range": [1, 3],
        "inc_strategy": "linear",
        "inc_val": 1,
        "type": "int",
        "default": 2
      },
      {
        "param": "num-bases",
        "range": [0, 8],
        "inc_strategy": "linear",
        "inc_val": 2,
        "type": "int",
        "default": 0
      }
    ],
    "fixed-param": [
      {
        "param": "concat-node-embed",
        "type": "bool",
        "default": true
      },
      {
        "param": "use-self-loop",
        "type": "bool",
        "default": true
      },
      {
        "param": "low-mem",
        "type": "bool",
        "default": true
      },
      {
        "param": "l2norm",
        "type": "float",
        "default": 0
      }
    ]
  }
]
```

```
}
```

Elementos de un archivo **`model-HPO-configuration.json`**

El archivo incluye un objeto de JSON con una única matriz de nivel superior denominado `models` que incluye un único objeto de configuración de modelo. Al personalizar el archivo, asegúrese de que la matriz de `models` solo tenga un objeto de configuración de modelo. Si el archivo incluye más de un objeto de configuración de modelo, se producirá un error en el trabajo de ajuste y aparecerá una advertencia.

El objeto de configuración de modelo incluye los siguientes elementos de nivel superior:

- **`model`**: (cadena) el tipo de modelo que se va a entrenar (no lo modifique). Los valores válidos son:
  - `"rgcn"`: este es el valor predeterminado para las tareas de clasificación y regresión de nodos y para las tareas heterogéneas de predicción de enlaces.
  - `"transe"`: este es el valor predeterminado para las tareas de predicción de enlaces de KGE.
  - `"distmult"`: este es un tipo de modelo alternativo para las tareas de predicción de enlaces de KGE.
  - `"rotate"`: este es un tipo de modelo alternativo para las tareas de predicción de enlaces de KGE.

  Por regla general, no modifique directamente el valor de `model`, ya que los diferentes tipos de modelos suelen tener hiperparámetros aplicables bastante diferentes, lo que puede provocar un error de análisis una vez iniciado el trabajo de entrenamiento.

  Para cambiar el tipo de modelo, utilice el parámetro `modelName` de la API de modelTraining en lugar de cambiarlo en el archivo `model-HPO-configuration.json`.

  Una forma de cambiar el tipo de modelo y realizar cambios detallados en los hiperparámetros consiste en copiar la plantilla predeterminada de configuración del modelo para el modelo que desee usar y pegarla en el archivo `model-HPO-configuration.json`. Hay una carpeta con el nombre `hpo-configuration-templates` en la misma ubicación de Amazon S3 que el archivo `model-HPO-configuration.json` si el tipo de tarea inferido admite varios modelos. Esta carpeta incluye todas las configuraciones predeterminadas de hiperparámetros para los demás modelos que se aplican a la tarea.

  Por ejemplo, si desea cambiar las configuraciones del modelo y de los hiperparámetros de una tarea de predicción de enlaces de KGE del modelo predeterminado`transe` a un modelo

`distmult`, simplemente pegue el contenido del archivo `hpo-configuration-templates/distmult.json` en el archivo `model-HPO-configuration.json` y, a continuación, edite los hiperparámetros según sea necesario.

> **Note**
>
> Si establece el parámetro `modelName` en la API de `modelTraining` y también cambia la opción `model` y la especificación del hiperparámetro en el archivo `model-HPO-configuration.json`, y estos son diferentes, el valor `model` del archivo `model-HPO-configuration.json` tendrá prioridad y se omitirá `modelName`.

- **`task_type`**: (cadena) el tipo de tarea de machine learning inferido o transferido directamente al trabajo de procesamiento de datos (no lo modifique). Los valores válidos son:
  - `"node_class"`
  - `"node_regression"`
  - `"link_prediction"`

  El trabajo de procesamiento de datos infiere el tipo de tarea examinando las propiedades del conjunto de datos exportado y el archivo de configuración del trabajo de entrenamiento generado.

  Este valor no debe modificarse. Si desea entrenar otra tarea, debe ejecutar un nuevo trabajo de procesamiento de datos. Si el valor `task_type` no es el que esperaba, debe comprobar las entradas del trabajo de procesamiento de datos para asegurarse de que sean correctas. Esto incluye los parámetros de la API de `modelTraining` y del archivo de configuración del trabajo de entrenamiento generado por el proceso de exportación de datos.
- **`eval_metric`**: (cadena) la métrica de evaluación debe usarse para evaluar el rendimiento del modelo y seleccionar el modelo con mejor rendimiento en todas las ejecuciones de HPO. Los valores válidos son:
  - `"acc"`: precisión de clasificación estándar. Este es el valor predeterminado para las tareas de clasificación de una sola etiqueta, a menos que se encuentren etiquetas desequilibradas durante el procesamiento de los datos, en cuyo caso el valor predeterminado es `"F1"`.
  - `"acc_topk"`: el número de veces que la etiqueta correcta figura entre las principales predicciones de **k**. Para establecer el valor **k**, también puede proporcionar el valor `topk` como clave adicional.
  - `"F1"`: el valor F1.
  - `"mse"`: métrica de error cuadrático medio, para tareas de regresión.

- `"mrr"`: métrica de rango recíproco medio.
- `"precision"`: la precisión del modelo, calculada como la proporción entre los positivos verdaderos y los positivos pronosticados: `= true-positives / (true-positives + false-positives)`.
- `"recall"`: la exhaustividad del modelo, calculada como la proporción entre los positivos verdaderos y los positivos pronosticados: `= true-positives / (true-positives + false-negatives)`.
- `"roc_auc"`: el área debajo de la curva ROC. Este es el valor predeterminado para la clasificación de varias etiquetas.

Por ejemplo, para cambiar la métrica a `F1`, cambie el valor `eval_metric` de la siguiente manera:

```
"  eval_metric": {
    "metric": "F1",
  },
```

O bien, si desea cambiar la métrica a una puntuación de precisión `topk`, debería cambiar el valor `eval_metric` de la siguiente manera:

```
  "eval_metric": {
    "metric": "acc_topk",
    "topk": 2
  },
```

- **`eval_frequency`**: (objeto) especifica con qué frecuencia se debe comprobar el rendimiento del modelo en el conjunto de validación durante el entrenamiento. En función del rendimiento de la validación, se puede iniciar una detención temprana y guardar el mejor modelo.

  El objeto `eval_frequency` incluye dos elementos: `"type"` y `"value"`. Por ejemplo:

```
  "eval_frequency": {
    "type":  "evaluate_every_pct",
    "value":  0.1
  },
```

  Los valores válidos de `type` son:

  - **`evaluate_every_pct`**: especifica el porcentaje de entrenamiento que debe completarse en cada evaluación.

En el caso de `evaluate_every_pct`, el campo `"value"` incluye un número de coma flotante entre cero y uno que expresa ese porcentaje.

- **`evaluate_every_batch`**: especifica el número de lotes entrenamiento que debe completarse en cada evaluación.

  En el caso de `evaluate_every_batch`, el campo `"value"` incluye un número entero que expresa el recuento de lotes.
- **`evaluate_every_epoch`**: especifica el número de épocas por evaluación, donde una nueva época comienza a medianoche.

  En el caso de `evaluate_every_epoch`, el campo `"value"` incluye un número entero que expresa el recuento de épocas.

El ajuste predeterminado de `eval_frequency` es:

```
"eval_frequency": {
  "type":  "evaluate_every_epoch",
  "value":  1
},
```

- **`1-tier-param`**: (obligatorio) una matriz de hiperparámetros de nivel 1.

  Si no desea ajustar ningún hiperparámetro, puede hacerlo en una matriz vacía. Esto no afecta al número total de trabajos de entrenamiento lanzados por el trabajo de ajuste de hiperparámetros de SageMaker. Tan solo significa que todos los trabajos de entrenamiento, si hay más de 1 pero menos de 10, se ejecutarán con el mismo conjunto de hiperparámetros.

  Por otro lado, si desea tratar todos los hiperparámetros ajustables con la misma importancia, puede colocar todos los hiperparámetros en esta matriz.
- **`2-tier-param`**: (obligatorio) una matriz de hiperparámetros de nivel 2.

  Estos parámetros solo se ajustan si la opción `maxHPONumberOfTrainingJobs` tiene un valor superior a 10. De lo contrario, se fijan a los valores predeterminados.

  Si tiene un presupuesto de entrenamiento de un máximo de 10 trabajos de entrenamiento o no quiere hiperparámetros de nivel 2 por algún otro motivo, pero desea ajustar todos los hiperparámetros ajustables, puede hacerlo en una matriz vacía.
- **`3-tier-param`**: (obligatorio) una matriz de hiperparámetros de nivel 3.

Estos parámetros solo se ajustan si la opción `maxHPONumberOfTrainingJobs` tiene un valor superior a 50. De lo contrario, se fijan a los valores predeterminados.

Si no desea ajustar los hiperparámetros de nivel 3, puede hacerlo en una matriz vacía.

- **`fixed-param`**: (obligatorio) una matriz de hiperparámetros fijos que solo toman sus valores predeterminados y no varían en los distintos trabajos de entrenamiento.

  Si desea modificar todos los hiperparámetros, puede hacerlo en una matriz vacía y establecer el valor para `maxHPONumberOfTrainingJobs` lo suficientemente grande como para modificar todos los niveles o hacer que todos los hiperparámetros sean de nivel 1.

El objeto de JSON que representa cada hiperparámetro de `1-tier-param`, `2-tier-param`, `3-tier-param` y `fixed-param` incluye los siguientes elementos:

- **`param`**: (cadena) el nombre del hiperparámetro (no lo modifique).

  Consulte la lista de nombres de hiperparámetros válidos en Neptune ML.

- **`type`**: (cadena) el tipo del hiperparámetro (no lo modifique).

  Los tipos válidos son `bool`, `int` y `float`.

- **`default`**: (cadena) El valor predeterminado del hiperparámetro.

  Puede establecer un nuevo valor predeterminado.

Los hiperparámetros ajustables también pueden incluir los siguientes elementos:

- **`range`**: (matriz) el rango de un hiperparámetro ajustable continuo.

  Debe ser una matriz con dos valores, es decir, el mínimo y el máximo del rango (`[min, max]`).

- **`options`**: (matriz) las opciones de un hiperparámetro ajustable categórico.

  Esta matriz debe incluir todas las opciones que se deben tener en cuenta:

  ```
  "options" : [value1, value2, ... valuen]
  ```

- **`inc_strategy`**: (cadena) el tipo de cambio incremental para rangos de hiperparámetros ajustables continuos (no lo modifique).

Los valores válidos son `log`, `linear` y `power2`. Esto solo se aplica cuando se establece la clave de rango.

Si se modifica, es posible que no se utilice todo el rango del hiperparámetro para el ajuste.

- **`inc_val`**: (flotante) la diferencia entre los incrementos sucesivos de los hiperparámetros ajustables continuo (no lo modifique).

  Esto solo se aplica cuando se establece la clave de rango.

  Si se modifica, es posible que no se utilice todo el rango del hiperparámetro para el ajuste.
- **`node_strategy`**: (cadena) indica que el rango efectivo de este hiperparámetro debe cambiar en función del número de nodos del gráfico (no lo modifique).

  Los valores válidos son `"perM"` (por millón), `"per10M"` (por 10 millones) y `"per100M"` (por 100 millones).

  En lugar de cambiar este valor, cambie la opción `range`.
- **`edge_strategy`**: (cadena) indica que el rango efectivo de este hiperparámetro debe cambiar en función del número de bordes del gráfico (no lo modifique).

  Los valores válidos son `"perM"` (por millón), `"per10M"` (por 10 millones) y `"per100M"` (por 100 millones).

  En lugar de cambiar este valor, cambie la opción `range`.

Lista de todos los hiperparámetros de Neptune ML

En la siguiente lista se incluyen todos los hiperparámetros que se pueden configurar en cualquier lugar de Neptune ML, para cualquier tipo de modelo y tarea. Como no todos son aplicables a todos los tipos de modelo, es importante que solo establezca los hiperparámetros en el archivo `model-HPO-configuration.json` que aparece en la plantilla del modelo que esté utilizando.

- **`batch-size`**: el tamaño del lote de nodos de destino que se utilizan en un pase hacia adelante. Tipo: `int`.

  Si se establece en un valor mucho mayor, se pueden producir problemas de memoria durante el entrenamiento en instancias de GPU.

- **`concat-node-embed`**: indica si se debe obtener la representación inicial de un nodo concatenando sus características procesadas con incrustaciones de nodos iniciales que se puedan aprender con el fin de aumentar la expresividad del modelo. Tipo: `bool`.
- **`dropout`**: la probabilidad de abandono aplicada a las capas de abandono. Tipo: `float`.
- **`edge-num-hidden`**: el tamaño de la capa oculta o el número de unidades del módulo de características de borde. Solo se usa cuando el valor `use-edge-features` está establecido en `True`. Tipo: flotante.
- **`enable-early-stop`**: cambia si se utiliza o no la característica de detención temprana. Tipo: `bool`. Valor predeterminado: `true`.

  Utilice este parámetro booleano para desactivar la característica de detención temprana.
- **`fanout`**: el número de vecinos que se deben muestrear para un nodo de destino durante el muestreo de vecinos. Tipo: `int`.

  Este valor está estrechamente relacionado con el mismo nivel de hiperparámetros `num-layers` y siempre debe estar en el mismo. Esto se debe a que puede especificar una distribución ramificada para cada posible capa de GNN.

  Como este hiperparámetro puede hacer que el rendimiento del modelo varíe considerablemente, debe ser fijo o establecerse como un hiperparámetro de nivel 2 o 3. Si se establece en un valor grande se pueden producir problemas de memoria durante el entrenamiento en instancias de GPU.
- **`gamma`**: el valor del margen de la función de puntuación. Tipo: `float`.

  Esto se aplica únicamente a los modelos de predicción de enlaces de `KGE`.
- **`l2norm`**: el valor de decadencia de ponderación utilizado en el optimizador, que impone una penalización de normalización L2 a las ponderaciones. Tipo: `bool`.
- **`layer-norm`**: indica si se debe utilizar la normalización de capas para los modelos `rgcn`. Tipo: `bool`.
- **`low-mem`**: indica si se debe utilizar una implementación con poca memoria de la función de transferencias de mensajes de relación en detrimento de la velocidad. Tipo: `bool`.
- **`lr`**: la tasa de aprendizaje. Tipo: `float`.

  Debe configurarse como un hiperparámetro de nivel 1.

- **`neg-share`**: en la predicción de enlaces, indica si los bordes muestreados positivos pueden compartir ejemplos de bordes negativos. Tipo: `bool`.
- **`num-bases`**: el número de bases para la descomposición de bases en un modelo `rgcn`. El uso de un valor de `num-bases` que sea inferior al número de tipos de bordes del gráfico actúa como regularizador del modelo `rgcn`. Tipo: `int`.
- **`num-epochs`**: el número de épocas del entrenamiento que se va a ejecutar. Tipo: `int`.

  Una época es un recorrido completo de entrenamiento por el gráfico.
- **`num-hidden`**: el tamaño de la capa oculta o el número de unidades. Tipo: `int`.

  Esto también establece el tamaño de incrustación inicial para los nodos sin características.

  Si se establece en un valor mucho mayor sin reducir la opción `batch-size`, se pueden producir problemas de memoria durante el entrenamiento en la instancia de GPU.
- **`num-layer`**: el número de capas GNN en el modelo. Tipo: `int`.

  Este valor está estrechamente relacionado con el parámetro de distribución ramificada y debe aparecer después de que la distribución ramificada se establezca en la misma capa de hiperparámetro.

  Dado que esto puede hacer que el rendimiento del modelo varíe considerablemente, debe ser fijo o establecerse como un hiperparámetro de nivel 2 o 3.
- **`num-negs`**: en la predicción de enlaces, el número de muestras negativas por muestra positiva. Tipo: `int`.
- **`per-feat-name-embed`**: indica si se debe incrustar cada característica transformándola de forma independiente antes de combinar características. Tipo: `bool`.

  Si se establece en `true`, cada característica por nodo se transforma de forma independiente en un tamaño de dimensión fijo antes de que todas las características transformadas del nodo se concatenen y se transformen posteriormente en la dimensión `num_hidden`.

  Cuando se establece en `false`, las características se concatenan sin ninguna transformación específica de la característica.
- **`regularization-coef`**: en la predicción de enlaces, el coeficiente de pérdida de regularización. Tipo: `float`.
- **`rel-part`**: indica si se debe utilizar la partición de relaciones para la predicción de enlaces de `KGE`. Tipo: `bool`.

- **`sparse-lr`**: la tasa de aprendizaje de las incrustaciones de nodos que se pueden aprender. Tipo: `float`.

  Las incrustaciones de nodos iniciales que se pueden aprender se utilizan para los nodos sin características o cuando esté establecida la opción `concat-node-embed`. Los parámetros de la capa de incrustación de nodos dispersos que se pueden aprender se entrenan mediante un optimizador independiente que puede tener una tasa de aprendizaje diferente.
- **`use-class-weight`**: indica si se deben aplicar ponderaciones de clase a las tareas de clasificación desequilibradas. Si se establece en `true`, los recuentos de etiquetas se utilizan para establecer una ponderación para cada etiqueta de clase. Tipo: `bool`.
- **`use-edge-features`**: indica si se deben utilizar las características de borde al transferir los mensajes. Si se establece en `true`, se añade un módulo de características de borde personalizado a la capa RGCN para los tipos de bordes que tienen características. Tipo: `bool`.
- **`use-self-loop`**: indica si se deben incluir bucles automáticos en el entrenamiento de un modelo `rgcn`. Tipo: `bool`.
- **`window-for-early-stop`**: controla el número de puntuaciones de validación más recientes para obtener una media con el fin de decidir si se debe detener antes de tiempo. El valor predeterminado es 3. type=int. Véase también Detención temprana del proceso de entrenamiento de modelos en Neptune ML. Tipo: `int`. Valor predeterminado: `3`.

  Consulte .

Personalización de hiperparámetros en Neptune ML

Al editar el archivo `model-HPO-configuration.json`, los tipos de cambios más habituales que hay que realizar son los siguientes:

- Edite los valores mínimo o máximo para de los hiperparámetros `range`.
- Para establecer un hiperparámetro en un valor fijo, muévalo a la sección `fixed-param` y establezca su valor predeterminado en el valor fijo que desee que adquiera.
- Para cambiar la prioridad de un hiperparámetro, colóquelo en un determinado nivel, edite su rango y asegúrese de que su valor predeterminado esté establecido de forma adecuada.

Prácticas recomendadas de entrenamiento de modelos

Hay cosas que puede hacer para mejorar el rendimiento de los modelos de Neptune ML.

Selección de la propiedad de nodo correcta

Es posible que no todas las propiedades del gráfico sean significativas o relevantes para las tareas de machine learning. Debe excluirse cualquier propiedad irrelevante durante la exportación de datos.

A continuación le presentamos algunas prácticas recomendadas:

- Recurra a expertos en la materia para que le ayuden a evaluar la importancia de las características y la viabilidad de utilizarlas para hacer predicciones.
- Elimine las características que considere redundantes o irrelevantes para reducir el ruido en los datos y las correlaciones sin importancia.
- Realice iteraciones a medida que crea el modelo. Ajuste las características, las combinaciones de características y ajuste los objetivos a medida que avanza.

En la sección Procesamiento de características de la Guía para desarrolladores de Amazon Machine Learning se proporcionan directrices adicionales para el procesamiento de características que son relevantes para Neptune ML.

Gestión de puntos de datos atípicos

Un valor atípico es un punto de datos que difiere bastante del resto de datos. Los valores atípicos de los datos pueden confundir o engañar al proceso de entrenamiento, lo que se traduce en un tiempo de entrenamiento más prolongado o en modelos menos precisos. A menos que sean realmente importantes, debe eliminar los valores atípicos antes de exportar los datos.

Eliminación de los nodos y bordes duplicados

Los gráficos almacenados en Neptune pueden tener nodos o bordes duplicados. Estos elementos redundantes introducirán ruido durante el entrenamiento de modelos de ML. Elimine los nodos o bordes duplicados antes de exportar los datos.

Ajuste de la estructura del gráfico

Al exportar el gráfico, puede cambiar la forma en que se procesan las características y cómo se crea el gráfico para mejorar el rendimiento del modelo.

A continuación le presentamos algunas prácticas recomendadas:

- Cuando una propiedad de borde tiene el significado de categorías de bordes, en algunos casos, merece la pena convertirla en tipos de borde.
- La política de normalización predeterminada que se utiliza para una propiedad numérica es `min-max`, pero, en algunos casos, otras políticas de normalización funcionan mejor. Puede preprocesar la propiedad y cambiar la política de normalización, tal y como se explica en Elementos de un archivo `model-HPO-configuration.json`.
- El proceso de exportación genera automáticamente tipos de características en función de los tipos de propiedad. Por ejemplo, trata las propiedades de `String` como características categóricas y las propiedades de `Float` y `Int` como características numéricas. Si fuera necesario, puede modificar el tipo de característica después de la exportación (consulte Elementos de un archivo `model-HPO-configuration.json`).

Ajuste de los rangos y valores predeterminados de los hiperparámetros

La operación de procesamiento de datos infiere los rangos de configuración de los hiperparámetros a partir del gráfico. Si los rangos de hiperparámetros y los valores predeterminados del modelo generados no funcionan bien con los datos del gráfico, puede editar el archivo de configuración de HPO para crear su propia estrategia de ajuste de hiperparámetros.

A continuación le presentamos algunas prácticas recomendadas:

- Al ampliar el gráfico, es posible que el tamaño predeterminado de la dimensión oculta no sea lo suficientemente grande como para incluir toda la información. Puede cambiar el hiperparámetro `num-hidden` para controlar el tamaño de la dimensión oculta.
- En el caso de los modelos de incrustación de gráficos de conocimientos (KGE), es posible que desee cambiar el modelo específico que se utiliza en función del presupuesto y la estructura del gráfico.

  Los modelos de `TrainsE` tienen dificultades para tratar relaciones de uno a muchos (1-N), de muchos a uno (N-1) y de muchos a muchos (N-N). Los modelos de `DistMult` tienen dificultades para tratar las relaciones simétricas. `RotatE` es una buena opción para modelar todo tipo de relaciones, pero es más caro que `TrainsE` y `DistMult` durante el entrenamiento.
- En algunos casos, cuando la identificación del nodo y la información de las características del nodo son importantes, debe utilizar `` `concat-node-embed` `` para indicar al modelo de Neptune ML que

obtenga la representación inicial de un nodo mediante la concatenación de sus características con sus incrustaciones iniciales.

- Si obtiene un rendimiento razonablemente bueno con algunos hiperparámetros, puede ajustar el espacio de búsqueda de hiperparámetros en función de estos resultados.

Detención temprana del proceso de entrenamiento de modelos en Neptune ML

La detención temprana puede reducir considerablemente el tiempo de ejecución del entrenamiento de modelos y los costos asociados sin degradar el rendimiento de los modelos. También evita que el modelo se sobreajuste a los datos de entrenamiento.

La detención temprana depende de las mediciones periódicas del rendimiento del conjunto de validación. Inicialmente, el rendimiento mejora a medida que avanza el entrenamiento, pero cuando el modelo comienza a sobreajustarse, vuelve a disminuir. La característica de detención temprana identifica el punto en el que el modelo comienza a sobreajustarse y detiene el entrenamiento del modelo en ese momento.

Neptune ML supervisa las llamadas a las métricas de validación y compara la métrica de validación más reciente con el promedio de las métricas de validación de las últimas evaluaciones de **n**, donde **n** es un número establecido mediante el parámetro `window-for-early-stop`. En cuanto la métrica de validación es inferior al promedio, Neptune ML detiene el entrenamiento del modelo y guarda el mejor modelo hasta el momento.

Puede controlar la detención temprana mediante los siguientes parámetros:

- **`window-for-early-stop`**: el valor de este parámetro es un número entero que especifica el número de puntuaciones de validación recientes que se van a promediar cuando se decide si realizar una detención temprana. El valor predeterminado es `3`.
- **`enable-early-stop`**: utilice este parámetro booleano para desactivar la característica de detención temprana. De forma predeterminada su valor es `true`.

Detención temprana del proceso de HPO en Neptune ML

La característica de detención temprana de Neptune ML también detiene los trabajos de entrenamiento que no tienen un buen rendimiento en comparación con otros trabajos, mediante la característica de inicio en caliente de HPO de SageMaker. Esto también puede reducir los costos y mejorar la calidad de HPO.

Consulte Ejecución de un trabajo de ajuste de hiperparámetros de inicio en caliente para obtener una descripción de su funcionamiento.

El inicio en caliente permite transferir la información aprendida en trabajos de entrenamiento anteriores a trabajos de entrenamiento posteriores y ofrece dos ventajas bien diferenciadas:

- En primer lugar, los resultados de los trabajos de entrenamiento anteriores se utilizan para seleccionar combinaciones correctas de hiperparámetros para realizar búsquedas en el nuevo trabajo de ajuste.
- En segundo lugar, permite una detención temprana para acceder a más ejecuciones del modelo, lo que reduce el tiempo de ajuste.

Esta característica se habilita automáticamente en Neptune ML y le permite lograr un equilibrio entre el tiempo de entrenamiento del modelo y el rendimiento. Si está satisfecho con el rendimiento del modelo actual, puede utilizar ese modelo. De lo contrario, ejecute más HPO que se inicien en caliente con los resultados de ejecuciones anteriores para descubrir un modelo mejor.

Obtener servicios de soporte profesional

AWS ofrece servicios de soporte profesional para ayudarle con los problemas del machine learning en los proyectos de Neptune. Si se queda bloqueado, póngase en contacto con AWS Support.

Uso de un modelo entrenado para generar nuevos artefactos de modelo

Con el comando de transformación de modelos de Neptune ML, puede calcular artefactos de modelos, como incrustaciones de nodos, en datos de gráficos procesados mediante parámetros de modelo previamente entrenados.

Transformación de modelos para la inferencia incremental

En el flujo de trabajo de inferencia incremental de modelos, una vez que haya procesado los datos de gráficos actualizados que ha exportado de Neptune, puede iniciar un trabajo de transformación de modelo mediante un comando curl (o awscurl), tal y como se indica a continuación:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "mlModelTrainingJobId": "(the ML model training job-id)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

A continuación, puede transferir el ID de este trabajo a la llamada a la API de create-endpoints para crear un nuevo punto de conexión o actualizar uno existente con los nuevos artefactos del modelo generados por este trabajo. Esto permite que el punto de conexión nuevo o actualizado proporcione predicciones del modelo para los datos de gráficos actualizados.

Transformación de modelos para cualquier trabajo de entrenamiento

También puede proporcionar un parámetro `trainingJobName` para generar artefactos de modelos para cualquiera de los trabajos de entrenamiento de SageMaker lanzados durante el entrenamiento de modelos de Neptune ML. Dado que un trabajo de entrenamiento de modelos de Neptune ML puede lanzar muchos trabajos de entrenamiento de SageMaker, esto le ofrece la flexibilidad de crear un punto de conexión de inferencia basado en cualquiera de esos trabajos de entrenamiento de SageMaker.

Por ejemplo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
      }'
```

Si el trabajo de entrenamiento original era para un modelo personalizado proporcionado por el usuario, debe incluir un objeto `customModelTransformParameters` al invocar una transformación de modelo. Consulte Modelos personalizados de Neptune ML para obtener información sobre cómo implementar y utilizar un modelo personalizado.

> **Note**
>
> El comando `modeltransform` siempre ejecuta la transformación del modelo en el mejor trabajo de entrenamiento de SageMaker para ese entrenamiento.

Consulte El comando modeltransform para obtener más información sobre los trabajos de transformación de modelos.

Artefactos producidos por el entrenamiento de modelos en Neptune ML

Tras el entrenamiento del modelo, Neptune ML utiliza los parámetros del modelo mejor entrenados para generar los artefactos del modelo que son necesarios para lanzar el punto de conexión de inferencia y proporcionar predicciones del modelo. Estos artefactos se empaquetan según el trabajo de entrenamiento y se almacenan en la ubicación de salida de Amazon S3 del mejor trabajo de entrenamiento de SageMaker.

En las siguientes secciones se describe lo que se incluye en los artefactos de modelos para las distintas tareas y cómo el comando de transformación de modelos utiliza un modelo entrenado preexistente para generar artefactos, incluso a partir de datos de gráficos nuevos.

Artefactos generados para diferentes tareas

El contenido de los artefactos de modelos generados por el proceso de entrenamiento depende de la tarea de machine learning de destino:

- Clasificación y regresión de los nodos: para la predicción de las propiedades de los nodos, los artefactos incluyen los parámetros del modelo, las incrustaciones de nodos del codificador GNN, las predicciones del modelo para los nodos del gráfico de entrenamiento y algunos archivos de configuración para el punto de conexión de inferencia. En las tareas de clasificación y regresión de nodos, las predicciones del modelo se calculan previamente para los nodos presentes durante el entrenamiento con el fin de reducir la latencia de las consultas.
- Clasificación y regresión de bordes: para la predicción de las propiedades de las bordes, los artefactos también incluyen los parámetros del modelo y las incrustaciones de nodos. Los parámetros del decodificador del modelo son especialmente importantes para la inferencia, ya que calculamos la clasificación de los bordes o las predicciones de regresión de los bordes aplicando el decodificador del modelo a las incrustaciones del vértice de origen y destino de un borde.
- Predicción de enlaces: para la predicción de enlaces, además de los artefactos generados para la predicción de propiedades de borde, el gráfico DGL también se incluye como artefacto, ya que la predicción de enlaces requiere que el gráfico de entrenamiento realice las predicciones. El objetivo de la predicción de enlaces es predecir los vértices de destino que es probable que se combinen con un vértice de origen para formar un borde de un determinado tipo en el gráfico. Para ello, se combinan la incrustación de nodos del vértice de origen y una representación aprendida del tipo de borde con las incrustaciones de nodos de todos los vértices de destino posibles para obtener

una puntuación de probabilidad de borde para cada uno de los vértices de destino. A continuación, las puntuaciones se ordenan para clasificar los posibles vértices de destino y devolver los mejores candidatos.

Para cada uno de los tipos de tareas, las ponderaciones del modelo de red neuronal de gráficos se guardan en el artefacto del modelo. Esto permite a Neptune ML calcular salidas de modelos nuevos a medida que cambia el gráfico (inferencia inductiva), además de utilizar predicciones e incrustaciones precalculadas (inferencia transductiva) para reducir la latencia.

Creación de nuevos artefactos de modelos

Los artefactos de modelos generados después del entrenamiento del modelo en Neptune ML están directamente relacionados con el proceso de entrenamiento. Esto significa que las incrustaciones y predicciones precalculadas solo existen para las entidades que estaban en el gráfico de entrenamiento original. Aunque el modo de inferencia inductiva para los puntos de conexión de Neptune ML puede calcular predicciones para nuevas entidades en tiempo real, es posible que desee generar predicciones por lotes sobre nuevas entidades sin consultar un punto de conexión.

Para obtener las predicciones de modelos por lotes para las nuevas entidades que se han añadido al gráfico, es necesario volver a calcular los artefactos de modelos nuevos para los nuevos datos de gráfico. Esto se logra mediante el comando `modeltransform`. Utilice el comando `modeltransform` cuando solo desee realizar predicciones por lotes sin configurar un punto de conexión, o cuando desee que se generen todas las predicciones de forma que pueda volver a escribirlas en el gráfico.

Dado que el entrenamiento de modelos realiza de forma implícita una transformación del modelo al final del proceso de entrenamiento, los artefactos de modelos siempre se vuelven a calcular a partir de los datos del gráfico de entrenamiento mediante un trabajo de entrenamiento. Sin embargo, el comando `modeltransform` también puede calcular los artefactos de modelos en los datos del gráfico que no se usaron para entrenar un modelo. Para ello, los nuevos datos de gráficos deben procesarse con las mismas codificaciones de características que los datos de gráficos originales y deben seguir el mismo esquema de gráfico.

Para ello, cree primero un nuevo trabajo de procesamiento de datos que sea un clon del trabajo de procesamiento de datos ejecutado con los datos del gráfico de entrenamiento original y ejecútelo con los nuevos datos del gráfico (consulte Procesamiento de datos de gráficos actualizados para Neptune ML). A continuación, llame al comando `modeltransform` con el nuevo

`dataProcessingJobId` y el antiguo `modelTrainingJobId` para volver a calcular los artefactos del modelo a partir de los datos del gráfico actualizados.

Para la predicción de las propiedades de los nodos, las incrustaciones y predicciones de los nodos se vuelven a calcular a partir de los nuevos datos del gráfico, incluso también para los nodos que estaban presentes en el gráfico de entrenamiento original.

Para la predicción de las propiedades de los bordes y los enlaces, las incrustaciones de los nodos también se vuelven a calcular y, de forma similar, anulan cualquier incrustación de nodos existente. Para volver a calcular las incrustaciones de nodos, Neptune ML aplica el codificador GNN aprendido del modelo entrenado anterior a los nodos de los nuevos datos de gráficos con sus nuevas características.

En el caso de los nodos que no tengan características, se vuelven a utilizar las representaciones iniciales aprendidas del entrenamiento del modelo original. En el caso de los nodos nuevos que no tengan características y no estaban presentes en el gráfico de entrenamiento original, Neptune ML inicializa su representación como el promedio de las representaciones de nodos iniciales aprendidas de ese tipo de nodo presentes en el gráfico de entrenamiento original. Esto puede provocar una disminución en el rendimiento de las predicciones del modelo si hay muchos nodos nuevos que no tengan características, ya que todos se inicializarán con la incrustación inicial promedio de ese tipo de nodo.

Si el modelo se entrena con el valor `concat-node-embed` establecido en true, las representaciones de los nodos iniciales se crean concatenando las características del nodo con la representación inicial que se puede aprender. Por lo tanto, para el gráfico actualizado, la representación del nodo inicial de los nodos nuevos también utiliza el promedio de incrustaciones de nodos iniciales, concatenadas con las nuevas características de los nodos.

Además, actualmente no se admiten las eliminaciones de nodos. Si se han eliminado los nodos del gráfico actualizado, debe volver a entrenar el modelo a partir de los datos del gráfico actualizado.

Al volver a calcular los artefactos del modelo, se vuelven a utilizar los parámetros del modelo aprendidos en un gráfico nuevo, y solo debe realizarse cuando el nuevo gráfico sea muy similar al antiguo. Si el nuevo gráfico no es lo suficientemente similar, es necesario volver a entrenar el modelo para obtener un rendimiento similar con los datos del nuevo gráfico. Lo que se considere suficientemente similar depende de la estructura de los datos del gráfico, pero como regla general, debe volver a entrenar el modelo si los nuevos datos difieren entre un 10 y un 20 % de los datos del gráfico de entrenamiento original.

En el caso de los gráficos en los que todos los nodos tengan características, se aplica el extremo superior del umbral (un 20 % de diferencia), pero en el caso de los gráficos en los que muchos nodos no tengan características y los nuevos nodos añadidos al gráfico no tengan propiedades, el extremo inferior (un 10 % de diferencia) puede ser incluso demasiado alto.

Consulte El comando modeltransform para obtener más información sobre los trabajos de transformación de modelos.

Modelos personalizados de Neptune ML

Neptune ML le permite definir sus propias implementaciones de modelos personalizados mediante Python. Puede entrenar e implementar modelos personalizados con la infraestructura de Neptune ML de la misma manera que lo hace con los modelos integrados, y usarlos para obtener predicciones mediante consultas de gráficos.

> **Note**
>
> Por el momento, los modelos personalizados no admiten la inferencia inductiva en tiempo real.

Para empezar a implementar su propio modelo personalizado en Python, siga los ejemplos del kit de herramientas de Neptune ML y utilice los componentes del modelo que se proporcionan en el kit de herramientas de Neptune ML. En las siguientes secciones ofrecemos más información.

Contenido

Información general sobre los modelos personalizados de Neptune ML

Cuándo usar un modelo personalizado en Neptune ML

Los modelos integrados de Neptune ML gestionan todas las tareas estándar compatibles con Neptune ML, pero puede haber casos en los que desee tener un control más detallado sobre el modelo para una tarea concreta o necesite personalizar el proceso de entrenamiento de modelos. Por ejemplo, un modelo personalizado es adecuado en las siguientes situaciones:

- La codificación de características para las características de texto de los modelos de texto muy grandes deben ejecutarse en la GPU.
- Desea utilizar su propio modelo de red neuronal gráfica (GNN) personalizado y desarrollado en la biblioteca Deep Graph Library (DGL).
- Desea utilizar modelos tabulares o modelos de ensamblaje para la clasificación y regresión de nodos.

Flujo de trabajo para desarrollar y usar un modelo personalizado en Neptune ML

El soporte de modelos personalizados en Neptune ML se ha diseñado para integrarse sin problemas en los flujos de trabajo existentes de Neptune ML. Funciona mediante la ejecución de código personalizado en el módulo de origen de la infraestructura de Neptune ML para entrenar el modelo. Al igual que en el caso de un modo integrado, Neptune ML lanza automáticamente un trabajo de ajuste de hiperparámetros de SageMaker y selecciona el mejor modelo según la métrica de evaluación. A continuación, utiliza la implementación proporcionada en el módulo de origen para generar artefactos de modelos para su implementación.

La exportación de datos, la configuración de entrenamientos y el preprocesamiento de datos son los mismos para un modelo personalizado que para uno integrado.

Una vez realizado el preprocesamiento de los datos, puede desarrollar y probar de forma iterativa e interactiva la implementación de su modelo personalizado mediante Python. Cuando el modelo esté listo para la producción, puede cargar el módulo de Python resultante en Amazon S3 de la siguiente manera:

```
aws s3 cp --recursive (source path to module) s3://(bucket name)/(destination path for
 your module)
```

A continuación, puede usar el flujo de trabajo de datos normal predeterminado o incremental para implementar el modelo en producción, con algunas diferencias.

Para el entrenamiento de modelos con un modelo personalizado, debe proporcionar un objeto `customModelTrainingParameters` de JSON a la API de entrenamiento de modelos de Neptune ML para garantizar que se utilice el código personalizado. Los campos del objeto `customModelTrainingParameters` son los siguientes:

- **`sourceS3DirectoryPath`**: (obligatorio) la ruta a la ubicación de Amazon S3 donde se encuentra el módulo de Python que implementa el modelo. Debe apuntar a una ubicación válida de Amazon S3 existente que incluya, como mínimo, un script de entrenamiento, un script de transformación y un archivo `model-hpo-configuration.json`.
- **`trainingEntryPointScript`**: (opcional) el nombre del punto de entrada en el módulo de un script que realiza el entrenamiento de modelos y utiliza los hiperparámetros como argumentos de la línea de comandos, incluidos los hiperparámetros fijos.

  Valor predeterminado: `training.py`.
- **`transformEntryPointScript`**: (opcional) el nombre del punto de entrada en el módulo de un script que debe ejecutarse después de identificar el mejor modelo de la búsqueda de hiperparámetros, con el fin de calcular los artefactos de modelos necesarios para su implementación. Debería poder ejecutarse sin argumentos de línea de comandos.

  Valor predeterminado: `transform.py`.

Por ejemplo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
```

```
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Del mismo modo, para habilitar una transformación de modelo personalizada, debe proporcionar un objeto `customModelTransformParameters` de JSON a la API de transformación de Neptune ML, con valores de campo que sean compatibles con los parámetros del modelo guardados del trabajo de entrenamiento. El objeto `customModelTransformParameters` incluye estos campos:

- **`sourceS3DirectoryPath`**: (obligatorio) la ruta a la ubicación de Amazon S3 donde se encuentra el módulo de Python que implementa el modelo. Debe apuntar a una ubicación válida de Amazon S3 existente que incluya, como mínimo, un script de entrenamiento, un script de transformación y un archivo `model-hpo-configuration.json`.
- **`transformEntryPointScript`**: (opcional) el nombre del punto de entrada en el módulo de un script que debe ejecutarse después de identificar el mejor modelo de la búsqueda de hiperparámetros, con el fin de calcular los artefactos de modelos necesarios para su implementación. Debería poder ejecutarse sin argumentos de línea de comandos.

  Valor predeterminado: `transform.py`.

Por ejemplo:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Desarrollo de modelos personalizados en Neptune ML

Una buena forma de iniciar el desarrollo de modelos personalizados es seguir los ejemplos del kit de herramientas de Neptune ML para estructurar y escribir el módulo de entrenamiento. El kit de herramientas de Neptune ML también implementa componentes de modelos de ML de gráficos modularizados en el modelzoo que puede apilar y usar para crear un modelo personalizado.

Además, el kit de herramientas proporciona funciones de utilidad que le ayudan a generar los artefactos necesarios durante el entrenamiento y la transformación de modelos. Puede importar este paquete de Python en la implementación personalizada. Todas las funciones o módulos incluidos en el kit de herramientas también están disponibles en el entorno de entrenamiento de Neptune ML.

Si su módulo de Python tiene dependencias externas adicionales, puede incluirlas creando un `requirements.txt` archivo en el directorio del módulo. Los paquetes que figuran en el archivo `requirements.txt` se instalarán antes de ejecutar el script de entrenamiento.

Como mínimo, el módulo de Python que implementa el modelo personalizado debe incluir lo siguiente:

- Un punto de entrada de script de entrenamiento
- Un punto de entrada de script de transformación
- Un archivo `model-hpo-configuration.json`

Desarrollo de scripts de entrenamiento de modelos personalizados en Neptune ML

El script de entrenamiento del modelo personalizado debe ser un script de Python ejecutable, como el ejemplo `train.py` del kit de herramientas de Neptune ML. Debe aceptar nombres y valores de hiperparámetros como argumentos de línea de comandos. Durante el entrenamiento de modelos, los nombres de los hiperparámetros se obtienen del archivo `model-hpo-configuration.json`. Los valores de los hiperparámetros se encuentran dentro del rango de hiperparámetros válido si el hiperparámetro se puede ajustar, o toman el valor de hiperparámetro predeterminado si no se puede ajustar.

El script de entrenamiento se ejecuta en una instancia de entrenamiento de SageMaker con una sintaxis como la siguiente:

```
python3 (script entry point) --(1st parameter) (1st value) --(2nd parameter) (2nd
 value) (...)
```

Para todas las tareas, Neptune ML AutoTrainer envía varios parámetros necesarios al script de entrenamiento, además de los hiperparámetros que especifique, y el script debe ser capaz de gestionar estos parámetros adicionales para que funcione correctamente.

Estos parámetros adicionales obligatorios varían un poco según la tarea:

Para la clasificación o la regresión de nodos

- **`task`**: el tipo de tarea utilizado internamente por Neptune ML. Para la clasificación de nodos, es `node_class`, y para la regresión de nodos, es `node_regression`.
- **`model`**: el nombre del modelo utilizado internamente por Neptune ML, que es `custom` en este caso.
- **`name`**: el nombre de la tarea utilizada internamente por Neptune ML, que es `node_class-custom` para la clasificación de nodos, y `node_regression-custom` para la regresión de nodos.
- **`target_ntype`**: el nombre del tipo de nodo para la clasificación o la regresión.
- **`property`**: el nombre de la propiedad de nodo para la clasificación o la regresión.

Para la predicción de enlaces

- **`task`**: el tipo de tarea utilizado internamente por Neptune ML. Para la predicción de enlaces, es `link_predict`.
- **`model`**: el nombre del modelo utilizado internamente por Neptune ML, que es `custom` en este caso.
- **`name`**: el nombre de la tarea utilizada internamente por Neptune ML, que es `link_predict-custom` en este caso.

Para la clasificación o la regresión de bordes

- **`task`**: el tipo de tarea utilizado internamente por Neptune ML. Para la clasificación de bordes, es `edge_class`, y para la regresión de bordes, es `edge_regression`.
- **`model`**: el nombre del modelo utilizado internamente por Neptune ML, que es `custom` en este caso.
- **`name`**: el nombre de la tarea utilizada internamente por Neptune ML, que es `edge_class-custom` para la clasificación de bordes, y `edge_regression-custom` para la regresión de bordes.
- **`target_etype`**: el nombre del tipo de borde para la clasificación o la regresión.

- **`property`**: el nombre de la propiedad de borde para la clasificación o la regresión.

El script debe guardar los parámetros del modelo, así como cualquier otro artefacto que sea necesario al final del entrenamiento.

Puede utilizar las funciones de utilidad del kit de herramientas de Neptune ML para determinar la ubicación de los datos de gráficos procesados, la ubicación en la que deben guardarse los parámetros del modelo y qué dispositivos de GPU están disponibles en la instancia de entrenamiento. Consulte el ejemplo de script de entrenamiento train.py para ver ejemplos de cómo utilizar estas funciones de utilidad.

Desarrollo de scripts de transformación de modelos personalizados en Neptune ML

Se necesita un script de transformación para sacar partido del flujo de trabajo incremental de Neptune ML para la inferencia de modelos en gráficos en constante evolución sin volver a entrenar el modelo. Aunque el script de entrenamiento genere todos los artefactos necesarios para la implementación del modelo, tendrá que proporcionar un script de transformación si quiere generar modelos actualizados sin volver a entrenar el modelo.

> **Note**
>
> Por el momento, los modelos personalizados no admiten la inferencia inductiva en tiempo real.

El script de transformación del modelo personalizado debe ser un script de Python ejecutable, como el script de ejemplo transform.py del kit de herramientas de Neptune ML. Dado que este script se invoca durante el entrenamiento de modelos sin argumentos de línea de comandos, todos los argumentos de línea de comandos que el script acepte deben tener valores predeterminados.

El script se ejecuta en una instancia de entrenamiento de SageMaker con una sintaxis como la siguiente:

```
python3 (your transform script entry point)
```

El script de transformación necesitará varios datos, como, por ejemplo:

- La ubicación de los datos de gráficos procesados.

- La ubicación en la que se guardan los parámetros del modelo y en la que se deben guardar los nuevos artefactos del modelo.
- Los dispositivos disponibles en la instancia.
- Los hiperparámetros que generaron el mejor modelo.

Estas entradas se obtienen mediante las funciones de utilidad de Neptune ML a las que puede llamar el script. Consulte el ejemplo de script transform.py del kit de herramientas para ver ejemplos de cómo hacerlo.

El script debe guardar las incrustaciones de nodos, los mapeados de identificador de nodo y cualquier otro artefacto necesario para la implementación del modelo de cada tarea. Consulte la documentación de artefactos de modelos para obtener más información sobre los artefactos de modelos necesarios para las diferentes tareas de Neptune ML.

Archivo **`model-hpo-configuration.json`** personalizado de Neptune ML

El archivo `model-hpo-configuration.json` define los hiperparámetros para el modelo personalizado. Tiene el mismo formato que el archivo `model-hpo-configuration.json` utilizado con los modelos integrados de Neptune ML y tiene prioridad con respecto a la versión que Neptune ML genera automáticamente y se carga en la ubicación de los datos procesados.

Al añadir un hiperparámetro nuevo al modelo, también debe añadir una entrada para el hiperparámetro en este archivo para que el hiperparámetro se transfiera al script de entrenamiento.

Debe proporcionar un rango para un hiperparámetro si desea que sea ajustable y configurarlo como un parámetro `tier-1`, `tier-2` o `tier-3`. El hiperparámetro se ajustará si el número total de trabajos de entrenamiento configurados permite ajustar los hiperparámetros de su nivel. En el caso de un parámetro que no se pueda ajustar, debe proporcionar un valor predeterminado y añadir el hiperparámetro a la sección `fixed-param` del archivo. Consulte el archivo `model-hpo-configuration.json` de ejemplo del kit de herramientas para ver un ejemplo de cómo hacerlo.

También debe proporcionar la definición de métrica que utilizará el trabajo de optimización de hiperparámetros de SageMaker para evaluar los modelos candidatos entrenados. Para ello, añada un objeto `eval_metric` de JSON al archivo `model-hpo-configuration.json` de la siguiente manera:

```
"eval_metric": {
  "tuning_objective": {
```

```
      "MetricName": "(metric_name)",
      "Type": "Maximize"
  },
  "metric_definitions": [
    {
      "Name": "(metric_name)",
      "Regex": "(metric regular expression)"
    }
  ]
},
```

La matriz de `metric_definitions` del objeto `eval_metric` muestra los objetos de definición de métrica para cada métrica que desee que SageMaker extraiga de la instancia de entrenamiento. Cada objeto de definición de métrica tiene una clave `Name` que le permite proporcionar un nombre a la métrica (por ejemplo, “precisión”, “f1”, etc.). La clave `Regex` le permite proporcionar una cadena de expresión regular que coincide con la forma en que se imprime esa métrica en concreto en los registros de entrenamiento. Consulte la página Ajuste de hiperparámetros de SageMaker para obtener más información sobre cómo definir las métricas.

A continuación, el objeto `tuning_objective` de `eval_metric` le permite especificar las métricas de `metric_definitions` que deben utilizarse como métrica de evaluación que sirva como métrica objetivo para la optimización de hiperparámetros. El valor de `MetricName` debe coincidir con el valor de un `Name` en una de las definiciones de `metric_definitions`. El valor de `Type` debe ser “Maximizar” o “Minimizar”, en función de si la métrica debe interpretarse como mayor es mejor (por ejemplo, “precisión”) o menor es mejor (como “error cuadrático medio”).

Los errores que se produzcan en esta sección del archivo `model-hpo-configuration.json` pueden provocar errores en el trabajo de la API de entrenamiento del modelo de Neptune ML, ya que el trabajo de ajuste de hiperparámetros de SageMaker no podrá seleccionar el mejor modelo.

Pruebas locales de la implementación de un modelo personalizado en Neptune ML

Puede utilizar el entorno de Conda del kit de herramientas de Neptune ML para ejecutar el código localmente con el fin de probar y validar el modelo. Si está desarrollando en una instancia de cuaderno de Neptune, este entorno estará preinstalado en la instancia de cuaderno de Neptune. Si está desarrollando en otra instancia, debe seguir las instrucciones de configuración local del kit de herramientas de Neptune ML.

El entorno de Conda reproduce con precisión el entorno en el que se ejecutará el modelo cuando llame a la API de entrenamiento de modelos. Todos los ejemplos de scripts de entrenamiento y

de transformación permiten utilizar un indicador `--local` de línea de comandos para ejecutar los scripts en un entorno local y facilitar su depuración. Esta práctica se recomienda cuando tenga que desarrollar su propio modelo, ya que le permite probar la implementación del modelo de forma interactiva e iterativa. Durante el entrenamiento de modelos en el entorno de entrenamiento de producción de Neptune ML, se omite este parámetro.

Creación de un punto de conexión de inferencia para consultar

Un punto de conexión de inferencia le permite consultar un modelo específico que se creó durante el proceso de entrenamiento del modelo. El punto de conexión se relaciona con el modelo de mejor rendimiento de un tipo determinado que el proceso de entrenamiento fue capaz de generar. A continuación, el punto de conexión puede aceptar las consultas de Gremlin de Neptune y devolver las predicciones de ese modelo para las entradas de las consultas. Una vez creado el punto de conexión de inferencia, permanecerá activo hasta que lo elimine.

Administración de puntos de conexión de inferencia para Neptune ML

Una vez que haya completado el entrenamiento del modelo con los datos que ha exportado desde Neptune, puede crear un punto de conexión de inferencia mediante un comando `curl` (o `awscurl`) como el siguiente:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

También puede crear un punto de conexión de inferencia a partir de un modelo creado por un trabajo de transformación de modelo completado, prácticamente de la misma manera:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-transform job-id of a completed job)"
      }'
```

Las instrucciones sobre cómo utilizar estos comandos se explican en El comando endpoints, junto con información sobre cómo obtener el estado de un punto de conexión, cómo eliminar un punto de conexión y cómo enumerar todos los puntos de conexión de inferencia.

Consultas de inferencia en Neptune ML

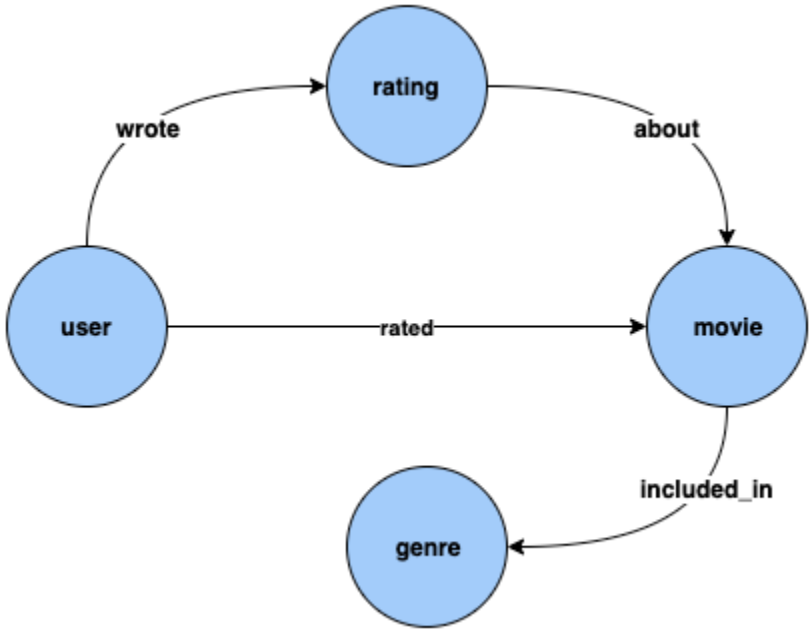
Puede utilizar Gremlin o SPARQL para consular un punto de conexión de inferencia de Neptune ML. Sin embargo, la inferencia inductiva en tiempo real solo se admite actualmente para consultas de Gremlin.

Consultas de inferencia de Gremlin en Neptune ML

Tal y como se describe en Capacidades de Neptune ML, Neptune ML admite modelos de entrenamiento que pueden realizar los siguientes tipos de tareas de inferencia:

- Clasificación de nodos: predice la característica categórica de una propiedad de vértice.
- Regresión de nodos: predice una propiedad numérica de un vértice.
- Clasificación de bordes: predice la característica categórica de una propiedad de borde.
- Regresión de bordes: predice una propiedad numérica de un borde.
- Predicción de enlaces: predice los nodos de destino con un nodo de origen y un borde de salida, o los nodos de origen con un nodo de destino y un borde de entrada.

Podemos ilustrar estas diferentes tareas con ejemplos que utilizan el conjunto de datos de MovieLens 100k proporcionado por GroupLens Research. Este conjunto de datos consta de películas, usuarios y clasificaciones de las películas por parte de los usuarios, a partir de las cales, hemos creado un gráfico de propiedades como el siguiente:



Clasificación de nodos: en el conjunto de datos anterior, `Genre` es un tipo de vértice que está conectado al tipo de vértice `Movie` mediante el borde `included_in`. Sin embargo, si retocamos el conjunto de datos para que `Genre` sea una característica categórica para el tipo de vértice `Movie`, el problema de inferir `Genre` para las nuevas películas que se añadan a nuestro gráfico de conocimientos se puede resolver mediante modelos de clasificación de nodos.

Regresión de nodos: si tenemos en cuenta el tipo de vértice `Rating`, que tiene propiedades como `timestamp` y `score`, el problema de inferir el valor numérico `Score` para una `Rating` se puede resolver mediante modelos de regresión de nodos.

Clasificación de bordes: del mismo modo, en el caso de un borde `Rated`, si tenemos una propiedad `Scale` que puede tener uno de los valores `Love`, `Like`, `Dislike`, `Neutral` o `Hate`, el problema de inferir `Scale` del borde `Rated` para nuevas películas o clasificaciones se puede resolver mediante modelos de clasificación de bordes.

Regresión de bordes: del mismo modo, para el mismo borde `Rated`, si tenemos una propiedad `Score` que incluye un valor numérico para la clasificación, esto se puede inferir de los modelos de regresión de bordes.

Predicción de enlaces: problemas, como encontrar los diez principales usuarios que tienen más probabilidades de puntuar una determinada película o encontrar las diez principales películas que un determinado usuario tiene más probabilidades de valorar, se consideran parte de la predicción de enlaces.

> **Note**
>
> Para los casos de uso de Neptune ML, tenemos un conjunto muy completo de cuadernos diseñados para proporcionarle una comprensión práctica de cada caso de uso. Puede crear estos cuadernos junto con el clúster de Neptune cuando utilice la plantilla de AWS CloudFormation de Neptune ML para crear un clúster de Neptune ML. Estos cuadernos también están disponibles en github.

Temas

Predicados de Neptune ML utilizados en las consultas de inferencia de Gremlin

`Neptune#ml.deterministic`

Este predicado es una opción para las consultas de inferencia inductiva, es decir, para las consultas que incluyen el predicado `Neptune#ml.inductiveInference`.

Cuando se utiliza la inferencia inductiva, el motor de Neptune crea el subgráfico adecuado para evaluar el modelo GNN entrenado, y los requisitos de este subgráfico dependen de los parámetros del modelo final. En concreto, el parámetro `num-layer` determina el número de saltos de recorridos desde los nodos o bordes de destino, y el parámetro `fanouts` especifica el número de vecinos que hay que muestrear en cada salto (consulte Parámetros de HPO).

De forma predeterminada, las consultas de inferencia inductiva se ejecutan en modo no determinista, en el que Neptune crea la vecindad de forma aleatoria. Al realizar predicciones, este muestreo normal de vecinos aleatorios a veces da como resultado predicciones diferentes.

Cuando se incluye `Neptune#ml.deterministic` en una consulta de inferencia inductiva, el motor de Neptune intenta muestrear los vecinos de forma determinista para que las diversas invocaciones de la misma consulta devuelvan siempre los mismos resultados. Sin embargo, no se puede garantizar que los resultados sean completamente deterministas, ya que los cambios en el gráfico subyacente y los artefactos de los sistemas distribuidos aún pueden provocar fluctuaciones.

El predicado `Neptune#ml.deterministic` se incluye en una consulta de la siguiente manera:

```
.with("Neptune#ml.deterministic")
```

Si el predicado `Neptune#ml.deterministic` se incluye en una consulta que no incluye también `Neptune#ml.inductiveInference`, simplemente se ignora.

`Neptune#ml.disableInductiveInferenceMetadataCache`

Este predicado es una opción para las consultas de inferencia inductiva, es decir, para las consultas que incluyen el predicado `Neptune#ml.inductiveInference`.

Para las consultas de inferencia inductiva, Neptune utiliza un archivo de metadatos almacenado en Amazon S3 para decidir el número de saltos y la distribución ramificada al crear el vecindario. Neptune normalmente almacena en caché los metadatos de este modelo para evitar tener que recuperar una y otra vez el archivo de Amazon

S3. El almacenamiento en caché se puede deshabilitar si se incluye el predicado `Neptune#ml.disableInductiveInferenceMetadataCache` en la consulta. Aunque Neptune puede tardar más en recuperar los metadatos directamente de Amazon S3, resulta útil cuando el punto de conexión de SageMaker se ha actualizado tras volver a entrenarse o transformarse y la caché está obsoleta.

El predicado `Neptune#ml.disableInductiveInferenceMetadataCache` se incluye en una consulta de la siguiente manera:

```
.with("Neptune#ml.disableInductiveInferenceMetadataCache")
```

A continuación, se muestra cómo podría ser una consulta en un cuaderno de Jupyter:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ep1")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .with("Neptune#ml.disableInductiveInferenceMetadataCache")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

`Neptune#ml.endpoint`

El predicado `Neptune#ml.endpoint` se utiliza en un paso `with()` para especificar el punto de conexión de inferencia, si es necesario:

```
 .with("Neptune#ml.endpoint", "the model's SageMaker inference endpoint")
```

Puede identificar el punto de conexión tanto por su `id` o como por su URL. Por ejemplo:

```
 .with( "Neptune#ml.endpoint", "node-classification-movie-lens-endpoint" )
```

O bien:

```
 .with( "Neptune#ml.endpoint", "https://runtime.sagemaker.us-east-1.amazonaws.com/
endpoints/node-classification-movie-lens-endpoint/invocations" )
```

> **Note**
>
> Si establece el parámetro `neptune_ml_endpoint` en el grupo de parámetros del clúster de base de datos de Neptune en el `id` o la URL del punto de conexión, no necesita incluir el predicado `Neptune#ml.endpoint` en cada consulta.

`Neptune#ml.iamRoleArn`

`Neptune#ml.iamRoleArn` se utiliza en un paso `with()` para especificar el ARN del rol de IAM de ejecución de SageMaker, si es necesario:

```
.with("Neptune#ml.iamRoleArn", "the ARN for the SageMaker execution IAM role")
```

Para obtener información sobre cómo crear el rol de IAM de ejecución de SageMaker, consulte Para crear un rol de `NeptuneSageMakerIAMRole` personalizado.

> **Note**
>
> Si establece el parámetro `neptune_ml_iam_role` en el grupo de parámetros del clúster de base de datos de Neptune en el ARN del rol de IAM de ejecución de SageMaker, no necesita incluir el predicado `Neptune#ml.iamRoleArn` en cada consulta.

Neptune#ml.inductiveInference

La inferencia transductiva está habilitada de forma predeterminada en Gremlin. Para realizar una consulta de inferencia inductiva en tiempo real, incluya el predicado `Neptune#ml.inductiveInference` de la siguiente manera:

```
.with("Neptune#ml.inductiveInference")
```

Si el gráfico es dinámico, la inferencia inductiva suele ser la mejor opción, pero si el gráfico es estático, la inferencia transductiva es más rápida y eficaz.

`Neptune#ml.limit`

El predicado `Neptune#ml.limit` es opcional y limita el número de resultados devueltos por la entidad:

```
.with( "Neptune#ml.limit", 2 )
```

De forma predeterminada, el límite es 1 y el número máximo que se puede establecer es 100.

Neptune#ml.threshold

El predicado `Neptune#ml.threshold` establece de forma opcional un umbral límite para las puntuaciones de los resultados:

```
.with( "Neptune#ml.threshold", 0.5D )
```

Esto le permite descartar todos los resultados con puntuaciones por debajo del umbral especificado.

Neptune#ml.classification

El predicado `Neptune#ml.classification` se adjunta al paso `properties()` para establecer que las propiedades deben obtenerse del punto de conexión de SageMaker del modelo de clasificación de nodos:

```
.properties( "property key of the node classification
model" ).with( "Neptune#ml.classification" )
```

Neptune#ml.regression

El predicado `Neptune#ml.regression` se adjunta al paso `properties()` para establecer que las propiedades deben obtenerse del punto de conexión de SageMaker del modelo de regresión de nodos:

```
.properties( "property key of the node regression
model" ).with( "Neptune#ml.regression" )
```

Neptune#ml.prediction

El predicado `Neptune#ml.prediction` se adjunta a los pasos `in()` y `out()` pasos para establecer que se trata de una consulta de predicción de enlaces:

```
.in("edge label of the link prediction
model").with("Neptune#ml.prediction").hasLabel("target node label")
```

`Neptune#ml.score`

El predicado `Neptune#ml.score` se utiliza en las consultas de clasificación de nodos o bordes de Gremlin para obtener una puntuación de confianza en el machine learning. El predicado `Neptune#ml.score` debe pasarse junto con el predicado de la consulta en el paso `properties()` para obtener una puntuación de confianza en el ML para las consultas de clasificación de nodos o bordes.

Puede encontrar un ejemplo de clasificación de nodos con otros ejemplos de clasificación de nodos y un ejemplo de clasificación de bordes en la sección Clasificación de bordes.

Consultas de clasificación de nodos de Gremlin en Neptune ML

Para la clasificación de nodos de Gremlin en Neptune ML:

- El modelo se entrena en una propiedad de los vértices. El conjunto de valores únicos de esta propiedad se denomina conjunto de clases de nodos o, simplemente, clases.
- La clase de nodo o el valor de la propiedad categórica de la propiedad de un vértice se pueden inferir del modelo de clasificación de nodos. Esto es útil cuando esta propiedad aún no está asociada al vértice.
- Para obtener una o varias clases de un modelo de clasificación de nodos, debe usar el paso `with()` con el predicado `Neptune#ml.classification` para configurar el paso `properties()`. El formato de salida es similar al que cabría esperar si se trataran de propiedades de vértices.

> **Note**
>
> La clasificación de nodos solo funciona con valores de propiedades de cadena. Esto significa que no se admiten valores de propiedades numéricas como `0` o `1`, aunque sí se admiten los equivalentes de cadena `"0"` y `"1"`. Del mismo modo, los valores de la propiedad Boolean `true` y `false` no funcionan, pero sí lo hacen `"true"` y `"false"`.

Este es un ejemplo de una consulta de clasificación de nodos:

```
g.with( "Neptune#ml.endpoint","node-classification-movie-lens-endpoint" )
 .with( "Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role" )
```

```
  .with( "Neptune#ml.limit", 2 )
  .with( "Neptune#ml.threshold", 0.5D )
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre").with("Neptune#ml.classification")
```

La salida de esta consulta sería similar a la siguiente:

```
==>vp[genre->Action]
==>vp[genre->Crime]
==>vp[genre->Comedy]
```

En la consulta anterior, los pasos `V()` y `properties()` se utilizan de la siguiente manera:

El paso `V()` incluye el conjunto de vértices para el que desea obtener las clases del modelo de clasificación de nodos:

```
  .V( "movie_1", "movie_2", "movie_3" )
```

El paso `properties()` incluye la clave en la que se entrenó el modelo, así como el valor `.with("Neptune#ml.classification")` para indicar que se trata de una consulta de inferencia de ML de clasificación de nodos.

Por el momento, no se admiten varias claves de propiedad en un paso `properties().with("Neptune#ml.classification")`. Por ejemplo, la siguiente consulta da lugar a una excepción:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V( "movie_1", "movie_2", "movie_3" )
  .properties("genre", "other_label").with("Neptune#ml.classification")
```

Para obtener información sobre el mensaje de error específico, consulte la lista de excepciones de Neptune ML.

Se puede usar un paso `properties().with("Neptune#ml.classification")` en combinación con cualquiera de los pasos siguientes:

- `value()`
- `value().is()`

- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Otras consultas de clasificación de nodos

Si tanto el punto de conexión de inferencia como el correspondiente rol de IAM se han guardado en el grupo de parámetros del clúster de base de datos, una consulta de clasificación de nodos puede ser tan sencilla como esta:

```
g.V("movie_1", "movie_2",
 "movie_3").properties("genre").with("Neptune#ml.classification")
```

Puede combinar las propiedades y las clases de los vértices en una consulta mediante el paso `union()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .union(
   properties("genre").with("Neptune#ml.classification"),
   properties("genre")
 )
```

También puede realizar una consulta ilimitada como la siguiente:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V()
 .properties("genre").with("Neptune#ml.classification")
```

Puede recuperar las clases de nodos junto con los vértices mediante el paso `select()` junto con el paso `as()`:

```
g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" ).as("vertex")
 .properties("genre").with("Neptune#ml.classification").as("properties")
 .select("vertex","properties")
```

También puede filtrar por clases de nodos, tal y como se muestra en estos ejemplos:

```
g.with("Neptune#ml.endpoint", "node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
 .has(value, "Horror")

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
 .has(value, P.eq("Action"))

g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre").with("Neptune#ml.classification")
 .has(value, P.within("Action", "Horror"))
```

Puede obtener una puntuación de confianza en la clasificación de nodos mediante el predicado `Neptune#ml.score`:

```
 g.with("Neptune#ml.endpoint","node-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V( "movie_1", "movie_2", "movie_3" )
 .properties("genre", "Neptune#ml.score").with("Neptune#ml.classification")
```

La respuesta sería similar a:

```
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.01234567]
==>vp[genre->Crime]
==>vp[Neptune#ml.score->0.543210]
==>vp[genre->Comedy]
```

```
==>vp[Neptune#ml.score->0.10101]
```

Uso de la inferencia inductiva en una consulta de clasificación de nodos

Supongamos que tuviera que añadir un nuevo nodo a un gráfico existente, en un cuaderno de Jupyter, de la siguiente manera:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV').to('oldV1')
 .addE('eLabel2').from('oldV2').to('newV')
```

A continuación, podría usar una consulta de inferencia inductiva para obtener un género y una puntuación de confianza que reflejaran el nuevo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("genre", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Sin embargo, si ejecutara la consulta varias veces, es posible que obtuviera resultados algo diferentes:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.21365921]
```

Podría hacer que la misma consulta fuera determinista:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
```

```
  .V('101').properties("genre", "Neptune#ml.score")
  .with("Neptune#ml.classification")
  .with("Neptune#ml.inductiveInference")
  .with("Neptune#ml.deterministic")
```

En ese caso, los resultados serían aproximadamente los mismos cada vez:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
```

Consultas de regresión de nodos de Gremlin en Neptune ML

La regresión de nodos es similar a la clasificación de nodos, excepto el valor inferido del modelo de regresión para cada nodo que es numérico. Puede utilizar las mismas consultas de Gremlin para la regresión de nodos que para la clasificación de nodos, excepto por las siguientes diferencias:

- De nuevo, en Neptune ML, los nodos hacen referencia a vértices.
- El paso `properties()` adopta el formato `properties().with("Neptune#ml.regression")` en lugar de `properties().with("Neptune#ml.classification")`.
- Los predicados `"Neptune#ml.limit` y `"Neptune#ml.threshold"` no son aplicables.
- Al filtrar el valor, debe especificar un valor numérico.

Este es un ejemplo de una consulta de clasificación de vértices:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn", "arn:aws:iam::0123456789:role/sagemaker-role")
  .V("movie_1","movie_2","movie_3")
  .properties("revenue").with("Neptune#ml.regression")
```

Puede filtrar el valor inferido mediante un modelo de regresión, tal y como se muestra en los siguientes ejemplos:

```
g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
```

```
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("movie_1","movie_2","movie_3")
  .properties("revenue").with("Neptune#ml.regression")
  .value().is(P.gte(1600000))

g.with("Neptune#ml.endpoint","node-regression-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("movie_1","movie_2","movie_3")
  .properties("revenue").with("Neptune#ml.regression")
  .hasValue(P.lte(1600000D))
```

Uso de la inferencia inductiva en una consulta de regresión de nodos

Supongamos que tuviera que añadir un nuevo nodo a un gráfico existente, en un cuaderno de Jupyter, de la siguiente manera:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV')
  .V('1').as('oldV1')
  .V('2').as('oldV2')
  .addE('eLabel1').from('newV').to('oldV1')
  .addE('eLabel2').from('oldV2').to('newV')
```

A continuación, podría usar una consulta de inferencia inductiva para obtener una calificación que tuviera en cuenta el nuevo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nr-ep")
  .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
  .V('101').properties("rating")
  .with("Neptune#ml.regression")
  .with("Neptune#ml.inductiveInference")
```

Dado que la consulta no es determinista, es posible que arrojara resultados algo diferentes si la ejecutara varias veces, en función de la vecindad:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->8.9]
```

Si necesitara resultados más coherentes, podría hacer que la consulta fuera determinista:

```
%%gremlin
g.with("Neptune#ml.endpoint", "nc-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').properties("rating")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Ahora los resultados serán aproximadamente los mismos cada vez:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->9.1]
```

Consultas de clasificación de bordes de Gremlin en Neptune ML

Para la clasificación de bordes de Gremlin en Neptune ML:

- El modelo se entrena en una propiedad de los bordes. El conjunto de valores únicos de esta propiedad se denomina conjunto de clases.
- El valor de clase o propiedad categórica de un borde se puede inferir del modelo de clasificación de bordes, lo que resulta útil cuando esta propiedad aún no está asociada al borde.
- Para obtener una o varias clases de un modelo de clasificación de bordes, debe usar el paso `with()` con el predicado `"Neptune#ml.classification"` para configurar el paso `properties()`. El formato de salida es similar al que cabría esperar si se trataran de propiedades de bordes.

> **Note**
>
> La clasificación de bordes solo funciona con valores de propiedades de cadena. Esto significa que no se admiten valores de propiedades numéricas como `0` o `1`, aunque sí se admiten los equivalentes de cadena `"0"` y `"1"`. Del mismo modo, los valores de la propiedad Boolean `true` y `false` no funcionan, pero sí lo hacen `"true"` y `"false"`.

A continuación, se muestra un ejemplo de una consulta de clasificación de bordes que solicita una puntuación de confianza mediante el predicado `Neptune#ml.score`:

```
g.with("Neptune#ml.endpoint","edge-classification-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "Neptune#ml.score").with("Neptune#ml.classification")
```

La respuesta sería similar a:

```
==>p[knows_by->"Family"]
==>p[Neptune#ml.score->0.01234567]
==>p[knows_by->"Friends"]
==>p[Neptune#ml.score->0.543210]
==>p[knows_by->"Colleagues"]
==>p[Neptune#ml.score->0.10101]
```

Sintaxis de una consulta de clasificación de bordes de Gremlin

En el caso de un gráfico sencillo en el que `User` es el nodo principal y final y `Relationship` es el borde que los conecta: un ejemplo de consulta de clasificación de bordes es:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by").with("Neptune#ml.classification")
```

La salida de esta consulta sería similar a la siguiente:

```
==>p[knows_by->"Family"]
==>p[knows_by->"Friends"]
==>p[knows_by->"Colleagues"]
```

En la consulta anterior, los pasos `E()` y `properties()` se utilizan de la siguiente manera:

- El paso `E()` incluye el conjunto de bordes para el que desea obtener las clases del modelo de clasificación de bordes:

  ```
  .E("relationship_1","relationship_2","relationship_3")
  ```

- El paso `properties()` incluye la clave en la que se entrenó el modelo, así como el valor `.with("Neptune#ml.classification")` para indicar que se trata de una consulta de inferencia de ML de clasificación de bordes.

Por el momento, no se admiten varias claves de propiedad en un paso `properties().with("Neptune#ml.classification")`. Por ejemplo, la siguiente consulta da lugar a una excepción:

```
g.with("Neptune#ml.endpoint","edge-classification-social-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("relationship_1","relationship_2","relationship_3")
 .properties("knows_by", "other_label").with("Neptune#ml.classification")
```

Para ver mensajes de error específicos, consulte Lista de excepciones para las consultas de inferencia de Gremlin en Neptune ML.

Se puede usar un paso `properties().with("Neptune#ml.classification")` en combinación con cualquiera de los pasos siguientes:

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Uso de la inferencia inductiva en una consulta de clasificación de bordes

Supongamos que tuviera que añadir un nuevo borde a un gráfico existente, en un cuaderno de Jupyter, de la siguiente manera:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
```

```
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

A continuación, podría usar una consulta de inferencia inductiva para obtener una escala que tuviera en cuenta el nuevo borde:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
```

Dado que la consulta no es determinista, los resultados variarían algo si se ejecutara varias veces, en función de la vecindad aleatoria:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.21365921]
```

Si necesitara resultados más coherentes, podría hacer que la consulta fuera determinista:

```
%%gremlin
g.with("Neptune#ml.endpoint", "ec-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("scale", "Neptune#ml.score")
 .with("Neptune#ml.classification")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Ahora los resultados serán más o menos los mismos cada vez que ejecute la consulta:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
```

```
==>vp[Neptune#ml.score->0.12345678]
```

Consultas de regresión de bordes de Gremlin en Neptune ML

La regresión de bordes es similar a la clasificación de bordes, excepto el valor inferido del modelo de ML que es numérico. En el caso de la regresión de bordes, Neptune ML admite las mismas consultas que para la clasificación.

Los puntos clave a tener en cuenta son:

- Debe usar el predicado `"Neptune#ml.regression"` de ML para configurar el paso `properties()` para este caso de uso.
- Los predicados `"Neptune#ml.limit"` y `"Neptune#ml.threshold"` no son aplicables en este caso de uso.
- Para filtrar el valor, debe especificarlo como numérico.

Sintaxis de una consulta de regresión de bordes de Gremlin

En el caso de un gráfico sencillo en el que `User` es el nodo principal, `Movie` es el nodo final y `Rated` es el borde que los conecta: a continuación se muestra un ejemplo de consulta de regresión de bordes que encuentra el valor de clasificación numérica, denominado puntuación, para el borde `Rated`:

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
```

También puede filtrar un valor inferido del modelo de regresión de ML. En el caso de los bordes `Rated` existentes (de `User` a `Movie`) identificadas por `"rating_1"`, `"rating_2"` y `"rating_3"`, donde la propiedad de borde `Score` esté presente para estas clasificaciones, puede utilizar una consulta como la siguiente para inferir `Score` para los bordes en los que es mayor o igual a 9:

```
g.with("Neptune#ml.endpoint","edge-regression-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .E("rating_1","rating_2","rating_3")
 .properties("score").with("Neptune#ml.regression")
 .value().is(P.gte(9))
```

Uso de la inferencia inductiva en una consulta de regresión de bordes

Supongamos que tuviera que añadir un nuevo borde a un gráfico existente, en un cuaderno de Jupyter, de la siguiente manera:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

A continuación, podría usar una consulta de inferencia inductiva para obtener una puntuación que tuviera en cuenta el nuevo borde:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
```

Dado que la consulta no es determinista, los resultados variarían algo si se ejecutara varias veces, en función de la vecindad aleatoria:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->91]
```

Si necesitara resultados más coherentes, podría hacer que la consulta fuera determinista:

```
%%gremlin
g.with("Neptune#ml.endpoint", "er-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .E('e101').properties("score")
 .with("Neptune#ml.regression")
 .with("Neptune#ml.inductiveInference")
 .with("Neptune#ml.deterministic")
```

Ahora los resultados serán más o menos los mismos cada vez que ejecute la consulta:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->96]
```

Consultas de predicción de enlaces de Gremlin mediante modelos de predicción de enlaces en Neptune ML

Los modelos de predicción de enlaces pueden resolver problemas como los siguientes:

- Predicción del nodo principal: con un vértice y un tipo de borde, ¿desde qué vértices es probable que se enlace este vértice?
- Predicción del nodo final: con un vértice y un tipo de borde, ¿desde qué vértices es probable que se enlace este vértice?

> **Note**
>
> Aún no se admite la predicción de bordes en Neptune ML.

Para los ejemplos siguientes, supongamos que tenemos un gráfico sencillo con los vértices `User` y `Movie` que estén vinculados por el borde `Rated`.

Este es un ejemplo de consulta de predicción del nodo principal, que se utiliza para predecir cuáles son los cinco principales usuarios que tienen más probabilidades de valorar las películas `"movie_1"`, `"movie_2"` y `"movie_3"`:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .with("Neptune#ml.limit", 5)
 .V("movie_1", "movie_2", "movie_3")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Esta es una similar para la predicción del nodo final, que se utiliza para predecir cuáles son las cinco principales películas que el usuario `"user_1"` tiene más probabilidades de valorar:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
```

```
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("user_1")
  .out("rated").with("Neptune#ml.prediction").hasLabel("movie")
```

Se necesitan tanto la etiqueta de borde como la de vértice previsto. Si se omite alguna de ellas, se produce una excepción. Por ejemplo, la siguiente consulta sin una etiqueta de vértice previsto produce una excepción:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("user_1")
  .out("rated").with("Neptune#ml.prediction")
```

Del mismo modo, la siguiente consulta sin una etiqueta de borde genera una excepción:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("user_1")
  .out().with("Neptune#ml.prediction").hasLabel("movie")
```

Para obtener información sobre los mensajes de error específicos que generan estas excepciones, consulte la lista de excepciones de Neptune ML.

Otras consultas de predicción de enlaces

Puede usar el paso `select()` con el paso `as(` para generar los vértices previstos junto con los vértices de entrada:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("movie_1").as("source")
  .in("rated").with("Neptune#ml.prediction").hasLabel("user").as("target")
  .select("source","target")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
  .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
  .V("user_1").as("source")
  .out("rated").with("Neptune#ml.prediction").hasLabel("movie").as("target")
  .select("source","target")
```

Puede realizar consultas ilimitadas, como, por ejemplo:

```
g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("user_1")
 .out("rated").with("Neptune#ml.prediction").hasLabel("movie")

g.with("Neptune#ml.endpoint","node-prediction-movie-lens-endpoint")
 .with("Neptune#ml.iamRoleArn","arn:aws:iam::0123456789:role/sagemaker-role")
 .V("movie_1")
 .in("rated").with("Neptune#ml.prediction").hasLabel("user")
```

Uso de la inferencia inductiva en una consulta de predicción de enlaces

Supongamos que tuviera que añadir un nuevo nodo a un gráfico existente, en un cuaderno de Jupyter, de la siguiente manera:

```
%%gremlin
g.addV('label1').property(id,'101').as('newV1')
 .addV('label2').property(id,'102').as('newV2')
 .V('1').as('oldV1')
 .V('2').as('oldV2')
 .addE('eLabel1').from('newV1').to('oldV1')
 .addE('eLabel2').from('oldV2').to('newV2')
```

A continuación, podría usar una consulta de inferencia inductiva para predecir el nodo principal, teniendo en cuenta el nuevo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('101').out("eLabel1")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label2")
```

Resultado:

```
==>V[2]
```

Asimismo, podría usar una consulta de inferencia inductiva para predecir el nodo final, teniendo en cuenta el nuevo nodo:

```
%%gremlin
g.with("Neptune#ml.endpoint", "lp-ep")
 .with("Neptune#ml.iamRoleArn", "arn:aws:iam::123456789012:role/NeptuneMLRole")
 .V('102').in("eLabel2")
 .with("Neptune#ml.prediction")
 .with("Neptune#ml.inductiveInference")
 .hasLabel("label1")
```

Resultado:

```
==>V[1]
```

Lista de excepciones para las consultas de inferencia de Gremlin en Neptune ML

- **`BadRequestException`**: no se pueden cargar las credenciales del rol proporcionado.

  Mensaje: `Unable to load credentials for role: the specified IAM Role ARN.`
- **`BadRequestException`**: el rol de IAM especificado no está autorizado a invocar el punto de conexión de SageMaker.

  Mensaje: `User: the specified IAM Role ARN is not authorized to perform: sagemaker:InvokeEndpoint on resource: the specified endpoint.`
- **`BadRequestException`**: el punto de conexión especificado no existe.

  Mensaje: `Endpoint the specified endpoint not found.`
- **`InternalFailureException`**: no se pueden obtener los metadatos de inferencia inductiva en tiempo real de Neptune ML desde Amazon S3.

  Mensaje: `Unable to fetch Neptune ML - Real-Time Inductive Inference metadata from S3. Check the permissions of the S3 bucket or if the Neptune instance can connect to S3.`
- **`InternalFailureException`**: Neptune ML no puede encontrar el archivo de metadatos de inferencia inductiva en tiempo real en Amazon S3.

  Mensaje: `Neptune ML cannot find the metadata file for Real-Time Inductive Inference in S3.`
- **`InvalidParameterException`**: el punto de conexión especificado no es válido desde el punto de vista sintáctico.

Mensaje: `Invalid endpoint provided for external service query.`

- **`InvalidParameterException`**: el ARN del rol de IAM de ejecución de SageMaker especificado no es válido desde el punto de vista sintáctico.

  Mensaje: `Invalid IAM role ARN provided for external service query.`

- **`InvalidParameterException`**: se especifican varias claves de propiedad en el paso `properties()` de una consulta.

  Mensaje: `ML inference queries are currently supported for one property key.`

- **`InvalidParameterException`**: se especifican varias etiquetas de borde en una consulta.

  Mensaje: `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`**: se especifican varias restricciones de etiquetas de vértices en una consulta.

  Mensaje: `ML inference are currently supported only with one vertex label constraint.`

- **`InvalidParameterException`**: los predicados `Neptune#ml.classification` y `Neptune#ml.regression` están presentes en la misma consulta.

  Mensaje: `Both regression and classification ML predicates cannot be specified in the query.`

- **`InvalidParameterException`**: se ha especificado más de una etiqueta de borde en los pasos `in()` o `out()` de una consulta de predicción de enlaces.

  Mensaje: `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`**: se ha especificado más de una clave de propiedad con Neptune#ml.score.

  Mensaje: `Neptune ML inference queries are currently supported for one property key and one Neptune#ml.score property key.`

- **`MissingParameterException`**: no se ha especificado el punto de conexión en la consulta ni como parámetro de clúster de base de datos.

  Mensaje: `No endpoint provided for external service query.`

- MissingParameterException: no se ha especificado el rol de IAM de ejecución de SageMaker en la consulta ni como parámetro de clúster de base de datos.

  Mensaje: `No IAM role ARN provided for external service query.`

- **`MissingParameterException`**: falta la clave de propiedad en el paso `properties()` de una consulta.

  Mensaje: `Property key needs to be specified using properties() step for ML inference queries.`

- **`MissingParameterException`**: no se ha especificado ninguna etiqueta de borde en los pasos `in()` o `out()` de una consulta de predicción de enlaces.

  Mensaje: `Edge label needs to be specified while using in() or out() step for ML inference queries.`

- **`MissingParameterException`**: no se ha especificado ninguna clave de propiedad con Neptune#ml.score.

  Mensaje: `Property key needs to be specified along with Neptune#ml.score property key while using the properties() step for Neptune ML inference queries.`

- **`UnsupportedOperationException`**: el paso `both()` se utiliza en una consulta de predicción de enlaces.

  Mensaje: `ML inference queries are currently not supported with both() step.`

- **`UnsupportedOperationException`**: no se ha especificado ninguna etiqueta de vértice prevista en el paso `has()` con los pasos `in()` o `out()` en una consulta de predicción de enlaces.

  Mensaje: `Predicted vertex label needs to be specified using has() step for ML inference queries.`

- **`UnsupportedOperationException`**: las consultas de inferencia inductiva de ML de Gremlin no son compatibles actualmente con pasos no optimizados.

  Mensaje: `Neptune ML - Real-Time Inductive Inference queries are currently not supported with Gremlin steps which are not optimized for Neptune. Check the Neptune User Guide for a list of Neptune-optimized steps.`

- **`UnsupportedOperationException`**: las consultas de inferencia de Neptune ML no son compatibles actualmente en un paso `repeat`.

Mensaje: `Neptune ML inference queries are currently not supported inside a repeat step.`

- **`UnsupportedOperationException`**: no se admite más de una consulta de inferencia de Neptune ML por consulta de Gremlin.

  Mensaje: `Neptune ML inference queries are currently supported only with one ML inference query per gremlin query.`

Consultas de inferencia de SPARQL en Neptune ML

Neptune ML mapea el gráfico de RDF en un gráfico de propiedades para modelar la tarea de ML. Por el momento, admite los siguientes casos de uso:

- Clasificación de objetos: predice la característica categórica de un objeto.
- Regresión de objetos: predice una propiedad numérica de un objeto.
- Predicción de objetos: predice un objeto a partir de un sujeto y una relación.
- Predicción de sujetos: predice un sujeto a partir de un objeto y una relación.

> **Note**
>
> Neptune ML no admite casos de uso de clasificación y regresión de sujetos con SPARQL.

Predicados de Neptune ML utilizados en las consultas de inferencia de SPARQL

Los siguientes predicados se utilizan con la inferencia de SPARQL:

Predicado **`neptune-ml:timeout`**

Especifica el tiempo de espera de la conexión con el servidor remoto. No debe confundirse con el tiempo de espera de la solicitud de consulta, que es el tiempo máximo que el servidor puede tardar en satisfacer una solicitud.

Tenga en cuenta que si el tiempo de espera de la consulta se produce antes de que se agote el tiempo de espera del servicio especificado por el predicado `neptune-ml:timeout`, la conexión del servicio también se cancelará.

Predicado **`neptune-ml:outputClass`**

El predicado `neptune-ml:outputClass` solo se usa para definir la clase del objeto previsto para la predicción del objeto o del sujeto previsto para la predicción del sujeto.

Predicado **`neptune-ml:outputScore`**

El predicado `neptune-ml:outputScore` es un número positivo que representa la probabilidad de que el resultado de un modelo de machine learning sea correcto.

Predicado **`neptune-ml:modelType`**

El predicado `neptune-ml:modelType` especifica el tipo de modelo de machine learning que se está entrenando:

- `OBJECT_CLASSIFICATION`
- `OBJECT_REGRESSION`
- `OBJECT_PREDICTION`
- `SUBJECT_PREDICTION`

Predicado **`neptune-ml:input`**

El predicado `neptune-ml:input` hace referencia a la lista de URI que se utilizan como entradas para Neptune ML.

Predicado **`neptune-ml:output`**

El predicado `neptune-ml:output` hace referencia a la lista de conjuntos de enlaces en los que Neptune ML devuelve resultados.

Predicado **`neptune-ml:predicate`**

El predicado `neptune-ml:predicate` se usa de manera diferente en función de la tarea que se esté realizando:

- Para la predicción de un objeto o un sujeto: define el tipo de predicado (el tipo de borde o relación).
- Para la clasificación y regresión de objetos: define el valor literal (propiedad) que queremos predecir.

Predicado **`neptune-ml:batchSize`**

`neptune-ml:batchSize` especifica el tamaño de entrada para la llamada de servicio remoto.

Ejemplos de clasificación de objetos de SPARQL

Para la clasificación de objetos de SPARQL en Neptune ML, el modelo se entrena en uno de los valores de los predicados. Esto es útil cuando ese predicado aún no está presente en un determinado sujeto.

Solo se pueden inferir valores de predicados categóricos con el modelo de clasificación de objetos.

La siguiente consulta busca predecir el valor del predicado <http://www.example.org/team> para todas las entradas del tipo `foaf:Person`:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Esta consulta se puede personalizar de la siguiente manera:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_CLASSIFICATION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/team> ;
                      neptune-ml:output ?output .
  }
}
```

Ejemplos de regresión de objetos de SPARQL

La regresión de objetos es similar a la clasificación de nodos, excepto el valor de predicado numérico inferido del modelo de regresión de cada nodo. Puede utilizar las mismas consultas de SPARQL para la regresión de objetos que para la clasificación de objetos, con la excepción de que los predicados `the Neptune#ml.limit` y `Neptune#ml.threshold` no son aplicables.

La siguiente consulta busca predecir el valor del predicado <http://www.example.org/accountbalance> para todas las entradas del tipo `foaf:Person`:

```
SELECT * WHERE { ?input a foaf:Person .
```

```
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/accountbalance> ;
                      neptune-ml:output ?output .
  }
}
```

Esta consulta se puede personalizar de la siguiente manera:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-account-balance-endpoint' ;
                      neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                      neptune-ml:batchSize "40"^^xsd:integer ;
                      neptune-ml:timeout "1000"^^xsd:integer ;

                      neptune-ml:modelType 'OBJECT_REGRESSION' ;
                      neptune-ml:input ?input ;
                      neptune-ml:predicate <http://www.example.org/accountbalance> ;
                      neptune-ml:output ?output .
  }
}
```

Ejemplo de predicción de objetos de SPARQL

La predicción de objetos predice el valor del objeto para un sujeto y un predicado determinados.

La siguiente consulta de predicción de objetos busca predecir qué película le gustaría a la entrada del tipo `foaf:Person`:

```
?x a foaf:Person .
?x   <http://www.example.org/likes> ?m .
?m a <http://www.example.org/movie> .

## Query
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:modelType 'OBJECT_PREDICTION' ;
                      neptune-ml:input ?input ;
```

```
                    neptune-ml:predicate <http://www.example.org/likes> ;
                    neptune-ml:output ?output ;
                    neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

La consulta en sí misma se puede personalizar de la siguiente manera:

```
SELECT * WHERE { ?input a foaf:Person .
  SERVICE neptune-ml:inference {
    neptune-ml:config neptune-ml:endpoint 'node-prediction-user-movie-prediction-
endpoint' ;
                    neptune-ml:iamRoleArn 'arn:aws:iam::0123456789:role/sagemaker-
role' ;

                    neptune-ml:limit "5"^^xsd:integer ;
                    neptune-ml:batchSize "40"^^xsd:integer ;
                    neptune-ml:threshold "0.1"^^xsd:double ;
                    neptune-ml:timeout "1000"^^xsd:integer ;
                    neptune-ml:outputScore ?score ;

                    neptune-ml:modelType 'OBJECT_PREDICTION' ;
                    neptune-ml:input ?input ;
                    neptune-ml:predicate <http://www.example.org/likes> ;
                    neptune-ml:output ?output ;
                    neptune-ml:outputClass <http://www.example.org/movie> .
  }
}
```

Ejemplo de predicción de sujetos de SPARQL

La predicción de sujetos predice el valor del sujetos para un objeto y un predicado determinados.

Por ejemplo, la siguiente consulta predice quién (del tipo `foaf:User`) verá una película determinada:

```
SELECT * WHERE { ?input (a foaf:Movie) .
    SERVICE neptune-ml:inference {
        neptune-ml:config neptune-ml:modelType 'SUBJECT_PREDICTION' ;
                          neptune-ml:input ?input ;
                          neptune-ml:predicate <http://aws.amazon.com/neptune/csv2rdf/
object_Property/rated> ;
```

```
                        neptune-ml:output ?output ;
                        neptune-ml:outputClass <http://aws.amazon.com/neptune/
csv2rdf/class/User> ;       }
}
```

Lista de excepciones para las consultas de inferencia de SPARQL en Neptune ML

- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at least 1 value for the parameter *(parameter name)*, found zero.
- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects at most 1 value for the parameter *(parameter name)*, found *(a number)* values.
- **BadRequestException**: Mensaje: Invalid predicate *(predicate name)* provided for external service http://aws.amazon.com/neptune/vocab/v01/services/ml#inference query.
- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the predicate *(predicate name)* to be defined.
- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the value of (parameter) *(parameter name)* to be a variable, found: *(type)*"
- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the input *(parameter name)* to be a constant, found: *(type)*.
- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference is expected to return only 1 value.
- **BadRequestException**: Mensaje: "The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference only allows StatementPatternNodes.
- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference does not allow the predicate *(predicate name)*.
- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates cannot be variables, found: *(type)*.

- **BadRequestException**: Mensaje: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates are expected to be part of the namespace *(namespace name)*, found: *(namespace name)*.

Referencia de la API de administración de Neptune ML

Contenido

Procesamiento de datos mediante el comando **`dataprocessing`**

El comando `dataprocessing` de Neptune ML se utiliza para crear un trabajo de procesamiento de datos, comprobar su estado, detenerlo o enumerar todos los trabajos de procesamiento de datos activos.

Creación de un trabajo de procesamiento de datos mediante el comando **`dataprocessing`** de Neptune ML

Un comando típico `dataprocessing` de Neptune ML para crear un nuevo trabajo tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for the new job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
      }'
```

Un comando para iniciar el reprocesamiento incremental tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/dataprocessing \
  -H 'Content-Type: application/json' \
  -d '{
        "inputDataS3Location" : "s3://(Amazon S3 bucket name)/(path to your input
 folder)",
        "id" : "(a job ID for this job)",
        "processedDataS3Location" : "s3://(S3 bucket name)/(path to your output
 folder)"
        "previousDataProcessingJobId" : "(the job ID of a previously completed job to
 update)"
}'
```

Parámetros para la creación de trabajos de **`dataprocessing`**

- **`id`**: (opcional) un identificador único para el trabajo nuevo.

Tipo: cadena. Valor predeterminado: un UUID generado automáticamente.

- **`previousDataProcessingJobId`**: (opcional) el ID de trabajo de un trabajo de procesamiento de datos completado que se ejecuta en una versión anterior de los datos.

  Tipo: cadena. Valor predeterminado: ninguno.

  Nota: Úselo para el procesamiento incremental de datos, para actualizar el modelo cuando los datos del gráfico cambien (pero no cuando se eliminen los datos).

- **`inputDataS3Location`**: (obligatorio) el URI de la ubicación de Amazon S3 en la que desea que SageMaker descargue los datos necesarios para ejecutar el trabajo de procesamiento de datos.

  Tipo: cadena.

- **`processedDataS3Location`**: (obligatorio) el URI de la ubicación de Amazon S3 en la que desea que SageMaker guarde los resultados de un trabajo de procesamiento de datos.

  Tipo: cadena.

- **`sagemakerIamRoleArn`**: (opcional) el ARN de un rol de IAM para la ejecución de SageMaker.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- **`neptuneIamRoleArn`**: (opcional) el nombre de recurso de Amazon (ARN) de un rol de IAM que Amazon SageMaker puede asumir para realizar tareas en su nombre.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- **`processingInstanceType`**: (opcional) el tipo de instancia de ML que se utiliza durante el procesamiento de datos. Su memoria debe ser lo suficientemente grande como para incluir el conjunto de datos procesado.

  Tipo: cadena. Valor predeterminado: el tipo `ml.r5` de menor tamaño cuya memoria es diez veces mayor que el tamaño de los datos de gráficos exportados en el disco.

  Nota: Neptune ML puede seleccionar el tipo de instancia automáticamente. Consulte Selección de una instancia para el procesamiento de datos.

- **`processingInstanceVolumeSizeInGB`**: (opcional) el tamaño del volumen del disco de la instancia de procesamiento. Tanto los datos de entrada como los datos procesados se almacenan

en el disco, por lo que el tamaño del volumen debe ser lo suficientemente grande como para incluir ambos conjuntos de datos.

Tipo: número entero. Valor predeterminado: `0`.

Nota: Si no se especifica o el valor es 0, Neptune ML elige el tamaño del volumen automáticamente en función del tamaño de los datos.

- **`processingTimeOutInSeconds`**: (opcional) tiempo de espera en segundos para el trabajo de procesamiento de datos.

  Tipo: número entero. Valor predeterminado: `86,400` (un día).

- **`modelType`**: (opcional) uno de los dos tipos de modelos que Neptune ML admite actualmente: modelos de subgráficos heterogéneos (`heterogeneous`) y gráficos de conocimientos (`kge`).

  Tipo: cadena. Valor predeterminado: ninguno.

  Nota: Si no se especifica, Neptune ML elige automáticamente el modelo en función de los datos.

- **`configFileName`**: (opcional) un archivo de especificación de datos que describe cómo cargar los datos de gráficos exportados para el entrenamiento. El kit de herramientas de exportación de Neptune genera automáticamente el archivo.

  Tipo: cadena. Valor predeterminado: `training-data-configuration.json`.

- **`subnets`**: (opcional) los ID de las subredes de la VPC de Neptune.

  Tipo: lista de cadenas. Valor predeterminado: ninguno.

- **`securityGroupIds`**: (opcional) los ID del grupo de seguridad de la VPC.

  Tipo: lista de cadenas. Valor predeterminado: ninguno.

- **`volumeEncryptionKMSKey`**: (opcional) la clave de AWS Key Management Service (AWS KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan el trabajo de procesamiento.

  Tipo: cadena. Valor predeterminado: ninguno.

- **`enableInterContainerTrafficEncryption`**: (opcional) habilite o deshabilite el cifrado del tráfico entre contenedores en trabajos de entrenamiento o de ajuste de hiperparámetros.

  Tipo: booleano. Valor predeterminado: true.

> **Note**
>
> El parámetro `enableInterContainerTrafficEncryption` solo está disponible en la versión 1.2.0.2.R3 del motor.

- **`s3OutputEncryptionKMSKey`**: (opcional) la clave AWS Key Management Service (AWS KMS) que SageMaker utiliza para cifrar el resultado del trabajo de entrenamiento.

  Tipo: cadena. Valor predeterminado: ninguno.

Obtención del estado de un trabajo de procesamiento de datos mediante el comando **`dataprocessing`** de Neptune ML

Un ejemplo del comando `dataprocessing` de Neptune ML para el estado de un trabajo:

```
curl -s \
  "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)" \
  | python -m json.tool
```

Parámetros para el estado del trabajo de **`dataprocessing`**

- **`id`**: (obligatorio) el identificador único del trabajo de procesamiento de datos.

  Tipo: cadena.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Detención de un trabajo de procesamiento de datos mediante el comando **`dataprocessing`** de Neptune ML

Un ejemplo del comando `dataprocessing` de Neptune ML para detener un trabajo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)"
```

Otro ejemplo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)?clean=true"
```

Parámetros para el trabajo de detención de **`dataprocessing`**

- **`id`**: (obligatorio) el identificador único del trabajo de procesamiento de datos.

  Tipo: cadena.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.
- **`clean`**: (opcional) este indicador especifica que todos los artefactos de Amazon S3 deben eliminarse cuando se detiene el trabajo.

  Tipo: booleano. Valor predeterminado: `FALSE`.

Enumeración de trabajos de procesamiento de datos activos mediante el comando **`dataprocessing`** de Neptune ML

Un ejemplo del comando `dataprocessing` de Neptune ML para enumerar los trabajos activos:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing"
```

Otro ejemplo:

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing?maxItems=3"
```

Parámetros para los trabajos de enumeración de **`dataprocessing`**

- **`maxItems`**: (opcional) el número máximo de elementos que devolver.

  Tipo: número entero. Valor predeterminado: `10`. Valor máximo permitido: `1024`.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Entrenamiento de modelos con el comando **`modeltraining`**

El comando `modeltraining` de Neptune ML se utiliza para crear un trabajo de entrenamiento de modelos, comprobar su estado, detenerlo o enumerar todos los trabajos de entrenamiento de modelos activos.

Creación de un trabajo de entrenamiento de modelos mediante el comando **`modeltraining`** de Neptune ML

Un comando `modeltraining` de Neptune ML para crear un trabajo completamente nuevo tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
      }'
```

Un comando `modeltraining` de Neptune ML para crear un trabajo de actualización para el entrenamiento incremental de modelos tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "previousModelTrainingJobId" : "(the job ID of a completed model-training job
 to update)",
      }'
```

Un comando `modeltraining` de Neptune ML para crear un nuevo trabajo con una implementación de modelo personalizada proporcionada por el usuario tiene el siguiente aspecto:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltraining
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "dataProcessingJobId" : "(the data-processing job-id of a completed job)",
        "trainModelS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-graph-
autotrainer"
        "modelName": "custom",
        "customModelTrainingParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "trainingEntryPointScript": "(your training script entry-point name in the
 Python module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parámetros para la creación de trabajos de **`modeltraining`**

- **`id`**: (opcional) un identificador único para el trabajo nuevo.

  Tipo: cadena. Valor predeterminado: un UUID generado automáticamente.

- **`dataProcessingJobId`**: (obligatorio) el ID del trabajo de procesamiento de datos completado que ha creado los datos con los que se trabajará en el entrenamiento.

  Tipo: cadena.

- **`trainModelS3Location`**: (obligatorio) la ubicación en Amazon S3 donde se van a almacenar los artefactos del modelo.

  Tipo: cadena.

- **`previousModelTrainingJobId`**: (opcional) el ID del trabajo de entrenamiento de modelos completado que desee actualizar de forma incremental en función de los datos actualizados.

  Tipo: cadena. Valor predeterminado: ninguno.

- **`sagemakerIamRoleArn`**: (opcional) el ARN de un rol de IAM para la ejecución de SageMaker.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- **`modelName`**: (opcional) el tipo de modelo para el entrenamiento. De forma predeterminada, el modelo de ML se basa automáticamente en el `modelType` utilizado en el procesamiento de datos, pero aquí puede especificar otro tipo de modelo.

  Tipo: cadena. Valor predeterminado: `rgcn` para gráficos heterogéneos y `kge` para gráficos de conocimientos. Valores válidos: para gráficos heterogéneos: `rgcn`. Para gráficos de `kge`: `transe`, `distmult` o `rotate`. Para la implementación de modelos personalizados: `custom`.

- **`baseProcessingInstanceType`**: (opcional) el tipo de instancia de ML que se utiliza para preparar y administrar el entrenamiento de modelos de ML.

  Tipo: cadena. Nota: Se trata de una instancia de CPU que se elige en función de los requisitos de memoria para procesar los datos y el modelo de entrenamiento. Consulte Selección de una instancia para el entrenamiento y la transformación de modelos.

- **`trainingInstanceType`**: (opcional) el tipo de instancia de ML que se utiliza para el entrenamiento de modelos. Todos los modelos de Neptune ML admiten el entrenamiento con CPU, GPU y varias GPU.

  Tipo: cadena. Valor predeterminado: `ml.p3.2xlarge`.

  Nota: La elección del tipo de instancia adecuado para el entrenamiento depende del tipo de tarea, del tamaño del gráfico y del presupuesto. Consulte Selección de una instancia para el entrenamiento y la transformación de modelos.

- **`trainingInstanceVolumeSizeInGB`**: (opcional) el tamaño del volumen del disco de la instancia de entrenamiento. Tanto los datos de entrada como los datos de salida se almacenan en el disco, por lo que el tamaño del volumen debe ser lo suficientemente grande como para incluir ambos conjuntos de datos.

  Tipo: número entero. Valor predeterminado: `0`.

  Nota: Si no se especifica o el valor es 0, Neptune ML selecciona un tamaño de volumen de disco en función de la recomendación generada en el paso de procesamiento de datos. Consulte Selección de una instancia para el entrenamiento y la transformación de modelos.

- **`trainingTimeOutInSeconds`**: (opcional) tiempo de espera en segundos para el trabajo de entrenamiento.

  Tipo: número entero. Valor predeterminado: `86,400` (un día).

- **`maxHPONumberOfTrainingJobs`**: número total máximo de trabajos de entrenamiento que se deben iniciar para el trabajo de ajuste de hiperparámetros.

  Tipo: número entero. Valor predeterminado: 2.

  Nota: Neptune ML ajusta automáticamente los hiperparámetros del modelo de machine learning. Para obtener un modelo que funcione bien, utilice al menos 10 trabajos (dicho de otro modo, establezca `maxHPONumberOfTrainingJobs` en 10). Por lo general, cuantos más ajustes se realicen, mejores serán los resultados.

- **`maxHPOParallelTrainingJobs`**: número máximo de trabajos de entrenamiento en paralelo que se deben iniciar para el trabajo de ajuste de hiperparámetros.

  Tipo: número entero. Valor predeterminado: 2.

  Nota: El número de trabajos en paralelo que puede ejecutar está limitado por los recursos disponibles en la instancia de entrenamiento.

- **`subnets`**: (opcional) los ID de las subredes de la VPC de Neptune.

  Tipo: lista de cadenas. Valor predeterminado: ninguno.

- **`securityGroupIds`**: (opcional) los ID del grupo de seguridad de la VPC.

  Tipo: lista de cadenas. Valor predeterminado: ninguno.

- **`volumeEncryptionKMSKey`**: (opcional) la clave de AWS Key Management Service (AWS KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan el trabajo de entrenamiento.

  Tipo: cadena. Valor predeterminado: ninguno.

- **`s3OutputEncryptionKMSKey`**: (opcional) la clave AWS Key Management Service (AWS KMS) que SageMaker utiliza para cifrar el resultado del trabajo de procesamiento.

  Tipo: cadena. Valor predeterminado: ninguno.

- **`enableInterContainerTrafficEncryption`**: (opcional) habilite o deshabilite el cifrado del tráfico entre contenedores en trabajos de entrenamiento o de ajuste de hiperparámetros.

Tipo: booleano. Valor predeterminado: true.

> **Note**
>
> El parámetro `enableInterContainerTrafficEncryption` solo está disponible en la versión 1.2.0.2.R3 del motor.

- **`enableManagedSpotTraining`**: (opcional) optimiza el costo de entrenar modelos de machine learning mediante el uso de instancias de spot de Amazon Elastic Compute Cloud. Para obtener más información, consulte Entrenamiento de spot administrado en Amazon SageMaker.

  Tipo: booleano. Valor predeterminado: false.

- **`customModelTrainingParameters`**: (opcional) la configuración para el entrenamiento de modelos personalizados. Este campo es un objeto de JSON con los siguientes campos:

  - **`sourceS3DirectoryPath`**: (obligatorio) la ruta a la ubicación de Amazon S3 donde se encuentra el módulo de Python que implementa el modelo. Debe apuntar a una ubicación válida de Amazon S3 existente que incluya, como mínimo, un script de entrenamiento, un script de transformación y un archivo `model-hpo-configuration.json`.
  - **`trainingEntryPointScript`**: (opcional) el nombre del punto de entrada en el módulo de un script que realiza el entrenamiento de modelos y utiliza los hiperparámetros como argumentos de la línea de comandos, incluidos los hiperparámetros fijos.

    Valor predeterminado: `training.py`.
  - **`transformEntryPointScript`**: (opcional) el nombre del punto de entrada en el módulo de un script que debe ejecutarse después de identificar el mejor modelo de la búsqueda de hiperparámetros, con el fin de calcular los artefactos de modelos necesarios para su implementación. Debería poder ejecutarse sin argumentos de línea de comandos.

    Valor predeterminado: `transform.py`.

- **`maxWaitTime`**: (Opcional) el tiempo máximo de espera, en segundos, al realizar el entrenamiento del modelo mediante instancias de spot. Debe ser superior a `trainingTimeOutInSeconds`.

  Tipo: número entero.

Obtención del estado de un trabajo de entrenamiento de modelos mediante el comando **`modeltraining`** de Neptune ML

Un ejemplo del comando `modeltraining` de Neptune ML para el estado de un trabajo:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)" \
  | python -m json.tool
```

Parámetros para el estado del trabajo de **`modeltraining`**

- **`id`**: (obligatorio) el identificador único del trabajo de entrenamiento de modelos.

  Tipo: cadena.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Detención de un trabajo de entrenamiento de modelos mediante el comando **`modeltraining`** de Neptune ML

Un ejemplo del comando `modeltraining` de Neptune ML para detener un trabajo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)"
```

Otro ejemplo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)?clean=true"
```

Parámetros para el trabajo de detención de **`modeltraining`**

- **`id`**: (obligatorio) el identificador único del trabajo de entrenamiento de modelos.

  Tipo: cadena.

- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.
- **`clean`**: (opcional) este indicador especifica que todos los artefactos de Amazon S3 deben eliminarse cuando se detiene el trabajo.

  Tipo: booleano. Valor predeterminado: `FALSE`.

Enumeración de trabajos de entrenamiento de modelos activos mediante el comando **`modeltraining`** de Neptune ML

Un ejemplo del comando `modeltraining` de Neptune ML para enumerar los trabajos activos:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining" | python -m json.tool
```

Otro ejemplo:

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining?maxItems=3" | python -m
 json.tool
```

Parámetros para los trabajos de enumeración de **`modeltraining`**

- **`maxItems`**: (opcional) el número máximo de elementos que devolver.

  Tipo: número entero. Valor predeterminado: `10`. Valor máximo permitido: `1024`.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Transformación de modelos mediante el comando **`modeltransform`**

El comando `modeltransform` de Neptune ML se utiliza para crear un trabajo de transformación de modelos, comprobar su estado, detenerlo o enumerar todos los trabajos de transformación de modelos activos.

Creación de un trabajo de transformación de modelos mediante el comando **`modeltransform`** de Neptune ML

Un comando `modeltransform` de Neptune ML para crear un trabajo de transformación incremental, sin tener que volver a entrenar los modelos, tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "dataProcessingJobId" : "(the job-id of a completed data-processing job)",
        "mlModelTrainingJobId" : "(the job-id of a completed model-training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform"
      }'
```

Un comando `modeltransform` de Neptune ML para crear un trabajo a partir de un trabajo de entrenamiento de SageMaker completado tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-transform job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform",
        "baseProcessingInstanceType" : ""
      }'
```

Un comando `modeltransform` de Neptune ML para crear un trabajo que utiliza una implementación de modelos personalizados tiene el siguiente aspecto:

```
curl \
```

```
  -X POST https://(your Neptune endpoint)/ml/modeltransform
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique model-training job ID)",
        "trainingJobName" : "(name of a completed SageMaker training job)",
        "modelTransformOutputS3Location" : "s3://(your Amazon S3 bucket)/neptune-model-
transform/"
        "customModelTransformParameters" : {
          "sourceS3DirectoryPath": "s3://(your Amazon S3 bucket)/(path to your Python
 module)",
          "transformEntryPointScript": "(your transform script entry-point name in the
 Python module)"
        }
      }'
```

Parámetros para la creación de trabajos de **`modeltransform`**

- **`id`**: (opcional) un identificador único para el trabajo nuevo.

  Tipo: cadena. Valor predeterminado: un UUID generado automáticamente.

- **`dataProcessingJobId`**: el ID del trabajo de un trabajo de procesamiento de datos completado.

  Tipo: cadena.

  Nota: Debe incluir los valores `dataProcessingJobId` y `mlModelTrainingJobId`, o `trainingJobName`.

- **`mlModelTrainingJobId`**: el ID de trabajo de un trabajo de entrenamiento de modelos completado.

  Tipo: cadena.

  Nota: Debe incluir los valores `dataProcessingJobId` y `mlModelTrainingJobId`, o `trainingJobName`.

- **`trainingJobName`**: el nombre de un trabajo de entrenamiento completado de SageMaker.

  Tipo: cadena.

  Nota: Debe incluir los parámetros `dataProcessingJobId` y `mlModelTrainingJobId`, o el parámetro `trainingJobName`.

- **`sagemakerIamRoleArn`**: (opcional) el ARN de un rol de IAM para la ejecución de SageMaker.

Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- **`customModelTransformParameters`** : (opcional) información de configuración para la transformación de un modelo mediante un modelo personalizado. El objeto `customModelTransformParameters` incluye los siguientes campos, que deben tener valores compatibles con los parámetros del modelo guardados del trabajo de entrenamiento:

  - **`sourceS3DirectoryPath`**: (obligatorio) la ruta a la ubicación de Amazon S3 donde se encuentra el módulo de Python que implementa el modelo. Debe apuntar a una ubicación válida de Amazon S3 existente que incluya, como mínimo, un script de entrenamiento, un script de transformación y un archivo `model-hpo-configuration.json`.
  - **`transformEntryPointScript`**: (opcional) el nombre del punto de entrada en el módulo de un script que debe ejecutarse después de identificar el mejor modelo de la búsqueda de hiperparámetros, con el fin de calcular los artefactos de modelos necesarios para su implementación. Debería poder ejecutarse sin argumentos de línea de comandos.

    Valor predeterminado: `transform.py`.

- **`baseProcessingInstanceType`**: (opcional) el tipo de instancia de ML que se utiliza para preparar y administrar el entrenamiento de modelos de ML.

  Tipo: cadena. Nota: Se trata de una instancia de CPU que se elige en función de los requisitos de memoria para procesar los datos y el modelo de transformación. Consulte Selección de una instancia para el entrenamiento y la transformación de modelos.

- **`baseProcessingInstanceVolumeSizeInGB`**: (opcional) el tamaño del volumen del disco de la instancia de entrenamiento. Tanto los datos de entrada como los datos de salida se almacenan en el disco, por lo que el tamaño del volumen debe ser lo suficientemente grande como para incluir ambos conjuntos de datos.

  Tipo: número entero. Valor predeterminado: `0`.

Nota: Si no se especifica o el valor es 0, Neptune ML selecciona un tamaño de volumen de disco en función de la recomendación generada en el paso de procesamiento de datos. Consulte Selección de una instancia para el entrenamiento y la transformación de modelos.

- **`subnets`**: (opcional) los ID de las subredes de la VPC de Neptune.

  Tipo: lista de cadenas. Valor predeterminado: ninguno.

- **`securityGroupIds`**: (opcional) los ID del grupo de seguridad de la VPC.

  Tipo: lista de cadenas. Valor predeterminado: ninguno.

- **`volumeEncryptionKMSKey`**: (opcional) la clave de AWS Key Management Service (AWS KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan el trabajo de transformación.

  Tipo: cadena. Valor predeterminado: ninguno.

- **`enableInterContainerTrafficEncryption`**: (opcional) habilite o deshabilite el cifrado del tráfico entre contenedores en trabajos de entrenamiento o de ajuste de hiperparámetros.

  Tipo: booleano. Valor predeterminado: true.

  > **Note**
  >
  > El parámetro `enableInterContainerTrafficEncryption` solo está disponible en la versión 1.2.0.2.R3 del motor.

- **`s3OutputEncryptionKMSKey`**: (opcional) la clave AWS Key Management Service (AWS KMS) que SageMaker utiliza para cifrar el resultado del trabajo de procesamiento.

  Tipo: cadena. Valor predeterminado: ninguno.

Obtención del estado de un trabajo de transformación de modelos mediante el comando **`modeltransform`** de Neptune ML

Un ejemplo del comando `modeltransform` de Neptune ML para el estado de un trabajo:

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)" \
  | python -m json.tool
```

Parámetros para el estado del trabajo de **`modeltransform`**

- **`id`**: (obligatorio) el identificador único del trabajo de transformación de modelos.

  Tipo: cadena.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Detención de un trabajo de transformación de modelos mediante el comando **`modeltransform`** de Neptune ML

Un ejemplo del comando `modeltransform` de Neptune ML para detener un trabajo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)"
```

Otro ejemplo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)?clean=true"
```

Parámetros para el trabajo de detención de **`modeltransform`**

- **`id`**: (obligatorio) el identificador único del trabajo de transformación de modelos.

  Tipo: cadena.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.
- **`clean`**: (opcional) este indicador especifica que todos los artefactos de Amazon S3 deben eliminarse cuando se detiene el trabajo.

  Tipo: booleano. Valor predeterminado: `FALSE`.

Enumeración de trabajos de transformación de modelos activos mediante el comando **`modeltransform`** de Neptune ML

Un ejemplo del comando `modeltransform` de Neptune ML para enumerar los trabajos activos:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform" | python -m json.tool
```

Otro ejemplo:

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform?maxItems=3" | python -m
 json.tool
```

Parámetros para los trabajos de enumeración de **`modeltransform`**

- **`maxItems`**: (opcional) el número máximo de elementos que devolver.

  Tipo: número entero. Valor predeterminado: `10`. Valor máximo permitido: `1024`.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Administración de los puntos de conexión de inferencia mediante el comando **endpoints**

El comando `endpoints` de Neptune ML se utiliza para crear un punto de conexión de inferencia, comprobar su estado, eliminarlo o enumerar los puntos de conexión de inferencia existentes.

Creación de un punto de conexión de inferencia mediante el comando **endpoints** de Neptune ML

Un comando `endpoints` de Neptune ML para crear un punto de conexión de inferencia a partir de un modelo creado por un trabajo de entrenamiento tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Un comando `endpoints` de Neptune ML para actualizar un punto de conexión de inferencia existente a partir de un modelo creado por un trabajo de entrenamiento tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTrainingJobId": "(the model-training job-id of a completed job)"
      }'
```

Un comando `endpoints` de Neptune ML para crear un punto de conexión de inferencia a partir de un trabajo de transformación de modelos tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
```

```
      }'
```

Un comando `endpoints` de Neptune ML para actualizar un punto de conexión de inferencia existente a partir de un trabajo de transformación de modelos tiene el siguiente aspecto:

```
curl \
  -X POST https://(your Neptune endpoint)/ml/endpoints
  -H 'Content-Type: application/json' \
  -d '{
        "id" : "(a unique ID for the new endpoint)",
        "update" : "true",
        "mlModelTransformJobId": "(the model-training job-id of a completed job)"
      }'
```

Parámetros para la creación de puntos de conexión de inferencia **`endpoints`**

- **`id`**: (opcional) un identificador único para el nuevo punto de conexión de inferencia.

  Tipo: cadena. Valor predeterminado: un nombre con marca temporal generado automáticamente.
- **`mlModelTrainingJobId`**: el ID del trabajo de entrenamiento de modelos completado que ha creado el modelo al que apuntará el punto de conexión de inferencia.

  Tipo: cadena.

  Nota: Debe proporcionar el valor `mlModelTrainingJobId` o el valor `mlModelTransformJobId`.
- **`mlModelTransformJobId`**: el ID de trabajo de un trabajo de transformación de modelos completado.

  Tipo: cadena.

  Nota: Debe proporcionar el valor `mlModelTrainingJobId` o el valor `mlModelTransformJobId`.
- **`update`**: (opcional) si está presente, este parámetro indica que se trata de una solicitud de actualización.

  Tipo: booleano. Valor predeterminado: `false`

  Nota: Debe proporcionar el valor `mlModelTrainingJobId` o el valor `mlModelTransformJobId`.

- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.
- **`modelName`**: (opcional) tipo de modelo para el entrenamiento. De forma predeterminada, el modelo de ML se basa automáticamente en el `modelType` utilizado en el procesamiento de datos, pero aquí puede especificar otro tipo de modelo.

  Tipo: cadena. Valor predeterminado: `rgcn` para gráficos heterogéneos y `kge` para gráficos de conocimientos. Valores válidos: para gráficos heterogéneos: `rgcn`. Para gráficos de conocimientos: `kge`, `transe`, `distmult` o `rotate`.
- **`instanceType`**: (opcional) el tipo de instancia de ML que se utiliza para el mantenimiento en línea.

  Tipo: cadena. Valor predeterminado: `ml.m5.xlarge`.

  Nota: La selección de la instancia de ML para un punto de conexión de inferencia depende del tipo de tarea, del tamaño del gráfico y del presupuesto. Consulte Selección de una instancia para un punto de conexión de inferencia.
- **`instanceCount`**: (opcional) el número mínimo de instancias de Amazon EC2 que se deben implementar en un punto de conexión para la predicción.

  Tipo: número entero. Valor predeterminado: `1`.
- **`volumeEncryptionKMSKey`**: (opcional) la clave de AWS Key Management Service (AWS KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan los puntos de conexión.

  Tipo: cadena. Valor predeterminado: ninguno.

Obtención del estado de un punto de conexión de inferencia mediante el comando **`endpoints`** de Neptune ML

Un ejemplo del comando `endpoints` de Neptune ML para el estado de un punto de conexión de instancia:

```
curl -s \
  "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)" \
```

```
  | python -m json.tool
```

Parámetros para el estado del punto de conexión de instancia **`endpoints`**

- **`id`**: (obligatorio) el identificador único del punto de conexión de inferencia.

  Tipo: cadena.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Eliminación de un punto de conexión de instancia mediante el comando **`endpoints`** de Neptune ML

Un ejemplo del comando `endpoints` de Neptune ML para eliminar un punto de conexión de instancia:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)"
```

Otro ejemplo:

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)?
clean=true"
```

Parámetros para la eliminación de puntos de conexión de inferencia **`endpoints`**

- **`id`**: (obligatorio) el identificador único del punto de conexión de inferencia.

  Tipo: cadena.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- **`clean`**: (opcional) indica que también se deben eliminar todos los artefactos relacionados con este punto de conexión.

  Tipo: booleano. Valor predeterminado: `FALSE`.

Enumeración de puntos de conexión de inferencia mediante el comando **`endpoints`** de Neptune ML

Un comando `endpoints` de Neptune ML para enumerar los puntos de conexión de inferencia tiene el siguiente aspecto:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints" \
  | python -m json.tool
```

Otro ejemplo:

```
curl -s "https://(your Neptune endpoint)/ml/endpoints?maxItems=3" \
  | python -m json.tool
```

Parámetros para la enumeración de puntos de conexión de inferencia **`dataprocessing`**

- **`maxItems`**: (opcional) el número máximo de elementos que devolver.

  Tipo: número entero. Valor predeterminado: `10`. Valor máximo permitido: `1024`.
- **`neptuneIamRoleArn`**: (opcional) el ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3.

  Tipo: cadena. Nota: Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Errores y excepciones de la API de administración de Neptune ML

Todas las excepciones de la API de administración de Neptune ML devuelven un código HTTP 400. Tras recibir alguna de estas excepciones, no se debe volver a intentar el comando que generó la excepción.

- **`MissingParameterException`**: Mensaje de error:

  ```
  Required credentials are missing. Please add IAM role to the cluster or pass as a parameter to this request.
  ```

- **`InvalidParameterException`** Mensajes de error:
  - `Invalid ML instance type.`
  - `Invalid ID provided. ID can be 1-48 alphanumeric characters.`
  - `Invalid ID provided. Must contain only letters, digits, or hyphens.`
  - `Invalid ID provided. Please check whether a resource with the given ID exists.`
  - `Another resource with same ID already exists. Please use a new ID.`
  - `Failed to stop the job because it has already completed or failed.`
- **`BadRequestException`** Mensajes de error:
  - `Invalid S3 URL or incorrect S3 permissions. Please check your S3 configuration.`
  - `Provided ModelTraining job has not completed.`
  - `Provided SageMaker Training job has not completed.`
  - `Provided MLDataProcessing job is not completed.`
  - `Provided MLModelTraining job doesn't exist.`
  - `Provided ModelTransformJob doesn't exist.`
  - `Unable to find SageMaker resource. Please check your input.`

Límites de Neptune ML

- Los tipos de inferencia admitidos actualmente son la clasificación de nodos, la regresión de nodos, la clasificación de bordes, la regresión de bordes y la predicción de enlaces (consulte Capacidades de Neptune ML).
- El tamaño máximo del gráfico que puede admitir Neptune ML depende de la cantidad de memoria y almacenamiento necesarios durante la preparación de los datos, el entrenamiento del modelo y la inferencia.
  - El tamaño máximo de memoria de una instancia de procesamiento de datos de SageMaker es de 768 GB. Como resultado, se produce un error en la etapa de procesamiento de datos si se necesitan más de 768 GB de memoria.
  - El tamaño máximo de memoria de una instancia de entrenamiento de SageMaker es de 732 GB. Como resultado, se produce un error en la etapa de entrenamiento si se necesitan más de 732 GB de memoria.
- El tamaño máximo de una carga útil de inferencia para un punto de conexión de SageMaker es de 6 MiB. Como resultado, se produce un error en la inferencia inductiva si la carga útil del subgráfico supera este tamaño.
- Por el momento, Neptune ML solo está disponible en las regiones en las que se admiten Neptune y los demás servicios de los que depende (como AWS Lambda, Amazon API Gateway y Amazon SageMaker).

  Existen diferencias en China (Pekín) y China (Ningxia) relacionadas con el uso predeterminado de la autenticación de IAM, tal y como se explica aquí junto con otras diferencias.
- Los puntos de conexión de inferencia de predicción de enlaces lanzados por Neptune ML actualmente solo pueden predecir posibles enlaces con nodos que estaban presentes en el gráfico durante el entrenamiento.

  Por ejemplo, supongamos que tenemos un gráfico con los vértices `User` y `Movie` y los bordes `Rated`. Con el modelo de recomendación de predicción de enlaces de Neptune ML correspondiente, puede añadir un nuevo usuario al gráfico y hacer que el modelo pronostique películas para él, pero el modelo solo puede recomendar películas que estuvieron presentes durante el entrenamiento del modelo. Aunque la incrustación de nodos `User` se calcula en tiempo real mediante su subgráfico local y el modelo GNN y, por lo tanto, puede cambiar con el tiempo a medida que los usuarios valoran las películas, se compara con las incrustaciones de películas estáticas y precalculadas para la recomendación final.

- Los modelos KGE compatibles con Neptune ML solo funcionan para tareas de predicción de enlaces, y las representaciones son específicas de los tipos de bordes y vértices presentes en el gráfico durante el entrenamiento. Esto significa que todos los tipos de bordes y vértices a los que se hace referencia en una consulta de inferencia deben haber estado presentes en el gráfico durante el entrenamiento. No se pueden hacer predicciones para nuevos tipos de bordes o vértices sin volver a entrenar el modelo.

Limitaciones de los recursos de SageMaker

Según las actividades y el uso de recursos con el paso del tiempo, es posible que reciba mensajes que indique que ha superado su cuota (ResourceLimitExceeded) y que necesita escalar verticalmente sus recursos de SageMaker. Siga los pasos del procedimiento Solicitar un aumento de la cuota de servicio para los recursos de SageMaker de esta página para solicitar un aumento de la cuota a AWS Support.

Los nombres de los recursos de SageMaker corresponden a las etapas de Neptune ML, como, por ejemplo:

- `ProcessingJob` de SageMaker lo utilizan los trabajos de procesamiento de datos, entrenamiento de modelos y transformación de modelos de Neptune.
- `HyperParameterTuningJob` de SageMaker lo utilizan los trabajos de entrenamiento de modelos de Neptune.
- `TrainingJob` de SageMaker lo utilizan los trabajos de entrenamiento de modelos de Neptune.
- `Endpoint` de SageMaker lo utilizan los puntos de conexión de inferencia de Neptune.

Monitorización de recursos de Amazon Neptune

Amazon Neptune admite varios métodos para monitorizar el rendimiento y el uso de la base de datos:

- Estado de la instancia: compruebe el estado del motor de base de datos de gráficos del clúster de Neptune, averigüe qué versión del motor está instalada y obtenga otra información relacionada con la instancia mediante la API de estado de la instancia.
- API de resumen de gráficos: la API de resumen de gráficos le permite conocer rápidamente el tamaño y el contenido de los datos de sus gráficos.

> **Note**
>
> Como la API de resumen de gráficos se basa en las estadísticas del DFE, solo está disponible cuando las estadísticas están habilitadas, lo que no ocurre en los tipos de instancias T3 y T4g.

- Amazon CloudWatch: Neptune envía automáticamente las métricas a las alarmas CloudWatch y también las admite CloudWatch . Para obtener más información, consulte the section called “Usando CloudWatch”.
- Archivos de registro de auditoría: consulte, descargue o vea los archivos de registro de base de datos con la consola de Neptune. Para obtener más información, consulte the section called “Registros de auditoría con Neptune”.
- Publicación de registros en Amazon CloudWatch Logs: puede configurar un clúster de base de datos de Neptune para publicar datos de registros de auditoría en un grupo de registros de Amazon CloudWatch Logs. Con CloudWatch Logs, puede realizar un análisis en tiempo real de los datos de registro, usarlos CloudWatch para crear alarmas y ver métricas, y usar CloudWatch Logs para almacenar sus registros de registro en un almacenamiento de alta durabilidad. Consulte Registros de Neptune CloudWatch .
- AWS CloudTrail— Neptune admite el registro de API mediante. CloudTrail Para obtener más información, consulte the section called “Registro de llamadas a la API de Neptune con AWS CloudTrail”.
- Suscripciones a notificaciones de eventos: suscríbase a los eventos de Neptune para mantenerse informado de lo que está sucediendo. Para obtener más información, consulte the section called “Notificaciones de eventos”.

- Etiquetado: utilice etiquetas para añadir metadatos a sus recursos de Neptune y realizar un seguimiento del uso en función de las etiquetas. Para obtener más información, consulte the section called “Etiquetado de recursos de Neptune”.

Temas

Compruebe el estado de una instancia de Neptune

Amazon Neptune proporciona un mecanismo para comprobar el estado de la base de datos de gráficos en el host. También es una buena forma para confirmar que puede conectarse a una instancia.

Para comprobar el estado de una instancia y obtener el estado del clúster de base de datos mediante `curl`:

```
curl -G https://your-neptune-endpoint:port/status
```

O bien, a partir de la versión 1.2.1.0.R6 del motor, puede utilizar el siguiente comando de la CLI en su lugar:

```
aws neptunedata get-engine-status
```

Si la instancia es correcta, el comando `status` devuelve un objeto JSON con los siguientes campos:

- **`status`**: se establece en `"healthy"` si la instancia no está experimentando problemas.

Si la instancia se está recuperando de un bloqueo o se reinicia y hay transacciones activas en ejecución desde el último cierre del servidor, `status` se establece en `"recovery"`.

- **`startTime`**: se establece en la hora UTC a la que se inició el proceso actual del servidor.
- **`dbEngineVersion`**: se establece en la versión del motor de Neptune que se ejecuta en el clúster de base de datos.

  Si esta versión del motor se ha parcheado de forma manual desde que se lanzó, el número de versión está precedido por `"Patch-"`.
- **`role`**: se establece en `"reader"` si la instancia es una réplica de lectura o en `"writer"` si la instancia es la instancia principal.
- **`dfeQueryEngine`**: se establece en `"enabled"` si el motor DFE está totalmente habilitado o en `viaQueryHint` (valor predeterminado) si el motor DFE solo se usa con consultas que tienen la sugerencia de consulta `useDFE` establecida en `true` (`viaQueryHint` es el valor predeterminado).
- **`gremlin`**: incluye información sobre el lenguaje de consultas de Gremlin disponible en el clúster. En concreto, contiene un `version` campo que especifica la TinkerPop versión actual que utiliza el motor.
- **`sparql`**: incluye información sobre el lenguaje de consultas de SPARQL disponible en el clúster. Específicamente, incluye un campo de `version` que especifica la versión actual de SPARQL que utiliza el motor.
- **`opencypher`** incluye información sobre el lenguaje de consultas de openCypher disponible en el clúster. Específicamente, incluye un campo de `version` que especifica la versión actual de openCypher que utiliza el motor.
- **`labMode`**: incluye los ajustes de Modo lab que utiliza el motor.
- **`rollingBackTrxCount`**: si hay transacciones que se están restaurando, este campo se establece en el número de dichas transacciones. Si no hay ninguna transacción, el campo no aparecerá.
- **`rollingBackTrxEarliestStartTime`**: se establece en la hora de inicio de la primera transacción que se va a restaurar. Si no se está revirtiendo ninguna transacción, este campo no aparecerá.
- **`features`**: incluye información de estado sobre las características habilitadas en el clúster de base de datos.

- **`lookupCache`**: el estado actual del Caché de búsqueda. Este campo solo aparece en los tipos de instancias R5d, ya que son las únicas instancias en las que puede existir una caché de búsqueda. El campo es un objeto JSON con el siguiente formato:

  ```
  "lookupCache": {
    "status": "current lookup cache status"
  }
  ```

  En una instancia R5d:

  - Si la caché de búsqueda está habilitada, el estado aparece como `"Available"`.
  - Si la caché de búsqueda está deshabilitada, el estado aparece como `"Disabled"`.
  - Si se ha alcanzado el límite de disco en la instancia, el estado aparece como `"Read Only Mode - Storage Limit Reached"`.

- **`ResultCache`**: el estado actual del Almacenamiento en caché de resultados de las consultas. Este campo es un objeto JSON con el siguiente formato:

  ```
  "ResultCache": {
    "status": "current results cache status"
  }
  ```

  - Si se ha habilitado la caché de resultados, el estado aparece como `"Available"`.
  - Si la caché está deshabilitada, el estado aparece como `"Disabled"`.

- **`IAMAuthentication`**— Especifica si la autenticación AWS Identity and Access Management (IAM) está habilitada o no en el clúster de base de datos:
  - Si se ha habilitado la autenticación de IAM, el estado aparece como `"enabled"`.
  - Si se ha deshabilitado la autenticación de IAM, el estado aparece como `"disabled"`.
- **`Streams`**: especifica si los flujos de Neptune se han habilitado o no en su clúster de base de datos:
  - Si los flujos se han habilitado, el estado aparece como `"enabled"`.
  - Si los flujos se han deshabilitado, el estado aparece como `"disabled"`.
- **`AuditLog`**: igual a `enabled` si los registros de auditoría están habilitados; de lo contrario, es `disabled`.
- **`SlowQueryLogs`**: igual a `info` o debug si el registro de consultas lentas está habilitado; de lo contrario, es `disabled`.

  - **`QueryTimeout`**: el valor, en milisegundos, del tiempo de espera de la consulta.
- **`settings`**: configuración aplicada a la instancia:
  - **`clusterQueryTimeoutInMs`**: el valor, en milisegundos, del tiempo de espera de la consulta, establecido para todo el clúster.
  - **`SlowQueryLogsThreshold`**: el valor, en milisegundos, del tiempo de espera de la consulta, establecido para todo el clúster.
- **`serverlessConfiguration`**: configuración sin servidor para un clúster si se ejecuta sin servidor:
  - **`minCapacity`**: el tamaño más pequeño al que puede reducirse una instancia sin servidor de su clúster de base de datos, en unidades de capacidad de Neptune (NCU).
  - **`maxCapacity`**: el tamaño más grande al que puede reducirse una instancia sin servidor de su clúster de base de datos, en unidades de capacidad de Neptune (NCU).

Ejemplo del resultado del comando de estado de instancia

A continuación, se muestra un ejemplo del resultado del comando de estado de instancia (en este caso, se ejecuta en una instancia de R5d):

```
{
  'status': 'healthy',
  'startTime': 'Thu Aug 24 21:47:12 UTC 2023',
  'dbEngineVersion': '1.2.1.0.R4',
  'role': 'writer',
  'dfeQueryEngine': 'viaQueryHint',
  'gremlin': {'version': 'tinkerpop-3.6.2'},
  'sparql': {'version': 'sparql-1.1'},
  'opencypher': {'version': 'Neptune-9.0.20190305-1.0'},
  'labMode': {
    'ObjectIndex': 'disabled',
    'ReadWriteConflictDetection': 'enabled'
  },
  'features': {
    'SlowQueryLogs': 'disabled',
    'ResultCache': {'status': 'disabled'},
    'IAMAuthentication': 'disabled',
    'Streams': 'disabled',
    'AuditLog': 'disabled'
  },
```

```
  'settings': {
    'clusterQueryTimeoutInMs': '120000',
    'SlowQueryLogsThreshold': '5000'
  },
  'serverlessConfiguration': {
    'minCapacity': '1.0',
    'maxCapacity': '128.0'
  }
}
```

Si existe algún problema con la instancia, el comando de estado devuelve el código de error HTTP `500`. Si el host no está disponible, se agota el tiempo de espera de la solicitud. Asegúrese de tener acceso a la instancia desde la nube virtual privada (VPC) y de que los grupos de seguridad le permiten el acceso a ella.

Monitorización de Neptune con Amazon CloudWatch

Amazon Neptune y Amazon CloudWatch están integrados para que pueda recopilar y analizar las métricas de rendimiento. Puede supervisar estas métricas mediante la CloudWatch consola, la AWS Command Line Interface (AWS CLI) o la CloudWatch API.

CloudWatch también te permite configurar alarmas para que puedas recibir notificaciones si el valor de una métrica supera un umbral que especifiques. Incluso puede configurar CloudWatch eventos para tomar medidas correctivas en caso de que se produzca una infracción. Para obtener más información sobre el uso CloudWatch y las alarmas, consulte la CloudWatch documentación.

Temas

- Visualización CloudWatch de datos (consola)
- Visualización CloudWatch de datos (AWS CLI)
- Visualización de CloudWatch datos (API)
- Utilización CloudWatch para supervisar el rendimiento de las instancias de base de datos en Neptune
- Métricas de Neptune CloudWatch
- Dimensiones de Neptune CloudWatch

Visualización CloudWatch de datos (consola)

Para ver CloudWatch los datos de un clúster de Neptune (consola)

1. Inicie sesión en la CloudWatch consola AWS Management Console y ábrala en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, seleccione Métricas.
3. En el panel Todas las métricas, elija Neptune y, a continuación, elija DB. ClusterIdentifier
4. En el panel superior, desplácese hacia abajo para ver la lista completa de métricas correspondientes a su clúster. Las opciones de métricas de Neptune disponibles aparecen en la lista Viendo.

Para seleccionar o anular la selección de una métrica individual, en el panel de resultados, marque la casilla situada junto al nombre del recurso y la métrica. Los gráficos que muestran las métricas de los elementos seleccionados aparecen en la parte inferior de la consola. Para obtener más información sobre los CloudWatch gráficos, consulta Graph Metrics en la Guía del CloudWatch usuario de Amazon.

Visualización CloudWatch de datos (AWS CLI)

Para ver CloudWatch los datos de un cúmulo de Neptuno ()AWS CLI

1. Instale el AWS CLI. Para conocer las instrucciones, consulte la Guía del usuario de AWS Command Line Interface.
2. Utilice el AWS CLI para buscar información. Los CloudWatch parámetros relevantes para Neptune se enumeran en. Métricas de Neptune CloudWatch

   El siguiente ejemplo recupera CloudWatch las métricas del número de solicitudes de Gremlin por segundo para el clúster. `gremlin-cluster`

```
aws cloudwatch get-metric-statistics \
    --namespace AWS/Neptune  --metric-name GremlinRequestsPerSec \
    --dimensions Name=DBClusterIdentifier,Value=gremlin-cluster \
    --start-time 2018-03-03T00:00:00Z --end-time 2018-03-04T00:00:00Z \
    --period 60 --statistics=Average
```

Visualización de CloudWatch datos (API)

CloudWatch también admite una `Query` acción para que puedas solicitar información mediante programación. Para obtener más información, consulta la documentación de la API de CloudWatch Query y la referencia de la CloudWatch API de Amazon.

Cuando una CloudWatch acción requiera un parámetro específico para la supervisión de Neptuno, por ejemplo`MetricName`, utilice los valores que aparecen en. Métricas de Neptune CloudWatch

El siguiente ejemplo muestra una CloudWatch solicitud de bajo nivel que utiliza los siguientes parámetros:

- `Statistics.member.1 = Average`
- `Dimensions.member.1 = DBClusterIdentifier=gremlin-cluster`
- `Namespace = AWS/Neptune`
- `StartTime = 2013-11-14T00:00:00Z`
- `EndTime = 2013-11-16T00:00:00Z`
- `Period = 60`
- `MetricName = GremlinRequestsPerSec`

Este es el aspecto de la CloudWatch solicitud. Sin embargo, solo se muestra el formato de la solicitud; debe construir su propia solicitud en función de sus métricas y plazo de tiempo.

```
https://monitoring.amazonaws.com/
      ?SignatureVersion=2
      &Action=GremlinRequestsPerSec
      &Version=2010-08-01
      &StartTime=2018-03-03T00:00:00
      &EndTime=2018-03-04T00:00:00
      &Period=60
      &Statistics.member.1=Average
      &Dimensions.member.1=DBClusterIdentifier=gremlin-cluster
      &Namespace=AWS/Neptune
      &MetricName=GremlinRequests
      &Timestamp=2018-03-04T17%3A48%3A21.746Z
      &AWSAccessKeyId=AWS Access Key ID;
      &Signature=signature
```

Utilización CloudWatch para supervisar el rendimiento de las instancias de base de datos en Neptune

Puede utilizar CloudWatch las métricas de Neptune para supervisar lo que ocurre en sus instancias de base de datos y realizar un seguimiento de la longitud de la cola de consultas según lo observa la base de datos. Las siguientes métricas son particularmente útiles:

- **`CPUUtilization`**: muestra el porcentaje de utilización de CPU.
- **`VolumeWriteIOPs`**: número medio de operaciones de E/S de escritura en el disco en el volumen de clúster indicadas a intervalos de 5 minutos.
- **`MainRequestQueuePendingRequests`**: muestra el número de solicitudes que espera en la cola de entrada pendientes de ejecución.

También puede averiguar cuántas solicitudes están pendientes en el servidor utilizando el punto de conexión de estado de la consulta de Gremlin con el parámetro `includeWaiting`. Esto le mostrará el estado de todas las consultas en espera.

Los siguientes indicadores pueden ayudarle a ajustar sus estrategias de aprovisionamiento y consulta de Neptune para mejorar la eficiencia y el rendimiento:

- Una latencia constante, una `CPUUtilization` alta, unos `VolumeWriteIOPs` altos y unas `MainRequestQueuePendingRequests` bajas en conjunto muestran que el servidor está trabajando activamente en el procesamiento de solicitudes de escritura simultáneas a un ritmo sostenible, con pocas esperas de E/S.
- Una latencia constante, una `CPUUtilization` baja, unos `VolumeWriteIOPs` bajos y unas `MainRequestQueuePendingRequests` nulas en conjunto indican que hay un exceso de capacidad en la instancia de base de datos principal para procesar las solicitudes de escritura.
- Una `CPUUtilization` alta y unos `VolumeWriteIOPs` altos, pero una latencia y unas `MainRequestQueuePendingRequests` variables, en conjunto, indican que está enviando más trabajo del que el servidor puede procesar en un intervalo determinado. Considere la posibilidad de crear solicitudes por lotes o cambiar su tamaño para que realicen la misma cantidad de trabajo con menos sobrecarga transaccional o de escalar la instancia principal para aumentar el número de subprocesos de consulta capaces de procesar solicitudes de escritura de forma simultánea.
- Una `CPUUtilization` baja con unos `VolumeWriteIOPs` altos significa que los subprocesos de consulta están esperando a que se completen las operaciones de E/S de la capa de almacenamiento. Si observa latencias variables y algunas aumentan en

`MainRequestQueuePendingRequests`, considere la posibilidad de crear o cambiar el tamaño de las solicitudes por lotes para realizar la misma cantidad de trabajo con menos sobrecarga transaccional.

Métricas de Neptune CloudWatch

> **Note**
>
> Amazon Neptune envía las métricas CloudWatch solo cuando tienen un valor distinto de cero.
> Para todas las métricas de Neptune, la granularidad de acumulación es de 5 minutos.

Temas

Métricas de Neptune CloudWatch

En la siguiente tabla se enumeran las CloudWatch métricas que admite Neptune.

> **Note**
>
> Todas las métricas acumuladas se restablecen a cero cada vez que el servidor se reinicia, ya sea para realizar tareas de mantenimiento, para reiniciarse o para recuperarse de una caída.

Métricas de Neptune CloudWatch

Métrica	Descripción
`BackupRetentionPeriodStorageUsed`	La cantidad total de almacenamiento de copias de seguridad, en bytes, que se utiliza para permitir el periodo de retención de copias de seguridad del clúster de base de datos de Neptune. Se incluye en el total registrado por la métrica `TotalBackupStorageBilled` .

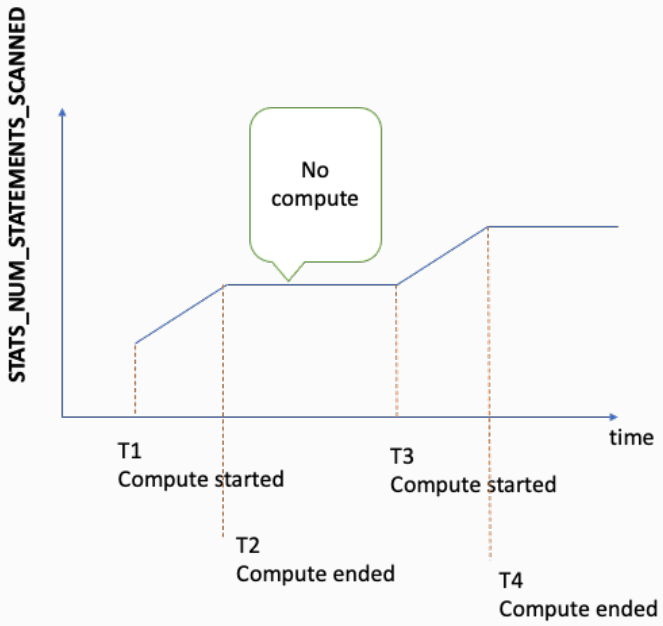
Métrica	Descripción
BufferCacheHitRatio	Porcentaje de solicitudes que se responden desde la caché de búfer. Esta métrica puede resultar útil para diagnosticar la latencia de las consultas, ya que la falta de memoria caché provoca una latencia significativa. Si la relación de aciertos de la caché es inferior al 99,9 %, considere la posibilidad de actualizar el tipo de instancia para almacenar en caché más datos en memoria.
ClusterReplicaLag	Para una réplica de lectura, retardo en milisegundos que se da cuando la replicación actualiza desde la instancia principal.
ClusterReplicaLagMaximum	Retardo máximo en milisegundos entre la instancia principal y cada instancia de base de datos de Neptune del clúster de base de datos.
ClusterReplicaLagMinimum	Retardo mínimo en milisegundos entre la instancia principal y cada instancia de base de datos de Neptune del clúster de base de datos.
CPUUtilization	El porcentaje de utilización de CPU.
EngineUptime	Cantidad de tiempo en segundos que la instancia lleva en ejecución.
FreeableMemory	Cantidad de memoria de acceso aleatorio disponible en bytes.
GlobalDbDataTransferBytes	El número de bytes de datos de redo log transferidos de la base de datos global Región de AWS de Neptune Región de AWS a una secundaria.

Métrica	Descripción
GlobalDbReplicatedWriteIO	El número de operaciones de E/S de escritura replicadas desde la Región de AWS principal de la base de datos global hasta el volumen de clúster de una Región de AWS secundaria. Los cálculos de facturación de cada clúster de base de datos de una base de datos global de Neptune utilizan la métrica VolumeWriteIOPS para contabilizar las escrituras realizadas dentro del clúster. En el caso del clúster de base de datos principal, los cálculos de facturación utilizan GlobalDbReplicatedWriteIO para contabilizar la replicación entre regiones en clústeres de bases de datos secundarios.
GlobalDbProgressLag	El número de milisegundos que un clúster secundario está por detrás del clúster principal tanto para las transacciones de usuario como para las del sistema.
GremlinRequestsPerSec	Número de solicitudes por segundo al motor Gremlin.
GremlinWebSocketOpenConnections	El número de WebSocket conexiones abiertas a Neptune.
LoaderRequestsPerSec	Número de solicitudes del programa de carga por segundo.
MainRequestQueuePendingRequests	El número de solicitudes que espera en la cola de entrada pendientes de ejecución. Neptune comienza a limitar las solicitudes cuando superan la capacidad máxima de cola.

Métrica	Descripción
`NCUUtilization`	Solo se aplica a una instancia de base de datos o un clúster de base de datos de Neptune sin servidor. A nivel de instancia, informa de un porcentaje calculado como el número de unidades de capacidad de Neptune (NCU) que utiliza actualmente la instancia en cuestión, dividido por la configuración de capacidad máxima de la NCU para el clúster. Una NCU, o unidad de capacidad de Neptune, consta de 2 GiB (gibibyte) de memoria (RAM), junto con la capacidad del procesador virtual (vCPU) y la red asociadas. A nivel de clúster, `NCUUtilization` indica el porcentaje de capacidad máxima que utiliza el clúster en su conjunto.
`NetworkThroughput`	Cantidad de rendimiento de red en bytes por segundo recibida de los clientes y transmitida a ellos por cada instancia del clúster de base de datos de Neptune. Este rendimiento no incluye el tráfico de red entre las instancias del clúster de bases de datos y el volumen de clúster.
`NetworkTransmitThroughput`	Cantidad de rendimiento de red de salida en bytes por segundo transmitida a los clientes por cada instancia del clúster de base de datos de Neptune. Este rendimiento no incluye el tráfico de red entre las instancias del clúster de bases de datos y el volumen de clúster.
`NumTxCommitted`	Número de transacciones confirmadas correctamente por segundo.
`NumTxOpened`	Número de transacciones abiertas en el servidor por segundo.

Métrica	Descripción
`NumTxRolledBack`	Para consultas de escritura, número de transacciones por segundo restauradas en el servidor debido a errores. En el caso de las consultas de solo lectura, esta métrica es igual al número de transacciones de solo lectura completadas por segundo.
`OpenCypherRequestsPerSec`	Número de solicitudes por segundo (tanto HTTPS como Bolt) al motor de openCypher.
`OpenCypherBoltOpenConnections`	Número de conexiones Bolt abiertas a Neptune.
`ServerlessDatabaseCapacity`	Como métrica de nivel de instancia, `ServerlessDatabaseCapacity` informa de la capacidad de instancia actual de una instancia de Neptune sin servidor determina da, en NCU. Una NCU, o unidad de capacidad de Neptune, consta de 2 GiB (gibibyte) de memoria (RAM), junto con la capacidad del procesador virtual (vCPU) y la red asociadas. A nivel de clúster, `ServerlessDatabase Capacity` informa de la media de todos los valores de `ServerlessDatabase Capacity` de las instancias de base de datos del clúster.
`SnapshotStorageUsed`	La cantidad total de almacenamiento de copias de seguridad consumida por todas las instantáneas de un clúster de base de datos de Neptune determinado fuera de su periodo de retención de copia de seguridad en bytes. Se incluye en el total registrado por la métrica `TotalBackupStorageBilled` .

Métrica	Descripción
`SparqlRequestsPerSec`	Número de solicitudes al motor de SPARQL por segundo.

Métrica	Descripción
`StatsNumStatementsScanned`	El número total de instrucciones escaneadas en busca de estadísticas de DFE desde que se inició el servidor. Cada vez que se activa el cálculo de estadísti cas, este número aumenta y, cuando no se realiza ningún cálculo, permanece estático. Como resultado, si lo representa en un gráfico a lo largo del tiempo, puede saber cuándo se realizó el cálculo y cuándo no:  Al observar la pendiente del gráfico en los períodos en los que la métrica aumenta, también puede ver lo rápido que se realizó el cálculo. Si no existe esa métrica, significa que la característica de estadísticas está deshabili tada en el clúster de base de datos o que la versión del motor que está ejecutando no tiene la característica de estadísticas. Si el valor

Métrica	Descripción
	de la métrica es cero, significa que no se ha realizado ningún cálculo estadístico.
`TotalBackupStorageBilled`	La cantidad total de almacenamiento de copias de seguridad facturada para un clúster de base de datos de Neptune determina do en bytes. Incluye el almacenamiento de copias de seguridad medido por las métricas `BackupRetentionPeriodStorageUsed` y `SnapshotStorageUsed` .
`TotalRequestsPerSec`	Número total de solicitudes por segundo dirigidas al servidor desde todos los orígenes.
`TotalClientErrorsPerSec`	Número total por segundo de solicitudes que han dado error debido a problemas del lado del cliente.
`TotalServerErrorsPerSec`	Número total por segundo de solicitudes en las que se ha producido un error en el servidor debido a errores internos.

Métrica	Descripción
`UndoLogListSize`	El recuento de registros de deshacer en la lista de registros de deshacer. Los registros de deshacer contienen registros de transacciones confirmadas que caducan cuando todas las transacciones activas son más recientes que la fecha de confirmación. Los registros caducados se depuran periódica mente. Los registros de las operaciones de eliminación pueden tardar más en depurarse que los registros de otros tipos de transacci ones. La depuración la realiza exclusivamente la instancia de escritura del clúster de base de datos, por lo que la velocidad de depuración depende del tipo de instancia de escritura. Si el valor de `UndoLogListSize` es alto y está aumentando en su clúster de base de datos, actualice la instancia de escritor para aumentar la tasa de depuración. Además, si está actualizando a la versión del motor `1.2.0.0` o posterior desde una versión anterior a la `1.2.0.0`, asegúrese primero de que el valor de `UndoLogListSize` esté cerca de 0. Como las versiones del motor `1.2.0.0` y posteriores utilizan un formato diferente para los registros de deshacer, la actualización solo puede comenzar una vez que los registros de deshacer anteriores se hayan depurado por completo. Para obtener más información, consulte Actualización a la versión 1.2.0.0 o posterior.

Métrica	Descripción
`VolumeBytesUsed`	La cantidad total de almacenamiento asignado al clúster de base de datos de Neptune, en bytes. Esta es la cantidad de almacenamiento por la que se le factura. Es la cantidad máxima de almacenamiento asignada al clúster de base de datos en cualquier momento de su existencia, no la cantidad que está utilizando actualmente (consulte Facturación del almacenamiento de Neptune).
`VolumeReadIOPs`	El número total de operaciones de E/S de lectura facturadas desde un volumen de clúster, informadas en intervalos de 5 minutos. Las operaciones de lectura facturadas se calculan en el nivel del volumen de clúster, agrupadas desde todas las instancias del clúster de base de datos de Neptune y notificadas a continuación a intervalos de 5 minutos.
`VolumeWriteIOPs`	El número total de operaciones de E/S del disco de escritura en el volumen del clúster, registrado en intervalos de 5 minutos.

CloudWatch Métricas que ahora están en desuso en Neptune

Estas métricas están ahora en desuso. Siguen siendo compatibles, pero podrían eliminarse en el futuro cuando haya nuevas y mejores métricas disponibles.

Métrica	Descripción
`GremlinHttp1xx`	Número de respuestas HTTP 1xx del punto de enlace de Gremlin por segundo. Le recomendamos que utilice la nueva métrica combinada `Http1xx` en su lugar.

Métrica	Descripción
`GremlinHttp2xx`	Número de respuestas HTTP 2xx del punto de enlace de Gremlin por segundo. Le recomendamos que utilice la nueva métrica combinada `Http2xx` en su lugar.
`GremlinHttp4xx`	Número de errores HTTP 4xx del punto de enlace de Gremlin por segundo. Le recomendamos que utilice la nueva métrica combinada `Http4xx` en su lugar.
`GremlinHttp5xx`	Número de errores HTTP 5xx del punto de enlace de Gremlin por segundo. Le recomendamos que utilice la nueva métrica combinada `Http5xx` en su lugar.
`GremlinErrors`	Número de errores en recorridos Gremlin.
`GremlinRequests`	Número de solicitudes al motor de Gremlin.
`GremlinWebSocketSuccess`	Número de WebSocket conexiones correctas al punto final de Gremlin por segundo.
`GremlinWebSocketClientErrors`	Número de errores de WebSocket cliente en el punto final de Gremlin por segundo.
`GremlinWebSocketServerErrors`	Número de errores del WebSocket servidor en el punto final de Gremlin por segundo.
`GremlinWebSocketAvailableConnections`	Número de WebSocket conexiones potenciales disponibles actualmente.

Métrica	Descripción
`Http100`	Número de respuestas HTTP 100 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http1xx` en su lugar.
`Http101`	Número de respuestas HTTP 101 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http1xx` en su lugar.
`Http1xx`	Número de respuestas HTTP 1xx del punto de enlace por segundo.
`Http200`	Número de respuestas HTTP 200 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http2xx` en su lugar.
`Http2xx`	Número de respuestas HTTP 2xx del punto de enlace por segundo.
`Http400`	Número de errores HTTP 400 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http4xx` en su lugar.
`Http403`	Número de errores HTTP 403 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http4xx` en su lugar.

Métrica	Descripción
`Http405`	Número de errores HTTP 405 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http4xx` en su lugar.
`Http413`	Número de errores HTTP 413 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http4xx` en su lugar.
`Http429`	Número de errores HTTP 429 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http4xx` en su lugar.
`Http4xx`	Número de errores HTTP 4xx del punto de enlace por segundo.
`Http500`	Número de errores HTTP 500 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http5xx` en su lugar.
`Http501`	Número de errores HTTP 501 del punto de enlace por segundo. Le recomendamos que utilice la nueva métrica combinada `Http5xx` en su lugar.
`Http5xx`	Número de errores HTTP 5xx del punto de enlace por segundo.
`LoaderErrors`	Número de errores de las solicitudes del programa de carga.

Métrica	Descripción
`LoaderRequests`	Número de solicitudes del programa de carga.
`SparqlHttp1xx`	Número de respuestas HTTP 1xx del punto de enlace de SPARQL por segundo. Le recomendamos que utilice la nueva métrica combinada `Http1xx` en su lugar.
`SparqlHttp2xx`	Número de respuestas HTTP 2xx del punto de enlace de SPARQL por segundo. Le recomendamos que utilice la nueva métrica combinada `Http2xx` en su lugar.
`SparqlHttp4xx`	Número de errores HTTP 4xx del punto de enlace de SPARQL por segundo. Le recomendamos que utilice la nueva métrica combinada `Http4xx` en su lugar.
`SparqlHttp5xx`	Número de errores HTTP 5xx del punto de enlace de SPARQL por segundo. Le recomendamos que utilice la nueva métrica combinada `Http5xx` en su lugar.
`SparqlErrors`	Número de errores en las consultas SPARQL.
`SparqlRequests`	Número de solicitudes al motor de SPARQL.
`StatusErrors`	Número de errores del punto de enlace de estado.
`StatusRequests`	Número de solicitudes al punto de enlace de estado.

Dimensiones de Neptune CloudWatch

Las métricas de Amazon Neptune se identifican por los valores de la cuenta, el nombre del gráfico o la operación. Puedes usar la CloudWatch consola de Amazon para recuperar los datos de Neptune junto con cualquiera de las dimensiones de la siguiente tabla.

Dimensión	Descripción
`DBInstanceIdentifier`	Filtra los datos solicitados para una instancia de base de datos específica en un clúster.
`DBClusterIdentifier`	Filtra los datos solicitados que son específicos del clúster de base de datos de Neptune.
`DBClusterIdentifier` , `EngineName`	Filtra los datos por clúster. El nombre de motor de todas las instancias de Neptune es `neptune`.
`DBClusterIdentifier` , `Role`	Filtra los datos solicitados para un clúster de base de datos de Neptune específico, agrupando las métricas por rol de instancia (WRITER/READER). Por ejemplo, puede agregar métricas para todas las instancias READER que pertenezcan a un clúster.
`DBClusterIdentifier` , `SourceRegion`	Filtra los datos por clúster principal en una región principal de base de datos global.
`DatabaseClass`	Filtra los datos solicitados para todas las instancias de una clase de base de datos. Por ejemplo, puede agregar métricas para todas las instancias que pertenezcan a la clase de base de datos db.r4.large.
`EngineName`	El nombre de motor de todas las instancias de Neptune es `neptune`.

Dimensión	Descripción
`GlobalDbDBClusterIdentifier` , `SecondaryRegion`	Filtra los datos por el clúster secundario de una base de datos global específica en una región secundaria.

Uso de registros de auditoría con un clúster de Amazon Neptune

Para auditar la actividad del clúster de base de datos de Amazon Neptune, habilite la recopilación de registros de auditoría estableciendo un parámetro de clúster de base de datos. Si se habilitan los logs de auditoría, puede utilizarlos para registrar cualquier combinación de eventos compatibles. Puede ver o descargar los registros de auditoría para revisarlos.

Habilitación de los registros de auditoría de Neptune

Utilice el parámetro `neptune_enable_audit_log` para habilitar (`1`) o deshabilitar (`0`) los logs de auditoría.

Establezca este parámetro en en el grupo de parámetros que utiliza el clúster de base de datos. Puede utilizar el procedimiento que se muestra en Edición de un grupo de parámetros de clúster de base de datos o de un grupo de parámetros de base de datos para modificar el parámetro mediante el AWS Management Console comando modify-db-cluster-parameter-group o el AWS CLI comando de la API ModifyDB Group para modificar el parámetro mediante programación. ClusterParameter

Debe reiniciar las instancias de base de datos después de modificar este parámetro para aplicar el cambio.

Visualización de registros de auditoría de Neptune mediante la consola

Puede ver y descargar los registros de auditoría mediante la AWS Management Console. En la página Instances (Instancias), elija la instancia de base de datos para mostrar sus detalles y, a continuación, desplácese hasta la sección Logs.

Para descargar un archivo de log, selecciónelo en la sección Logs y, a continuación, elija Download (Descargar).

Detalles de registros de auditoría de Neptune

Los archivos de log están en formato UTF-8. Los logs se escriben en varios archivos, cuyo número varía en función del tamaño de la instancia. Para ver los eventos más recientes, es posible que tenga que revisar todos los archivos de log de auditoría.

Las entradas del registro no están en orden secuencial. Puede usar el valor `timestamp` para ordenarlas.

Los archivos de log se rotan cuando llegan a 100 MB combinados. Este límite no se puede configurar.

Los archivos de log de auditoría incluyen la siguiente información delimitada por comas en las filas en el orden siguiente:

Campo	Descripción
Timestamp	Marca temporal de Unix para el evento registrado con una precisión de microsegundos.
ClientHost	El nombre de host o la IP desde donde se ha conectado el usuario.
ServerHost	El nombre de host o la IP de la instancia para la que se ha registrado el evento.
ConnectionType	El tipo de conexión. Puede ser `Websocket` , `HTTP_POST` , `HTTP_GET` o `Bolt`.

ARN del IAM del intermediario	El ARN del rol de IAM o usuario de IAM utilizado para firmar la solicitud. Vacío si la autenticación de IAM está deshabilitada. Su formato es el siguiente:
	`arn:`*`partition`*` :`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
	Por ejemplo:
	`arn:aws:iam::123456789012:user/Anna`
	`arn:aws:sts::123456789012:assumed-role/AWSNeptuneNotebookRole/SageMaker`

Campo	Descripción
Contexto de autenticación	Contiene un objeto JSON serializado que tiene información de autenticación. El campo `authenticationSucceeded` es `True` si el usuario se ha autentica do. Vacío si la autenticación de IAM está deshabilitada.
HttpHeader	La información del encabezado HTTP. Puede contener una consulta. WebSocket Conexiones de empalme y atornillado vacías.
Carga	La consulta de Gremlin, SPARQL u openCypher.

Publicación de Neptune Logs en Amazon Logs CloudWatch

Puede configurar un clúster de base de datos de Neptune para publicar datos de registro de auditoría o datos de registro de consultas lentas en un grupo de registros de Amazon Logs. CloudWatch Con CloudWatch Logs, puede realizar un análisis en tiempo real de los datos de registro y utilizarlos CloudWatch para crear alarmas y ver métricas. Puede usar CloudWatch los registros para almacenar sus registros en un almacenamiento de alta durabilidad.

Para publicar los registros de auditoría en CloudWatch Logs, los registros de auditoría deben estar habilitados de forma explícita (consulteHabilitación de los logs de auditoría). Del mismo modo, para publicar registros de consultas lentas en Logs, los CloudWatch registros de consultas lentas deben estar habilitados de forma explícita (consulte). Uso del registro de consultas lentas de Amazon Neptune

> **Note**
>
> Tenga en cuenta lo siguiente:
>
> - Se aplican cargos adicionales al publicar registros en. CloudWatch Consulta la página CloudWatch de precios para obtener más información.
> - No puedes publicar registros en CloudWatch Logs de la región de China (Pekín) o China (Ningxia).
> - Si la exportación de datos de registros está deshabilitada, Neptune no elimina los grupos de registros o flujos de registros. Si la exportación de datos de registro está desactivada, los datos de registro existentes permanecerán disponibles en CloudWatch los registros,

en función de la conservación de los registros, y se seguirán cobrando por los datos de los registros de auditoría almacenados. Puede eliminar flujos de registros y grupos de registros mediante la consola de CloudWatch registros AWS CLI, la API de registros o la API de CloudWatch registros.

Uso de la consola para publicar registros de Neptune en registros CloudWatch

Para publicar los registros de Neptune en Logs desde la consola CloudWatch

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos de Neptune para el que desee publicar los datos de registro.
4. Para Actions (Acciones), elija Modify (Modificar).
5. En la sección Exportaciones de registros, elija los registros que desee empezar a publicar en CloudWatch Logs.
6. Elija Continue (Continuar), seguido de Modify DB Cluster (Modificar clúster de base de datos) en la página de resumen.

Uso de la CLI para publicar los registros de auditoría de Neptune en Logs CloudWatch

Puede crear un nuevo clúster de base de datos que publique los registros de auditoría en CloudWatch Logs mediante el AWS CLI `create-db-cluster` comando con los siguientes parámetros:

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["audit"]'
```

Puede configurar un clúster de base de datos existente para publicar los registros de auditoría en CloudWatch Logs mediante el AWS CLI `modify-db-cluster` comando con los siguientes parámetros:

```
aws neptune modify-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Uso de la CLI para publicar registros de consultas lentas de Neptune en Logs CloudWatch

También puede crear un nuevo clúster de base de datos que publique registros de consultas lentas en Logs mediante el AWS CLI `create-db-cluster` comando con los siguientes parámetros: CloudWatch

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["slowquery"]'
```

Del mismo modo, puede configurar un clúster de base de datos existente para publicar registros de consultas lentas en Logs mediante el AWS CLI `modify-db-cluster` comando con los siguientes parámetros: CloudWatch

```
aws neptune modify-db-cluster --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["slowquery"]}'
```

Supervisión de eventos de Neptune Log en Amazon CloudWatch

Tras habilitar los registros de Neptune, puede supervisar los eventos de registro en Amazon CloudWatch Logs. Se crea un nuevo grupo de registro automáticamente para el clúster de base de datos de Neptune con el siguiente prefijo, donde *`cluster-name`* representa el nombre del clúster de base de datos y *`log_type`* representa el tipo de registro:

```
/aws/neptune/cluster-name/log_type
```

Por ejemplo, si configura la función de exportación para que incluya el log de auditoría para un clúster de base de datos denominado `mydbcluster`, los datos del log se almacenan en el grupo de logs `/aws/neptune/mydbcluster/audit`.

Todos los eventos de todas las instancias de base de datos en un clúster de base de datos se envían a un grupo de logs utilizando flujos de logs distintos.

Si ya existe un grupo de registros con el nombre especificado, Neptune utilizará dicho grupo de registros para exportar los datos de registros para el clúster de base de datos de Neptune. Puede utilizar la configuración automatizada, por ejemplo AWS CloudFormation, para crear grupos de registros con períodos de retención de registros predefinidos, filtros de métricas y acceso de clientes. De lo contrario, se crea automáticamente un nuevo grupo de registros utilizando el período de retención de registros predeterminado, «Nunca caduca», en CloudWatch los registros.

Puede usar la consola de CloudWatch registros AWS CLI, la API de registros o la API de CloudWatch registros para cambiar el período de retención de registros. Para obtener más información sobre cómo cambiar los períodos de retención de CloudWatch registros en los registros, consulte Cambiar la retención de datos de registro en CloudWatch los registros.

Puede utilizar la consola de CloudWatch registros AWS CLI, la API de registros o la API de CloudWatch registros para buscar información en los eventos de registro de un clúster de base de datos. Para obtener más información sobre la búsqueda y el filtrado de datos de registro, consulte Búsqueda y filtrado de datos de registros.

Habilitación de Amazon CloudWatch Logs para una libreta Neptune

CloudWatch Los registros de los cuadernos de Neptune están deshabilitados de forma predeterminada. Siga estos pasos para habilitarlos, con fines de depuración u otros fines:

Uso del AWS Management Console para habilitar los CloudWatch registros de un cuaderno de Neptune

1. Abre la SageMaker consola de Amazon en https://console.aws.amazon.com/sagemaker/.
2. En el panel de navegación de la izquierda, seleccione Cuaderno y, a continuación, Instancias de cuaderno. Busque el nombre del cuaderno de Neptune para el que desea habilitar los registros.
3. Vaya a la página de detalles y elija el nombre de la instancia de cuaderno que se menciona en el paso anterior.
4. Si la instancia del cuaderno se está ejecutando, seleccione el botón Detener, situado en la parte superior derecha de la página de detalles del cuaderno.

5. En Permisos y cifrado hay un campo para el ARN del rol de IAM. Seleccione el enlace de este campo para ir al rol de IAM de este cuaderno.
6. Cree la política siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogDelivery",
        "logs:Describe*",
        "logs:GetLogDelivery",
        "logs:GetLogEvents",
        "logs:ListLogDeliveries",
        "logs:PutLogEvents",
        "logs:PutResourcePolicy",
        "logs:UpdateLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

7. Guarde esta nueva política y asóciela al rol de IAM del paso 8.
8. Selecciona Iniciar en la parte superior derecha de la página de detalles de la instancia del SageMaker bloc de notas.
9. Cuando los registros comiencen a fluir, debería ver el enlace Ver registros debajo del campo denominado Configuración del ciclo de vida, cerca de la parte inferior izquierda de la sección Configuración de instancias de cuaderno de la página de detalles.

Si el bloc de notas no se inicia, aparecerá un mensaje en la página de detalles del bloc de notas de la SageMaker consola que indica que la instancia del bloc de notas ha tardado más de 5 minutos en iniciarse. Los CloudWatch registros relacionados con este problema se encuentran bajo el nombre:*(your-notebook-name)*`/LifecycleConfigOnStart`.

Consulte Registrar SageMaker eventos de Amazon con Amazon CloudWatch para obtener más información, si es necesario.

Uso del registro de consultas lentas de Amazon Neptune

La identificación, depuración y optimización de una consulta de ejecución lenta puede resultar difícil. Cuando el registro de consultas lentas de Neptune está habilitado, los atributos de todas las consultas de larga duración se registran automáticamente para facilitar este proceso.

> **Note**
>
> El registro de consultas lentas se introdujo en la versión 1.2.1.0 del motor de Neptune.

Para habilitar el registro de consultas lentas, utilice el parámetro de clúster de base de datos neptune_enable_slow_query_log. Este parámetro está establecido en `disabled` de forma predeterminada. Al establecerlo en `info` o `debug`, se habilita el registro de consultas lentas. La configuración de `info` registra algunos atributos útiles de cada consulta de ejecución lenta, mientras que la configuración de `debug` registra todos los atributos disponibles.

Para establecer el umbral de lo que se considera una consulta de ejecución lenta, utilice el parámetro de clúster de base de datos neptune_slow_query_log_threshold para especificar el número de milisegundos tras los cuales una consulta en ejecución se considera lenta y se registra cuando el registro de consultas lentas está activado. El valor predeterminado es de 5000 milisegundos (5 segundos).

Puede configurar estos parámetros del clúster de base de datos en el AWS Management Console comando AWS CLImodify-db-cluster-parameter-group o mediante la función de administración de grupos ModifyDB. ClusterParameter

> **Note**
>
> Los parámetros de registro de consultas lentas son dinámicos, lo que significa que cambiar sus valores no requiere ni provoca el reinicio del clúster de base de datos.

Para ver los registros de consultas lentas en el AWS Management Console

Puede ver y descargar los registros de consultas lentas en el, de la AWS Management Console siguiente manera:

En la página Instancias, elija la instancia de base de datos y, a continuación, desplácese hasta la sección Registros. A continuación, puede seleccionar un archivo de registro allí y luego elegir Descargar para descargarlo.

Los archivos generados por el registro de consultas lentas de Neptune

Los archivos de registro generados por el registro de consultas lentas en Neptune tienen las siguientes características:

- Los archivos están codificados en UTF-8.
- Las consultas y sus atributos se registran en formato JSON.
- Los atributos nulos y vacíos no se registran, excepto los datos `queryTime`.
- Los registros abarcan varios archivos, cuyo número varía en función del tamaño de la instancia.
- Las entradas del registro no están en orden secuencial. Puede usar sus valores `timestamp` para ordenarlas.
- Para ver los eventos más recientes, es posible que sea necesario revisar todos los archivos de consultas lentas.
- Los archivos de registro se rotan cuando llegan a 100 MiB combinados. Este límite no se puede configurar.

Atributos de consultas registrados en el modo **`info`**

Los siguientes atributos se registran para las consultas lentas cuando el parámetro del clúster de base de datos `neptune_enable_slow_query_log` se establece en `info`:

Grupo	Atributo	Descripción
solicitud ResponseMetadata	`requestId`	Identificador de solicitud de la consulta.
	`requestType`	Tipo de solicitud, como HTTP o WebSocket.

Grupo	Atributo	Descripción
	`responseStatusCode`	Código de estado de respuesta de consulta, como 200.
	`exceptionClass`	Clase de excepción del error devuelto tras la ejecución de la consulta.
queryStats	`query`	Cadena de consulta.
	`queryFingerprint`	Huella digital de la consulta.
	`queryLanguage`	Lenguaje de consulta, como Gremlin, SPARQL u openCypher.
memoryStats	`allocatedPermits`	Permisos asignados a la consulta.
	`approximateUsedMemoryBytes`	Memoria aproximada utilizada por la consulta durante la ejecución.
queryTime	`startTime`	Hora de inicio de la consulta (UTC).
	`overallRunTimeMs`	Tiempo total de la consulta en milisegundos.
	`parsingTimeMs`	Tiempo de análisis de la consulta en milisegundos.
	`waitingTimeMs`	Tiempo de espera en cola de consultas de Gremlin/SPARQL/openCypher, en milisegundos

Grupo	Atributo	Descripción
	`executionTimeMs`	Tiempo de ejecución de la consulta en milisegundos.
	`serializationTimeMs`	Tiempo de serialización de la consulta en milisegundos.
statementCounters	`scanned`	Número de instrucciones escaneadas.
	`written`	Número de instrucciones escritas.
	`deleted`	Número de instrucciones eliminadas.
transactionCounters	`committed`	Número de transacciones confirmadas.
	`rolledBack`	Número de transacciones revertidas.
vertexCounters	`added`	Número de vértices añadidos.
	`removed`	Número de vértices eliminados.
	`propertiesAdded`	Número de propiedades de vértices añadidas.
	`propertiesRemoved`	Número de propiedades de vértices eliminadas.
edgeCounters	`added`	Número de bordes añadidos.
	`removed`	Número de bordes eliminados.
	`propertiesAdded`	Número de propiedades de borde añadidas.

Grupo	Atributo	Descripción
	`propertiesRemoved`	Número de propiedades de borde eliminadas.
resultCache	`hitCount`	Número de aciertos de la caché de resultados.
	`missCount`	Número de errores de la caché de resultados.
	`putCount`	Número de colocaciones en la caché de resultados.
concurrentExecution	`acceptedQueryCountAtStart`	Se aceptan consultas paralelas con la ejecución de la consulta actual al inicio.
	`runningQueryCountAtStart`	Consultas paralelas que se ejecutan con la ejecución de la consulta actual al inicio.
	`acceptedQueryCountAtEnd`	Se aceptan consultas paralelas con la ejecución de la consulta actual al final.
	`runningQueryCountAtEnd`	Consultas paralelas que se ejecutan con la ejecución de la consulta actual al final.
queryBatch	`queryProcessingBatchSize`	Tamaño del lote durante el procesamiento de consultas.
	`querySerialisationBatchSize`	Tamaño del lote durante la serialización de consultas.

Atributos de consultas registrados en el modo **debug**

Cuando el parámetro del clúster de base de datos `neptune_enable_slow_query_log` se ha establecido en `debug`, se registran los siguientes atributos del contador de almacenamiento además de los atributos que se registran en el modo `info`:

Atributo	Descripción
`statementsScannedInAllIndexes`	Las instrucciones se escanean en todos los índices.
`statementsScannedSPOGIndex`	Instrucciones escaneadas en el índice SPOG.
`statementsScannedPOGSIndex`	Instrucciones escaneadas en el índice POGS.
`statementsScannedGPSOIndex`	Instrucciones escaneadas en el índice GPSO.
`statementsScannedOSGPIndex`	Instrucciones escaneadas en el índice OSGP.
`statementsScannedInChunk`	Instrucciones escaneadas juntas en fragmento s.
`postFilteredStatementScans`	Las instrucciones que quedan en el posfiltrado después de escanearlas.
`distinctStatementScans`	Se han escaneado distintas instrucciones.
`statementsReadInAllIndexes`	Las instrucciones leídas en el posfiltrado después de escanearlas en todos los índices.
`statementsReadSPOGIndex`	Las instrucciones leídas en el posfiltrado después de escanearlas en el índice SPOG.
`statementsReadPOGSIndex`	Las instrucciones leídas en el posfiltrado después de escanearlas en el índice POGS.
`statementsReadGPSOIndex`	Las instrucciones leídas en el posfiltrado después de escanearlas en el índice GPSO.

Atributo	Descripción
`statementsReadOSGPIndex`	Las instrucciones leídas en el posfiltrado después de escanearlas en el índice OSGP.
`accessPathSearches`	Número de búsquedas de rutas de acceso.
`fullyBoundedAccessPathSearches`	Número de búsquedas de rutas de acceso clave totalmente delimitadas.
`accessPathSearchedByPrefix`	Número de rutas de acceso buscadas por prefijo.
`searchesWhereRecordsWereFound`	Número de búsquedas que tenían 1 o más registros como salida.
`searchesWhereRecordsWereNotFound`	Número de búsquedas que no tenían registros como salida.
`totalRecordsFoundInSearches`	Total de registros encontrados en todas las búsquedas.
`statementsInsertedInAllIndexes`	Número de instrucciones insertadas en todos los índices.
`statementsUpdatedInAllIndexes`	Número de instrucciones actualizadas en todos los índices.
`statementsDeletedInAllIndexes`	Número de instrucciones eliminadas en todos los índices.
`predicateCount`	Número de predicados.
`dictionaryReadsFromValueToIdTable`	Número de lecturas del diccionario desde el valor hasta la tabla de identificadores.
`dictionaryReadsFromIdToValueTable`	Número de lecturas del diccionario de la tabla de identificadores de valores.

Atributo	Descripción
`dictionaryWritesToValueToIdTable`	Número de escrituras del diccionario en el valor de la tabla de identificadores.
`dictionaryWritesToIdToValueTable`	Número de escrituras del diccionario en el identificador de la tabla de valores.
`rangeCountsInAllIndexes`	Número de recuentos de rangos en todos los índices.
`deadlockCount`	Número de interbloqueos en la consulta.
`singleCardinalityInserts`	Número de inserciones de cardinalidad única realizadas.
`singleCardinalityInsertDeletions`	Número de instrucciones eliminadas durante la inserción de cardinalidad única.

Ejemplo de registro de depuración para una consulta lenta

La siguiente consulta de Gremlin podría tardar más en ejecutarse que el umbral establecido para las consultas lentas:

```
gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by('
```

En ese caso, si el registro de consultas lentas estuviera habilitado en el modo de depuración, se registrarían los siguientes atributos para la consulta, de una forma como esta:

```
{
  "requestResponseMetadata": {
    "requestId": "5311e493-0e98-457e-9131-d250a2ce1e12",
    "requestType": "HTTP_GET",
    "responseStatusCode": 200
  },
  "queryStats": {
    "query":
 "gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by
    "queryFingerprint":
 "g.V().has(string0,string1).repeat(__.out().simplePath()).until(__.has(string0,string2)).path(
```

```
    "queryLanguage": "Gremlin"
  },
  "memoryStats": {
    "allocatedPermits": 20,
    "approximateUsedMemoryBytes": 14838
  },
  "queryTimeStats": {
    "startTime": "23/02/2023 11:42:52.657",
    "overallRunTimeMs": 2249,
    "executionTimeMs": 2229,
    "serializationTimeMs": 13
  },
  "statementCounters": {
    "read": 69979
  },
  "transactionCounters": {
    "committed": 1
  },
  "concurrentExecutionStats": {
    "acceptedQueryCountAtStart": 1
  },
  "queryBatchStats": {
    "queryProcessingBatchSize": 1000,
    "querySerialisationBatchSize": 1000
  },
  "storageCounters": {
    "statementsScannedInAllIndexes": 69979,
    "statementsScannedSPOGIndex": 44936,
    "statementsScannedPOGSIndex": 4,
    "statementsScannedGPSOIndex": 25039,
    "statementsReadInAllIndexes": 68566,
    "statementsReadSPOGIndex": 43544,
    "statementsReadPOGSIndex": 2,
    "statementsReadGPSOIndex": 25020,
    "accessPathSearches": 27,
    "fullyBoundedAccessPathSearches": 27,
    "dictionaryReadsFromValueToIdTable": 10,
    "dictionaryReadsFromIdToValueTable": 17,
    "rangeCountsInAllIndexes": 4
  }
}
```

Registro de llamadas a la API de Amazon Neptune con AWS CloudTrail

Amazon Neptune está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en Neptune. CloudTrail captura las llamadas a la API de Neptune como eventos, incluidas las llamadas desde la consola de Neptune y las llamadas de código a las API de Neptune.

CloudTrail solo registra los eventos de las llamadas a la API de administración de Neptune, como la creación de una instancia o un clúster. Si desea auditar los cambios en el gráfico, puede utilizar registros de auditoría. Para obtener más información, consulte Uso de registros de auditoría con un clúster de Amazon Neptune.

> **Important**
>
> Las llamadas a la consola y a la API de Amazon Neptune se registran como llamadas realizadas a la API de Amazon Relational Database Service (Amazon RDS). AWS CLI

Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Neptune. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Neptune, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la Guía del AWS CloudTrail usuario.

Información sobre Neptune en CloudTrail

CloudTrail está activado en su AWS cuenta al crear la cuenta. Cuando se produce actividad en Amazon Neptune, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar los eventos recientes en su AWS cuenta. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para tener un registro continuo de los eventos en tu AWS cuenta, incluidos los eventos de Neptune, crea un sendero. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De manera predeterminada, cuando crea un registro de seguimiento en la consola, el registro de

seguimiento se aplica a todas las regiones. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para obtener más información, consulte:

- Introducción a la creación de registros de seguimiento
- CloudTrail Integraciones y servicios compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Si se realiza una acción en nombre de su AWS cuenta mediante la consola de Neptune, la interfaz de línea de comandos de Neptune o las API del SDK de Neptune, AWS CloudTrail registra la acción como llamadas realizadas a la API de Amazon RDS. Por ejemplo, si utiliza la consola de Neptune para modificar una instancia de base de datos o llama al comando AWS CLImodify-db-instance, el AWS CloudTrail registro muestra una llamada a la acción ModifyDBInstance de la API de Amazon RDS. Para obtener una lista de las acciones de la API de Neptune que se registran AWS CloudTrail, consulta la Referencia de la API de Neptune.

> **Note**
>
> AWS CloudTrail solo registra los eventos de las llamadas a la API de administración de Neptune, como la creación de una instancia o un clúster. Si desea auditar los cambios en el gráfico, puede utilizar registros de auditoría. Para obtener más información, consulte Uso de registros de auditoría con un clúster de Amazon Neptune.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales raíz o del usuario de IAM.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el elemento CloudTrail UserIdentity.

Descripción de las entradas de archivos de registros de Neptune

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

El siguiente ejemplo muestra un CloudTrail registro de un usuario que creó una instantánea de una instancia de base de datos y, a continuación, la eliminó mediante la consola Neptune. La consola se identifica mediante el elemento `userAgent`. Las llamadas a la API realizadas por la consola (`CreateDBSnapshot` y `DeleteDBInstance`) se encuentran en el elemento `eventName` de cada registro. Encontrará la información sobre el usuario (`Alice`) en el elemento `userIdentity`.

```
{
  Records:[
  {
    "awsRegion":"us-west-2",
    "eventName":"CreateDBSnapshot",
    "eventSource":"",
    "eventTime":"2014-01-14T16:23:49Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
```

```
    }
  },
  {
    "awsRegion":"us-west-2",
    "eventName":"DeleteDBInstance",
    "eventSource":"",
    "eventTime":"2014-01-14T16:28:27Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-
kaos_fleet-1108-prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  }
  ]
}
```

Uso de las notificaciones de eventos de Neptune

Temas

- Categorías y mensajes de eventos de Amazon Neptune
- Suscripción a la notificación de eventos de Neptune
- Administración de suscripciones a notificaciones de eventos de Neptune

Amazon Neptune utiliza Amazon Simple Notification Service (Amazon SNS) para proporcionar notificaciones cuando se produce un evento de Neptune. Estas notificaciones pueden estar en cualquier formato compatible con Amazon SNS para una AWS región, como un correo electrónico, un mensaje de texto o una llamada a un punto de enlace HTTP.

Neptune agrupa estos eventos en categorías a las que puede suscribirse para recibir una notificación cada vez que se produzca un evento en esa categoría. Puede suscribirse a una categoría de eventos para una instancia de base de datos, un clúster de base de datos, una instantánea de base de datos, una instantánea de clúster de base de datos, o para un grupo de parámetros de base de datos. Por ejemplo, si se suscribe a la categoría Backup de una instancia de base de datos determinada, recibe una notificación cada vez que se produzca un evento relacionado con las copias de seguridad que afecte a la instancia de base de datos. También recibirá una notificación cuando cambie una suscripción de notificación de eventos.

Los eventos se producen en el clúster de base de datos y en el nivel de instancia de base de datos, por lo que puede recibir eventos si se suscribe a un clúster de base de datos o a una instancia de base de datos.

Las notificaciones de eventos se envían a las direcciones que se proporcionan al crear la suscripción. Es posible que le interese crear distintas suscripciones como, por ejemplo, una que reciba todas las notificaciones de eventos y otra que incluya únicamente los eventos críticos para las instancias de base de datos de producción. Puede desactivar fácilmente las notificaciones sin eliminar una suscripción. Para ello, defina el botón de opción Activado en No en la consola de Neptune.

> **Important**
>
> Amazon Neptune no garantiza el orden de los eventos enviados en una secuencia de eventos. El orden de los eventos está sujeto a cambio.

Neptune utiliza el nombre de recurso de Amazon (ARN) de un tema de Amazon SNS para identificar cada suscripción. La consola de Neptune crea el ARN automáticamente cuando se crea la suscripción.

La facturación de notificaciones de eventos de Neptune se efectúa a través de Amazon SNS. Se aplican las tarifas de Amazon SNS cuando se utiliza la notificación de eventos. Para obtener más información, consulte los precios de Amazon Simple Notification Service.

Categorías y mensajes de eventos de Amazon Neptune

Neptune genera un número significativo de eventos en categorías a las que puede suscribirse a través de la consola de Neptune. Cada categoría se aplica a un tipo de origen, que puede ser una instancia de base de datos, una instantánea de base de datos o un grupo de parámetros de base de datos.

> **Note**
>
> Neptune utiliza las definiciones de eventos e identificadores de Amazon RDS existentes.

Eventos de Neptune que se originan en instancias de base de datos

En la siguiente tabla, se muestra una lista de los eventos, por categoría de evento, que pueden producirse cuando el tipo de origen es una instancia de base de datos.

Categoría	ID de evento de Amazon RDS	Descripción
availability	RDS-EVENT-0006	Instancia de base de datos reiniciada.
	RDS-EVENT-0004	Instancia de base de datos cerrada.
	RDS-EVENT-0022	Se ha producido un error al reiniciar el motor de Neptune.
backup	RDS-EVENT-0001	Se está realizando una copia de seguridad de la instancia de base de datos.
	RDS-EVENT-0002	Copia de seguridad de instancia de base de datos finalizada.

Categoría	ID de evento de Amazon RDS	Descripción
configuration change	RDS-EVENT-0009	La instancia de base de datos se ha añadido a un grupo de seguridad.
	RDS-EVENT-0024	La instancia de base de datos se está convirtiendo en una instancia de base de datos Multi-AZ.
	RDS-EVENT-0030	La instancia de base de datos se está convirtiendo en una instancia de base de datos Single-AZ.
	RDS-EVENT-0012	Se está aplicando la modificación a la clase de instancia de base de datos.
	RDS-EVENT-0018	Se está modificando la configuración de almacenamiento actual de esta instancia de base de datos.
	RDS-EVENT-0011	Se ha modificado un grupo de parámetros de esta instancia de base de datos.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0092	Ha finalizado la actualización de un grupo de parámetros de esta instancia de base de datos.
	RDS-EVENT-0028	Se han desactiva do las copias de seguridad automátic as en esta instancia de base de datos.
	RDS-EVENT-0032	Se han activado las copias de seguridad automáticas en esta instancia de base de datos.
	RDS-EVENT-0025	La instancia de base de datos se ha convertido en una instancia de base de datos Multi-AZ.
	RDS-EVENT-0029	La instancia de base de datos se ha convertido en una instancia de base de datos Single-AZ.
	RDS-EVENT-0014	Se ha modificado la clase de instancia de base de datos de esta instancia de base de datos.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0017	Se ha modificado la configuración de almacenamiento actual de esta instancia de base de datos.
	RDS-EVENT-0010	La instancia de base de datos se ha eliminado de un grupo de seguridad.
creación	RDS-EVENT-0005	Instancia de base de datos creada.
deletion	RDS-EVENT-0003	Se ha eliminado la instancia de base de datos.
failover	RDS-EVENT-0034	Neptune no está intentando realizar la conmutación por error solicitada porque recientemente se ha producido una conmutación por error en la instancia de base de datos.
	RDS-EVENT-0013	Se ha iniciado una conmutación por error Multi-AZ que ha dado como resultado la promoción de una instancia en espera.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0015	Ha finalizado una conmutación por error Multi-AZ que ha dado como resultado la promoción de una instancia en espera. El DNS puede tardar varios minutos en realizar la transfere ncia a la nueva instancia de base de datos principal.
	RDS-EVENT-0065	La instancia se ha recuperado de una conmutación por error parcial.
	RDS-EVENT-0049	Ha finalizado una conmutación por error Multi-AZ.
	RDS-EVENT-0050	Se ha iniciado una activación Multi-AZ después de una recuperación correcta de la instancia.
	RDS-EVENT-0051	Ha finalizado una activación Multi-AZ. La base de datos ya debería estar accessible.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0031	Se ha producido un error en la instancia de base de datos debido a una configuración incompatible o a un problema de almacenamiento subyacente. Inicie una point-in-time-restore para la instancia de base de datos.
	RDS-EVENT-0036	La instancia de base de datos está en una red incompatible. Algunos de los ID de subred especificados no son válidos o no existen.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0035	La instancia de base de datos tiene parámetros no válidos. Por ejemplo, no se ha podido iniciar la instancia de base de datos porque un parámetro relacionado con la memoria tiene un valor demasiado alto para esta clase de instancia, por lo que el cliente debería modificar el parámetro de memoria y reiniciar la instancia de base de datos.
	RDS-EVENT-0082	Neptune no ha podido copiar los datos de copia de seguridad de un bucket de Amazon S3. Es probable que los permisos que tiene Neptune para obtener acceso al bucket de Amazon S3 se hayan configurado incorrectamente.

Categoría	ID de evento de Amazon RDS	Descripción
low storage	RDS-EVENT-0089	La instancia de base de datos ha consumido más del 90 % del almacenamiento asignado. El espacio de almacenamiento de una instancia de base de datos se puede monitorizar con la métrica Free Storage Space.
	RDS-EVENT-0007	Se ha agotado el almacenamiento asignado a la instancia de base de datos. Para solucionar este problema, debe asignar almacenamiento adicional a la instancia de base de datos.
maintenance	RDS-EVENT-0026	Se está realizando el mantenimiento sin conexión de la instancia de base de datos. La instancia de base de datos no está disponible en este momento.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0027	Ha finalizado el mantenimiento sin conexión de la instancia de base de datos. La instancia de base de datos ya está disponible.
	RDS-EVENT-0047	Ha finalizado la aplicación de parches a la instancia de base de datos.
notification	RDS-EVENT-0044	Notificación emitida por el operador. Para obtener más informaci ón, consulte el mensaje del evento.
	RDS-EVENT-0048	Se ha retrasado la aplicación de parches a la instancia de base de datos.
	RDS-EVENT-0087	Se ha detenido la instancia de base de datos.
	RDS-EVENT-0088	Se ha iniciado la instancia de base de datos.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0154	La instancia de base de datos se está iniciando debido a que se supera el tiempo máximo permitido para estar detenida.
	RDS-EVENT-0158	La instancia de base de datos se encuentra en un estado que no se puede actualizar.
	RDS-EVENT-0173	Se ha parcheado la instancia de base de datos.
read replica	RDS-EVENT-0045	Se ha producido un error en el proceso de replicación de una réplica de lectura. Para obtener más información, consulte el mensaje del evento.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0046	La réplica de lectura ha reanudado la replicación. Este mensaje aparece cuando se crea por primera vez una réplica de lectura o como mensaje de monitoreo que confirma que la replicación funciona correctamente. Si este mensaje es posterior a una notificación RDS-EVENT-0045, significa que la replicación se ha reanudado después de que se detuvo o tras producirse un error.
	RDS-EVENT-0057	La replicación de la réplica de lectura ha terminado.
	RDS-EVENT-0062	La replicación de la réplica de lectura se ha detenido manualmente.
	RDS-EVENT-0063	La replicación de la réplica de lectura se ha restablecido.

Categoría	ID de evento de Amazon RDS	Descripción
recovery	RDS-EVENT-0020	Se ha iniciado la recuperación de la instancia de base de datos. El tiempo de recuperación dependerá de la cantidad de datos que deban recuperarse.
	RDS-EVENT-0021	Ha finalizado la recuperación de la instancia de base de datos.
	RDS-EVENT-0023	Se ha solicitado una copia de seguridad manual, pero Neptune está creando una instantánea de base de datos. Envíe de nuevo la solicitud cuando Neptune haya terminado la instantánea de base de datos.
	RDS-EVENT-0052	Se ha iniciado la recuperación de la instancia Multi-AZ. El tiempo de recuperación dependerá de la cantidad de datos que deban recuperarse.

Categoría	ID de evento de Amazon RDS	Descripción
	RDS-EVENT-0053	Ha finalizado la recuperación de la instancia Multi-AZ.
restoration	RDS-EVENT-0008	La instancia de base de datos se ha restaurado a partir de una instantánea de base de datos.
	RDS-EVENT-0019	La instancia de base de datos se restauró a partir de una point-in-time copia de seguridad.

Eventos de Neptune que se originan en un clúster de base de datos

En la siguiente tabla se muestra una lista de los eventos, por categoría de evento, que pueden producirse cuando el tipo de origen es un clúster de base de datos.

Categoría	ID de evento de RDS	Descripción
conmutación por error	RDS-EVENT-0069	Ha fallado la conmutación por error de un clúster de bases de datos.
	RDS-EVENT-0070	Se ha reiniciado la conmutación por error de un clúster de bases de datos.
	RDS-EVENT-0071	Ha finalizado la conmutación por

Categoría	ID de evento de RDS	Descripción
		error de un clúster de bases de datos.
	RDS-EVENT-0072	Se ha iniciado una conmutación por error para el clúster de bases de datos dentro de la misma zona de disponibilidad.
	RDS-EVENT-0073	Se ha iniciado una conmutación por error para el clúster de bases de datos entre zonas de disponibi lidad distintas.
	RDS-EVENT-0083	Neptune no ha podido copiar los datos de copia de seguridad de un bucket de Amazon S3. Es probable que los permisos que tiene Neptune para obtener acceso al bucket de Amazon S3 se hayan configurado incorrectamente.
maintenance	RDS-EVENT-0156	El clúster de bases de datos tiene disponible una actualización de versiones secundarias de motor de base de datos.

Categoría	ID de evento de RDS	Descripción
notification	RDS-EVENT-0076	La migración a un cluster de base de datos de Neptune ha fallado.
	RDS-EVENT-0077	Ha fallado un intento de convertir una tabla de la base de datos de origen al formato de la base de datos durante la migración a un clúster de base de datos de Neptune.
	RDS-EVENT-0150	El clúster de bases de datos se ha detenido
	RDS-EVENT-0151	El clúster de bases de datos se ha iniciado
	RDS-EVENT-0152	La detención del clúster de bases de datos ha producido un error.
	RDS-EVENT-0153	El clúster de bases de datos se está iniciando debido a que se supera el tiempo máximo permitido para estar detenida.

Eventos de Neptune que se originan en una instantánea de clúster de base de datos

En la siguiente tabla, se muestra la categoría de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es una instantánea de clúster de base de datos de Neptune.

Categoría	ID de evento de RDS	Descripción
backup	RDS-EVENT-0074	Se ha iniciado la creación de una instantánea manual de clúster de bases de datos.
backup	RDS-EVENT-0075	Se ha creado una instantánea manual de clúster de bases de datos.
notification	RDS-EVENT-0162	Error en la tarea de exportación de instantánea del clúster de bases de datos.
notification	RDS-EVENT-0163	Se ha cancelado la tarea de exportaci ón de instantánea de clúster de bases de datos.
notification	RDS-EVENT-0164	Se ha completado la tarea de exportaci ón de instantánea de clúster de bases de datos.

Categoría	ID de evento de RDS	Descripción
backup	RDS-EVENT-0168	Creación de instantáneas de clúster automatizadas.
backup	RDS-EVENT-0169	Se ha creado una instantánea de clúster automatizada.
creation	RDS-EVENT-0170	Se ha creado un clúster de bases de datos.
deletion	RDS-EVENT-0171	Se ha eliminado un clúster de base de datos.
notification	RDS-EVENT-0172	Se ha cambiado el nombre del clúster de bases de datos de [nombre antiguo del clúster de bases de datos] a [nombre nuevo del clúster de bases de datos].

Eventos de Neptune que se originan en el grupo de parámetros del clúster de base de datos

En la siguiente tabla se muestra la categoría de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es un grupo de parámetros de clúster de base de datos.

Categoría	ID de evento de RDS	Descripción
configuration change	RDS-EVENT-0037	El grupo de parámetros se ha modificado.

Eventos de Neptune que se originan en un grupo de seguridad

En la siguiente tabla se muestran las categorías de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es un grupo de seguridad.

Categoría	ID de evento de RDS	Descripción
configuration change	RDS-EVENT-0038	El grupo de seguridad se ha modificado.
failure	RDS-EVENT-0039	El grupo de seguridad propiedad de [usuario] no existe; se ha revocado la autorizac ión para el grupo de seguridad.

Suscripción a la notificación de eventos de Neptune

Puede utilizar la consola de Neptune para suscribirse a las notificaciones de eventos de la siguiente manera:

Para suscribirse a una notificación de eventos de Neptune

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos).
3. En la página Event Subscriptions (Suscripciones de eventos) seleccione Create Event Subscription (Crear suscripción de eventos).
4. En el cuadro de diálogo Create Event Subscription (Crear suscripción de eventos), haga lo siguiente:

    a. En Name (Nombre) escriba un nombre para la suscripción de notificación de evento.

    b. Para Send notifications to (Enviar notificaciones a), elija un ARN de Amazon SNS existente correspondiente a un tema de Amazon SNS o elija create topic (crear tema) para escribir el nombre de un tema y una lista de destinatarios.

    c. En Source type (Tipo de origen) elija un tipo de origen.

d. Elija Yes (Sí) para activar la suscripción. Si desea crear la suscripción, pero que todavía no envíen notificaciones, seleccione No.

e. En función del tipo de origen que haya seleccionado, seleccione las categorías y orígenes del evento de las que desea recibir notificaciones.

f. Seleccione Crear.

Administración de suscripciones a notificaciones de eventos de Neptune

Si selecciona Suscripciones a eventos en el panel de navegación de la consola de Neptune, puede ver las categorías de suscripciones y una lista de sus suscripciones actuales.

También puede modificar o eliminar una suscripción específica.

Modificación de suscripciones a notificaciones de eventos de Neptune

Para modificar sus suscripciones actuales de notificación de eventos de Neptune

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos). El panel Event subscriptions (Suscripciones de eventos) muestra todas sus suscripciones a notificaciones de eventos.
3. En el panel Event subscriptions (Suscripciones de eventos), elija la suscripción que desea modificar y elija Edit (Editar).
4. Realice los cambios que desee en la suscripción en las secciones Target (Objetivo) o Source (Fuente). Puede añadir o eliminar identificadores de origen activándolos o desactivándolos en la sección Origen.
5. Elija Editar. La consola de Neptune indica que se está modificando la suscripción.

Eliminación de una suscripción de notificación de eventos de Neptune

Puede eliminar una suscripción cuando ya no la necesite. Los suscriptores del tema dejarán de recibir notificaciones de los eventos especificados en la suscripción.

Para eliminar una suscripción de notificación de eventos de Neptune

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos).
3. En el panel Suscripciones a eventos, seleccione la suscripción que desea eliminar.
4. Elija Eliminar.
5. La consola de Neptune indica que se está eliminando la suscripción.

Etiquetado de recursos de Amazon Neptune

Puede utilizar etiquetas de Neptune para agregar metadatos a los recursos de Neptune. Además, puede usar etiquetas con políticas AWS Identity and Access Management (IAM) para administrar el acceso a los recursos de Neptune y controlar qué acciones se pueden aplicar a esos recursos. Por último, estas etiquetas se pueden utilizar para hacer un seguimiento de costos, agrupando los gastos correspondientes a los recursos que tienen etiquetas similares.

Todos los recursos de Neptune se pueden etiquetar, incluidos los siguientes:

- Instancias de base de datos
- Clústeres de base de datos
- Réplicas de lectura
- Instantáneas de base de datos
- Instantáneas de clúster de base de datos
- Suscripciones de eventos
- Grupos de parámetros de base de datos
- Grupos de parámetros de clúster de bases de datos
- Grupos de subred de base de datos

Información general de las etiquetas de recursos de Neptune

Las etiquetas de Amazon Neptune son pares de nombre-valor que el usuario define y asocia a un recurso de Neptune. El nombre es la clave. Si lo desea puede proporcionar un valor para la clave o no. También puede usar etiquetas para asignar información arbitraria a un recurso de Neptune.

Puede usar claves de etiqueta, por ejemplo, para definir una categoría, y el valor de la etiqueta puede ser un elemento dentro de esa categoría. Por ejemplo, podría definir una clave de etiqueta "proyecto" y un valor de etiqueta "Salix" para indicar que el recurso de Neptune está asignado al proyecto Salix. Asimismo, puede utilizar etiquetas para designar recursos de Neptune para pruebas o para producción mediante una clave como `environment=test` o `environment=production`. Recomendamos utilizar un conjunto coherente de claves de etiqueta que facilite el seguimiento de los metadatos asociados a los recursos de Neptune.

Utilice etiquetas para organizar su AWS factura y reflejar su propia estructura de costes. Para ello, inscríbase para recibir su Cuenta de AWS factura con los valores clave de las etiquetas incluidos. A continuación, para ver los costos de los recursos combinados, organice la información de facturación de acuerdo con los recursos con los mismos valores de clave de etiquetas. Por ejemplo, puede etiquetar varios recursos con un nombre de aplicación específico y luego organizar su información de facturación para ver el costo total de la aplicación en distintos servicios. Para obtener más información, consulte Uso de etiquetas de asignación de costos en la Guía del usuario de AWS Billing .

Cada recurso de Neptune tiene un conjunto de etiquetas con todas las etiquetas asignadas a ese recurso de Neptune. Un conjunto de etiquetas puede contener hasta 10 etiquetas, y también puede estar vacío. Si agrega una etiqueta a un recurso de Neptune con la misma clave que una etiqueta existente en el recurso, el valor nuevo sobrescribirá al antiguo.

AWS no aplica ningún significado semántico a sus etiquetas; las etiquetas se interpretan estrictamente como cadenas de caracteres. Neptune puede definir etiquetas en una instancia de base de datos u otros recursos de Neptune, con arreglo a la configuración utilizada al crear el recurso. Por ejemplo, Neptune podría agregar una etiqueta en la que se indica que una instancia de base de datos es para producción o para la realización de pruebas.

- La clave de la etiqueta es el nombre obligatorio de la etiqueta. El valor de cadena puede tener una longitud de entre 1 y 128 caracteres Unicode y no puede llevar el prefijo "`aws:`" ni "`rds:`". La cadena puede contener únicamente los siguientes caracteres del conjunto Unicode: letras, dígitos, espacio en blanco, '_', '.', '/', '=', '+', '-' (regex Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").
- El valor de etiqueta es un valor de cadena optativo en la etiqueta. El valor de cadena puede tener una longitud de entre 1 y 256 caracteres Unicode y no puede llevar el prefijo "`aws:`". La cadena puede contener únicamente los siguientes caracteres del conjunto Unicode: letras, dígitos, espacio en blanco, '_', '.', '/', '=', '+', '-' (regex Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").

Los valores no deben ser únicos dentro de un conjunto de etiquetas y también pueden ser nulos. Por ejemplo, es posible tener en un conjunto de etiquetas los pares clave-valor `project/Trinity` y `cost-center/Trinity`.

> **Note**
>
> Puede añadir una etiqueta a una instantánea. Sin embargo, la factura no reflejará esta agrupación.

Puede usar la AWS Management Console API de Neptune o la AWS CLI API de Neptune para agregar, enumerar y eliminar etiquetas en los recursos de Neptune. Al utilizar la API de Neptune AWS CLI o la API, debe proporcionar el nombre de recurso de Amazon (ARN) del recurso de Neptune con el que desee trabajar. Para obtener más información sobre cómo crear un ARN, consulte Creación de un ARN para Neptune.

Las etiquetas se almacenan en caché con fines de autorización. Por este motivo, cuando se actualizan o se agregan valores a las etiquetas de recursos de Neptune, estos pueden tardar varios minutos en estar disponibles.

Copia de etiquetas en Neptune

Cuando crea o restaura una instancia de base de datos, puede especificar que las etiquetas de dicha instancia se copien en instantáneas de la instancia de base de datos. La copia de las etiquetas garantiza que los metadatos para las instantáneas de base de datos coincidan con los de la instancia de base de datos de origen y que cualquier política de acceso para la instantánea de base de datos también coincida con la de la instancia de base de datos de origen. Las etiquetas no se copian de forma predeterminada.

Puede especificar que las etiquetas se copien en las instantáneas de base de datos para las siguientes acciones:

- Creación de una instancia de base de datos.
- Restauración de una instancia de base de datos.
- Creación de una réplica de lectura.
- Copia de una instantánea de base de datos.

> **Note**
>
> Si incluye un valor para el `--tag-key` parámetro del AWS CLI comando create-db-cluster-snapshot (o proporciona al menos una etiqueta a la CreateDBClusterSnapshot acción de la API), Neptune no copia las etiquetas de la instancia de base de datos de origen a la nueva instantánea de base de datos. Esto es válido incluso si la instancia de base de datos de origen tiene la opción `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) habilitada. Esto significa que puede crear una copia de una instancia de base de datos a partir de una instantánea de base de datos y evitar tener que agregar etiquetas que no se aplican a la instancia de base de datos nueva. Tras crear la instantánea de base de datos mediante el AWS CLI `create-db-cluster-snapshot` comando (o la acción de la API de `CreateDBClusterSnapshot` Neptune), puede añadir etiquetas tal y como se describe más adelante en este tema.

Etiquetado en Neptuno usando el AWS Management Console

El proceso para etiquetar un recurso de Amazon Neptune es similar para todos los recursos. El siguiente procedimiento muestra cómo etiquetar una instancia de base de datos de Neptune.

Para agregar una etiqueta a una instancia de base de datos

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Instancias.

   > **Note**
   >
   > Para filtrar la lista de instancias de base de datos en el panel Instances (Instancias), escriba una cadena de texto en el cuadro Filter instances (Instancias de filtro). Solo aparecen instancias de base de datos que contienen la cadena.

3. Elija la instancia de base de datos que desea etiquetar.
4. Elija Instance actions y, a continuación, See details.
5. En la sección de detalles, desplácese hasta la sección Tags (Etiquetas).
6. Elija Add (Añadir). Aparece la ventana Add tags (Añadir etiquetas).
7. Escriba un valor para Tag key (Clave de etiqueta) y Value (Valor).

8. Para añadir otra etiqueta, puede elegir Add another Tag (Añadir otra etiqueta) y escribir un valor para Tag key (Clave de etiqueta) y Value (Valor).

   Repita este paso tantas veces como sea necesario.
9. Elija Add (Añadir).

Para eliminar una etiqueta de una instancia de base de datos

1. Inicie sesión en la consola AWS de administración y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Instancias.

   > **Note**
   >
   > Para filtrar la lista de instancias de base de datos en el panel Instances (Instancias), escriba una cadena de texto en el cuadro Filter instances (Instancias de filtro). Solo aparecen instancias de base de datos que contienen la cadena.
3. Elija la instancia de base de datos que desea etiquetar.
4. Elija Instance actions y, a continuación, See details.
5. En la sección de detalles, desplácese hasta la sección Tags (Etiquetas).
6. Elija la etiqueta que desea eliminar.
7. Elija Remove (Eliminar) y, a continuación, Remove (Eliminar) en la ventana Remove tags (Eliminar etiquetas).

Etiquetado en Neptuno usando el AWS CLI

Puede utilizar la AWS CLI para agregar, enumerar o eliminar etiquetas de una instancia de base de datos de Neptune.

- Para añadir una o más etiquetas a un recurso de Neptune, utilice el AWS CLI comando. `add-tags-to-resource`
- Para enumerar las etiquetas de un recurso de Neptune, utilice el AWS CLI comando. `list-tags-for-resource`
- Para eliminar una o más etiquetas de un recurso de Neptune, utilice el AWS CLI comando. `remove-tags-from-resource`

Para obtener más información sobre cómo crear el nombre de recurso de Amazon (ARN) necesario, consulte Creación de un ARN para Neptune.

Etiquetado en Neptune usando la API

Puede utilizar la API de Neptune para agregar, enumerar o eliminar etiquetas de una instancia de base de datos.

- Para añadir una etiqueta a un recurso de Neptune, utilice la operación `AddTagsToResource`.
- Para enumerar las etiquetas asignadas a un recurso de Neptune, utilice la operación `ListTagsForResource`.
- Para eliminar etiquetas de un recurso de Neptune, utilice la operación `RemoveTagsFromResource`.

Para obtener más información acerca de cómo crear el ARN requerido, consulte Creación de un ARN para Neptune.

Cuando se trabaja con XML mediante la API de Neptune, las etiquetas utilizan el esquema siguiente:

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

La tabla siguiente proporciona una lista de las etiquetas XML permitidas y sus características. Los valores de `Key` y `Value` distinguen entre mayúsculas y minúsculas. Por ejemplo, `project=Trinity` y `PROJECT=Trinity` son dos etiquetas diferentes.

Elemento de etiquetado	Descripción
`TagSet`	Los conjuntos de etiquetas son un contenedor de todas las etiquetas asignadas a un recurso de Neptune. Solo puede haber un conjunto de etiquetas por recurso. Solo se puede trabajar con un `TagSet` mediante la API de Neptune.
`Tag`	Las etiquetas son pares clave-valor que define el usuario. En un conjunto de etiquetas puede haber entre 1 y 50 etiquetas.
`Key`	La clave es el nombre obligatorio de la etiqueta. El valor de cadena puede tener una longitud de entre 1 y 128 caracteres Unicode y no puede llevar el prefijo "`rds:`" ni "`aws:`". La cadena puede contener únicamente los siguientes caracteres del conjunto Unicode: letras, dígitos, espacio en blanco, '_', '.', '/', '=', '+', '-' (regex Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). Las claves deben ser únicas dentro de un conjunto de etiquetas. Por ejemplo, en un conjunto de etiquetas no puede haber claves iguales pero con valores diferentes, como `project/Trinity` y `project/Xanadu`.
Valor	El valor es la parte opcional de la etiqueta. El valor de cadena puede tener una longitud de entre 1 y 256 caracteres Unicode y no puede llevar el prefijo "`rds:`" ni "`aws:`". La cadena puede contener únicamente los siguientes caracteres del conjunto Unicode: letras, dígitos, espacio en blanco, '_', '.', '/', '=', '+', '-' (regex Java: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`"). Los valores no tienen que ser únicos dentro de un conjunto de etiquetas y también pueden ser nulos. Por ejemplo, es posible tener en un conjunto de etiquetas los pares clave-valor `project/Trinity` y `cost-center/Trinity`.

Uso de ARN administrativos en Amazon Neptune

Cada recurso que se crea en Amazon Web Services se identifica de forma inequívoca mediante un nombre de recurso de Amazon (ARN). Para determinadas operaciones de Amazon Neptune, es necesario identificar de forma inequívoca un recurso de Neptune mediante su ARN.

> **Important**
>
> Amazon Neptune comparte el formato de los ARN de Amazon RDS para las acciones administrativas que utilizan el Referencia de la API de administración. Los ARN administrativos de Neptune contienen `rds` y no `neptune-db`. Para ver los ARN del plano de datos que identifican recursos de datos de Neptune, consulte Specifying data resources.

Temas

Creación de un ARN para Neptune

Puede crear un ARN para un recurso de Amazon Neptune utilizando la siguiente sintaxis. Tenga en cuenta que Neptune comparte el formato de los ARN de Amazon RDS.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

En la siguiente tabla se muestra el formato que debe utilizar al crear un ARN para un tipo de recurso administrativo de Neptune.

Tipo de recurso	Formato de ARN
Instancia de base de datos	arn:aws:rds:*<region>*:*<account>* :db:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :db:my-instance-1`
Clúster de base de datos	arn:aws:rds:*<region>*:*<account>* :cluster: *<name>*

Tipo de recurso	Formato de ARN
	Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-cluster-1`
Suscripción a eventos	arn:aws:rds:*<region>*:*<account>* :es:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
Grupo de parámetros de base de datos	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Grupo de parámetros de clúster de base de datos	arn:aws:rds:*<region>*:*<account>* :cluster-pg: *<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Instantánea de clúster de bases de datos	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot: *<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-snap-20160809`

Tipo de recurso	Formato de ARN
Grupo de subred de base de datos	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet -10`

Obtención de un ARN existente en Amazon Neptune

Puede obtener el ARN de un recurso de Neptune mediante la API AWS Management ConsoleAWS CLI, AWS Command Line Interface () o Neptune.

Obtener un ARN existente mediante el AWS Management Console

Para obtener un ARN mediante la consola, vaya al recurso cuyo ARN desea obtener y consulte los detalles de ese recurso. Por ejemplo, para obtener el ARN de una instancia de base de datos, elija Instances (Instancias) en el panel de navegación y elija la instancia que desee en la lista. El ARN se encuentra en la sección Instance Details (Detalles de la instancia).

Obtener un ARN existente mediante el AWS CLI

Para usar el AWS CLI para obtener un ARN para un recurso de Neptune en particular, usa el `describe` comando para ese recurso. En la tabla siguiente se muestran cada AWS CLI comando y la propiedad ARN que se utiliza con el comando para obtener un ARN.

AWS CLI Comando	Propiedad ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DB Arn ParameterGroup
describe-db-cluster-parameter-groups	DB ClusterParameter GroupArn
describe-db-instances	DB InstanceArn

AWS CLI Comando	Propiedad ARN
describe-events	SourceArn
describe-db-subnet-groups	DB SubnetGroup Arn
describe-db-clusters	DB ClusterArn
describe-db-cluster-snapshots	DB ClusterSnapshot Arn

Por ejemplo, el AWS CLI comando siguiente obtiene el ARN de una instancia de base de datos.

Example

Para Linux, OS X o Unix:

```
aws neptune describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

Para Windows:

```
aws neptune describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

Obtención de un ARN existente mediante la API

Para obtener el ARN de un recurso concreto de Neptune, llame a las siguientes acciones de la API y utilice las propiedades de ARN mostradas.

Acción de la API de Neptune	Propiedad ARN
DescribeEventSuscripciones	EventSubscriptionArn
DescribeCertificates	CertificateArn
Descrito B ParameterGroups	DB Arn ParameterGroup

Acción de la API de Neptune	Propiedad ARN
Grupos B descritos ClusterParameter	Base de datos ClusterParameter GroupArn
DescribeDBInstances	DB InstanceArn
DescribeEvents	SourceArn
Descrito B SubnetGroups	DB Arn SubnetGroup
DescribeDBClusters	DB ClusterArn
Descrito B ClusterSnapshots	DB Arn ClusterSnapshot

Copia de seguridad y restauración de un clúster de base de datos de Amazon Neptune

En esta sección, se muestra cómo puede crear copias de seguridad y restauraciones de clústeres de base de datos de Amazon Neptune.

Temas

Información general sobre la copia de seguridad y restauración de un clúster de base de datos de Neptune

En esta sección, se proporciona información de nivel superior sobre las copias de seguridad y la restauración de datos en Amazon Neptune.

Temas

Tolerancia a errores para un clúster de base de datos de Neptune

Un clúster de base de datos de Neptune ofrece tolerancia a errores por diseño. El volumen del clúster abarca varias zonas de disponibilidad en una única región de AWS y cada zona de disponibilidad contiene una copia de los datos del volumen del clúster. Esta funcionalidad significa que el clúster de base de datos puede tolerar un error de una zona de disponibilidad sin perder datos y con tan solo una interrupción breve del servicio.

Si se produce un error en la instancia principal de un clúster de base de datos, Neptune conmuta por error automáticamente a una nueva instancia principal de una de las dos formas siguientes:

- Promoviendo una réplica de Neptune ya existente a nueva instancia principal
- Creando una nueva instancia principal

Si el clúster de base de datos tiene una o varias réplicas de Neptune, se promueve una réplica de Neptune a instancia principal durante un evento de error. Un evento de error provoca una interrupción breve durante la cual las operaciones de lectura y escritura generan errores con una excepción. Sin embargo, el servicio se suele restaurar en menos de 120 segundos y, en muchos casos, en menos de 60 segundos. Para aumentar la disponibilidad de su clúster de base de datos, es recomendable que cree al menos una o varias réplicas de Neptune en dos o más zonas de disponibilidad diferentes.

Puede personalizar el orden en que se promueven las réplicas de Neptune a instancia principal tras un error mediante la asignación de una prioridad a cada réplica. Las prioridades van desde 0 para la prioridad más alta hasta 15 para la más baja. Si la instancia principal experimenta un error, Neptune promueve la réplica de Neptune con la prioridad más alta a nueva instancia principal. Puede modificar la prioridad de una réplica de Neptune en cualquier momento. Al modificar la prioridad, no se activa una conmutación por error.

Puede usar la AWS CLI para establecer la prioridad de conmutación por error de una instancia de base de datos de la siguiente manera:

```
aws neptune modify-db-instance --db-instance-identifier (the instance ID) --promotion-
tier (the failover priority value)
```

Puede haber más de una réplica de Neptune con la misma prioridad, lo que genera niveles de promoción. Si dos o más réplicas de Neptune comparten la misma prioridad, Neptune promueve la réplica que tiene un tamaño mayor. Si dos o más réplicas de Neptune tienen la misma prioridad y el mismo tamaño, Neptune promueve una réplica arbitraria del mismo nivel de promoción.

Si el clúster de base de datos no contiene ninguna réplica de Neptune, la instancia principal se vuelve a crear durante un evento de error. Un evento de error provoca una interrupción durante la cual las operaciones de lectura y escritura generan errores con una excepción. El servicio se restaura cuando se crea la nueva instancia principal, un proceso que normalmente dura menos de 10 minutos. Promover una réplica de Neptune a instancia principal es mucho más rápido que crear una nueva instancia principal.

Copias de seguridad de Neptune

Neptune crea copias de seguridad del volumen de clúster automáticamente y retiene los datos de restauración durante el periodo de retención de copia de seguridad. Las copias de seguridad de Neptune son continuas y progresivas para que se puedan restaurar con rapidez a cualquier punto durante el periodo de retención de copia de seguridad. No se produce ningún impacto en el desempeño ni ninguna interrupción del servicio de base de datos durante la escritura de los datos de copia de seguridad. Puede especificar un periodo de retención de copia de seguridad de 1 a 35 días cuando cree o modifique un clúster de bases de datos.

Para controlar el uso de almacenamiento de copias de seguridad, puede reducir el intervalo de retención de copia de seguridad, eliminar las instantáneas manuales anteriores cuando ya no las necesite o ambas cosas. Para ayudar a administrar los costos, puede monitorizar la cantidad de almacenamiento consumido por las copias de seguridad continuas y las instantáneas manuales

que persistan más allá del periodo de retención. Puede reducir el intervalo de retención de copia de seguridad y eliminar las instantáneas manuales cuando ya no las necesite.

Si desea conservar una copia de seguridad más allá del periodo de retención de copia de seguridad, también puede crear un snapshot de los datos del volumen de clúster. Al almacenar instantáneas, se generan los cargos de almacenamiento estándar para Neptune. Para obtener más información acerca de los precios de almacenamiento de instancias, consulte Precios de Amazon Neptune.

Neptune conserva los datos de restauración incrementales durante todo el periodo de retención de la copia de seguridad. Por lo tanto, solo tiene que crear una instantánea de los datos que desee conservar después del periodo de retención de copia de seguridad. Puede crear un nuevo clúster de bases de datos a partir de la instantánea.

Important

Si elimina un clúster de base de datos, todas sus copias de seguridad automatizadas se eliminan al mismo tiempo y no se pueden recuperar. Esto significa que, a no ser que decida crear una instantánea de base de datos final, no podrá restaurar posteriormente la instancia de base de datos a su estado final. Las instantáneas manuales no se eliminan cuando se elimina el clúster.

Note

- Para todos los clústeres de base de datos de Amazon Neptune, el periodo de retención de copia de seguridad predeterminado es de un día con independencia del procedimiento empleado para crear el clúster.
- No es posible deshabilitar las copias de seguridad automatizadas en Neptune. El clúster de base de datos administra el período de retención de copia de seguridad para Neptune.

Métricas de CloudWatch que son útiles para administrar el almacenamiento de copias de seguridad de Neptune

Puede utilizar las métricas de Amazon CloudWatch `TotalBackupStorageBilled`, `SnapshotStorageUsed` y `BackupRetentionPeriodStorageUsed` para revisar y monitorizar la

cantidad de almacenamiento utilizado por sus copias de seguridad de Neptune, tal y como se indica a continuación:

- `BackupRetentionPeriodStorageUsed` representa la cantidad de almacenamiento de copias de seguridad, en bytes, utilizado para almacenar copias de seguridad continuas en el momento actual. Este valor depende del tamaño del volumen del clúster y de la cantidad de cambios que realice durante el periodo de retención. Sin embargo, a efectos de facturación, no supera el tamaño del volumen del clúster acumulado durante el periodo de retención. Por ejemplo, si el tamaño de `VolumeBytesUsed` del clúster es de 107.374.182.400 GiB (100 GB) y el periodo de retención es de dos días, el valor máximo de `BackupRetentionPeriodStorageUsed` es 214.748.364.800 (100 GiB + 100 GiB).
- `SnapshotStorageUsed` representa la cantidad de almacenamiento de copias de seguridad utilizado, en bytes, para almacenar instantáneas manuales más allá del periodo de retención de copia de seguridad. Las instantáneas manuales no cuentan para el almacenamiento de copias de seguridad de instantáneas mientras su marca de tiempo de creación se encuentra dentro del período de retención. Todas las instantáneas automáticas tampoco cuentan para el almacenamiento de copias de seguridad. El tamaño de cada instantánea es el tamaño del volumen del clúster en el momento en que se realiza la instantánea. El valor de `SnapshotStorageUsed` depende del número de instantáneas que mantiene y del tamaño de cada instantánea. Suponga, por ejemplo, que tiene una instantánea manual fuera del periodo de retención y que el tamaño de `VolumeBytesUsed` del clúster era de 100 GiB cuando se realizó la instantánea. La cantidad de SnapshotStorageUsed es 107.374.182.400 bytes (100 GiB).
- `TotalBackupStorageBilled` representa la suma, en bytes, de `BackupRetentionPeriodStorageUsed` y `SnapshotStorageUsed`, menos una cantidad de almacenamiento gratuito de copias de seguridad equivalente al tamaño del volumen de un clúster de un día. El almacenamiento de copia de seguridad gratuito es igual al tamaño de volumen más reciente. Por ejemplo, si el tamaño de `VolumeBytesUsed` del clúster es de 100 GiB, el período de retención es de dos días y tiene una instantánea manual fuera del período de retención, el `TotalBackupStorageBilled` es de 214.748.364.800 bytes (200 GiB + 100 GiB - 100 GiB).

Puede monitorizar un clúster de Neptune y crear informes que utilicen las métricas de CloudWatch con la consola de CloudWatch. Para obtener más información acerca de cómo utilizar métricas de CloudWatch, consulte Monitorización de Neptune y la tabla de métricas en Métricas de Neptune CloudWatch .

Restauración de datos a partir de una copia de seguridad de Neptune

Puede recuperar sus datos creando un nuevo clúster de base de datos de Neptune a partir de los datos de copias de seguridad conservados por Neptune o de una instantánea de clúster de base de datos que haya guardado. Puede restaurar rápidamente una nueva copia del clúster de base de datos creada a partir de los datos de backup de cualquier momento dado durante el periodo de retención de copia de seguridad. La naturaleza continua e incremental de las copias de seguridad de Neptune durante el periodo de retención de copia de seguridad implica que no es necesario realizar instantáneas de los datos con demasiada frecuencia para mejorar los tiempos de restauración.

Para determinar el primer o el último momento para el que se puede restaurar una instancia de base de datos, busque los valores `Latest Restorable Time` o `Earliest Restorable Time` en la consola de Neptune. El último momento que se puede restaurar para un clúster de base de datos es el punto más reciente en el que se puede restaurar dicho clúster, normalmente en los cinco minutos previos a la hora actual. El momento más antiguo que se puede restaurar especifica el primer momento del periodo de retención de copia de seguridad para el que se puede restaurar el volumen del clúster.

Puede determinar cuándo se ha completado la restauración de un clúster de base de datos comprobando los valores `Latest Restorable Time` y `Earliest Restorable Time`. Los valores `Latest Restorable Time` y `Earliest Restorable Time` devolverán NULL hasta que la operación de restauración se haya completado. No puede solicitar una operación de backup o restauración si `Latest Restorable Time` o `Earliest Restorable Time` devuelven el valor NULL.

Para restaurar una instancia de base de datos a un momento especificado usando la AWS Management Console

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Instances (Instancias). Elija la instancia principal del clúster de base de datos que desea restaurar.
3. Elija Instance actions y, a continuación, Restore to point in time.

   En la ventana Launch DB Instance (Lanzar instancia de base de datos), elija Custom (Personalizado) en Restore time (Tiempo de restauración).
4. Especifique la fecha y la hora que desea restaurar en Custom (Personalizado).

5. Escriba el nombre de la instancia de base de datos nueva restaurada en DB instance identifier (Identificador de instancias de bases de datos) en Settings (Configuración).
6. Elija Launch DB Instance (Lanzar instancia de base de datos) para lanzar la instancia de base de datos restaurada.

   Se crea una nueva instancia de base de datos con el nombre que ha especificado y se crea un nuevo clúster de base de datos. El nombre del clúster de base de datos es el nombre de la nueva instancia de base de datos seguido de `-cluster`. Por ejemplo, si el nombre de la nueva instancia de base de datos es `myrestoreddb`, el nombre del nuevo clúster de base de datos es `myrestoreddb-cluster`.

Periodo de copia de seguridad en Neptune

Los backups automatizados se producen a diario durante la ventana de copia de seguridad preferida. Si la la copia de seguridad requiere más tiempo del asignado a la ventana de copia de seguridad, la copia de seguridad continúa cuando finaliza la ventana hasta que se completa. La ventana de copia de seguridad no se puede solapar con la ventana de mantenimiento semanal para la instancia de base de datos.

Durante la ventana de copia de seguridad automático, las E/S de almacenamiento pueden quedar suspendidas brevemente mientras se inicializa el proceso de copia de seguridad (normalmente durante unos pocos segundos). Pueden producirse latencias elevadas durante unos minutos mientras se realizan los backups para las implementaciones Multi-AZ.

Normalmente, el plano de control de Amazon RDS subyacente a Neptune selecciona de forma aleatoria el periodo de la copia de seguridad a partir de un bloque temporal de ocho horas por región. Los bloques de tiempo de cada región desde la que se asignan los periodos predeterminados de las copias de seguridad se documentan en la sección Backup Window de la Guía de usuario de Amazon RDS.

Creación de una instantánea de clúster de base de datos en Neptune

Neptune crea una instantánea del volumen de almacenamiento del clúster de base de datos. Para ello, realiza una copia de seguridad de todo el clúster, no solo de las bases de datos individuales. Cuando cree una instantánea de clúster de base de datos, tiene que identificar el clúster de base de datos cuya copia de seguridad va a realizar. Asigne a continuación un nombre a la instantánea de base de datos para poder restaurar desde esta más adelante. La cantidad de tiempo que tarda en crearse una instantánea de clúster de base de datos varía con el tamaño de sus bases de datos. La instantánea incluye todo el volumen de almacenamiento. Por tanto, el tamaño de los archivos (como archivos temporales) también afecta a la cantidad de tiempo que tarda en crearse la instantánea.

Puede crear una instantánea de clúster de base de datos usando la AWS Management Console, la AWS CLI o la API de Neptune.

Uso de la consola para crear una instantánea de clúster de base de datos

Para crear una instantánea de clúster de base de datos

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. En la lista de instancias de base de datos, seleccione la instancia principal para el clúster de base de datos.
4. Elija Instance actions (Acciones de instancias) y, a continuación, Take snapshot (Tomar instantánea).

   Aparece la ventana Take DB Snapshot.
5. Introduzca el nombre de la instantánea del clúster de base de datos en el cuadro de texto Snapshot Name (Nombre de instantánea).
6. Elija Take Snapshot (Tomar instantánea).

Restauración de una instantánea de clúster de base de datos

Cuando se crea una instantánea de Amazon Neptune de un clúster de base de datos, Neptune crea una instantánea del volumen de almacenamiento del clúster. Para ello, realiza una copia de seguridad de todos los datos, no solo de las instancias individuales. Para crear posteriormente un clúster de base de datos nuevo, restaure esa instantánea de clúster de base de datos. Al restaurar el clúster de base de datos, debe indicar el nombre de la instantánea de clúster de base de datos desde la que se restaura y, a continuación, proporcionar un nombre para el nuevo clúster de base de datos que se crea con la operación de restauración.

Contenido

Aspectos a tener en cuenta acerca de cómo restaurar un clúster de base de datos de Neptune a partir de una instantánea

No puede realizar la restauración en un clúster de base de datos existente

El proceso de restauración siempre crea un clúster de base de datos nuevo, por lo que no se puede restaurar en un clúster de base de datos que ya existe.

No se restaura ninguna instancia

Un clúster de base de datos nuevo que se crea por medio de una restauración no tiene ninguna instancia asociada.

En cuanto finalice la restauración y su nuevo clúster de base de datos esté disponible, cree de forma explícita las instancias que va a necesitar. Para ello, puede utilizar la consola de Neptune o la API de CreateDBInstance.

No se restaura ningún grupo de parámetros personalizado

Un clúster de base de datos nuevo creado por medio de una restauración tiene asociado automáticamente el grupo de parámetros de base de datos predeterminado.

En cuanto termine la restauración y el nuevo clúster de base de datos esté disponible, asocie los grupos de parámetros de base de datos personalizados que utilizaba la instancia desde la que se ha restaurado. Para ello, utilice el comando Modificar de la consola de Neptune o la API de ModifyDBInstance.

> **Important**
>
> Se recomienda guardar un grupo de parámetros personalizado que se esté utilizando en un clúster de base de datos del que vaya a crear una instantánea. A continuación, cuando realice la restauración desde esa instantánea, podrá asociar fácilmente el grupo de parámetros correcto al clúster de base de datos restaurado.

No se restaura ningún grupo de seguridad personalizado

Un clúster de base de datos nuevo creado por medio de una restauración tiene asociado el grupo de seguridad predeterminado.

En cuanto termine la restauración y el nuevo clúster de base de datos esté disponible, asocie los grupos de seguridad personalizados que utilizaba la instancia desde la que se ha restaurado. Para ello, utilice el comando Modificar de la consola de Neptune o la API de ModifyDBInstance.

No puede realizar la restauración desde una instantánea cifrada compartida

No puede restaurar un clúster de base de datos desde una instantánea de clúster de base de datos que esté compartida y cifrada.

En su lugar, puede realizar una copia no compartida de la instantánea y restaurar desde la copia.

Un clúster de base de datos restaurado utiliza el mismo almacenamiento que antes

Al restaurar un clúster de base de datos a partir de una instantánea de clúster de base de datos, la cantidad de almacenamiento asignada al nuevo clúster es la misma que se asignó al clúster de base de datos desde el que se realizó la instantánea, independientemente de cuánto almacenamiento asignado se esté utilizando realmente.

En otras palabras, el límite máximo por el que se le factura no cambia. Para restablecer la marca de agua alta, es necesario exportar los datos del gráfico y volver a cargarlos en un nuevo clúster de base de datos (consulte Facturación del almacenamiento de Neptune).

Cómo realizar la restauración a partir de una instantánea

Puede restaurar un clúster de base de datos desde una instantánea de clúster de base de datos utilizando la AWS Management Console, la AWS CLI o la API de Neptune.

Uso de la consola para restaurar desde una instantánea

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Elija la instantánea de clúster de base de datos desde la que desea restaurar.
4. Seleccione Actions (Acciones), Restore Snapshot (Restaurar instantánea).
5. En la página de Restore DB Instance (Restaurar instancia de base de datos), en la casilla DB Instance Identifier (Identificador de instancias de bases de datos), introduzca el nombre del clúster de base de datos restaurado.
6. Elija Restore DB Instance (Restaurar instancia de base de datos).
7. Si desea restaurar la funcionalidad del clúster de base de datos para que concuerde con la de la instantánea de la que procede, debe modificar el clúster para que use el grupo de seguridad. Los pasos siguientes presuponen que el clúster de base de datos está en una nube virtual privada (VPC). Si el clúster de base de datos no está en una VPC, use la consola de Amazon EC2 para encontrar el grupo de seguridad que necesita para el clúster de base de datos.

    a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.

    b. En el panel de navegación, elija Security Groups (Grupos de seguridad).

c. Elija el grupo de seguridad que desea usar para los clústeres de base de datos. Si es necesario, añada reglas para vincular el grupo de seguridad a un grupo de seguridad de una instancia EC2.

Copia de una instantánea de clúster de base de datos

Con Neptune, puede copiar instantáneas de clúster de base de datos automatizadas o manuales. Después de copiar una instantánea, la copia es una instantánea manual.

Puede copiar una instantánea en la misma región de AWS y entre regiones de AWS.

La copia de una instantánea automatizada en otra cuenta de AWS consta de dos pasos: primero se crea una instantánea manual a partir de la automatizada y, a continuación, se copia la manual en la otra cuenta.

Como alternativa a la copia, también puede compartir instantáneas manuales con otras cuentas de AWS. Para obtener más información, consulte Compartir una instantánea de clúster de base de datos.

Temas

Limitaciones al copiar una instantánea

A continuación se indican algunas limitaciones al copiar instantáneas:

- Puede copiar una instantánea entre China (Pekín) y China (Ningxia), pero no puede copiar una instantánea entre estas regiones de China y otras regiones de AWS.
- Puede copiar una instantánea entre AWS GovCloud (Este de EE. UU.) y AWS GovCloud (Oeste de EE. UU.), pero no puede copiar una instantánea entre estas regiones de AWS GovCloud (US) y otras regiones de AWS.
- Si elimina una instantánea de origen antes de que la instantánea de destino esté disponible, la copia de la instantánea podría generar un error. Compruebe que la instantánea de destino tiene el estado `AVAILABLE` antes de eliminar una instantánea de origen.

- Puede tener hasta cinco solicitudes de copia de instantánea en curso en una única región por cuenta.
- Dependiendo de las regiones implicadas y de la cantidad de datos que se vayan a copiar, una copia de instantánea entre regiones puede tardar horas en completarse.

  Si hay un gran número de solicitudes de copia de instantánea entre regiones desde una región de AWS de origen, Neptune puede realizar nuevas solicitudes de copia entre regiones desde esa región de AWS de origen en una cola hasta que alguna de las copias en curso se complete. No se muestra ninguna información de progreso sobre las solicitudes de copia mientras están en esa cola. La información de progreso solo se muestra después de que comience la copia.

Retención de copias de instantáneas de clústeres de base de datos

Neptune elimina las instantáneas automatizadas de la siguiente manera:

- Al final de su periodo de retención.
- Al deshabilitar las instantáneas automatizadas para un clúster de base de datos.
- Al eliminar un clúster de base de datos.

Si desea conservar una instantánea automatizada durante un periodo más largo, cópiela para crear una instantánea manual que se conservará hasta que la elimine. Es posible que se apliquen costos de almacenamiento de Neptune a las instantáneas manuales si exceden el espacio de almacenamiento predeterminado.

Para obtener más información acerca de los costos de almacenamiento de copias de seguridad, consulte Precios de Neptune.

Administración del cifrado al copiar instantáneas

Puede copiar una instantánea que haya sido cifrada con una clave de cifrado de AWS KMS. Si copia una instantánea cifrada, la copia de la instantánea se debe cifrar también. Puede cifrar la copia con la misma clave de cifrado de AWS KMS que la instantánea original, o puede especificar una clave de cifrado de AWS KMS diferente.

No se puede cifrar un snapshot de clúster de base de datos sin cifrar durante el proceso de copia.

Para las instantáneas de clúster de base de datos de Amazon Neptune, tiene también la opción de dejar la instantánea del clúster de base de datos sin cifrar y especificar en su lugar una clave

de cifrado de AWS KMS al restaurar. El clúster de base de datos restaurado se cifra con la clave especificada.

Copia de instantáneas entre regiones de AWS

> **Note**
>
> Esta característica está disponible a partir de la versión 1.0.2.1 del motor de Neptune.

Al copiar una instantánea en una región de AWS que es distinta de la región de AWS de la instantánea de origen, la primera copia es una copia de la instantánea completa, incluso si copia una instantánea incremental. Una copia de la instantánea completa contiene todos los datos y metadatos necesarios para restaurar la instancia de base de datos. Tras la primera copia de la instantánea, puede copiar instantáneas incrementales de la misma instancia de base de datos en la misma región de destino dentro de la misma cuenta de AWS.

Una instantánea incremental contiene solo los datos que han cambiado tras la instantánea más reciente de la misma instancia de base de datos. La copia de instantáneas incrementales es más rápida y genera un costo de almacenamiento más bajo que la copia de instantáneas completa. La copia de instantáneas incrementales en las regiones AWS se admite tanto para las instantáneas sin cifrar como para las cifradas.

> **Important**
>
> En el caso de las instantáneas compartidas, no se admite la copia de instantáneas incrementales. En el caso de las instantáneas compartidas, todas las copias son instantáneas completas, incluso en la misma región.

Dependiendo de las regiones de AWS implicadas y de la cantidad de datos que se vayan a copiar, una copia de instantánea entre regiones puede tardar horas en completarse.

Copia de una instantánea de clúster de base de datos con la consola

Si el motor de base de datos de origen es Neptune, su instantánea es una instantánea de clúster de base de datos. Para cada cuenta de AWS, puede copiar hasta cinco instantáneas de clúster de base de datos a la vez por región de AWS. Puede copiar instantáneas de clúster de base de datos cifradas y sin cifrar.

Para obtener más información acerca de los precios de las transferencias de datos, consulte Precios de Neptune.

Para cancelar una operación de copia una vez que está en curso, elimine la instantánea del clúster de base de datos de destino mientras está en el estado copying (Copiando).

El siguiente procedimiento sirve para copiar instantáneas de clúster de base de datos cifradas o sin cifrar.

Para copiar una instantánea de clúster de base de datos

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Active la casilla de verificación de la instantánea del clúster de base de datos que desee copiar.
4. Elija Actions (Acciones) y, a continuación, elija Delete Snapshot (Eliminar instantánea). Aparece la página Make Copy of DB Snapshot.
5. Introduzca el nombre de la copia de la instantánea del clúster de base de datos en New DB Snapshot Identifier (Nuevo identificador de instantánea de base de datos).
6. Para copiar las etiquetas y los valores de la instantánea en la copia de la instantánea, elija Copy Tags.
7. En Enable Encryption, elija una de las siguientes opciones:
   - Elija Disable encryption (Deshablitar cifrado) si la instantánea de clúster de base de datos no está cifrada y no desea cifrar la copia.
   - Elija Enable encryption (Habilitar cifrado) si la instantánea de clúster de base de datos no está cifrada pero desea cifrar la copia. En este caso, en Clave maestra, especifique el identificador de la clave de AWS KMS que desea usar para cifrar la copia de la instantánea del clúster de base de datos.
   - Elija Enable encryption (Habilitar cifrado) si la instantánea de clúster de base de datos está cifrada. En ese caso, debe cifrar la copia, de modo que Yes ya está seleccionado. En Clave maestra, especifique el identificador de la clave de AWS KMS que se debe usar para cifrar la copia de la instantánea del clúster de base de datos.
8. Elija Copy Snapshot.

Copia de una instantánea de clúster de base de datos mediante la AWS CLI

Puede copiar una instantánea de base de datos usando el comando copy-db-cluster-snapshot de la AWS CLI.

Si desea copiar la instantánea en una nueva región de AWS, ejecute el comando en la nueva región .

Utilice las siguientes descripciones y ejemplos de parámetros para determinar qué parámetros se utilizarán al copiar una instantánea con la AWS CLI.

- `--source-db-cluster-snapshot-identifier`: identificador de la instantánea de base de datos de origen.
  - Si la instantánea de origen está en la misma región de AWS que la copia, especifique un identificador de instantánea de base de datos válido, como `neptune:instance1-snapshot-20130805`.
  - Si la instantánea de origen está en una región de AWS distinta de la de la copia, especifique un ARN de instantánea de base de datos válido, como `arn:aws:neptune:us-west-2:123456789012:snapshot:instance1-snapshot-20130805`.
  - Si va a copiar desde una instantánea de base de datos manual compartida, este parámetro debe ser el nombre de recurso de Amazon (ARN) de la instantánea de base de datos compartida.
  - Si va a copiar una instantánea cifrada, este parámetro debe estar en el formato de ARN de la región de AWS de origen y debe coincidir con el `SourceDBSnapshotIdentifier` en el parámetro `PreSignedUrl`.
- `--target-db-cluster-snapshot-identifier`: identificador de la nueva copia de la instantánea de base de datos cifrada.
- `--kms-key-id`: identificador de la clave de AWS KMS para una instantánea de base de datos cifrada. El ID de clave de AWS KMS es el nombre de recurso de Amazon (ARN), el identificador de clave de AWS KMS o el alias de clave de AWS KMS de la clave de cifrado de AWS KMS.
  - Si copia una instantánea de base de datos cifrada desde la cuenta de AWS, puede especificar un valor para este parámetro para cifrar la copia con una nueva clave de cifrado de AWS KMS. Si no especifica ningún valor para este parámetro, la copia de la instantánea de base de datos se cifrará con la misma clave de AWS KMS que la de la instantánea de base de datos de origen.
  - No se puede usar este parámetro para crear una copia cifrada de una instantánea sin cifrar. Si lo intenta, se generará un error.

  - Si copia una instantánea cifrada en otra región de AWS, debe especificar una clave de AWS KMS para la región de AWS de destino. Las claves de cifrado de AWS KMS son específicas de la región de AWS en la que se han creado y no se pueden usar claves de cifrado de una región de AWS en otra región de AWS.
- `--source-region`: identificador de la región de AWS donde está la instantánea de base de datos de origen. Si copia una instantánea cifrada en una región AWS diferente, debe especificar esta opción.
- `--region`: identificador de la región de AWS en la que va a copiar la instantánea. Si copia una instantánea cifrada en una región AWS diferente, debe especificar esta opción.

Example Origen sin cifrar, a la misma región

El código siguiente crea una copia de una instantánea, con el nuevo nombre `mydbsnapshotcopy`, desde la región de AWS `us-east-1` a la región `us-west-2`.

Para Linux, OS X o Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Para Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Example Origen sin cifrar, entre regiones

El código siguiente crea una copia de una instantánea, con el nuevo nombre `mydbsnapshotcopy`, desde la región de AWS `us-east-1` a la región `us-west-2`. Ejecute el comando en la región `us-west-2`.

Para Linux, OS X o Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
```

```
  --source-region us-east-1 \
  --region us-west-2
```

Para Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
east-1:123456789012:snapshot:instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
```

Example Origen cifrado, entre regiones

El siguiente ejemplo de código copia una instantánea de base de datos cifrada de la región de AWS `us-east-1` a la región `us-west-2`. Ejecute el comando en la región `us-west-2`.

Para Linux, OS X o Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy \
  --source-region us-east-1 \
  --region us-west-2
  --kms-key-id my_us_west_2_key
```

Para Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:neptune:us-
west-2:123456789012:snapshot:instance1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy ^
  --source-region us-east-1 ^
  --region us-west-2
  --kms-key-id my-us-west-2-key
```

Compartir una instantánea de clúster de base de datos

Al utilizar Neptune, puede compartir una instantánea manual de clúster de base de datos de las siguientes maneras:

- Al compartir una instantánea manual de clúster de base de datos, ya sea cifrada o no cifrada, habilita a las cuentas autorizadas de AWS a copiar la instantánea.
- Si se comparte una instantánea manual de un clúster de base de datos, ya sea cifrada o sin cifrar, las cuentas autorizadas de AWS podrán restaurar directamente un clúster de base de datos a partir de la instantánea en lugar de hacer una copia de ella y restaurarla.

> **Note**
>
> Para compartir una instantánea automatizada de clúster de base de datos, cree una instantánea manual de clúster de base de datos al copiar la instantánea automatizada y, a continuación, comparta esa copia.

Para obtener más información sobre cómo restaurar un clúster de base de datos a partir de una instantánea de clúster de base de datos, consulte Cómo realizar la restauración a partir de una instantánea.

Puede compartir una instantánea manual con un máximo de 20 cuentas de AWS. También puede compartir una instantánea manual sin cifrar como pública, lo que hace que esté disponible para todas las cuentas de AWS. Al compartir una instantánea como pública, tenga cuidado de que no incluya información confidencial.

> **Note**
>
> Cuando se restaura un clúster de base de datos a partir de una instantánea compartida utilizando la AWS Command Line Interface (AWS CLI) o la API de Neptune, se debe especificar el nombre de recurso de Amazon (ARN) de la instantánea compartida como identificador de instantánea.

Temas

- Compartir una instantánea de clúster de base de datos

Uso compartido de una instantánea de clúster de base de datos cifrada

Puede compartir instantáneas de clúster de base de datos que se han cifrado "en reposo" utilizando el algoritmo de cifrado AES-256. Para obtener más información, consulte Cifrado de los recursos de Neptune en reposo. Para ello, debe seguir estos pasos:

1. Comparta la clave de cifrado de AWS Key Management Service (AWS KMS) que se utilizó para cifrar la instantánea con las cuentas que desea que tengan acceso a la instantánea.

   Puede compartir las claves de cifrado de AWS KMS con otra cuenta de AWS añadiendo la otra cuenta a la política de claves de KMS. Para obtener información detallada sobre cómo actualizar una política de claves, consulte Políticas de claves en la Guía para desarrolladores de AWS KMS. Para ver un ejemplo de cómo crear una política de claves, consulte Creación de una política de IAM para permitir la copia de la instantánea cifrada más adelante en este tema.
2. Utilice la AWS Management Console, la AWS CLI o la API de Neptune para compartir la instantánea cifrada con las otras cuentas.

Estas restricciones se aplican al uso compartido de instantáneas cifradas:

- No se pueden compartir instantáneas cifradas como públicas.
- No se puede compartir una instantánea que se ha cifrado utilizando la clave de cifrado de AWS KMS predeterminada de la cuenta de AWS que compartió la instantánea.

Cómo permitir el acceso a una clave de cifrado de AWS KMS

Para que otra cuenta de AWS pueda copiar una instantánea cifrada de un clúster de base de datos que se comparta desde su cuenta, la cuenta con la que se comparte la instantánea debe tener acceso a la clave de KMS que cifró la instantánea. Para permitir que otra cuenta de AWS tenga acceso a una clave de AWS KMS, actualice la política de claves correspondiente a la clave de KMS con el ARN de la cuenta de AWS con la que está compartiendo como una `Principal` en la política de claves de KMS. A continuación, permita la acción `kms:CreateGrant`. Consulte Allowing users in other accounts to use a KMS key en la Guía para desarrolladores de AWS Key Management Service para conocer las instrucciones generales.

Después de dar a una cuenta de AWS acceso a su clave de cifrado de KMS, para copiar la instantánea cifrada, dicha cuenta de AWS debe crear un usuario de IAM si todavía no lo tiene. Las restricciones de seguridad de KMS no permiten el uso de una identidad de cuenta raíz de AWS para ello. La cuenta de AWS también debe asociar a ese usuario de IAM una política de IAM que le permita copiar una instantánea de clúster de base de datos cifrada utilizando la clave de KMS.

En el siguiente ejemplo de política de claves, el usuario `111122223333` es el propietario de la clave de cifrado de KMS, y el usuario `444455556666` es la cuenta con la que se comparte la clave. Esta política de claves actualizada da a la cuenta de AWS acceso a la clave de KMS al incluir el ARN de la identidad raíz de la cuenta de AWS del usuario `444455556666` como `Principal` para la política, y al permitir la acción `kms:CreateGrant`.

```
{
  "Id=": "key-policy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid=": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid=": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
```

```
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
    }
  ]
}
```

Creación de una política de IAM para permitir la copia de la instantánea cifrada

Después de que la cuenta externa de AWS tenga acceso a la clave de KMS, el propietario de esa cuenta puede crear una política que permita a un usuario de IAM creado para esa cuenta copiar una instantánea que haya sido cifrada con esa clave de KMS.

En el siguiente ejemplo, se muestra una política que puede asociarse a un usuario de IAM para la cuenta de AWS `444455556666`. Permite al usuario de IAM copiar una instantánea compartida de la cuenta de AWS `111122223333` que se ha cifrado con la clave de KMS `c989c1dd-a3f2-4a5d-8d96-e793d082ab26` en la región `us-west-2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid=": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid=": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
```

```
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Para obtener información detallada sobre cómo actualizar una política de claves, consulte Políticas de claves en la Guía para desarrolladores de AWS Key Management Service.

Compartir una instantánea de clúster de base de datos

Puede compartir una instantánea de clúster de base de datos usando la AWS Management Console, la AWS CLI o la API de Neptune.

Uso de la consola para compartir una instantánea de clúster de base de datos

Con la consola de Neptune, puede compartir una instantánea manual de clúster de base de datos con un máximo de 20 cuentas de AWS. También puede dejar de compartir una instantánea manual con una o varias cuentas.

Compartir una instantánea manual de un clúster de base de datos

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Elija la instantánea manual que desea compartir.
4. Elija Actions (Acciones), Share Snapshot (Compartir instantánea).
5. Elija una de las siguientes opciones para DB Snapshot Visibility (Visibilidad de instantánea de base de datos).

    - Si el origen no está cifrado, elija Público para permitir que todas las cuentas de AWSrestauren un clúster de base de datos a partir de su instantánea manual de clúster de base de datos. O

elija Privado para permitir que solo las cuentas de AWS que especifique puedan restaurar un clúster de base de datos a partir de su instantánea manual de clúster de base de datos.

> **Warning**
>
> Si establece DB snapshot visibility (Visibilidad de instantánea de base de datos) como Public (Pública), todas las cuentas de AWS pueden restaurar un clúster de base de datos a partir de una instantánea de clúster de base de datos manual y tener acceso a sus datos. No comparta como Public (Pública) ninguna instantánea de clúster de base de datos manual que contenga información confidencial.

- Si el original está cifrado, DB Snapshot Visibility (Visibilidad de instantánea de base de datos) se establece en Private (Privada), ya que las instantáneas cifradas no se pueden compartir como públicas.

6. En ID de cuenta de AWS, introduzca el identificador de cuenta de AWS correspondiente a una cuenta a la que desea permitir la restauración de un clúster de base de datos a partir de la instantánea manual. A continuación, elija Add (Añadir). Repita esta acción para incluir identificadores de cuenta de AWS adicionales, con un máximo de 20 cuentas de AWS.

   Si comete un error al añadir un identificador de cuenta de AWS a la lista de cuentas permitidas, puede eliminarlo de la lista seleccionando Delete (Eliminar) a la derecha del identificador incorrecto de la cuenta de AWS.
7. Después de añadir los identificadores de todas las cuentas de AWS a las que desea permitir la restauración de la instantánea manual, elija Guardar.

Para dejar de compartir una instantánea manual de clúster de base de datos con una cuenta de AWS

1. Abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Elija la instantánea manual que desea dejar de compartir.
4. Elija Actions (Acciones) y, a continuación, elija Share Snapshot (Compartir instantánea).
5. Para eliminar el permiso de una cuenta de AWS, elija Delete (Eliminar) para el identificador de cuenta de AWS correspondiente a esa cuenta en la lista de cuentas autorizadas.
6. Seleccione Save.

Eliminación de una instantánea de Neptune

Puede eliminar una instantánea de base de datos mediante la AWS Management Console, la AWS CLI o la API de administración de Neptune:

Eliminación mediante la consola

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Neptune en https://console.aws.amazon.com/neptune/home.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Elija la instantánea de base de datos que desee eliminar.
4. En Actions (Acciones), elija Delete Snapshot (Eliminar instantánea).
5. En la página de confirmación, elija Delete (Eliminar).

Eliminación mediante la AWS CLI

También puede eliminar una instantánea de base de datos mediante el comando de la AWS CLI delete_db_cluster_snapshot, utilizando el parámetro `--db-snapshot-identifier` para identificar la instantánea que desea eliminar:

Para Linux, OS X o Unix:

```
aws neptune delete-db-cluster-snapshot \
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Para Windows:

```
aws neptune delete-db-cluster-snapshot ^
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Eliminación mediante la API de administración de Neptune

Puede utilizar uno de los SDK para eliminar una instantánea de base de datos si llama a la API DeleteDBClusterSnapshot y utiliza los parámetros `DBSnapshotIdentifier` para identificar la instantánea de base de datos que desea eliminar.

Prácticas recomendadas: sacar el máximo partido de Neptune

Estas son algunas recomendaciones generales para trabajar con Amazon Neptune. Utilice esta información como referencia para encontrar rápidamente recomendaciones para usar Amazon Neptune y maximizar el rendimiento.

Contenido

- Cree objetos de cliente Java de Gremlin independientes para puntos de conexión de lectura y escritura
- Añada varios puntos de conexión de réplica de lectura a un grupo de conexiones Java de Gremlin
- Cierre el cliente para evitar el límite de conexiones
- Cree una nueva conexión después de una conmutación por error
- Establezca maxInProcessPerConnection y maxSimultaneousUsagePerConnection con el mismo valor.
- Envíe consultas al servidor como bytecode en lugar de como cadenas
- Consume siempre por completo el iterador ResultSet or devuelto por una consulta
- Añada vértices y bordes de forma masiva por lotes
- Deshabilitar el almacenamiento en caché de DNS en la máquina virtual de Java
- Opcionalmente, establezca tiempos de espera al nivel de consulta
- Solución de problemas de java.util.concurrent.TimeoutException

- Prácticas recomendadas de Neptune con openCypher y Bolt
  - En las consultas, utilice preferentemente bordes dirigidos en lugar de bidireccionales
  - Neptune no admite múltiples consultas simultáneas en una transacción
  - Cree una nueva conexión después de una conmutación por error
  - Gestión de conexiones para aplicaciones de larga duración
  - Manejo de conexiones para AWS Lambda
  - Cierre los objetos del controlador cuando haya terminado
  - Use modos de transacción explícitos para leer y escribir
    - Transacciones de solo lectura
    - Transacciones de solo lectura
  - Lógica de reintentos para las excepciones
  - Establezca varias propiedades a la vez mediante una sola cláusula SET
  - Utilice la cláusula SET para eliminar varias propiedades a la vez
  - Utilice consultas parametrizadas
  - Utilice mapas aplanados en lugar de mapas anidados en la cláusula UNWIND
  - Coloque los nodos más restrictivos en el lado izquierdo de las expresiones de ruta de longitud variable (VLP)

- Evite las comprobaciones redundantes de las etiquetas de los nodos mediante el uso de nombres de relación granulares
- Especifique las etiquetas de los bordes siempre que sea posible
- Evita usar la cláusula WITH siempre que sea posible
- Coloque los filtros restrictivos lo antes posible en la consulta
- Compruebe de forma explícita si existen propiedades
- No utilice una ruta con nombre (a menos que sea obligatoria)
- Evite COLLECT (DISTINCT ())
- Prefiera la función de propiedades a la búsqueda de propiedades individuales al recuperar todos los valores de las propiedades
- Realice cálculos estáticos fuera de la consulta
- Entradas por lotes utilizando UNWIND en lugar de declaraciones individuales
- Prefiera usar identificadores personalizados para el nodo o la relación
- Evite realizar ~id cálculos en la consulta

- Prácticas recomendadas de Neptune con SPARQL
  - Consulta de todos los gráficos con nombre de forma predeterminada
  - Especificación de un gráfico con nombre para la carga
  - Elegir entre FILTER, FILTER... IN y VALUES en sus consultas

Directrices operativas básicas de Amazon Neptune

A continuación, se detallan las directrices operativas básicas que debe seguir cuando trabaje con Neptune.

- Conozca las instancias de base de datos de Neptune para poder dimensionarlas correctamente en función de sus requisitos de rendimiento y casos de uso. Consulte Clústeres e instancias de base de datos de Amazon Neptune.
- Monitorice el uso de la memoria y de la CPU. Esto le servirá para saber cuándo migrar a una clase de instancia de base de datos con mayor capacidad de memoria o de CPU para conseguir el rendimiento de las consultas que necesita. Puede configurar Amazon CloudWatch para notificar cuándo cambian los patrones de uso o cuándo se está llegando al límite de capacidad de la implementación. Si lo hace, le podrá servir para mantener el rendimiento y la disponibilidad del

sistema. Consulte Monitorización de instancias y Monitorización de Neptune para obtener más información.

Dado que Neptune tiene su propio administrador de memoria, es normal observar un uso de la memoria relativamente bajo incluso cuando se utiliza mucha CPU. Encontrarse con excepciones de memoria insuficiente al ejecutar las consultas es el mejor indicador de que necesita aumentar la memoria que se puede liberar.

- Habilite las copias de seguridad automatizadas y configure la ventana de copia de seguridad para que se produzca cuando le convenga.
- Pruebe la conmutación por error de la instancia de base de datos para comprender cuánto tiempo tarda el proceso en su caso de uso. También ayuda a asegurarse de que la aplicación que obtiene acceso a su instancia de base de datos puede conectarse automáticamente a la nueva instancia de base de datos tras una conmutación por error.
- Si es posible, ejecute el cliente y el clúster de Neptune en la misma región y VPC, ya que las conexiones entre regiones con VPC interconexión pueden producir retrasos en los tiempos de respuesta de las consultas. Para obtener respuestas a consultas en milisegundos de un solo dígito, es necesario mantener el cliente y el clúster de Neptune en la misma región y VPC.
- Cuando cree una instancia de réplica de lectura, esta debe tener, como mínimo, el mismo tamaño que la instancia de escritor principal. Esto ayudará a mantener el retraso de replicación bajo control y evitará los reinicios de la réplica. Consulte Evite diferentes clases de instancia en un clúster.
- Antes de actualizar a una nueva versión principal del motor, asegúrese de probar la aplicación en ella. Para ello, puede clonar el clúster de base de datos para que el clúster clonado ejecute la nueva versión del motor y, a continuación, probar la aplicación en el clon.
- Para facilitar la conmutación por error, lo ideal es que todas las instancias tengan el mismo tamaño.

Temas

- Equilibrio de carga entre réplicas de lectura
- Carga más rápida usando una instancia temporal más grande
- Cambie el tamaño de la instancia de escritor mediante una conmutación por error a una réplica de lectura
- Reintente la carga tras el error de interrupción de la tarea de obtención previa de los datos

Prácticas recomendadas de seguridad de Amazon Neptune

Utilice cuentas de AWS Identity and Access Management (IAM) para controlar el acceso a las acciones de la API de Neptune. Controle las acciones que crean, modifican o eliminan recursos de Neptune (como instancias de bases de datos, grupos de seguridad, grupos de opciones o grupos de parámetros) y las acciones que realizan tareas administrativas frecuentes (como realizar copias de seguridad y restaurar instancias de bases de datos).

- Utilice credenciales temporales en lugar de persistentes siempre que sea posible.
- Asigne una cuenta de IAM individual a cada persona que administre recursos de Amazon Relational Database Service (Amazon RDS). Nunca utilice usuarios raíz de cuentas de AWS para administrar los recursos de Neptune. Cree un usuario de IAM para todos, incluido usted mismo.
- Asigne a cada usuario el conjunto mínimo de permisos requerido para realizar sus tareas.
- Use los grupos de IAM para administrar con eficacia los permisos para varios usuarios.
- Rote con regularidad sus credenciales de IAM.

Para obtener más información sobre el uso de IAM para acceder a los recursos de Neptune, consulte Seguridad en Amazon Neptune. Para obtener información general sobre cómo trabajar con IAM, consulte AWS Identity and Access Management y Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Evite diferentes clases de instancia en un clúster

Cuando el clúster de base de datos contiene instancias de clases diferentes, pueden surgir problemas con el paso del tiempo. El problema más común es que una instancia de lector pequeña puede entrar en un ciclo de reinicios repetidos debido al retardo en la replicación. Si un nodo de lector tiene una configuración de clase de instancia de base de datos más débil que la de una instancia de base de datos de escritor, el volumen de cambios puede ser demasiado grande para que el lector se adapte al ritmo.

> **Important**
>
> Para evitar reinicios repetidos a causa del retardo en la replicación, configure el clúster de base de datos de modo que todas las instancias tengan la misma clase (tamaño).

Puede ver el retardo entre la instancia de escritor (la principal) y los lectores de su clúster de base de datos mediante la métrica `ClusterReplicaLag` de Amazon CloudWatch. La métrica `VolumeWriteIOPs` también le permite detectar ráfagas de actividad de escritura en su clúster que pueden provocar un retardo en la replicación.

Evite reinicios repetidos durante la carga masiva

Si experimenta un ciclo de reinicios repetidos de réplicas de lectura debido a un retardo en la replicación durante una carga masiva, es probable que sus réplicas no puedan mantener el ritmo del escritor en el clúster de base de datos.

Escale los lectores a un tamaño superior al del escritor o quítelos temporalmente durante la carga masiva y vuelva a crearlos una vez finalizada la operación.

Habilite el índice OSGP si tiene un gran número de predicados

Si su modelo de datos contiene un gran número de predicados distintos (más de mil en la mayoría de los casos), es posible que el rendimiento se reduzca y aumenten los costos operativos.

Si es así, puede mejorar el rendimiento habilitando el índice OSGP. Consulte El índice OSGP.

Evite las transacciones de larga duración siempre que sea posible

Las transacciones de larga duración, ya sean de solo lectura o de lectura y escritura, pueden provocar problemas inesperados de los siguientes tipos:

Una transacción de larga duración en una instancia de lector o de escritor con escrituras simultáneas puede provocar una gran acumulación de diferentes versiones de los datos. Esto puede dar lugar a latencias más altas en las consultas de lectura que filtran una gran parte de sus resultados.

En algunos casos, las versiones acumuladas a lo largo de varias horas pueden provocar una limitación de las escrituras nuevas.

Una transacción de lectura-escritura de larga duración con muchas escrituras también puede provocar problemas si la instancia se reinicia. Si una instancia se reinicia tras un evento de mantenimiento o un bloqueo, se revierten todas las escrituras no confirmadas. Por lo general, estas operaciones de deshacer se ejecutan en segundo plano y no impiden que la instancia vuelva a funcionar, pero cualquier escritura nueva que entre en conflicto con las operaciones que se están revirtiendo fallará.

Por ejemplo, si se vuelve a intentar la misma consulta después de interrumpir la conexión en la ejecución anterior, es posible que se produzca un error al reiniciar la instancia.

El tiempo necesario para deshacer las operaciones es proporcional al tamaño de los cambios involucrados.

Prácticas recomendadas para utilizar métricas de Neptune

Para identificar los problemas de rendimiento causados por la falta de recursos y otros cuellos de botella frecuentes, puede monitorizar las métricas disponibles para su clúster de base de datos de Neptune.

Monitorice las métricas de desempeño con frecuencia para recopilar datos sobre los valores medios, máximos y mínimos de diversos intervalos de tiempo. Esto le servirá para identificar cuándo se degrada el rendimiento. Con estos datos, también puede definir alarmas de Amazon CloudWatch para recibir alertas cuando se alcancen umbrales de métricas concretos.

Cuando se configura un nuevo clúster de base de datos y se ejecuta con una carga de trabajo típica, intente capturar los valores medio, máximo y mínimo de todas las métricas de rendimiento en diversos intervalos (por ejemplo, una hora, 24 horas, una semana, dos semanas). Esto permite hacerse una idea de lo que es normal. Ayuda a obtener comparaciones para las horas con picos y valles de funcionamiento. Puede usar esta información para saber cuándo cae el rendimiento por debajo de los niveles estándar y puede configurar las alarmas según corresponda.

Consulte Monitorización de Neptune con Amazon CloudWatch para obtener información acerca de cómo ver las métricas de Neptune.

Las métricas más importantes con las que comenzar son las siguientes:

- BufferCacheHitRatio: porcentaje de solicitudes que se responden desde la caché del búfer. Los errores de caché añaden una latencia significativa a la ejecución de las consultas. Si la relación de aciertos de la caché es inferior al 99,9 % y la latencia es un problema para su aplicación, considere la posibilidad de actualizar el tipo de instancia para almacenar en caché más datos en memoria.

- Utilización de la CPU: porcentaje de la capacidad de procesamiento del equipo que está en uso. Unos valores elevados de consumo de CPU podrían ser adecuados, en función de los objetivos de rendimiento de las consultas.
- Memoria que se puede liberar: cuánta RAM está disponible en la instancia de base de datos en megabytes. Neptune tiene su propio administrador de memoria, por lo que esta métrica podría ser inferior a la esperada. Una buena señal de que debería plantearse la actualización de la clase de instancia a una con más RAM es si las consultas suelen lanzar con frecuencia excepciones de memoria insuficiente.

La línea roja de las métricas de la pestaña Monitoring (Monitorización) está marcada en el 75 % para las métricas de CPU y memoria. Si el consumo de memoria de instancia supera con frecuencia esta línea, significa que debe verificar la carga de trabajo y plantearse actualizar la instancia para mejorar el rendimiento de las consultas.

Prácticas recomendadas para el ajuste de consultas de Neptune

Una de las formas más eficaces de mejorar el rendimiento de Neptune es ajustar las consultas más utilizadas y que más recursos consumen para que ejecutarlas resulte más económico.

Para obtener información sobre cómo ajustar las consultas de Gremlin, consulte Sugerencias de consulta de Gremlin y Ajuste de las consultas de Gremlin. Para obtener información sobre cómo ajustar las consultas SPARQL, consulte Sugerencias de consulta SPARQL.

Equilibrio de carga entre réplicas de lectura

El enrutamiento de turnos rotativos para puntos de enlace de lectura funciona mediante el cambio del host al que apunta la entrada DNS. El cliente debe crear una nueva conexión y resolver el registro DNS para obtener una conexión a una nueva réplica de lectura, ya que las conexiones WebSocket a menudo se mantienen activas durante largos períodos.

Para obtener réplicas de lectura diferentes para las solicitudes sucesivas, asegúrese de que el cliente resuelva la entrada DNS cada vez que se conecte. Esto puede requerir cerrar la conexión y volver a conectarse al punto de enlace del lector.

También puede balancear la carga de las solicitudes en réplicas de lectura conectando los puntos de enlace de instancia explícitamente.

Carga más rápida usando una instancia temporal más grande

Su rendimiento de carga aumenta con tamaños de instancia más grandes. Si no utiliza un tipo de instancia grande, pero desea aumentar la velocidad de carga, puede utilizar una instancia mayor para cargarla y, a continuación, eliminarla.

> **Note**
>
> El siguiente procedimiento es para un nuevo clúster. Si tiene un clúster existente, puede agregar una nueva instancia más grande y, a continuación, promoverla a una instancia de base de datos principal.

Para cargar datos utilizando un tamaño de instancia mayor

1. Cree un clúster con una sola instancia `r5.12xlarge`. Esta instancia es la instancia de base de datos principal.
2. Cree una o varias réplicas de lectura del mismo tamaño (`r5.12xlarge`).

   Puede crear réplicas de lectura más pequeñas, pero si no son lo suficientemente grandes como para soportar las escrituras que realiza la instancia principal, es posible que tengan que reiniciarse con frecuencia. El tiempo de inactividad resultante reduce drásticamente el rendimiento.
3. En el comando de carga masiva, incluya `“parallelism” : “OVERSUBSCRIBE”` para indicarle a Neptune que utilice todos los recursos de CPU disponibles para la carga (consulte Parámetros de solicitudes del programa de carga de Neptune). En ese caso, la operación de carga se realizará tan rápido como lo permita la E/S, lo que generalmente requiere entre un 60 y un 70 % de los recursos de la CPU.
4. Cargue sus datos mediante el programa de carga de Neptune. El trabajo de carga se ejecuta en la instancia de base de datos principal.
5. Cuando terminen de cargarse los datos, asegúrese de escalar verticalmente todas las instancias del clúster al mismo tipo de instancia para evitar cargos adicionales y problemas de reinicio repetidos (consulte Evite distintos tamaños de instancias).

Cambie el tamaño de la instancia de escritor mediante una conmutación por error a una réplica de lectura

La mejor forma de cambiar el tamaño de una instancia del clúster de base de datos, incluida la instancia de escritor, es crear o modificar una instancia de réplica de lectura para que tenga el tamaño deseado y, a continuación, realizar una conmutación por error deliberada a esa réplica de lectura. El tiempo de inactividad registrado por la aplicación es solo el tiempo necesario para cambiar la dirección IP del escritor, que debería ser de unos 3 a 5 segundos.

La API de administración de Neptune que se utiliza para conmutar por error la instancia de escritor actual a una instancia de réplica de lectura es FailoverDBCluster de forma deliberada. Si utiliza el cliente Java de Gremlin, es posible que tenga que crear un nuevo objeto de cliente tras la conmutación por error para seleccionar la nueva dirección IP, tal y como se menciona aquí.

Asegúrese de cambiar todas las instancias al mismo tamaño para evitar un ciclo de reinicios repetidos, como se explica a continuación.

Reintente la carga tras el error de interrupción de la tarea de obtención previa de los datos

Cuando esté cargando datos en Neptune mediante el programa de carga masiva, ocasionalmente puede producirse un estado `LOAD_FAILED`, con los mensajes `PARSING_ERROR` y `Data prefetch task interrupted`, en respuesta a una solicitud de información detallada, como este:

```
"errorLogs" : [
  {
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Data prefetch task interrupted: Data prefetch task for 11467
 failed",
    "fileName" : "s3://some-source-bucket/some-source-file",
    "recordNum" : 0
  }
]
```

Si detecta este error, solo tiene que reintentar la solicitud de carga masiva.

El error se produce si ha habido una interrupción temporal no provocada normalmente por su solicitud o sus datos y, en general, puede resolverse ejecutando de nuevo la solicitud de carga masiva.

En caso de que esté usando la configuración predeterminada, es decir, `"mode":"AUTO"` y `"failOnError":"TRUE"`, el programa de carga omite los archivos que ya ha cargado correctamente y reanuda la carga de los archivos que aún no había cargado en el momento de la interrupción.

Prácticas recomendadas para utilizar Gremlin con Neptune

Siga estas recomendaciones cuando utilice el lenguaje de recorrido de gráficos Gremlin con Neptune. Para obtener información sobre el uso de Gremlin con Neptune, consulte the section called “Gremlin”.

Important

Se ha realizado un cambio en la versión 3.4.11 de TinkerPop que mejora la forma en que se procesan las consultas, pero, por el momento, a veces puede afectar gravemente al rendimiento de las consultas.
Por ejemplo, una consulta de este tipo podría ejecutarse de una forma mucho más lenta:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

Los vértices después del paso límite se obtienen ahora de una manera que no es la óptima debido al cambio de TinkerPop 3.4.11. Para evitarlo, puede modificar la consulta añadiendo el paso barrier() en cualquier punto después de `order().by()`. Por ejemplo:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 se ha habilitado en la versión 1.0.5.0 del motor de Neptune.

Temas

- Pruebe el código de Gremlin en el contexto en el que lo va a implementar
- Estructure las consultas de actualización o inserción para aprovechar el motor DFE
- Creación de escrituras de Gremlin eficientes de múltiples subprocesos
- Depuración de registros con la propiedad de hora de creación
- Uso del método datetime( ) para datos de tiempo de Groovy
- Uso de la fecha y hora nativas para los datos de tiempo de GLV

Pruebe el código de Gremlin en el contexto en el que lo va a implementar

En Gremlin, los clientes pueden enviar consultas al servidor de varias maneras: mediante WebSocket o Bytecode GLV, o mediante la consola de Gremlin con scripts basados en cadenas.

Es importante reconocer que la ejecución de las consultas de Gremlin puede variar según la forma en que se envíe la consulta. Una consulta que devuelve un resultado vacío puede considerarse correcta si se envía en el modo de Bytecode, pero puede considerarse que ha fallado si se envía en el modo de script. Por ejemplo, si incluye `next()` en una consulta en modo de script, `next()` se envía al servidor, pero al usar ByteCode, el cliente normalmente procesa `next()` automáticamente. En el primer caso, la consulta falla si no se encuentra ningún resultado, pero en el segundo, la consulta se realiza correctamente independientemente de que el conjunto de resultados esté vacío o no.

Si desarrolla y prueba su código en un solo contexto (por ejemplo, la consola de Gremlin, que normalmente envía consultas en forma de texto), pero luego implementa el código en un contexto diferente (por ejemplo, mediante el controlador de Java con Bytecode), podría tener problemas, ya que su código podría comportarse de manera diferente en producción que en su entorno de desarrollo.

> **Important**
>
> Asegúrese de probar el código de Gremlin en el contexto de GLV en el que se va a implementar para evitar resultados inesperados.

Estructure las consultas de actualización o inserción para aprovechar el motor DFE

Puede mejorar considerablemente el rendimiento de las consultas de actualización o inserción si aprovecha al máximo el motor DFE de Neptune.

En Realización de actualizaciones o inserciones eficientes con los pasos `mergeV()` y `mergeE()` de Gremlin, se explica cómo estructurar las consultas de actualización o inserción para utilizar el motor DFE de la forma más eficaz posible.

Creación de escrituras de Gremlin eficientes de múltiples subprocesos

Hay algunas directrices para la carga con múltiples subprocesos de datos en Neptune mediante Gremlin.

Si fuera posible, asigne a cada hilo un conjunto de vértices o bordes para su inserción o modificación que no colisionen. Por ejemplo, ID de direcciones de hilo 1, rango 1–50 000; ID de direcciones de hilo 2, rango 50 001–100 000, y así sucesivamente. Esto reduce el riesgo de alcanzar un `ConcurrentModificationException`. Para estar seguro, coloque un bloque `try/catch` alrededor de todas las escrituras. Si se produce cualquier error, puede volver a intentarlo después de un breve retraso.

Las operaciones de escritura por lotes en un tamaño de lote entre 50 y 100 (vértices o bordes) generalmente funcionan bien. Si está agregando una gran cantidad de propiedades para cada vértice, un número cercano al 50 en lugar de 100 podría ser una elección más acertada. Merece la pena experimentar. Por lo tanto, para las escrituras por lotes, puede utilizar algo así como esto:

```
g.addV('test').property(id,'1').as('a').
  addV('test').property(id,'2').
  addE('friend').to('a').
```

Esto se repite en cada operación por lotes.

El uso de lotes es bastante más eficiente que la adición de un vértice o borde por cada viaje de ida y vuelta de Gremlin al servidor.

Si está utilizando un cliente de la variante de lenguaje Gremlin (GLV), puede crear un lote mediante programación creando en primer lugar un recorrido. Añádala y, por último, realice una iteración sobre ella. Por ejemplo:

```
t.addV('test').property(id,'1').as('a')
t.addV('test').property(id,'2')
t.addE('friend').to('a')
t.iterate()
```

Si es posible, es mejor utilizar el cliente de la variante de lenguaje Gremlin. Sin embargo, puede hacer algo similar con un cliente que envía consultas como cadenas de texto concatenando cadenas para construir un lote.

Si utiliza una de las bibliotecas de clientes Gremlin en lugar de HTTP básicos de las consultas, todos los subprocesos deberán compartir el mismo cliente, clúster o grupo de conexiones. Es posible que tenga que ajustar la configuración para obtener el mejor rendimiento posible; por ejemplo, puede que tenga que ajustar el tamaño del grupo de conexiones y el número de subprocesos de trabajo que utiliza el cliente de Gremlin.

Depuración de registros con la propiedad de hora de creación

Puede depurar los registros viejos almacenando la hora de creación como una propiedad en los vértices y descartándolos periódicamente.

Si necesita almacenar datos durante un periodo específico y, a continuación, eliminarlos del gráfico (tiempo de vida de vértice), puede almacenar una propiedad de marca temporal en la creación del vértice. A continuación, puede emitir periódicamente una consulta `drop()` para todos los vértices que se crearon antes de una hora determinada; por ejemplo:

```
g.V().has("timestamp", lt(datetime('2018-10-11')))
```

Uso del método **`datetime( )`** para datos de tiempo de Groovy

Neptune proporciona el método `datetime` para especificar fechas y horas para las consultas enviadas en la variante Groovy de Gremlin. Esto incluye la consola de Gremlin, cadenas de texto que utilizan la API REST HTTP y cualquier otra serialización que utilice Groovy.

> **Important**
>
> Esto solo se aplica a aquellos casos en los que envíe la consulta de Gremlin como una cadena de texto. Si utiliza una variante del lenguaje Gremlin, debe usar las clases y funciones de fecha nativas para el lenguaje. Para obtener más información, consulte la siguiente sección, the section called "Fecha y hora nativas".

A partir de TinkerPop `3.5.2` (introducido en la versión 1.1.1.0 del motor de Neptune), `datetime` forma parte de TinkerPop.

Puede utilizar el método `datetime` para almacenar y comparar fechas:

```
g.V('3').property('date',datetime('2001-02-08'))
```

```
g.V().has('date',gt(datetime('2000-01-01')))
```

Uso de la fecha y hora nativas para los datos de tiempo de GLV

Si utiliza una variante del lenguaje Gremlin (GLV), debe usar las clases y funciones de fecha y hora nativas que proporciona el lenguaje de programación para los datos de tiempo de Gremlin.

Las bibliotecas oficiales TinkerPop Java, Node.js (JavaScript), Python o .NET son bibliotecas de variante del lenguaje Gremlin.

Important

Esto solo se aplica a las bibliotecas de la variante de lenguaje Gremlin (GLV). Si utiliza un método en el que envía la consulta de Gremlin como un cadena de texto, debe utilizar el método `datetime()` proporcionado por Neptune. Esto incluye la consola de Gremlin, cadenas de texto que utilizan la API REST HTTP y cualquier otra serialización que utilice Groovy. Para obtener más información, consulte la sección anterior, the section called "`datetime( )`".

Python

A continuación se muestra un ejemplo parcial en Python que crea una única propiedad llamada "`date`" para el vértice con un ID de "`3`". Define el valor que será una fecha generada utilizando el método `datetime.now()` de Python.

```
import datetime

g.V('3').property('date',datetime.datetime.now()).next()
```

Para ver un ejemplo completo de cómo conectarse a Neptune mediante Python, consulte Uso de Python para conectarse a una instancia de base de datos de Neptune

Node.js (JavaScript)

A continuación se muestra un ejemplo parcial en JavaScript que crea una única propiedad llamada "`date`" para el vértice con un ID de "`3`". Define el valor que será una fecha generada utilizando el constructor `Date()` de Node.js.

```
g.V('3').property('date', new Date()).next()
```

Para ver un ejemplo completo de cómo conectarse a Neptune mediante Node.js, consulte Uso de Node.js para conectarse a una instancia de base de datos de Neptune

Java

A continuación se muestra un ejemplo parcial en Java que crea una única propiedad llamada "`date`" para el vértice con un ID de "`3`". Define el valor que será una fecha generada utilizando el constructor `Date()` de Java.

```
import java.util.date

g.V('3').property('date', new Date()).next();
```

Para ver un ejemplo completo de cómo conectarse a Neptune mediante Java, consulte Uso de un cliente Java para conectarse a una instancia de base de datos de Neptune

.NET (C#)

A continuación se muestra un ejemplo parcial en C# que crea una única propiedad llamada "`date`" para el vértice con un ID de "`3`". Define el valor que será una fecha generada utilizando la propiedad *`DateTime.UtcNow`* de .NET.

```
Using System;

g.V('3').property('date', DateTime.UtcNow).next()
```

Para ver un ejemplo completo de cómo conectarse a Neptune mediante C#, consulte Uso de .NET para conectarse a una instancia de base de datos de Neptune

Prácticas recomendadas del uso del cliente Java de Gremlin con Neptune

Utilice la última versión compatible del cliente Apache TinkerPop Java

Si puede, utilice siempre la última versión del cliente Java Apache TinkerPop Gremlin compatible con la versión del motor que esté utilizando. Las versiones más recientes contienen numerosas correcciones de errores que pueden mejorar la estabilidad, el rendimiento y la facilidad de uso del cliente.

Consulte El cliente Apache Java Gremlin TinkerPop para obtener una lista de las versiones del cliente que son compatibles con diversas versiones del motor de Neptune.

Vuelva a utilizar el objeto del cliente en varios subprocesos

Volver a utilizar el mismo objeto de cliente (o `GraphTraversalSource`) en varios subprocesos. Es decir, cree una instancia compartida de una clase `org.apache.tinkerpop.gremlin.driver.Client` en su aplicación, en lugar de hacerlo en cada subproceso. El objeto `Client` es seguro para subprocesos y la sobrecarga para inicializar es considerable.

Esto también se aplica a `GraphTraversalSource`, que crea un objeto `Client` internamente. Por ejemplo, el siguiente código hace que se cree una instancia para un nuevo objeto `Client`:

```
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;

  /////

GraphTraversalSource traversal = traversal()
                                    .withRemote(DriverRemoteConnection.using(cluster));
```

Cree objetos de cliente Java de Gremlin independientes para puntos de conexión de lectura y escritura

Puede aumentar el rendimiento si solo escribe en el punto de enlace del escritor y lee desde uno o más puntos de enlace de solo lectura.

```
Client readerClient = Cluster.build("https://reader-endpoint")
```

```
            ...
            .connect()

Client writerClient = Cluster.build("https://writer-endpoint")
            ...
            .connect()
```

Añada varios puntos de conexión de réplica de lectura a un grupo de conexiones Java de Gremlin

Al crear un objeto `Cluster` de Java de Gremlin, puede utilizar el método `.addContactPoint()` para agregar varias instancias de réplicas de lectura a los puntos de contacto del grupo de conexiones.

```
Cluster.Builder readerBuilder = Cluster.build()
            .port(8182)
            .minConnectionPoolSize(…)
            .maxConnectionPoolSize(…)
            ……
            .addContactPoint("reader-endpoint-1")
            .addContactPoint("reader-endpoint-2")
```

Cierre el cliente para evitar el límite de conexiones

Es importante cerrar el cliente cuando haya terminado con él para asegurarse de que el servidor cierre las WebSocket conexiones y de que se liberen todos los recursos asociados a las conexiones. Esto ocurre automáticamente si cierra el clúster mediante `Cluster.close( )`, ya que `client.close( )` se llama internamente.

Si el cliente no está cerrado correctamente, Neptune finaliza todas las WebSocket conexiones inactivas después de 20 a 25 minutos. Sin embargo, si no cierras WebSocket las conexiones de forma explícita cuando terminas de usarlas y el número de conexiones activas alcanza el límite de conexiones WebSocket simultáneas, se rechazarán las conexiones adicionales con un código de error HTTP429. En ese momento, deberá reiniciar la instancia de Neptune para cerrar las conexiones.

La recomendación de llamar a `cluster.close()` no se aplica a las funciones AWS Lambda de Java. Para obtener más información, consulte Administración de las conexiones WebSocket de Gremlin en las funciones de AWS Lambda.

Cree una nueva conexión después de una conmutación por error

En caso de una conmutación por error, el controlador de Gremlin podría continuar conectándose al escritor antiguo porque el nombre DNS del clúster está resuelto a una dirección IP. Si esto sucede, puede crear un nuevo objeto `Client` después de la conmutación por error.

Establezca **`maxInProcessPerConnection`** y **`maxSimultaneousUsagePerConnection`** con el mismo valor.

`maxInProcessPerConnection`Tanto el parámetro como el `maxSimultaneousUsagePerConnection` parámetro están relacionados con el número máximo de consultas simultáneas que se pueden enviar en una sola WebSocket conexión. Internamente, estos parámetros están correlacionados y la modificación de uno sin el otra podría dar lugar a que un cliente reciba un tiempo de espera al intentar recuperar una conexión desde el grupo de conexión del cliente.

Le recomendamos que mantenga los valores predeterminados mínimo en proceso y de uso simultáneo, y defina `maxInProcessPerConnection` y `maxSimultaneousUsagePerConnection` con el mismo valor.

El valor con el que establecer estos parámetros es una función de complejidad de consulta y el modelo de datos. Un caso de uso en el que la consulta devuelve una gran cantidad de datos requeriría más ancho de banda de conexión por consulta y, por tanto, debería tener unos valores más bajos para los parámetros y un valor más alto para `maxConnectionPoolSize`.

En cambio, en los casos en los que la consulta devuelve una menor cantidad de datos, `maxInProcessPerConnection` y `maxSimultaneousUsagePerConnection` deben establecerse en un valor más alto que `maxConnectionPoolSize`.

Envíe consultas al servidor como bytecode en lugar de como cadenas

Existen ventajas con el uso Bytecode en lugar de una cadena al enviar consultas:

- Detección temprana de sintaxis de consultas no válida: el uso de la variante Bytecode permite detectar la sintaxis de consultas no válidas en la fase de compilación. Si utiliza la variación basada en cadena, no detectará la sintaxis no válida hasta que la consulta se envíe al servidor y se devuelva un error.
- Evite las penalizaciones de rendimiento basadas en cadenas: cualquier consulta basada en cadenas, ya sea que utilice HTTP WebSockets o HTTP, da como resultado un vértice separado,

lo que implica que el objeto Vertex está formado por el identificador, la etiqueta y todas las propiedades asociadas al vértice (consulte Propiedades de los elementos).

Esto puede dar lugar a cálculos innecesarios en el servidor en casos en que las propiedades no son obligatorias. Por ejemplo, si el cliente está interesado en obtener el vértice con el ID "hakuna#1" utilizando la consulta `g.V("hakuna#1")`. Si la consulta se envía como envío basado en cadena, el servidor gastaría tiempo en recuperar el ID, la etiqueta y todas las propiedades de este vértice. Si la consulta se envía como un envío de Bytecode, el servidor solo dedica tiempo en recuperar el ID y la etiqueta del vértice.

En otras palabras, en lugar de enviar una consulta de esta manera:

```
 final Cluster cluster = Cluster.build("localhost")
                                .port(8182)
                                .maxInProcessPerConnection(32)
                                .maxSimultaneousUsagePerConnection(32)
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final Client client = cluster.connect();
     List<Result> results =
client.submit("g.V().has('name','pumba').out('friendOf').id()").all().get();
     System.out.println(verticesWithNamePumba);
 } finally {
     cluster.close();
 }
```

Envíe la consulta mediante Bytecode, tal y como se muestra a continuación:

```
 final Cluster cluster = Cluster.build("localhost")
                                .port(8182)
                                .maxInProcessPerConnection(32)
                                .maxSimultaneousUsagePerConnection(32)
                                .serializer(Serializers.GRAPHBINARY_V1D0)
                                .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
```

```
      List<Object> verticesWithNamePumba = g.V().has("name",
 "pumba").out("friendOf").id().toList();
      System.out.println(verticesWithNamePumba);
  } finally {
      cluster.close();
  }
```

Consume siempre por completo el iterador ResultSet or devuelto por una consulta

El objeto de cliente siempre debe consumir por completo `ResultSet` (en el caso de envío basado en cadena) o el iterador devuelto por `GraphTraversal`. Si los resultados de la consulta no se consumen por completo, el servidor los contiene, esperando a que el cliente termine de consumirlos.

Si la aplicación solo necesita un conjunto de resultados parcial, puede utilizar un paso `limit(X)` con su consulta para restringir el número de resultados que el servidor genera.

Añada vértices y bordes de forma masiva por lotes

Cada consulta a la base de datos de Neptune se ejecuta en el ámbito de una sola transacción, a menos que utilice una sesión. Esto significa que, si se necesita insertar una gran cantidad de datos con las consultas de Gremlin, agruparlos en lotes en un tamaño de 50-100 mejora el rendimiento al reducir el número de transacciones creadas para la carga.

Por ejemplo, agregar 5 vértices a la base de datos tendría el siguiente aspecto:

```
// Create a GraphTraversalSource for the remote connection
final GraphTraversalSource g =
 traversal().withRemote(DriverRemoteConnection.using(cluster));
// Add 5 vertices in a single query
g.addV("Person").property(T.id, "P1")
 .addV("Person").property(T.id, "P2")
 .addV("Person").property(T.id, "P3")
 .addV("Person").property(T.id, "P4")
 .addV("Person").property(T.id, "P5").iterate();
```

Deshabilitar el almacenamiento en caché de DNS en la máquina virtual de Java

En un entorno en el que desee equilibrar la carga de solicitudes en varias réplicas de lectura, debe inhabilitar el almacenamiento en caché de DNS en la máquina virtual de Java (JVM) y proporcionar el punto de enlace del lector de Neptune al crear el clúster. Desactivar la caché de DNS de la JVM garantiza que DNS se resuelva de nuevo para cada conexión nueva, de modo que las solicitudes se distribuyan en todas las réplicas de lectura. Puede hacerlo en el código de inicialización de su aplicación con la siguiente línea:

```
java.security.Security.setProperty("networkaddress.cache.ttl", "0");
```

Sin embargo, el código del cliente Java de Amazon Gremlin proporciona una solución más completa y sólida para el equilibrio de carga. GitHub El cliente Gremlin de Amazon Java conoce la topología de su clúster y distribuye de forma equitativa las conexiones y solicitudes entre un conjunto de instancias del clúster de Neptune. Consulte esta entrada del blog para ver un ejemplo de una función de Lambda de Java que usa ese cliente.

Opcionalmente, establezca tiempos de espera al nivel de consulta

Neptune ofrece la posibilidad de definir un tiempo de espera para sus consultas con la opción de grupo de parámetros `neptune_query_timeout` (consulte Parámetros). A partir de la versión 3.3.7 del cliente Java, sin embargo, también puede anular el tiempo de espera global, con un código como se indica a continuación:

```
 final Cluster cluster = Cluster.build("localhost")
                               .port(8182)
                               .maxInProcessPerConnection(32)
                               .maxSimultaneousUsagePerConnection(32)
                               .serializer(Serializers.GRAPHBINARY_V1D0)
                               .create();

 try {
     final GraphTraversalSource g =
traversal().withRemote(DriverRemoteConnection.using(cluster));
     List<Object> verticesWithNamePumba = g.with(ARGS_EVAL_TIMEOUT,
500L).V().has("name", "pumba").out("friendOf").id().toList();
     System.out.println(verticesWithNamePumba);
 } finally {
```

```
        cluster.close();
    }
```

O bien, para envío de consultas basado en cadenas, el código sería el siguiente:

```
RequestOptions options = RequestOptions.build().timeout(500).create();
List<Result> result = client.submit("g.V()", options).all().get();
```

> **Note**
>
> Es posible incurrir en costos inesperados si se establece un valor de tiempo de espera de consulta demasiado alto, especialmente en una instancia sin servidor. Si la configuración del tiempo de espera no es razonable, es posible que la consulta siga ejecutándose durante mucho más tiempo del esperado, lo que dará lugar a costos que no había previsto. Esto ocurre especialmente en una instancia sin servidor que podría escalarse verticalmente hasta convertirse en un tipo de instancia grande y caro mientras se ejecuta la consulta.
> Puede evitar gastos inesperados de este tipo si utiliza un valor de tiempo de espera de consulta que se adapte al tiempo de ejecución esperado y que solo provoque un tiempo de espera inusualmente largo.

Solución de problemas de `java.util.concurrent.TimeoutException`

El cliente Java de Gremlin lanza un mensaje `java.util.concurrent.TimeoutException` cuando se agota el tiempo de espera de una solicitud de Gremlin en el propio cliente mientras espera a que quede disponible un espacio en una de las conexiones. WebSocket Este tiempo de espera se controla mediante el parámetro configurable del lado del cliente `maxWaitForConnection`.

> **Note**
>
> Como las solicitudes cuyo tiempo de espera se agota en el cliente nunca se envían al servidor, no se reflejan en ninguna de las métricas recopiladas en el servidor, por ejemplo `GremlinRequestsPerSec`.

Este tipo de tiempo de espera se produce generalmente de una de estas dos formas:

- El servidor ha alcanzado su capacidad máxima. Si este es el caso, la cola del servidor se llena, lo que puedes detectar monitorizando la métrica de solicitudes. MainRequest QueuePending CloudWatch La cantidad de consultas paralelas que el servidor puede gestionar depende del tamaño de la instancia.

  Si la métrica `MainRequestQueuePendingRequests` no muestra una acumulación de solicitudes pendientes en el servidor, el servidor puede gestionar más solicitudes y el tiempo de espera se debe a una limitación del lado del cliente.

- Limitación de las solicitudes por parte del cliente. Por lo general, esto se puede solucionar cambiando los ajustes de la configuración del cliente.

  El número máximo de solicitudes paralelas que puede enviar el cliente se puede calcular de manera aproximada de la siguiente forma:

```
maxParallelQueries = maxConnectionPoolSize * Max( maxSimultaneousUsagePerConnection,
 maxInProcessPerConnection )
```

  Si se envía al cliente una cantidad superior a la especificada en `maxParallelQueries`, se producen excepciones `java.util.concurrent.TimeoutException`. Por lo general, hay varias maneras de solucionar esto:

  - Aumente la duración del tiempo de espera de la conexión. Si la latencia no es crucial para su aplicación, aumente la configuración de `maxWaitForConnection` del cliente. Por tanto, el cliente espera más antes de agotar el tiempo de espera, lo que a su vez puede aumentar la latencia.
  - Aumente el número máximo de solicitudes por conexión. Esto permite enviar más solicitudes con la misma WebSocket conexión. Para ello, aumente los ajustes de `maxSimultaneousUsagePerConnection` y `maxInProcessPerConnection` del cliente. Por lo general, estos ajustes deben tener el mismo valor.
  - Aumente el número de conexiones en el grupo de conexiones. Para ello, aumente el ajuste de `maxConnectionPoolSize` del cliente. El coste se traduce en un mayor consumo de recursos, ya que cada conexión utiliza memoria y un descriptor de archivos del sistema operativo, y requiere un SSL y WebSocket un protocolo de enlace durante la inicialización.

Prácticas recomendadas de Neptune con openCypher y Bolt

Siga estas prácticas recomendadas cuando use el lenguaje de consulta openCypher y el protocolo Bolt con Neptune. Para obtener más información sobre el uso de openCypher en Neptune, consulte Acceso al gráfico de Neptune con openCypher.

En las consultas, utilice preferentemente bordes dirigidos en lugar de bidireccionales

Cuando Neptune optimiza las consultas, los bordes bidireccionales dificultan la creación de planes de consulta óptimos. Cuando los planes no son óptimos, el motor debe realizar trabajos innecesarios y eso se traduce en un rendimiento inferior.

Por consiguiente, utilice bordes dirigidas en lugar de bidireccionales siempre que sea posible. Por ejemplo, utilice

```
MATCH p=(:airport {code: 'ANC'})-[:route]->(d) RETURN p)
```

en lugar de

```
MATCH p=(:airport {code: 'ANC'})-[:route]-(d) RETURN p)
```

En realidad, la mayoría de los modelos de datos no necesitan atravesar los bordes en ambas direcciones, por lo que se pueden lograr mejoras significativas de rendimiento en las consultas si se opta por utilizar bordes dirigidos.

Si su modelo de datos requiere atravesar bordes bidireccionales, haga que el primer nodo (lado izquierdo) del patrón `MATCH` sea el nodo con el filtrado más restrictivo.

Utilicemos como ejemplo: “Encuéntrame todas las `routes` de ida y vuelta al aeropuerto de ANC“. Escriba esta consulta para empezar en el aeropuerto de ANC, de esta forma:

```
MATCH p=(src:airport {code: 'ANC'})-[:route]-(d) RETURN p
```

El motor puede realizar la cantidad mínima de trabajo necesaria para satisfacer la consulta, ya que el nodo más restringido se coloca como el primer nodo (a la izquierda) del patrón. A continuación, el motor puede optimizar la consulta.

Esto es mucho mejor que filtrar el aeropuerto de ANC al final del patrón, de la siguiente manera:

```
MATCH p=(d)-[:route]-(src:airport {code: 'ANC'}) RETURN p
```

Cuando el nodo más restringido no se coloca primero en el patrón, el motor debe realizar un trabajo adicional porque no puede optimizar la consulta y tiene que realizar búsquedas adicionales para llegar a los resultados.

Neptune no admite múltiples consultas simultáneas en una transacción

Aunque el propio controlador de Bolt permite consultas simultáneas en una transacción, Neptune no admite múltiples consultas en una transacción que se ejecute simultáneamente. En su lugar, Neptune requiere que se ejecuten varias consultas en una transacción de forma secuencial y que los resultados de cada consulta se consuman por completo antes de que se inicie la siguiente consulta.

En el siguiente ejemplo, se muestra cómo usar Bolt para ejecutar varias consultas de forma secuencial en una transacción, de modo que los resultados de cada una de ellas se consuman por completo antes de que comience la siguiente:

```
final String query = "MATCH (n) RETURN n";

try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
  try (Session session = driver.session(readSessionConfig)) {
    try (Transaction trx = session.beginTransaction()) {
      final Result res_1 = trx.run(query);
      Assert.assertEquals(10000, res_1.list().size());
      final Result res_2 = trx.run(query);
      Assert.assertEquals(10000, res_2.list().size());
    }
  }
}
```

Cree una nueva conexión después de una conmutación por error

En caso de una conmutación por error, el controlador de Bolt puede seguir conectándose a la antigua instancia de escritor en lugar de a la nueva instancia activa, ya que el nombre DNS se ha resuelto en una dirección IP específica.

Para evitarlo, cierre el objeto `Driver` y vuelva a conectarlo después de cualquier conmutación por error.

Gestión de conexiones para aplicaciones de larga duración

Al crear aplicaciones de larga duración, como las que se ejecutan en contenedores o en instancias de Amazon EC2, cree una instancia de un objeto `Driver` una vez y, a continuación, reutilícelo durante toda la vida útil de la aplicación. El objeto `Driver` es seguro para subprocesos y la sobrecarga para inicializar es considerable.

Manejo de conexiones para AWS Lambda

No se recomienda el uso de destornilladores dentro de AWS Lambda las funciones debido a su sobrecarga de conexión y a los requisitos de administración. En su lugar, utilice el punto de conexión HTTPS.

Cierre los objetos del controlador cuando haya terminado

Asegúrese de cerrar el cliente cuando haya terminado con él para que el servidor cierre las conexiones de Bolt y se liberen todos los recursos asociados a las conexiones. Esto se realiza automáticamente si cierra el controlador utilizando `driver.close()`.

Si el controlador no se cierra correctamente, Neptune finaliza todas las conexiones de Bolt inactivas pasados 20 minutos, o después de 10 días si utiliza la autenticación de IAM.

Neptune no admite más de 1000 conexiones de Bolt simultáneas. Si no cierra las conexiones de forma explícita cuando termine de usarlas y el número de conexiones activas alcanza ese límite de 1000, cualquier nuevo intento de conexión fallará.

Use modos de transacción explícitos para leer y escribir

Cuando se utilizan transacciones con Neptune y el controlador de Bolt, es mejor establecer explícitamente el modo de acceso para las transacciones de lectura y escritura con la configuración correcta.

Transacciones de solo lectura

En el caso de las transacciones de solo lectura, si no se introduce la configuración de modo de acceso adecuada al crear la sesión, se utiliza el nivel de aislamiento predeterminado, que es el aislamiento de las consultas de mutación. Por ello, es importante que en las transacciones de solo lectura se establezca el modo de acceso a `read` de forma explícita.

Ejemplo de transacción de lectura con confirmación automática:

```
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.READ)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
  throw e;
} finally {
  driver.close()
}
```

Ejemplo de transacción de lectura:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.READ)
  .build();
driver.session(sessionConfig).readTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

En ambos casos, el aislamiento de SNAPSHOT se logra mediante la semántica de transacciones de solo lectura de Neptune.

Como las réplicas de lectura solo aceptan consultas de solo lectura, cualquier consulta que se envíe a una réplica de lectura se ejecuta en función de una semántica de aislamiento de SNAPSHOT.

No hay lecturas incorrectas ni lecturas no repetibles para las transacciones de solo lectura.

Transacciones de solo lectura

En el caso de las consultas de mutación, existen tres mecanismos diferentes para crear una transacción de escritura, cada uno de los cuales se ilustra a continuación:

Ejemplo de transacción de escritura implícita:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
driver.session(sessionConfig).writeTransaction(
  new TransactionWork<List<String>>() {
    @Override
    public List<String> execute(org.neo4j.driver.Transaction tx) {
      (Add your application code here)
    }
  }
);
```

Ejemplo de transacción de escritura con confirmación automática:

```
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.Write)
  .build();
Session session = driver.session(sessionConfig);
try {
  (Add your application code here)
} catch (final Exception e) {
    throw e;
} finally {
    driver.close()
}
```

Ejemplo de transacción de escritura explícita:

```
Driver driver = GraphDatabase.driver(url, auth, config);
SessionConfig sessionConfig = SessionConfig
  .builder()
  .withFetchSize(1000)
  .withDefaultAccessMode(AccessMode.WRITE)
  .build();
Transaction beginWriteTransaction = driver.session(sessionConfig).beginTransaction();
  (Add your application code here)
beginWriteTransaction.commit();
```

```
driver.close();
```

Niveles de aislamiento para transacciones de escritura

- Las lecturas realizadas como parte de las consultas de mutación se ejecutan de forma aislada en las transacciones de `READ COMMITTED`.
- No hay lecturas incorrectas en las lecturas realizadas como parte de las consultas de mutación.
- Los registros y rangos de registros se bloquean al leer una consulta de mutación.
- Cuando una transacción de mutación ha leído un rango del índice, existe una gran garantía de que este rango no se modificará mediante ninguna transacción simultánea hasta el final de la lectura.

Las consultas de mutación no son seguras para los subprocesos.

Para conocer los conflictos, consulte Resolución de conflictos mediante tiempos de espera de bloqueo.

Las consultas de mutación no se vuelven a intentar automáticamente en caso de error.

Lógica de reintentos para las excepciones

Para todas las excepciones que permiten un reintento, generalmente es mejor utilizar una estrategia de reintento y retroceso exponencial que proporcione tiempos de espera progresivamente más largos entre los reintentos, a fin de gestionar mejor los problemas transitorios, como los errores `ConcurrentModificationException`. A continuación, se muestra un ejemplo de patrón de reintento y retroceso exponencial:

```
public static void main() {
  try (Driver driver = getDriver(HOST_BOLT, getDefaultConfig())) {
    retriableOperation(driver, "CREATE (n {prop:'1'})")
        .withRetries(5)
        .withExponentialBackoff(true)
        .maxWaitTimeInMilliSec(500)
        .call();
  }
}

protected RetryableWrapper retriableOperation(final Driver driver, final String query){
  return new RetryableWrapper<Void>() {
    @Override
    public Void submit() {
```

```
      log.info("Performing graph Operation in a retry manner......");
      try (Session session = driver.session(writeSessionConfig)) {
        try (Transaction trx =  session.beginTransaction()) {
            trx.run(query).consume();
            trx.commit();
        }
      }
      return null;
    }

    @Override
    public boolean isRetryable(Exception e) {
      if (isCME(e)) {
        log.debug("Retrying on exception.... {}", e);
        return true;
      }
      return false;
    }

    private boolean isCME(Exception ex) {
      return ex.getMessage().contains("Operation failed due to conflicting concurrent
 operations");
    }
  };
}



/**
 * Wrapper which can retry on certain condition. Client can retry operation using this
 class.
 */
@Log4j2
@Getter
public abstract class RetryableWrapper<T> {

  private long retries = 5;
  private long maxWaitTimeInSec = 1;
  private boolean exponentialBackoff = true;

  /**
   * Override the method with custom implementation, which will be called in retryable
 block.
   */
```

```
  public abstract T submit() throws Exception;

  /**
   * Override with custom logic, on which exception to retry with.
   */
  public abstract boolean isRetryable(final Exception e);

  /**
   * Define the number of retries.
   *
   * @param retries -no of retries.
   */
  public RetryableWrapper<T> withRetries(final long retries) {
    this.retries = retries;
    return this;
  }

  /**
   * Max wait time before making the next call.
   *
   * @param time - max polling interval.
   */
  public RetryableWrapper<T> maxWaitTimeInMilliSec(final long time) {
    this.maxWaitTimeInSec = time;
    return this;
  }

  /**
   * ExponentialBackoff coefficient.
   */
  public RetryableWrapper<T> withExponentialBackoff(final boolean expo) {
    this.exponentialBackoff = expo;
    return this;
  }

  /**
   * Call client method which is wrapped in submit method.
   */
  public T call() throws Exception {
    int count = 0;
    Exception exceptionForMitigationPurpose = null;
    do {
      final long waitTime = exponentialBackoff ? Math.min(getWaitTimeExp(retries),
maxWaitTimeInSec) : 0;
```

```
      try {
          return submit();
      } catch (Exception e) {
        exceptionForMitigationPurpose = e;
        if (isRetryable(e) && count < retries) {
          Thread.sleep(waitTime);
          log.debug("Retrying on exception attempt - {} on exception cause - {}",
 count, e.getMessage());
        } else if (!isRetryable(e)) {
          log.error(e.getMessage());
          throw new RuntimeException(e);
        }
      }
    } while (++count < retries);

    throw new IOException(String.format(
          "Retry was unsuccessful.... attempts %d. Hence throwing exception " + "back
 to the caller...", count),
          exceptionForMitigationPurpose);
  }

  /*
   * Returns the next wait interval, in milliseconds, using an exponential backoff
   * algorithm.
   */
  private long getWaitTimeExp(final long retryCount) {
    if (0 == retryCount) {
      return 0;
    }
    return ((long) Math.pow(2, retryCount) * 100L);
  }
}
```

Establezca varias propiedades a la vez mediante una sola cláusula SET

En lugar de utilizar varias cláusulas SET para establecer propiedades individuales, utilice un mapa para establecer varias propiedades de una entidad a la vez.

Puede usar:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n += {property1 : 'value1',
property2 : 'value2',
```

```
property3 = 'value3'}
```

En lugar de:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n.property1 = 'value1'
SET n.property2 = 'value2'
SET n.property3 = 'value3'
```

La cláusula SET acepta una sola propiedad o un mapa. Si se actualizan varias propiedades en una sola entidad, el uso de una sola cláusula SET con un mapa permite que las actualizaciones se realicen en una sola operación en lugar de varias operaciones, que se pueden ejecutar de manera más eficaz.

Utilice la cláusula SET para eliminar varias propiedades a la vez

Cuando se utiliza el lenguaje OpenCypher, se utiliza REMOVE para eliminar propiedades de una entidad. En Neptune, cada propiedad que se elimina requiere una operación independiente, lo que añade latencia de consulta. En su lugar, puede usar SET con un mapa para establecer todos los valores de las propiedades`null`, lo que en Neptune equivale a eliminar propiedades. Neptune tendrá un mayor rendimiento cuando sea necesario eliminar varias propiedades de una sola entidad.

Uso:

```
WITH {prop1: null, prop2: null, prop3: null} as propertiesToRemove
MATCH (n)
SET n += propertiesToRemove
```

En lugar de:

```
MATCH (n)
REMOVE n.prop1, n.prop2, n.prop3
```

Utilice consultas parametrizadas

Se recomienda utilizar siempre consultas parametrizadas al realizar consultas con OpenCypher. El motor de consultas puede aprovechar las consultas parametrizadas repetidas para funciones como la memoria caché del plan de consultas, mientras que la invocación repetida de la misma estructura

parametrizada con diferentes parámetros puede aprovechar los planes almacenados en caché. El plan de consultas generado para las consultas parametrizadas se almacena en caché y se reutiliza solo cuando se completa en 100 ms y los tipos de parámetros son NUMBER, BOOLEAN o STRING.

Uso:

```
MATCH (n:foo) WHERE id(n) = $id RETURN n
```

Con los parámetros:

```
parameters={"id": "first"}
parameters={"id": "second"}
parameters={"id": "third"}
```

En lugar de:

```
MATCH (n:foo) WHERE id(n) = "first" RETURN n
MATCH (n:foo) WHERE id(n) = "second" RETURN n
MATCH (n:foo) WHERE id(n) = "third" RETURN n
```

Utilice mapas aplanados en lugar de mapas anidados en la cláusula UNWIND

Una estructura anidada profunda puede restringir la capacidad del motor de consultas para generar un plan de consultas óptimo. Para solucionar parcialmente este problema, los siguientes patrones definidos crearán planes óptimos para los siguientes escenarios:

- Escenario 1: RELÁJESE con una lista de literales cifrados, que incluye NUMBER, STRING y BOOLEAN.
- Escenario 2: relájese con una lista de mapas aplanados, que incluye solo literales cifrados (NUMBER, STRING, BOOLEAN) como valores.

Al escribir una consulta que contenga la cláusula UNWIND, utilice la recomendación anterior para mejorar el rendimiento.

Ejemplo de escenario 1:

```
UNWIND $ids as x
```

```
MATCH(t:ticket {`~id`: x})
```

Con parámetros:

```
parameters={
  "ids": [1, 2, 3]
}
```

Un ejemplo para el escenario 2 es generar una lista de nodos para CREARLOS o FUSIONARLOS. En lugar de emitir varias sentencias, utilice el siguiente patrón para definir las propiedades como un conjunto de mapas aplanados:

```
UNWIND $props as p
CREATE(t:ticket {title: p.title, severity:p.severity})
```

Con parámetros:

```
parameters={
  "props": [
    {"title": "food poisoning", "severity": "2"},
    {"title": "Simone is in office", "severity": "3"}
  ]
}
```

En lugar de objetos de nodo anidados como:

```
UNWIND $nodes as n
CREATE(t:ticket n.properties)
```

Con parámetros:

```
parameters={
  "nodes": [
    {"id": "ticket1", "properties": {"title": "food poisoning", "severity": "2"}},
    {"id": "ticket2", "properties": {"title": "Simone is in office", "severity": "3"}}
  ]
}
```

Coloque los nodos más restrictivos en el lado izquierdo de las expresiones de ruta de longitud variable (VLP)

En las consultas de ruta de longitud variable (VLP), el motor de consultas optimiza la evaluación al elegir iniciar el recorrido en el lado izquierdo o derecho de la expresión. La decisión se basa en la cardinalidad de los patrones de los lados izquierdo y derecho. La cardinalidad es el número de nodos que coinciden con el patrón especificado.

- Si el patrón correcto tiene una cardinalidad de uno, entonces el lado derecho será el punto de partida.
- Si el lado izquierdo y el derecho tienen una cardinalidad igual a uno, la expansión se comprueba en ambos lados y comienza en el lado con la expansión más pequeña. La expansión es el número de bordes salientes o entrantes del nodo de la izquierda y del nodo de la derecha de la expresión VLP. Esta parte de la optimización solo se usa si la relación VLP es unidireccional y se proporciona el tipo de relación.
- De lo contrario, el lado izquierdo será el punto de partida.

Para una cadena de expresiones VLP, esta optimización solo se puede aplicar a la primera expresión. Las demás VLP se evalúan empezando por el lado izquierdo. Como ejemplo, supongamos que la cardinalidad de (a), (b) es uno y la cardinalidad de (c) es mayor que uno.

- `(a)-[*1..]->(c)`: La evaluación comienza con (a).
- `(c)-[*1..]->(a)`: La evaluación comienza con (a).
- `(a)-[*1..]-(c)`: La evaluación comienza con (a).
- `(c)-[*1..]-(a)`: La evaluación comienza con (a).

Ahora supongamos que los bordes entrantes de (a) sean dos y los bordes salientes de (a) tres, los bordes entrantes de (b) cuatro y los bordes salientes de (b) cinco.

- `(a)-[*1..]->(b)`: La evaluación comienza con (a) ya que los bordes salientes de (a) son menores que los bordes entrantes de (b).
- `(a)<-[*1..]-(b)`: La evaluación comienza con (a) ya que los bordes entrantes de (a) son menores que los bordes salientes de (b).

Como regla general, coloque el patrón más restrictivo en el lado izquierdo de una expresión de VLP.

Evite las comprobaciones redundantes de las etiquetas de los nodos mediante el uso de nombres de relación granulares

Al optimizar el rendimiento, el uso de etiquetas de relación que sean exclusivas de los patrones de los nodos permite eliminar el filtrado de etiquetas en los nodos. Considere un modelo gráfico en el que la relación solo `likes` se utilice para definir una relación entre dos `person` nodos. Podríamos escribir la siguiente consulta para encontrar este patrón:

```
MATCH (n:person)-[:likes]->(m:person)
RETURN n, m
```

La `person` marca de etiquetas en n y m es redundante, ya que definimos la relación para que solo aparezca cuando ambas sean del mismo tipo`person`. Para optimizar el rendimiento, podemos escribir la consulta de la siguiente manera:

```
MATCH (n)-[:likes]->(m)
RETURN n, m
```

Este patrón también se puede aplicar cuando las propiedades son exclusivas de una sola etiqueta de nodo. Suponga que solo `person` los nodos tienen la propiedad`email`, por lo que comprobar que la etiqueta del nodo coincida `person` es redundante. Escribiendo esta consulta como:

```
MATCH (n:person)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Es menos eficiente que escribir esta consulta como:

```
MATCH (n)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Solo debe adoptar este patrón cuando el rendimiento sea importante y haya comprobado el proceso de modelado para garantizar que estas etiquetas de borde no se reutilicen en patrones que involucren otras etiquetas de nodos. Si posteriormente introduce una `email` propiedad en otra etiqueta de nodo, por ejemplo`company`, los resultados diferirán entre estas dos versiones de la consulta.

Especifique las etiquetas de los bordes siempre que sea posible

Se recomienda proporcionar una etiqueta de borde siempre que sea posible al especificar un borde en un patrón. Considere la siguiente consulta de ejemplo, que se utiliza para vincular a todas las personas que viven en una ciudad con todas las personas que la visitaron.

```
MATCH (person)-->(city {country: "US"})-->(anotherPerson)
RETURN person, anotherPerson
```

Si su modelo gráfico vincula a las personas con nodos distintos de las ciudades mediante varias etiquetas de borde, al no especificar la etiqueta final, Neptune tendrá que evaluar rutas adicionales que luego se descartarán. En la consulta anterior, como no se proporcionó una etiqueta de borde, el motor primero trabaja más y, a continuación, filtra los valores para obtener el resultado correcto. Una versión mejor de la consulta anterior podría ser:

```
MATCH (person)-[:livesIn]->(city {country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Esto no solo ayuda en la evaluación, sino que también permite al planificador de consultas crear mejores planes. Incluso puedes combinar esta práctica recomendada con comprobaciones redundantes de las etiquetas de los nodos para eliminar la marca de la etiqueta de la ciudad y escribir la consulta de la siguiente manera:

```
MATCH (person)-[:livesIn]->({country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Evita usar la cláusula WITH siempre que sea posible

La cláusula WITH de OpenCypher actúa como un límite en el que se encuentra todo lo que se ejecuta antes de ejecutarse y, a continuación, los valores resultantes se pasan al resto de la consulta. La cláusula WITH es necesaria cuando se requiere una agregación provisional o se desea limitar el número de resultados, pero aparte de eso, se debe intentar evitar el uso de la cláusula WITH. La orientación general consiste en eliminar estas sencillas cláusulas WITH (sin agregar, ordenar por o limitar) para que el planificador de consultas pueda trabajar en toda la consulta y crear un plan óptimo a nivel mundial. Como ejemplo, supongamos que escribiste una consulta para mostrar a todas las personas que viven en`India`:

```
MATCH (person)-[:lives_in]->(city)
```

```
WITH person, city
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

En la versión anterior, la cláusula WITH restringe la ubicación anterior `(person)-[:lives_in]->(city)` del patrón `(city)-[:part_of]->(country {name: 'India'})` (que es más restrictivo). Esto hace que el plan no sea óptimo. Una optimización de esta consulta sería eliminar la cláusula WITH y dejar que el planificador calcule el mejor plan.

```
MATCH (person)-[:lives_in]->(city)
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Coloque los filtros restrictivos lo antes posible en la consulta

En todos los escenarios, la colocación temprana de los filtros en la consulta ayuda a reducir las soluciones intermedias que debe tener en cuenta un plan de consultas. Esto significa que se necesita menos memoria y menos recursos de cómputo para ejecutar la consulta.

El siguiente ejemplo le ayuda a entender estos impactos. Supongamos que escribe una consulta para obtener todas las personas que viven allí`India`. Una versión de la consulta podría ser:

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WITH country, collect(n.firstName + " "  + n.lastName) AS result
WHERE country.name = 'India'
RETURN result
```

La versión anterior de la consulta no es la forma más óptima de lograr este caso de uso. El filtro `country.name = 'India'` aparece más adelante en el patrón de consulta. Primero recopilará a todas las personas y su lugar de residencia, las agrupará por país y, a continuación, filtrará solo por el grupo correspondiente`country.name = India`. La forma óptima de consultar solo a las personas que viven en `India` y, a continuación, realizar la agregación de recopilaciones.

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WHERE country.name = 'India'
RETURN collect(n.firstName + " "  + n.lastName) AS result
```

Una regla general es colocar un filtro lo antes posible después de introducir la variable.

Compruebe de forma explícita si existen propiedades

Según la semántica de OpenCypher, el acceso a una propiedad equivale a una unión opcional y debe conservar todas las filas aunque la propiedad no exista. Si, a partir del esquema gráfico, sabe que una propiedad determinada siempre existirá para esa entidad, comprobar de forma explícita la existencia de esa propiedad permite al motor de consultas crear planes óptimos y mejorar el rendimiento.

Considera un modelo gráfico en el que los nodos de un tipo `person` siempre tengan una propiedad`name`. En lugar de hacer esto:

```
MATCH (n:person)
RETURN n.name
```

Compruebe de forma explícita la existencia de la propiedad en la consulta con la opción NO ES NULA:

```
MATCH (n:person)
WHERE n.name IS NOT NULL
RETURN n.name
```

No utilice una ruta con nombre (a menos que sea obligatoria)

La ruta designada en una consulta siempre conlleva un coste adicional, lo que puede suponer una penalización en términos de mayor latencia y uso de memoria. Analice la siguiente consulta:

```
MATCH p = (n)-[:commentedOn]->(m)
WITH p, m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH p, m, n, distinct(o) as o1
RETURN p, m.name, n.name, o1.name
```

En la consulta anterior, suponiendo que solo queremos conocer las propiedades de los nodos, no es necesario utilizar la ruta «p». Al especificar la ruta nombrada como una variable, la operación de agregación mediante DISTINCT resultará costosa tanto en términos de tiempo como de uso de memoria. Una versión más optimizada de la consulta anterior podría ser:

```
MATCH (n)-[:commentedOn]->(m)
```

```
WITH m, n, n.score + m.score as total
WHERE total > 100
MATCH (m)-[:commentedON]->(o)
WITH m, n, distinct(o) as o1
RETURN m.name, n.name, o1.name
```

Evite COLLECT (DISTINCT ())

COLLECT (DISTINCT ()) se usa siempre que se va a formar una lista que contenga valores distintos. COLLECT es una función de agregación y la agrupación se realiza en función de la proyección de claves adicionales en la misma declaración. Cuando se usa distinct, la entrada se divide en varios fragmentos, donde cada fragmento indica un grupo para su reducción. El rendimiento se verá afectado a medida que aumente el número de grupos. En Neptune, es mucho más eficiente realizar DISTINCT antes de recopilar/formar la lista. Esto permite que la agrupación se realice directamente en las teclas de agrupación de todo el fragmento.

Analice la siguiente consulta:

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH n, collect(distinct(p.post_id)) as post_list
RETURN n, post_list
```

Una forma más óptima de escribir esta consulta es:

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH DISTINCT n, p.post_id as postId
WITH n, collect(postId) as post_list
RETURN n, post_list
```

Prefiera la función de propiedades a la búsqueda de propiedades individuales al recuperar todos los valores de las propiedades

La `properties()` función se utiliza para devolver un mapa que contiene todas las propiedades de una entidad y es mucho más eficaz que devolver las propiedades de forma individual.

Suponiendo que `Person` los nodos contienen 5 propiedades `firstNamelastName`,`age`,`dept`, y`company`, sería preferible realizar la siguiente consulta:

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
```

```
RETURN properties(n) as personDetails
```

En lugar de usar:

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN n.firstName, n.lastName, n.age, n.dept, n.company

=== OR ===

MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN {firstName: n.firstName, lastName: n.lastName, age: n.age,
department: n.dept, company: n.company} as personDetails
```

Realice cálculos estáticos fuera de la consulta

Se recomienda resolver los cálculos estáticos (operaciones simples matemáticas/de cadenas) en el lado del cliente. Considera este ejemplo en el que quieres encontrar a todas las personas con un año o menos de edad que el autor:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= ($age + 1)
RETURN m
```

En este caso, `$age` se inserta en la consulta mediante parámetros y, a continuación, se añade a un valor fijo. A continuación, se compara este valor con`p.age`. En su lugar, un enfoque mejor sería hacer la suma desde el lado del cliente y pasar el valor calculado como un parámetro $ageplusone. Esto ayuda al motor de consultas a crear planes optimizados y evita el cálculo estático para cada fila entrante. Siguiendo estas pautas, una versión más eficiente de la consulta sería:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= $ageplusone
RETURN m
```

Entradas por lotes utilizando UNWIND en lugar de declaraciones individuales

Siempre que sea necesario ejecutar la misma consulta para diferentes entradas, en lugar de ejecutar una consulta por entrada, sería mucho más eficaz ejecutar una consulta para un lote de entradas.

Si quieres fusionar un conjunto de nodos, una opción es ejecutar una consulta de fusión por entrada:

```
MERGE (n:Person {`~id`: $id})
SET n.name = $name, n.age = $age, n.employer = $employer
```

Con parámetros:

```
params = {id: '1', name: 'john', age: 25, employer: 'Amazon'}
```

La consulta anterior debe ejecutarse para cada entrada. Si bien este enfoque funciona, puede requerir la ejecución de muchas consultas para un conjunto grande de entradas. En este escenario, el procesamiento por lotes puede ayudar a reducir el número de consultas ejecutadas en el servidor, así como a mejorar el rendimiento general.

Use el siguiente patrón:

```
UNWIND $persons as person
MERGE (n:Person {`~id`: person.id})
SET n += person
```

Con parámetros:

```
params = {persons: [{id: '1', name: 'john', age: 25, employer: 'Amazon'},
{id: '2', name: 'jack', age: 28, employer: 'Amazon'},
{id: '3', name: 'alice', age: 24, employer: 'Amazon'}...]}
```

Se recomienda experimentar con diferentes tamaños de lote para determinar qué es lo que mejor se adapta a su carga de trabajo.

Prefiera usar identificadores personalizados para el nodo o la relación

Neptune permite a los usuarios asignar identificadores de forma explícita a los nodos y las relaciones. El ID debe ser único a nivel mundial en el conjunto de datos y determinista para que sea útil. Un identificador determinista se puede utilizar como mecanismo de búsqueda o filtrado, al igual que las propiedades; sin embargo, utilizar un identificador está mucho más optimizado desde la perspectiva de la ejecución de consultas que utilizar propiedades. El uso de identificadores personalizados tiene varias ventajas:

- Las propiedades de una entidad existente pueden ser nulas, pero el ID debe existir. Esto permite que el motor de consultas utilice una unión optimizada durante la ejecución.

- Cuando se ejecutan consultas de mutación simultáneas, las probabilidades de que se produzcan excepciones de modificación simultánea (CME) se reducen considerablemente cuando se utilizan los ID para acceder a los nodos, ya que los identificadores se bloquean menos que las propiedades debido a su unicidad obligatoria.
- El uso de identificadores evita la posibilidad de crear datos duplicados, ya que Neptune impone la unicidad de los identificadores, a diferencia de las propiedades.

En el siguiente ejemplo de consulta se utiliza un ID personalizado:

> **Note**
>
> La propiedad `~id` se usa para especificar el ID, mientras que solo `id` se almacena como cualquier otra propiedad.

```
CREATE (n:Person {`~id`: '1', name: 'alice'})
```

Sin usar un ID personalizado:

```
CREATE (n:Person {id: '1', name: 'alice'})
```

Si utiliza este último mecanismo, no se aplica la unicidad y puede ejecutar la consulta más adelante:

```
CREATE (n:Person {id: '1', name: 'john'})
```

Esto crea un segundo nodo con `id=1` nombre`john`. En este escenario, ahora tendrías dos nodos con `id=1` un nombre diferente: (alice y john).

Evite realizar **`~id`** cálculos en la consulta

Cuando utilice identificadores personalizados en las consultas, realice siempre cálculos estáticos fuera de las consultas y proporcione estos valores en los parámetros. Cuando se proporcionan valores estáticos, el motor puede optimizar mejor las búsquedas y evitar escanear y filtrar estos valores.

Si desea crear bordes entre los nodos que existen en la base de datos, una opción podría ser:

```
UNWIND $sections as section
```

```
MATCH (s:Section {`~id`: 'Sec-' + section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Con parámetros:

```
parameters={sections: [{id: '1'}, {id: '2'}]}
```

En la consulta anterior, `id` la sección se calcula en la consulta. Como el cálculo es dinámico, el motor no puede insertar los identificadores de forma estática y termina escaneando todos los nodos de la sección. A continuación, el motor realiza un filtrado posterior para los nodos necesarios. Esto puede resultar costoso si hay muchos nodos de sección en la base de datos.

Una mejor forma de lograrlo es añadir los identificadores `Sec-` que se van a transferir a la base de datos:

```
UNWIND $sections as section
MATCH (s:Section {`~id`: section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Con parámetros:

```
parameters={sections: [{id: 'Sec-1'}, {id: 'Sec-2'}]}
```

Prácticas recomendadas de Neptune con SPARQL

Siga estas prácticas recomendadas cuando utilice el lenguaje de consulta SPARQL con Neptune. Para obtener más información sobre el uso de SPARQL en Neptune, consulte Acceso al gráfico de Neptune con SPARQL.

Consulta de todos los gráficos con nombre de forma predeterminada

Amazon Neptune asocia cada triple con un gráfico con nombre. El gráfico predeterminado se define como la unión de todos los gráficos con nombre.

Si envía una consulta SPARQL sin especificar explícitamente un gráfico mediante la palabra clave `GRAPH` o construcciones como `FROM NAMED`, Neptune siempre tiene en cuenta todos los triples en su instancia de base de datos. Por ejemplo, la siguiente consulta devuelve todos los triples desde un punto de conexión de SPARQL de Neptune:

```
SELECT * WHERE { ?s ?p ?o }
```

Los triples que aparecen en más de un gráfico se devuelven solo una vez.

Para obtener información sobre la especificación de gráfico predeterminado, consulte la sección Conjunto de datos de RDF de la especificación del lenguaje de consulta SPARQL 1.1.

Especificación de un gráfico con nombre para la carga

Amazon Neptune asocia cada triple con un gráfico con nombre. Si no especifica un gráfico con nombre al cargar, insertar o actualizar triples, Neptune utiliza el gráfico con nombre alternativo definido por el URI, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Si utiliza el programa de carga masiva de Neptune, puede especificar el gráfico con nombre que se utilizará para todos los triples (o cuádruples con la cuarta posición en blanco) utilizando el parámetro `parserConfiguration: namedGraphUri`. Para obtener información acerca de la sintaxis del comando `Load` del programa de carga de Neptune, consulte the section called “Comando Loader”.

Elegir entre FILTER, FILTER... IN y VALUES en sus consultas

Existen tres formas básicas para introducir valores en las consultas SPARQL: `FILTER`, `FILTER...IN`, y `VALUES`.

Por ejemplo, suponiendo que desee buscar los amigos de varias personas dentro de una única consulta. Con `FILTER`, podría estructurar su consulta de la siguiente manera:

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. FILTER (?s = ex:person1 || ?s = ex:person2)}
```

Este devuelve todos los valores triples en el gráfico que tiene `?s` vinculado a `ex:person1` o `ex:person2` y que dispone de un borde saliente etiquetado como `foaf:knows`.

También puede crear una consulta mediante `FILTER...IN` que devuelve resultados equivalentes.

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
```

```
  WHERE {?s foaf:knows ?o. FILTER (?s IN (ex:person1, ex:person2))}
```

También puede crear una consulta mediante VALUES que, en este caso, también devuelve resultados equivalentes.

```
PREFIX ex: <https://www.example.com/>
PREFIX foaf : <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o
WHERE {?s foaf:knows ?o. VALUES ?s {ex:person1 ex:person2}}
```

Aunque en muchos casos estas consultas son equivalente semánticamente, existen algunos casos en los que las dos variantes de FILTER difieren de la variante VALUES:

- El primer caso es cuando introduce los valores duplicados; como, por ejemplo, introducir a la misma persona dos veces. En ese caso, la consulta VALUES incluye los valores duplicados en su resultado. Puede eliminar los duplicados de forma explícita mediante la incorporación de un DISTINCT a la cláusula SELECT. Sin embargo, podría haber situaciones en las que realmente quiere duplicados en los resultados de la consulta para la introducción de un valor redundante.

  Sin embargo, las versiones FILTER y FILTER...IN extraen solo una vez el valor cuando el mismo valor aparece varias veces.
- El segundo caso está relacionado con el hecho de que VALUES siempre realiza una coincidencia exacta, mientras que FILTER podría aplicar el tipo de promoción y realizar coincidencias parciales en algunos casos.

  Por ejemplo, cuando incluya una cláusula literal como `"2.0"^^xsd:float` en sus valores, una consulta VALUES coincide totalmente con este valor literal, incluidos el valor literal y el tipo de datos.

  En cambio, FILTER produce una coincidencia parcial para estos valores numéricos. Las coincidencias podría incluir valores literales con el mismo valor, pero con distintos tipos de valores numéricos, como `xsd:double`.

  > **Note**
  >
  > No hay ninguna diferencia entre el comportamiento deFILTER y VALUES cuando se enumeran los valores literales de las cadena o de los URI.

Las diferencias entre `FILTER` y `VALUES` pueden afectar a la optimización y a la estrategia de evaluación de consulta resultante. A menos que su caso de uso requiera una coincidencia parcial, le recomendamos que utilice `VALUES`, ya que evita examinar los casos especiales relacionados con la conversión de tipos. Como resultado, `VALUES` produce a menudo una consulta más eficaz, que se ejecuta más rápido y es más económica.

Límites de Amazon Neptune

Regiones

Amazon Neptune está disponible en las siguientes regiones: AWS

- Este de EE. UU. (Norte de Virginia): `us-east-1`
- Este de EE. UU. (Ohio): `us-east-2`
- Oeste de EE. UU. (Norte de California): `us-west-1`
- Oeste de EE. UU. (Oregón): `us-west-2`
- Canadá (centro): `ca-central-1`
- América del Sur (São Paulo): `sa-east-1`
- Europa (Estocolmo): `eu-north-1`
- Europa (Irlanda): `eu-west-1`
- Europa (Londres): `eu-west-2`
- Europa (París): `eu-west-3`
- Europa (Fráncfort): `eu-central-1`
- Medio Oriente (Baréin): `me-south-1`
- Medio Oriente (EAU): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- África (Ciudad del Cabo): `af-south-1`
- Asia Pacífico (Hong Kong): `ap-east-1`
- Asia-Pacífico (Tokio): `ap-northeast-1`
- Asia-Pacífico (Seúl): `ap-northeast-2`
- Asia-Pacífico (Osaka): `ap-northeast-3`
- Asia-Pacífico (Singapur): `ap-southeast-1`
- Asia-Pacífico (Sídney): `ap-southeast-2`
- Asia-Pacífico (Bombay): `ap-south-1`
- China (Pekín): `cn-north-1`

- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (EE. UU.-Oeste): `us-gov-west-1`
- AWS GovCloud (EE. UU.-Este): `us-gov-east-1`

Diferencias en las regiones de China

Como ocurre con muchos AWS servicios, Amazon Neptune funciona de forma ligeramente diferente en China (Pekín) y China (Ningxia) que en otras regiones. AWS

Por ejemplo, cuando Neptune ML utiliza Amazon API Gateway para crear su servicio de exportación, la autenticación de IAM está habilitada de forma predeterminada. En las regiones de China, el proceso para cambiar esa opción es ligeramente diferente al de otras regiones.

Estas y otras diferencias se explican aquí.

Tamaño máximo de los volúmenes del clúster de almacenamiento

El volumen de un clúster de Neptune puede crecer hasta un tamaño máximo de 128 tebibytes (TiB) en todas las regiones compatibles, excepto en China GovCloud y, donde el límite es de 64 TiB. Esto es válido para todas las versiones de motores que comiencen con Versión: 1.0.2.2 (09/03/2020). Consulte Almacenamiento, fiabilidad y disponibilidad de Amazon Neptune.

Tamaños de instancias de bases de datos compatibles

Neptune admite diferentes clases de instancias de base de datos en distintas AWS regiones. Para saber qué clases se admiten en una región determinada, consulte Precios de Amazon Neptune y elija la región que le interesa.

Límites para cada cuenta AWS

Para determinadas características de administración, Amazon Neptune utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS).

Cada AWS cuenta tiene límites para cada región en cuanto al número de recursos de Amazon Neptune y Amazon RDS que puede crear. Estos recursos incluyen las instancias y clústeres de base de datos.

Después de que alcance el límite de un recurso, las llamadas adicionales para crear ese recurso dejan de funcionar con una excepción.

Para obtener una lista de los límites que comparten Amazon Neptune y Amazon RDS, consulte Limits in Amazon RDS en la Guía del usuario de Amazon RDS.

Se necesita una VPC para la conexión a Neptune

Amazon Neptune es un servicio exclusivo de nube privada virtual (VPC).

Además, las instancias no permiten el acceso desde fuera de la VPC.

Neptune requiere SSL

A partir de la versión `1.0.4.0` del motor, Amazon Neptune solo permite conexiones de capa de conexión segura (SSL) a través de HTTPS a cualquier instancia o punto de conexión del clúster.

Neptune requiere la versión 1.2 de TLS, con los siguientes conjuntos de cifrado seguro:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Zonas de disponibilidad y grupos de subredes de base de datos

Amazon Neptune requiere un grupo de subredes de base de datos para cada clúster que tenga subredes en dos zonas de disponibilidad (AZ) admitidas.

Se recomienda utilizar tres o más subredes en distintas zonas de disponibilidad.

Tamaño máximo de las cargas de solicitudes HTTP (150 MB)

El tamaño total de las solicitudes HTTP de Gremlin y SPARQL debe ser inferior a 150 MB. Si una solicitud supera este tamaño, Neptune devuelve `HTTP 400: BadRequestException`.

Este límite no se aplica a las conexiones de WebSockets Gremlin.

Diferencias de implementación de Gremlin

La implementación de Gremlin de Amazon Neptune tiene detalles de implementación específicos que podrían ser diferentes de otras implementaciones de Gremlin.

Para obtener más información, consulte Conformidad con los estándares de Gremlin en Amazon Neptune.

Neptune no admite caracteres nulos en los datos de cadena

Neptune no admite caracteres nulos en las cadenas. Esto es así en los datos de gráficos de propiedades de Gremlin y openCypher y en los datos de RDF/SPARQL.

SPARQL UPDATE LOAD de URI

`SPARQL UPDATE LOAD` del URI solo funciona con recursos que se encuentran dentro de la misma VPC.

Esto incluye las direcciones URL de Amazon S3 de la misma región que el clúster en las que se haya creado un punto de conexión de VPC de Amazon S3.

La dirección URL de Amazon S3 tiene que ser HTTPS y debe incluir algún tipo de autenticación. Para obtener más información, consulte Authenticating Requests: Using Query Parameters en la referencia de la API de Amazon Simple Storage Service.

Para obtener información acerca de cómo crear un punto de enlace de la VPC, consulte Creación de un punto de conexión de VPC de Amazon S3.

Si necesita cargar datos desde un archivo, se recomienda usar la API del programa de carga de Amazon Neptune. Para obtener más información, consulte Uso del programa de carga masiva de Amazon Neptune para adquirir datos.

> **Note**
>
> La API del programa de carga de Amazon Neptune no es de tipo ACID.

Autenticación y control de acceso de IAM

En las versiones del motor de Neptune anteriores a la versión 1.2.0.0, la autenticación y el control de acceso de IAM solo se admiten a nivel de clúster de base de datos. Sin embargo, a partir de la versión en `1.2.0.0` adelante, puede controlar el acceso basado en consultas de forma más detallada mediante las claves de condición de las políticas de IAM. Para obtener más información, consulte Uso de acciones de consulta en las declaraciones de políticas de acceso a datos de Neptune y Descripción general de AWS Identity and Access Management (IAM) en Amazon Neptune.

La consola Amazon Neptune requiere NeptuneReadOnlyAccesspermisos. Para restringir el acceso a los usuarios de IAM, revoque este acceso. Para obtener más información, consulte AWS políticas administradas (predefinidas) para Amazon Neptune

Amazon Neptune no es compatible con el control de acceso basado en el nombre de usuario y la contraseña.

WebSocket conexiones simultáneas y tiempo máximo de conexión

Hay un límite en el número de WebSocket conexiones simultáneas por instancia de base de datos de Neptune. Cuando se alcanza ese límite, Neptune limita cualquier solicitud de apertura de una nueva WebSocket conexión para evitar que se agote toda la memoria acumulada asignada.

Para todos los tipos de instancias más grandes compatibles con Neptune y todas las instancias sin servidor, el número máximo de WebSocket conexiones simultáneas es de 32 000 (32 768).

El número máximo de WebSocket conexiones simultáneas para los tipos de instancias más pequeñas se indica en la siguiente tabla:

Tipo de instancia	Número máximo de conexiones simultáneas WebSocket
db.t3.medium	512
db.t4g.medium	512
db.r5.large	2048
db.r5d.large	2048

Tipo de instancia	Número máximo de conexiones simultáneas WebSocket
db.r5.xlarge	4.096
db.r5.2xlarge	8 192
db.r5d.2xlarge	8 192
db.r5.4xlarge	16.384
db.r5d.4xlarge	16.384
db.r6g.large	2048
db.r6gd.large	2048
db.r6g.xlarge	4.096
db.r6gd.xlarge	4.096
db.r6g.2xlarge	8 192
db.r6gd.2xlarge	8 192
db.r6g.4xlarge	16.384
db.r6gd.4xlarge	16.384
db.x2g.large	2048
db.x2gd.large	2048
db.x2g.xlarge	4.096
db.x2gd.xlarge	4.096
db.x2iedn.xlarge	4.096
db.x2g.2xlarge	8 192
db.x2gd.2xlarge	8 192

Tipo de instancia	Número máximo de conexiones simultáneas WebSocket
db.x2g.4xlarge	16.384
db.x2gd.4xlarge	16.384
db.x2iedn.2xlarge	16.384
db.x2iezn.2xlarge	16.384
sin servidor	32 768
(otros tipos de instancias de gran tamaño)	32 768

> **Note**
>
> A partir de la versión 1.1.0.0 del motor de Neptune, Neptune ya no admite los tipos de instancias R4.

Cuando un cliente cierra correctamente una conexión, el cierre se reflejará inmediatamente en el número de conexiones abiertas.

Si el cliente no cierra una conexión, la conexión puede cerrarse automáticamente después de un tiempo de espera de 20 a 25 minutos de inactividad (el tiempo de espera de inactividad es el tiempo transcurrido desde que se recibió el último mensaje del cliente). Sin embargo, mientras no se alcance el tiempo de espera de inactividad, Neptune mantiene la conexión abierta indefinidamente.

Cuando la autenticación de IAM está habilitada, una WebSocket conexión siempre se desconecta unos minutos más de 10 días después de su establecimiento, si aún no se ha cerrado para entonces.

Límites de las propiedades y etiquetas

No hay ningún límite en el número de vértices y bordes (ni de quads de RDF) que puede tener en un gráfico.

Tampoco hay límite en el número de propiedades o etiquetas que puede haber en cualquier vértice o borde.

Hay un límite de 55 MB en el tamaño de una propiedad o etiqueta individual. En términos de RDF, esto significa que el valor de cualquier columna (S, P, O o G) de un quad de RDF no puede superar los 55 MB.

Si tiene que asociar un objeto más grande, como una imagen, con un vértice o un nodo del gráfico, puede almacenarlo como un archivo en Amazon S3 y utilizar la ruta de Amazon S3 como propiedad o etiqueta.

Límites que afectan al programa de carga masiva de Neptune

No puede poner en cola más de 64 trabajos de carga masiva de Neptune a la vez.

Neptune solo realiza un seguimiento de los 1024 trabajos de carga masiva más recientes.

Neptune solo almacena los últimos 10 000 detalles de error por trabajo.

Trabajar con otros servicios de AWS

Puede usar Amazon Neptune junto con muchos otros servicios de AWS:

Integraciones de Neptune con otros servicios

- AWS Glue: AWS Glue es un servicio de integración de datos sin servidor que le ayuda a realizar trabajos de extracción, transformación y carga (ETL) en los datos.

  Neptune proporciona una biblioteca de código abierto, neptune-python-utilities, que simplifica el uso de Python y Gremlin en un trabajo de Glue. El conector Neo4j Spark también es compatible con la ejecución de trabajos de Scala y openCypher Glue.
- Amazon SageMaker: Amazon SageMaker es una plataforma de machine learning con todas las características para crear, entrenar e implementar modelos de machine learning de alta calidad.

  Neptune se integra con SageMaker de dos maneras principales:

  - Neptune proporciona un paquete Python de código abierto para los cuadernos de Jupyter que se puede encontrar en el proyecto de cuadernos de gráficos de Neptune en GitHub. Este paquete incluye un conjunto de comandos mágicos de Jupyter, cuadernos con tutoriales y ejemplos de código que se proporcionan en un entorno de codificación interactivo donde puede aprender sobre la tecnología de gráficos y Neptune. Neptune proporciona un entorno totalmente administrado para los cuadernos de Jupyter alojados por SageMaker y enlaza automáticamente a los cuadernos del proyecto de cuaderno de gráficos de Neptune de código abierto.
  - La característica Neptune ML permite crear y entrenar modelos útiles de machine learning en gráficos de gran tamaño en cuestión de horas en lugar de semanas. Para ello, Neptune ML utiliza la tecnología de redes neuronales de gráficos (GNN) con tecnología de Amazon SageMaker y la biblioteca Deep Graph Library (DGL).
- AWS Lambda: las funciones de AWS Lambda tienen muchos usos en las aplicaciones de Neptune.

  Para obtener información sobre cómo utilizar las funciones de Lambda con cualquiera de los conocidos controladores y variantes de lenguaje de Gremlin, así como ejemplos específicos de funciones de Lambda escritas en Java, JavaScript y Python, consulte Uso de funciones de AWS Lambda en Amazon Neptune.
- Amazon Athena: Amazon Athena es un servicio de consultas interactivo que facilita el análisis de datos en Amazon Simple Storage Service y otros orígenes de datos federados con SQL estándar.

Neptune proporciona un conector a Atenea le permite comunicarse con los datos almacenados en Neptune.

- AWS Database Migration Service (AWS DMS): AWS Database Migration Service es un servicio web de AWS que puede utilizar para migrar datos de una base de datos a otra.

  AWS DMS puede cargar datos en Neptune desde bases de datos de origen compatibles de forma rápida y segura. La base de datos de origen permanece totalmente operativa durante la migración, lo que minimiza el tiempo de inactividad de las aplicaciones que dependen de ella.
- AWS Backup: AWS Backup es un servicio de copia de seguridad completamente administrado que facilita la centralización y automatización de las copias de seguridad de datos en servicios de AWS en la nube y en las instalaciones.

  AWS Backup le permite crear instantáneas periódicas automatizadas de los clústeres de Neptune mediante su política de protección de datos centralizada en todos los servicios de AWS compatibles para bases de datos, almacenamiento y computación.
- AWS SDK para pandas: AWS SDK para pandas (anteriormente conocido como AWS Data Wrangler o `awswrangler`), es una iniciativa de Python de código abierto de AWSProfessional Service que amplía la potencia de la biblioteca de análisis de datos de Python `pandas` a AWS, conectando a `DataFrames` y a más de 30 servicios relacionados con los datos de AWS, incluido Neptune.

  Además de SDK, también hay un tutorial sobre cómo utilizarlo con Neptune, así como varios cuadernos de Neptune de ejemplo, es decir, Detección de círculos de fraude, Detección de identidades sintéticas y Análisis de logística.
- Controlador JDBC: el controlador JDBC de Neptune es compatible con consultas de openCypher, Gremlin, SQL-Gremlin y SPARQL.

  La conectividad JDBC facilita la conexión a Neptune con herramientas de inteligencia empresarial (BI), como, por ejemplo, Tableau.

Herramientas y utilidades de Neptune

Amazon Neptune ofrece una serie de herramientas y utilidades que pueden simplificar y automatizar el trabajo con un gráfico. A continuación, vamos a ver algunas:

Herramientas de Amazon Neptune

- Utilidad de Amazon Neptune para GraphQL: es una herramienta de línea de comandos de código abierto de Node.js que puede ayudarle a crear y mantener una API de GraphQL para una base de datos de gráficos de propiedades de Neptune. Es una forma sin código de crear un solucionador de GraphQL para consultas de GraphQL que tienen un número variable de parámetros de entrada y devuelven un número variable de campos anidados.

Utilidad de Amazon Neptune para GraphQL

La utilidad de Amazon Neptune para GraphQL es una herramienta de línea de comandos de código abierto de Node.js que puede ayudarle a crear y mantener una API de GraphQL para una base de datos de gráficos de propiedades de Neptune (aún no funciona con datos RDF). Es una forma sin código de crear un solucionador de GraphQL para consultas de GraphQL que tienen un número variable de parámetros de entrada y devuelven un número variable de campos anidados.

Se ha publicado como un proyecto de código abierto ubicado en https://github.com/aws/ amazon-neptune-for-graphql.

Puede instalar la utilidad con NPM de la siguiente manera (consulte Instalación y configuración para obtener más información):

```
npm i @aws/neptune-for-graphql -g
```

La utilidad puede descubrir el esquema de gráfico de un gráfico de propiedades de Neptune existente, incluidos los nodos, los bordes, las propiedades y la cardinalidad de los bordes. A continuación, genera un esquema de GraphQL con las directivas necesarias para mapear los tipos de GraphQL a los nodos y los bordes de la base de datos, y genera automáticamente el código de resolución. El código de resolución se ha diseñado para minimizar la latencia al devolver solo los datos solicitados por la consulta de GraphQL.

También puede empezar con un esquema de GraphQL existente y una base de datos de Neptune vacía, y dejar que la utilidad infiera las directivas necesarias para mapear ese esquema de GraphQL

a los nodos y los bordes de los datos que se van a cargar en la base de datos. O bien, puede empezar con un esquema y directivas de GraphQL que ya haya creado o modificado.

La utilidad es capaz de crear todos los AWS recursos que necesita para su canalización, incluida la AWS AppSync API, las funciones de IAM, la fuente de datos, el esquema y la resolución, y la función AWS Lambda que consulta Neptune.

> **Note**
>
> En los ejemplos de línea de comandos que se muestran aquí se supone que estamos utilizando una consola de Linux. Si utiliza Windows, sustituya las barras diagonales inversas ("\") al final de las líneas por signos de intercalación ("^").

Temas

Instalación y configuración de la utilidad de Amazon Neptune para GraphQL

Si va a utilizar la utilidad con una base de datos de Neptune existente, necesitará que pueda conectarse al punto de conexión de la base de datos. De forma predeterminada, solo se puede acceder a una base de datos de Neptune desde la VPC en la que se encuentra.

Como la utilidad es una herramienta de línea de comandos de Node.js, debe tener Node.js (versión 18 o posterior) instalado para que se ejecute la utilidad. Para instalar Node.js en una instancia de EC2 en la misma VPC que la base de datos de Neptune, siga las instrucciones que aparecen aquí. La instancia de tamaño mínimo para ejecutar la utilidad es t2.micro. Durante la creación de la instancia, seleccione la VPC de la base de datos de Neptune en el menú desplegable Grupos de seguridad comunes.

Sin embargo, a partir de la versión 1.2.0.0 del motor, puede crear un punto de conexión público para la base de datos de Neptune al que se pueda acceder desde fuera de la VPC. Si ha creado un punto

de conexión público, puede instalar Node.js y la utilidad en una máquina local. Para instalar Node.js en macOS o Windows, visite el sitio web Node.js para descargar el instalador.

Para instalar la propia utilidad en una instancia de EC2 o en una máquina local, utilice NPM:

```
npm install aws-neptune-for-graphql -g
```

A continuación, puede ejecutar el comando help de la utilidad para comprobar si se ha instalado correctamente:

```
neptune-for-graphql --help
```

Es posible que también desee instalar la AWS CLI para administrar los recursos de AWS.

Escaneo de datos en una base de datos de Neptune existente

Ya sea que esté familiarizado con GraphQL o no, el siguiente comando es la forma más rápida de crear una API de GraphQL. Esto supone que ha instalado y configurado la utilidad de Neptune para GraphQL tal y como se describe en la sección de instalación, de forma que esté conectada al punto de conexión de la base de datos de Neptune.

```
neptune-for-graphql \
  --input-graphdb-schema-neptune-endpoint (your neptune database endpoint):(port
 number) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (your new GraphQL API name) \
  --output-resolver-query-https
```

La utilidad analiza la base de datos para detectar el esquema de los nodos, los bordes y las propiedades que incluye. En función de este esquema, infiere un esquema de GraphQL con consultas y mutaciones asociadas. A continuación, crea una API de AppSync GraphQL y los AWS recursos necesarios para utilizarla. Estos recursos incluyen un par de roles de IAM y una función de Lambda que incluye el código de resolución de GraphQL.

Cuando la utilidad haya terminado, encontrarás una nueva API de GraphQL en la AppSync consola con el nombre que asignaste en el comando. Para probarla, usa la opción AppSync Consultas del menú.

Si vuelve a ejecutar el mismo comando después de añadir más datos a la base de datos, actualizará la AppSync API y el código Lambda en consecuencia.

Para liberar todos los recursos asociados con el comando, ejecute:

```
neptune-for-graphql \
  --remove-aws-pipeline-name (your new GraphQL API name from above)
```

Partir de un esquema de GraphQL sin directivas

Puede partir de una base de datos de Neptune vacía y utilizar un esquema de GraphQL sin directivas para crear los datos y consultarlos. El siguiente comando crea automáticamente los recursos de AWS para hacer esto:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file)
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint (your Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

El archivo de esquema de GraphQL debe incluir los tipos de esquema de GraphQL, tal y como se muestra en el siguiente ejemplo de TODO. La utilidad analiza el esquema y crea una versión ampliada en función de los tipos. Añade consultas y mutaciones para los nodos almacenados en la base de datos de gráficos y, si el esquema tiene tipos anidados, añade relaciones entre los tipos almacenados como bordes en la base de datos.

La utilidad crea una API de AppSync GraphQL y todos los AWS recursos necesarios. Estos recursos incluyen un par de roles de IAM y una función de Lambda que incluye el código de resolución de GraphQL. Cuando se complete el comando, encontrarás una nueva API de GraphQL con el nombre que especificaste en la AppSync consola. Para probarlo, usa Queries en el AppSync menú.

En el siguiente ejemplo se muestra cómo funciona:

Ejemplo de Todo, a partir de un esquema de GraphQL sin directivas

En este ejemplo partimos de un esquema Todo de GraphQL sin directivas, que puede encontrar en el directorio*???muestras???*. Incluye estos dos tipos:

```
type Todo {
  name: String
  description: String
  priority: Int
```

```
  status: String
  comments: [Comment]
}

type Comment {
  content: String
}
```

Este comando procesa el esquema Todo y un punto final de una base de datos de Neptune vacía para crear una API GraphQL en: AWS AppSync

```
neptune-for-graphql /
  --input-schema-file ./samples/todo.schema.graphql \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name TodoExample \
  --create-update-aws-pipeline-neptune-endpoint (empty Neptune database endpoint):(port
 number) \
  --output-resolver-query-https
```

La utilidad crea un nuevo archivo en la carpeta de salida llamada `TodoExample.source.graphql` y la API GraphQL en. AppSync La utilidad infiere lo siguiente:

- En el tipo Todo, se agregó `@relationship` un nuevo CommentEdge tipo. Esto indica al solucionador que conecte Todo a Comment mediante el borde de una base de datos de gráficos llamado CommentEdge.
- Se agregó una nueva entrada llamada TodoInput para facilitar las consultas y las mutaciones.
- Se añadieron dos consultas para cada tipo (Todo, Comment): una para recuperar un solo tipo mediante un `id` o cualquiera de los campos de tipo enumerados en la entrada, y la otra para recuperar varios valores, filtrados con la entrada de ese tipo.
- Se añadió tres mutaciones para cada tipo: crear, actualizar y eliminar. El tipo que se va a eliminar se especifica mediante un `id` o la entrada correspondiente a ese tipo. Estas mutaciones afectan a los datos almacenados en la base de datos de Neptune.
- Se añadieron dos mutaciones para las conexiones: conectar y eliminar. Toman como entrada los identificadores de los nodos de los vértices de origen y destino utilizados por Neptune y las conexiones son bordes en la base de datos.

El solucionador reconoce las consultas y las mutaciones por sus nombres, pero puede personalizarlas, tal y como se muestra a continuación.

Este es el contenido del archivo `TodoExample.source.graphql`:

```
type Todo {
  _id: ID! @id
  name: String
  description: String
  priority: Int
  status: String
  comments(filter: CommentInput, options: Options): [Comment] @relationship(type:
 "CommentEdge", direction: OUT)
  bestComment: Comment @relationship(type: "CommentEdge", direction: OUT)
  commentEdge: CommentEdge
}

type Comment {
  _id: ID! @id
  content: String
}

input Options {
  limit: Int
}

input TodoInput {
  _id: ID @id
  name: String
  description: String
  priority: Int
  status: String
}

type CommentEdge {
  _id: ID! @id
}

input CommentInput {
  _id: ID @id
  content: String
}

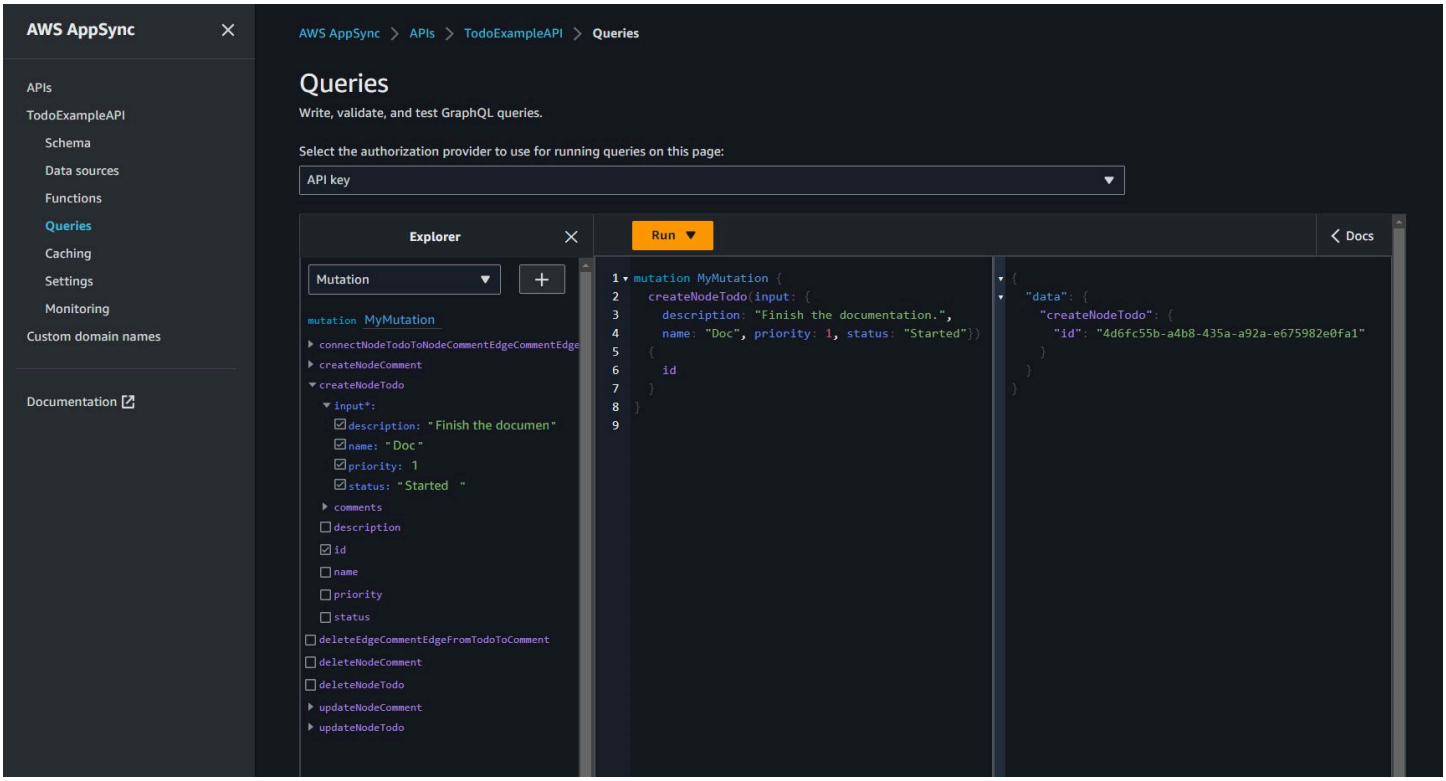
input Options {
  limit: Int
}
```

```
type Query {
  getNodeTodo(filter: TodoInput, options: Options): Todo
  getNodeTodos(filter: TodoInput): [Todo]
  getNodeComment(filter: CommentInput, options: Options): Comment
  getNodeComments(filter: CommentInput): [Comment]
}

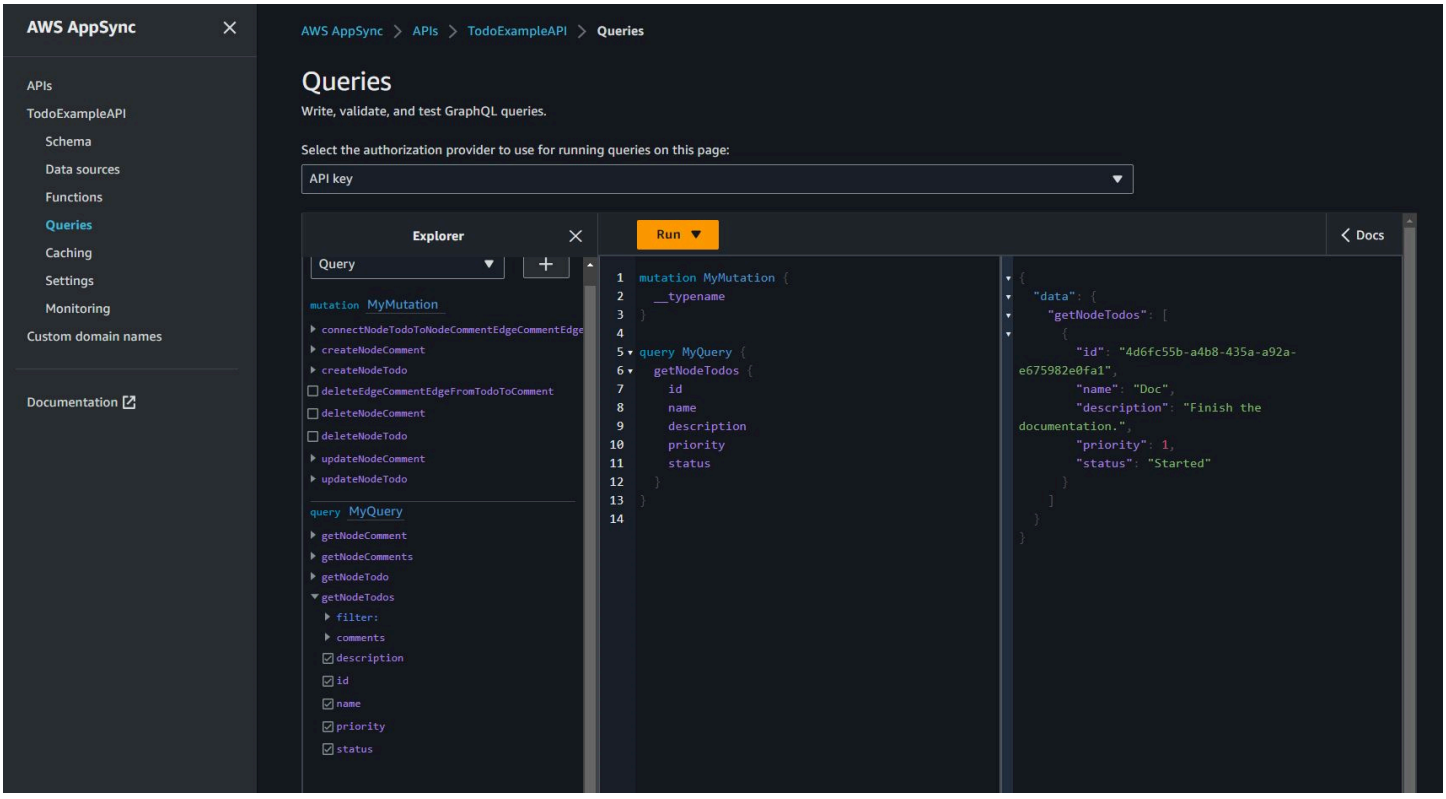
type Mutation {
  createNodeTodo(input: TodoInput!): Todo
  updateNodeTodo(input: TodoInput!): Todo
  deleteNodeTodo(_id: ID!): Boolean
  connectNodeTodoToNodeCommentEdgeCommentEdge(from_id: ID!, to_id: ID!): CommentEdge
  deleteEdgeCommentEdgeFromTodoToComment(from_id: ID!, to_id: ID!): Boolean
  createNodeComment(input: CommentInput!): Comment
  updateNodeComment(input: CommentInput!): Comment
  deleteNodeComment(_id: ID!): Boolean
}

schema {
  query: Query
  mutation: Mutation
}
```

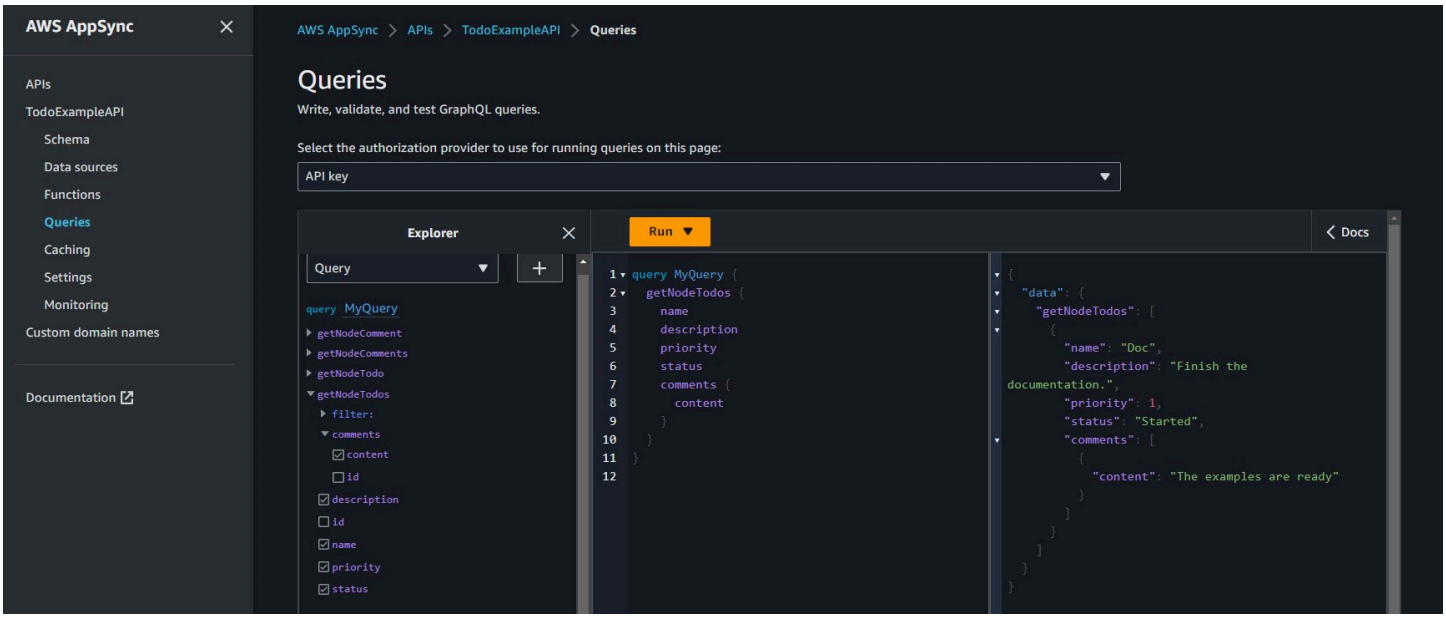
Ahora puede crear y consultar datos. Esta es una instantánea de la consola AppSync Queries utilizada para probar la nueva API de GraphQL, denominada `TodoExampleAPI` en este caso. En la ventana central, el explorador muestra una lista de consultas y mutaciones entre las que puede seleccionar una consulta, los parámetros de entrada y los campos de devolución. En esta captura de pantalla se muestra la creación de un tipo de nodo Todo mediante la mutación `createNodeTodo`:



En esta captura de pantalla se muestra la consulta de todos los nodos de Todo mediante la consulta `getNodeTodos`:



Después de crear un comentario con `createNodeComment`, puede usar la mutación `connectNodeTodoToNodeCommentEdgeCommentEdge` para conectarlos especificando sus identificadores. Esta es una consulta anidada para recuperar Todos y sus comentarios adjuntos:



Si desea realizar cambios en el archivo `TodoExample.source.graphql`, tal y como se describe en Trabajar con directivas, puede utilizar el esquema editado como entrada y volver a ejecutar la utilidad. A continuación, la utilidad modificará la API de GraphQL en consecuencia.

Trabajar con directivas para un esquema de GraphQL

Puede partir de un esquema de GraphQL que ya tenga directivas y utilizar un comando como el siguiente:

```
neptune-for-graphql \
  --input-schema-file (your GraphQL schema file with directives) \
  --create-update-aws-pipeline \
  --create-update-aws-pipeline-name (name for your new GraphQL API) \
  --create-update-aws-pipeline-neptune-endpoint  (empty Neptune database
 endpoint):(port number) \
  --output-resolver-query-https
```

Puede modificar las directivas que ha creado la utilidad o añadir sus propias directivas a un esquema de GraphQL. Estas son algunas de las formas de trabajar con las directivas:

Ejecutar la utilidad para que no genere mutaciones

Para evitar que la utilidad genere mutaciones en la API de GraphQL, utilice la opción `--output-schema-no-mutations` del comando `neptune-for-graphql`.

La directiva **`@alias`**

La directiva `@alias` se puede aplicar a campos o tipos de esquemas de GraphQL. Mapea diferentes nombres entre la base de datos de gráficos y el esquema de GraphQL. La sintaxis es la siguiente:

```
@alias(property: (property name))
```

En el siguiente ejemplo, `airport` es la etiqueta del nodo de la base de datos de gráficos mapeada al tipo `Airport` de GraphQL y `desc` es la propiedad del nodo de gráficos mapeada al campo `description` (consulte el ejemplo de rutas aéreas):

```
type Airport @alias(property: "airport") {
  city: String
  description: String @alias(property: "desc")
}
```

Tenga en cuenta que el formato estándar de GraphQL requiere nombres del tipo PascalCase y nombres de campo del tipo CamelCase.

La directiva **`@relationship`**

La directiva `@relationship` mapea los tipos de GraphQL anidados a los bordes de las bases de datos de gráficos. La sintaxis es la siguiente:

```
@relationship(edgeType: (edge name), direction: (IN or OUT))
```

A continuación se muestra un comando de ejemplo:

```
type Airport @alias(property: "airport") {
  ...
  continentContainsIn: Continent @relationship(edgeType: "contains", direction: IN)
  countryContainsIn: Country @relationship(edgeType: "contains", direction: IN)
  airportRoutesOut(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: OUT)
```

```
  airportRoutesIn(filter: AirportInput, options: Options): [Airport]
 @relationship(edgeType: "route", direction: IN)
}
```

Puede encontrar las directivas `@relationship` tanto en el ejemplo de Todo como en el ejemplo de rutas aéreas.

Las directivas **`@graphQuery`** y **`@cypher`**

Puede definir consultas de openCypher para resolver un valor de campo, añadir consultas o añadir mutaciones. Por ejemplo, esto añade un campo nuevo `outboundRoutesCount` al tipo `Airport` para contar las rutas de salida:

```
type Airport @alias(property: "airport") {
  ...
  outboundRoutesCount: Int @graphQuery(statement: "MATCH (this)-[r:route]->(a) RETURN
 count(r)")
}
```

A continuación se muestra un ejemplo de nuevas consultas y mutaciones:

```
type Query {
  getAirportConnection(fromCode: String!, toCode: String!): Airport \
    @cypher(statement: \
      "MATCH (:airport{code: '$fromCode'})-[:route]->(this:airport)-[:route]-
>(:airport{code:'$toCode'})")
}

type Mutation {
  createAirport(input: AirportInput!): Airport @graphQuery(statement: "CREATE
 (this:airport {$input}) RETURN this")
  addRoute(fromAirportCode:String, toAirportCode:String, dist:Int): Route \
    @graphQuery(statement: \
     "MATCH (from:airport{code:'$fromAirportCode'}),
 (to:airport{code:'$toAirportCode'}) \
      CREATE (from)-[this:route{dist:$dist}]->(to) \
      RETURN this")
}
```

Tenga en cuenta que si omite `RETURN`, el solucionador asume que la palabra clave `this` es el ámbito de devolución.

También puede añadir una consulta o mutación mediante una consulta de Gremlin:

```
type Query {
  getAirportWithGremlin(code:String): Airport \
    @graphQuery(statement: "g.V().has('airport', 'code', '$code').elementMap()")  #
 single node
  getAirportsWithGremlin: [Airport] \
    @graphQuery(statement: "g.V().hasLabel('airport').elementMap().fold()")       #
 list of nodes
  getCountriesCount: Int \
    @graphQuery(statement: "g.V().hasLabel('country').count()")                   #
 scalar
}
```

En este momento, las consultas de Gremlin se limitan a las que devuelven valores escalares, o `elementMap()` a un solo nodo o `elementMap().fold()` a una lista de nodos.

La directiva **`@id`**

La directiva `@id` identifica el campo mapeado a la entidad de la base de datos de gráficos `id`. Las bases de datos de gráficos como Amazon Neptune siempre tienen un único `id` para los nodos y los bordes que se asignan durante las importaciones masivas o que se generan automáticamente. Por ejemplo:

```
type Airport {
  _id: ID! @id
  city: String
  code: String
}
```

Nombres reservados de tipos, consultas y mutaciones

La utilidad genera automáticamente consultas y mutaciones para crear una API de GraphQL que funcione. El solucionador reconoce el patrón de estos nombres y lo reserva. Estos son algunos ejemplos del tipo `Airport` y del tipo de conexión `Route`:

El tipo `Options` está reservado.

```
input Options {
  limit: Int
}
```

Los parámetros de la función `filter` y `options` están reservados.

```
type Query {
  getNodeAirports(filter: AirportInput, options: Options): [Airport]
}
```

El prefijo getNode de los nombres de consulta está reservado y los prefijos de los nombres de mutaciones como `createNode`, `updateNode`, `deleteNode`, `connectNode`, `deleteNode`, `updateEdge` y `deleteEdge` están reservados.

```
type Query {
  getNodeAirport(id: ID, filter: AirportInput): Airport
  getNodeAirports(filter: AirportInput): [Airport]
}

type Mutation {
  createNodeAirport(input: AirportInput!): Airport
  updateNodeAirport(id: ID!, input: AirportInput!): Airport
  deleteNodeAirport(id: ID!): Boolean
  connectNodeAirportToNodeAirportEdgeRout(from: ID!, to: ID!, edge: RouteInput!): Route
  updateEdgeRouteFromAirportToAirport(from: ID!, to: ID!, edge: RouteInput!): Route
  deleteEdgeRouteFromAirportToAirport(from: ID!, to: ID!): Boolean
}
```

Aplicación de cambios al esquema de GraphQL

Puede modificar el esquema de origen de GraphQL y volver a ejecutar la utilidad, obteniendo el esquema más reciente de la base de datos de Neptune. Cada vez que la utilidad detecta un nuevo esquema en la base de datos, genera un nuevo esquema de GraphQL.

También puede editar manualmente el esquema de origen de GraphQL y volver a ejecutar la utilidad con el esquema de origen como entrada en lugar del punto de conexión de la base de datos de Neptune.

Por último, puede introducir los cambios en un archivo con este formato JSON:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
```

```
  }
]
```

Por ejemplo:

```
[
  {
    "type": "Airport",
    "field": "outboundRoutesCountAdd",
    "action": "add",
    "value":"outboundRoutesCountAdd: Int @graphQuery(statement: \"MATCH (this)-
[r:route]->(a) RETURN count(r)\")"
  },
  {
    "type": "Mutation",
    "field": "deleteNodeVersion",
    "action": "remove",
    "value": ""
  },
  {
    "type": "Mutation",
    "field": "createNodeVersion",
    "action": "remove",
    "value": ""
  }
]
```

A continuación, al ejecutar la utilidad en este archivo con el parámetro `--input-schema-changes-file` del comando, la utilidad aplicará los cambios a la vez.

Argumentos de línea de comandos para la utilidad de GraphQL

- **`--help, -h`**: devuelve el texto de ayuda de la utilidad de GraphQL a la consola.

- **`--input-schema`** ***`(schema text)`***: un esquema de GraphQL, con o sin directivas, para usarlo como entrada.

- **`--input-schema-file`** ***`(file URL)`***: la URL de un archivo que incluye un esquema de GraphQL para usarlo como entrada.

- **`--input-schema-changes-file` *`(file URL)`***: la URL de un archivo que incluye los cambios que desea realizar en un esquema de GraphQL. Si ejecuta la utilidad en una base de datos de Neptune varias veces y también cambia manualmente el esquema de origen de GraphQL, tal vez si añade una consulta personalizada, se perderán los cambios manuales. Para evitarlo, introduzca los cambios en un archivo de cambios y páselo con este argumento.

  El archivo de cambios utiliza el siguiente formato JSON:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

  Consulte el ejemplo de Todo para obtener más información.

- **`--input-graphdb-schema` *`(schema text)`***: en lugar de ejecutar la utilidad en una base de datos de Neptune, puede expresar un esquema de graphdb en formato escrito para usarlo como entrada. Un esquema de graphdb tiene un formato JSON como este:

```
{
  "nodeStructures": [
    { "label":"nodelabel1",
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    },
    { "label":"nodelabel2",
      "properties": [
          { "name":"name2", "type":"type1" }
      ]
    }
  ],
  "edgeStructures": [
    {
```

```
      "label":"label1",
      "directions": [
        { "from":"nodelabel1", "to":"nodelabel2", "relationship":"ONE-ONE|ONE-MANY|
MANY-MANY"  }
      ],
      "properties": [
        { "name":"name1", "type":"type1" }
      ]
    }
  ]
}
```

- **`--input-graphdb-schema-file` *`(file URL)`***: en lugar de ejecutar la utilidad en una base de datos de Neptune, puede guardar un esquema de graphdb en un archivo para usarlo como entrada. Consulte la opción anterior `--input-graphdb-schema` para ver un ejemplo del formato JSON para un archivo de esquema de graphdb.

- **`--input-graphdb-schema-neptune-endpoint` *`(endpoint URL)`***: el punto de conexión de la base de datos de Neptune del que la utilidad debe extraer el esquema de graphdb.

- **`--output-schema-file` *`(file name)`***: el nombre del archivo de salida del esquema de GraphQL. Si no se especifica, el valor predeterminado es `output.schema.graphql`, a menos que se haya establecido un nombre de canalización con `--create-update-aws-pipeline-name`, en cuyo caso, el nombre de archivo predeterminado es *`(pipline name)`*`.schema.graphql`.

- **`--output-source-schema-file` *`(file name)`***: el nombre del archivo de salida del esquema de GraphQL con directivas. Si no se especifica, el valor predeterminado es `output.source.schema.graphql`, a menos que se haya establecido un nombre de canalización con `--create-update-aws-pipeline-name`, en cuyo caso, el nombre predeterminado es *`(pipeline name)`*`.source.schema.graphql`.

- **`--output-schema-no-mutations`**: si este argumento está presente, la utilidad no genera mutaciones en la API de GraphQL, solo consultas.

- **`--output-neptune-schema-file`** ***`(file name)`***: el nombre del archivo de salida del esquema de graphdb de Neptune que detecta la utilidad. Si no se especifica, el valor predeterminado es `output.graphdb.json`, a menos que se haya establecido un nombre de canalización con `--create-update-aws-pipeline-name`, en cuyo caso, el nombre de archivo predeterminado es *`(pipeline name)`*`.graphdb.json`.

- **`--output-js-resolver-file`** ***`(file name)`***: el nombre del archivo de salida de una copia del código de resolución. Si no se especifica, el valor predeterminado es `output.resolver.graphql.js`, a menos que se haya establecido un nombre de canalización con `--create-update-aws-pipeline-name`, en cuyo caso, el nombre del archivo es *`(pipeline name)`*`.resolver.graphql.js`.

  Este archivo está comprimido en el paquete de código cargado en la función de Lambda que ejecuta el solucionador.

- **`--output-resolver-query-sdk`**: este argumento especifica que la función de Lambda de la utilidad debe consultar Neptune mediante el SDK de datos de Neptune, que ha estado disponible a partir de la versión 1.2.1.0.R5 del motor de Neptune (esta es la versión predeterminada). Sin embargo, si la utilidad detecta una versión anterior del motor de Neptune, sugiere utilizar en su lugar la opción HTTPS Lambda, que se puede invocar con el argumento `--output-resolver-query-https`.

- **`--output-resolver-query-https`**: este argumento especifica que la función de Lambda de la utilidad debe consultar a Neptune mediante la API de HTTPS de Neptune.

- **`--create-update-aws-pipeline`**— Este argumento desencadena la creación de los AWS recursos que debe utilizar la API GraphQL, incluida la API AppSync GraphQL y la Lambda que ejecuta el solucionador.

- **`--create-update-aws-pipeline-name` *`(pipeline name)`*** — Este argumento establece el nombre de la canalización, como la `pipeline-name` API AppSync o la `pipeline-name` función de la función Lambda. Si no se especifica un nombre, `--create-update-aws-pipeline` utiliza el nombre de la base de datos de `Neptune` .
- **`--create-update-aws-pipeline-region` *`(AWS region)`***: este argumento establece la región de AWS en la que se crea la canalización de la API de GraphQL. Si no se especifica, la región predeterminada es `us-east-1` o la región en la que se encuentra la base de datos de Neptune, extraída del punto de conexión de la base de datos.
- **`--create-update-aws-pipeline-neptune-endpoint` *`(endpoint URL)`***: este argumento establece el punto de conexión de la base de datos de Neptune que utiliza la función de Lambda para consultar la base de datos. Si no se establece, se utiliza el punto de conexión establecido por `--input-graphdb-schema-neptune-endpoint`.
- **`--remove-aws-pipeline-name` *`(pipeline name)`***: este argumento elimina una canalización creada con `--create-update-aws-pipeline`. Los recursos que se van a eliminar se enumeran en un archivo denominado *`(pipeline name)`*`.resources.json`.
- **`--output-aws-pipeline-cdk`** — Este argumento desencadena la creación de un archivo CDK que se puede utilizar para crear AWS los recursos para la API GraphQL, incluida la API AppSync GraphQL y la función Lambda que ejecuta el solucionador.
- **`--output-aws-pipeline-cdk-neptume-endpoint` *`(endpoint URL)`***: este argumento establece el punto de conexión de la base de datos de Neptune que utiliza la función de Lambda para consultar la base de datos de Neptune. Si no se establece, se utiliza el punto de conexión establecido por `--input-graphdb-schema-neptune-endpoint`.
- **`--output-aws-pipeline-cdk-name` *`(pipeline name)`*** — Este argumento establece el nombre de la canalización que deben utilizar la AppSync API y la función de nombre de canalización de Lambda. Si no se especifica, `--create-update-aws-pipeline` utiliza el nombre de la base de datos de Neptune.
- **`--output-aws-pipeline-cdk-region` *`(AWS region)`***: esta opción establece la región de AWS en la que se crea la canalización de la API de GraphQL. Si no se especifica, se establece de forma predeterminada en `us-east-1` o la región en la que se encuentra la base de datos de Neptune, extraída del punto de conexión de la base de datos.
- **`--output-aws-pipeline-cdk-file` *`(file name)`***: esta opción establece el nombre del archivo CDK. Si no se establece, el valor predeterminado es *`(pipeline name)`*`-cdk.js`.

Errores de servicio de Neptune

Amazon Neptune tiene dos conjuntos de errores diferentes:

- Los errores del motor de gráficos que son solo para los puntos de conexión del clúster de base de datos de Neptune.
- Los errores están asociados con las API para crear y modificar los recursos de Neptune con el SDK de AWS y la AWS Command Line Interface (AWS CLI).

Temas

Mensajes y códigos de error del motor de gráficos

Los puntos de conexión de Amazon Neptune devuelven los errores estándar de Gremlin y SPARQL que encuentran.

Dichos puntos de conexión pueden devolver también errores específicos de Neptune. En esta sección, se documentan los mensajes y códigos de error de Neptune, así como las acciones recomendadas.

> **Note**
>
> Estos errores son solo para puntos de conexión de clústeres de base de datos de Neptune. Las API para crear y modificar los recursos de Neptune con el SDK de AWS y la AWS CLI tienen un conjunto distinto de errores comunes. Para obtener información acerca de estos errores, consulte the section called “Errores de API”.

Formato de los errores del motor de gráficos

Los mensajes de error de Neptune devuelven un código de error HTTP relevante y una respuesta con formato JSON.

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: LDM6CJP8RMQ1FHKSC1RBVJFPNVV4KQNSO5AEMF66Q9ASUAAJG
Content-Type: application/x-amz-json-1.0
Content-Length: 465
Date: Thu, 15 Mar 2017 23:56:23 GMT

{
    "requestId": "0dbcded3-a9a1-4a25-b419-828c46342e47",
    "code": "ReadOnlyViolationException",
    "detailedMessage": "The request is rejected because it violates some read-only
 restriction, such as a designation of a replica as read-only."
}
```

Errores de consulta del motor de gráficos

La siguiente tabla contiene el código y el mensaje de error, así como el estado HTTP.

También indica si está permitido volver a intentar la solicitud. Por lo general, está permitido volver a intentar la solicitud si fuese posible que el nuevo intento se lleve a cabo correctamente.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`AccessDeniedException`	403	No	Authentication or authorization failure.
`BadRequestException`	400	No	The request could not be completed.
`BadRequestException`	400	No	Request size exceeds max allowed value of 157286400 bytes.
`CancelledByUserException`	500	Yes	The request processing was cancelled by an authorized client.
`ConcurrentModificationException`	500	Yes	The request processing did not succeed due to a modificat

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
			ion conflict. The client should retry the request.
`ConstraintViolationException`	400	Yes	The query engine discovered, during the execution of the request, that the completion of some operation is impossible without violating some data integrity constraints, such as persistence of in- and out-vertices while adding an edge. Such conditions are typically observed if there are concurrent modifications to the graph, and are transient. The client should retry the request.
`FailureByQueryException`	500	Yes	Calling fail() caused request processing to fail.
`InternalFailureException`	500	Yes	The request processing has failed.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`InvalidNumericDataException`	400	No	Invalid use of numeric data which cannot be represented in 64-bit storage size.
`InvalidParameterException`	400	No	An invalid or out-of-range value was supplied for some input parameter or invalid syntax in a supplied RDF file.
`MalformedQueryException`	400	No	The request is rejected because it contains a query that is syntactically incorrect or does not pass additional validation.
`MemoryLimitExceededException`	500	Yes	The request processing did not succeed due to lack of memory, but can be retried when the server is less busy.
`MethodNotAllowedException`	405	No	The request is rejected because the chosen HTTP method is not supported by the used endpoint.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`MissingParameterException`	400	No	A required parameter for the specified action is not supplied.
`QueryLimitExceededException`	500	Yes	The request processing did not succeed due to the lack of a limited resource, but can be retried when the server is less busy.
`QueryLimitException`	400	No	Size of query exceeds system limit.
`QueryTooLargeException`	400	No	The request was rejected because its body is too large.
`ReadOnlyViolationException`	400	No	The request is rejected because it violates some read-only restriction, such as a designation of a replica as read-only.
`ThrottlingException`	500	Yes	Rate of requests exceeds the maximum throughput. OK to retry.
`TimeLimitExceededException`	500	Yes	The request processing timed out.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`TooManyRequestsException` (Excepción de demasiadas solicitudes)	429	Yes	The rate of requests exceeds the maximum throughput. OK to retry.
`UnsupportedOperationException`	400	No	The request uses a currently unsupported feature or construct.

Errores de autenticación de IAM

Estos errores son específicos de los clústeres que tienen habilitada la autenticación de IAM.

La siguiente tabla contiene el código y el mensaje de error, así como el estado HTTP.

Neptune Service Error Code	HTTP status	Message
Incorrect IAM User/Policy	403	You do not have sufficient access to perform this action.
Incorrect or Missing Region	403	Credential should be scoped to a valid Region, not '*región*'.
Incorrect or Missing Service Name	403	Credential should be scoped to correct service: '`neptune-db `'.
Incorrect or Missing Host Header / Invalid Signature	403	The request signature we calculated does not match the signature you provided. Check your AWS Secret Access Key and signing method. Consult

Neptune Service Error Code	HTTP status	Message
		the service documentation for details. Host header is missing or hostname is incorrect.
Missing `X-Amz-Security-Token`	403	'`x-amz-security-token` ' is named as a `SignedHeader` , but it does not exist in the HTTP request
Missing Authorization Header	403	The request did not include the required authorization header, or it was malformed.
Missing Authentication Token	403	Missing Authentication Token.
Old Date	403	Signature expired: *20181011T213907Z* is now earlier than *20181011T213915Z* (*20181011T214415Z* - *5 min*.)
Future Date	403	Signature not yet current: *20500224T213559Z* is still later than *20181108T225925Z* (*20181108T225425Z* + *5 min*.)
Incorrect Date Format	403	Date must be in ISO-8601 'basic format'. Got '*date*'. See https://en.wikipedia.org/wiki/ISO_8601.
Unknown/Missing Access Key or Session Token	403	The security token included in the request is invalid.

Neptune Service Error Code	HTTP status	Message
Unknown/Missing Secret Key	403	The request signature we calculated does not match the signature you provided. Check your AWS Secret Access Key and signing method. Consult the service documentation for details. Host header is missing or hostname is incorrect.
`TooManyRequestsException (Excepción de demasiadas solicitudes)`	429	The rate of requests exceeds the maximum throughput. OK to retry.

Mensajes de error y códigos de la API de administración de clústeres de base de datos

Estos errores de Amazon Neptune están asociados con las API para crear y modificar los recursos de Neptune con el SDK de AWS y la AWS CLI.

La siguiente tabla contiene el código y el mensaje de error, así como el estado HTTP.

Neptune Service Error Code	HTTP status	Message
`AccessDeniedException`	403	No tiene acceso suficiente para realizar esta acción.
`IncompleteSignature`	400	La firma de solicitud no se ajusta a los estándares de AWS.
`InternalFailure`	500	El procesamiento de la solicitud ha devuelto un error debido a un error o una excepción desconocidos.

Neptune Service Error Code	HTTP status	Message
`InvalidAction`	400	La acción u operación solicitada no es válida. Compruebe que la acción se ha escrito correctamente.
`InvalidClientTokenId`	403	El certificado X.509 o el ID de clave de acceso de AWS proporcionado no existen en nuestros registros.
`InvalidParameterCombination`	400	Los parámetros que no deben utilizarse conjuntamente se utilizan de forma conjunta.
`InvalidParameterValue`	400	Se ha proporcionado un valor no válido o fuera de rango para el parámetro de entrada.
`InvalidQueryParameter`	400	Se ha proporcionado un valor no válido o fuera de rango para el parámetro de entrada.
`MalformedQueryString`	400	La cadena de consulta contiene un error de sintaxis.
`MissingAction`	400	Falta un parámetro obligatorio o una acción en la solicitud.
`MissingAuthenticationToken`	403	La solicitud debe contener un certificado X.509 o bien un ID de clave de acceso de AWS válido (registrado).
`MissingParameter`	400	No se ha facilitado un parámetro necesario para la acción especificada.

Neptune Service Error Code	HTTP status	Message
`OptInRequired`	403	El ID de clave de acceso de AWS necesita una suscripción al servicio.
`RequestExpired`	400	La solicitud llegó al servicio más de 15 minutos después de la marca de fecha de la solicitud o más de 15 minutos después de la fecha de vencimiento de la solicitud (por ejemplo, para las URL prefirmadas), o la marca de fecha de la solicitud correspon de a una hora futura en más de 15 minutos.
`ServiceUnavailable`	503	La solicitud no se ha ejecutado correctamente debido a un error temporal del servidor.
`ThrottlingException`	500	La solicitud se denegó debido a una limitación controlada.
`ValidationError`	400	La entrada no satisface las limitaciones que especifica un servicio de AWS.

Mensajes de errores y fuente del programa de carga de Neptune

El punto de enlace `status` del programa de carga de Neptune devuelve los siguientes mensajes. Para obtener más información, consulte API de obtención de estado.

La siguiente tabla contiene el código y la descripción de la fuente del programa de carga.

Error or Feed Code	Description
LOAD_NOT_STARTED	La carga se ha registrado, pero no se ha iniciado.
LOAD_IN_PROGRESS	Indica que la carga está en curso y especifica el número de archivos que se están cargando actualmente. Cuando el cargador analiza un archivo, crea uno o más fragmentos para cargarlos en paralelo. Como un solo archivo puede producir varios fragmentos, el número de archivos incluido en este mensaje suele ser inferior al número de subprocesos que se utilizan en el proceso de carga masiva.
LOAD_COMPLETED	La carga ha finalizado sin errores o con errores dentro de un umbral aceptable.
LOAD_CANCELLED_BY_USER	El usuario ha cancelado la carga.
LOAD_CANCELLED_DUE_TO_ERRORS	El sistema ha cancelado la carga debido a errores.
LOAD_UNEXPECTED_ERROR	Error de carga inesperado.
LOAD_FAILED	La carga no se ha realizado debido a uno o más errores.
LOAD_S3_READ_ERROR	Error de fuente debido a problemas de conectividad de Amazon S3 intermitentes o transitorios. Si cualquiera de las fuentes recibe este error, el estado general de carga se establece en LOAD_FAILED.
LOAD_S3_ACCESS_DENIED_ERROR	Se denegó el acceso al bucket de S3. Si cualquiera de las fuentes recibe este error,

Error or Feed Code	Description
	el estado general de carga se establece en LOAD_FAILED.
LOAD_COMMITTED_W_WRITE_CONFLICTS	Se han cargado datos con conflictos de escritura no resueltos. El programa de carga intentará resolver los conflictos de escritura en transacciones independientes y actualizar el estado de la fuente a medida que progresa la carga. Si el estado final de la fuente es LOAD_COMMITTED_W_WRITE_CONFLICTS, intentará reanudar la carga y es probable que esta se realice correctamente sin conflictos de escritura. Un conflicto de escritura no suele estar relacionado con datos de entrada incorrectos, pero las duplicaciones de datos pueden aumentar las posibilidades de este tipo de conflictos.
LOAD_DATA_DEADLOCK	Load was automatically rolled back due to deadlock.
LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETED	Se ha producido un error de fuente porque el archivo se ha eliminado o actualizado después de iniciar la carga.
LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED	La solicitud de carga no se ha ejecutado porque se ha producido un error en la comprobación de dependencias.
LOAD_IN_QUEUE	La solicitud de carga se ha puesto en cola y está a la espera de ejecutarse.

Error or Feed Code	Description
LOAD_FAILED_INVALID_REQUEST	Error en la carga porque la solicitud no era válida (por ejemplo, es posible que el origen/bucket especificado no exista o que el formato de archivo no sea válido).

Versiones del motor para Amazon Neptune

Amazon Neptune publica actualizaciones del motor con regularidad.

Puede determinar qué versión del motor tiene instalada actualmente utilizando la API instance-status o la consola de Neptune. El número de versión le indica si está ejecutando una versión principal original, una versión secundaria o una versión de parche. Para obtener más información sobre la numeración de versiones, consulte Números de la versión del motor.

Para obtener más información general sobre actualizaciones, consulte Mantenimiento de clústeres.

A partir de la versión 1.3.0.0 del motor, las versiones del motor tendrán la estructura que se muestra en la siguiente tabla. El número de versión secundario es el que se evaluará para su procesamiento de `AutoMinorVersionUpgrade`.

Versión	Versi de produ	Versi princ	Versión secundaria	Versiones de parche	Status	Released	Fin de vida útil	Actualización a:
1,32.1	1	3	2.	1	activa	2024-06-20	2025-11-30	N/A
1.3.2.0	1	3	2	0	activa	2024-06-10	2025-11-30	1.3.2.1
1.3.1.0	1	3	1	0	activa	2024-03-06	2025-11-30	1.3.2.1
1.3.0.0	1	3	0	0	activa	15-11-2020	2025-11-30	1.3.2.1

La siguiente tabla muestra todas las versiones del motor desde la 1.0.1.0, junto con información sobre la versión. end-of-life Puede utilizar las fechas para planificar sus ciclos de prueba y actualización.

Versión	Versión principal	Versión secundaria	Status	Released	Fin de vida útil	Actualización a:
1.2.1.1	1.2	1.1	activa	2024-03-11	2025-03-06	1.3.0.0
1.2.1.0	1.2	1.0	activa	2023-03-08	2025-03-06	1.3.0.0
1.2.0.2	1.2	0.2	activa	2022-11-16	2025-03-06	1.3.0.0
1.2.0.1	1.2	0.1	activa	2022-10-26	2025-03-06	1.3.0.0
1.2.0.0	1.2	0.0	activa	2022-07-21	2025-03-06	1.3.0.0
1.1.1.0	1.1	1.0	activa	2019-04-2022	2024-10-31	1.2.1.0
1.1.0.0	1.1	0.0	activa	2021-11-19	2024-10-31	1.1.1.0
1.0.5.1	1.0	5.1	en desuso	2021-10-01	2023-01-30	1.1.0.0
1.0.5.0	1.0	5.0	en desuso	2021-07-27	2023-01-30	1.1.0.0
1.0.4.2	1.0	4.2	en desuso	2021-06-01	2023-01-30	1.1.0.0
1.0.4.1	1.0	4.1	en desuso	2020-12-08	2023-01-30	1.1.0.0
1.0.4.0	1.0	4.0	en desuso	2020-10-12	2023-01-30	1.1.0.0
1.0.3.0	1.0	3.0	en desuso	2020-08-03	2023-01-30	1.1.0.0
1.0.2.2	1.0	2.2	en desuso	2020-03-09	2022-07-29	1.0.3.0
1.0.2.1	1.0	2.1	en desuso	2019-11-22	2022-07-29	1.0.3.0
1.0.2.0	1.0	2.0	en desuso	2019-11-08	2020-05-19	1.0.3.0
1.0.1.2	1.0	1.2	en desuso	2019-10-15	—	—
1.0.1.1	1.0	1.1	en desuso	2019-08-13	—	—

Versión	Versión principal	Versión secundaria	Status	Released	Fin de vida útil	Actualización a:
1.0.1.0.*	1.0	1.0.*	en desuso	02-07-2019 y fechas anteriores	—	—

Planificación de las principales versiones del motor end-of-life

Las versiones del motor de Neptune casi siempre llegan al final de su vida útil al final de un trimestre natural. Solo se realizan excepciones cuando surgen problemas importantes de seguridad o disponibilidad.

Cuando una versión del motor llegue al final de su vida útil, tendrá que actualizar la base de datos de Neptune a una versión más reciente.

En general, las versiones del motor de Neptune siguen estando disponibles de la siguiente manera:

- Versiones de motor secundarias: las versiones del motor secundarias permanecen disponibles durante al menos 6 meses después de su lanzamiento.
- Versiones de motor principales: las versiones del motor principales permanecen disponibles durante al menos 6 meses después de su lanzamiento.

Al menos 3 meses antes de que la versión del motor llegue al final de su vida útil, AWS enviará una notificación automática por correo electrónico a la dirección de correo electrónico asociada a su AWS cuenta y publicará el mismo mensaje en su AWS Health Dashboard. Esto le dará tiempo a planificar y prepararse para la actualización.

Cuando una versión del motor llegue al final de su vida útil, ya no podrá crear nuevos clústeres o instancias con esa versión. Tampoco se podrán crear instancias con esa versión mediante el escalado automático.

Una versión del motor que llegue al final de su vida útil se actualiza automáticamente durante un período de mantenimiento. El mensaje que se le envía tres meses antes del final de la vida útil de la versión de motor contiene detalles sobre lo que implica esa actualización automática, incluida la

versión a la que se actualizaría automáticamente, el impacto en sus clústeres de bases de datos y las acciones que recomendamos.

> **Important**
>
> Usted es responsable de mantener actualizadas las versiones de su motor de base de datos. AWS insta a todos los clientes a actualizar sus bases de datos a la última versión del motor para poder beneficiarse de las medidas de seguridad, privacidad y disponibilidad más actuales. Si utiliza su base de datos en un motor o software no compatibles después de la fecha de caducidad ("motor heredado"), es probable que corra riesgos operativos, de seguridad y de privacidad, lo que incluye sufrir períodos de inactividad.
> El funcionamiento de su base de datos en cualquier motor está sujeto al Acuerdo que rige su uso de los AWS Servicios. Los motores antiguos no están disponibles de forma general. AWS ya no es compatible con el Legacy Engine y AWS puede limitar el acceso o el uso de cualquier Legacy Engine en cualquier momento si se AWS determina que el Legacy Engine representa un riesgo de seguridad o responsabilidad, o un riesgo de daño, para los Servicios AWS, sus Filiales o cualquier tercero. Su decisión de seguir publicando su contenido en un motor heredado podría tener como consecuencia que su contenido deje de estar disponible, se dañe o sea irrecuperable. Las bases de datos que se ejecutan en un motor heredado están sujetas a las excepciones del acuerdo de nivel de servicio (SLA).
> LAS BASES DE DATOS Y EL SOFTWARE RELACIONADO QUE SE EJECUTAN EN UN MOTOR HEREDADO CONTIENEN ERRORES, DEFECTOS O COMPONENTES DAÑINOS. EN CONSECUENCIA, Y SIN PERJUICIO DE CUALQUIER DISPOSICIÓN EN CONTRARIO EN EL CONTRATO O EN LAS CONDICIONES DEL SERVICIO, AWS SUMINISTRA EL LEGACY ENGINE «TAL CUAL».

Amazon Neptune Engine versión 1.3.2.1 (2024-06-20)

A partir del 20 de junio de 2020, la versión 1.3.2.1 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> En la versión 1.3.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.3.0.0 a una versión de motor 1.3.0.0 o

posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.3`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1` o `neptune1.2`, y esos grupos de parámetros no funcionan con la versión 1.3.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

Defectos corregidos en esta versión del motor

Correcciones de openCypher

- Se detectó un error en la función de caché del plan de consultas para las consultas parametrizadas que contienen una `WITH` cláusula interna que tiene `SKIP` y como `LIMIT` parámetros. Los valores SKIP/LIMIT no estaban debidamente parametrizados y, como resultado, las ejecuciones posteriores del mismo plan de consultas en caché con valores de parámetros diferentes seguirían arrojando los mismos resultados que en la primera ejecución. Esto se ha corregido.

```
# insert some nodes
UNWIND range(1, 10) as i CREATE (s {name: i}) RETURN s

# sample query
MATCH (p)
WITH p ORDER BY p.name SKIP $s LIMIT $l
RETURN p.name as res

# first time executing with {"s": 2, "l": 1}
{
  "results" : [ {
    "res" : 3
  } ]
}

# second time executing with {"s": 2, "l": 10}
# due to bug, produces
{
  "results" : [ {
    "res" : 3
  } ]
}
# with fix, produces correct results:
```

```
{
  "results" : [ {
    "res" : 3
  }, {
    "res" : 4
  }, {
    "res" : 5
  }, {
    "res" : 6
  }, {
    "res" : 7
  }, {
    "res" : 8
  }, {
    "res" : 9
  }, {
    "res" : 10
  } ]
}%
```

- Se ha corregido un error que provocaba que las consultas de mutación parametrizadas emitieran un error `InternalFailureException` cuando el parámetro pasado no estaba ya presente en la base de datos.
- Se ha corregido un error que provocaba que las consultas de Bolt parametrizadas se bloquearan tras alcanzar una condición de carrera durante la limpieza de un recurso de consulta.

Los cambios de la versión 1.3.2.1 se arrastraron desde la versión 1.3.2.0

Mejoras heredadas de la versión 1.3.2.0 del motor

Mejoras generales

- Support para la versión 1.3 de TLS, incluidos los conjuntos de cifrado TLS_AES_128_GCM_SHA256 y TLS_AES_256_GCM_SHA384. El TLS 1.3 es una opción; el TLS 1.2 sigue siendo el mínimo.
- El soporte extendido de OpenCypher para el formato dateime está en lab_mode para esta versión. Le animamos a que lo pruebe.

Mejoras en Gremlin

- TinkerPop Actualización 3.7.x
  - Proporciona una gran expansión del lenguaje Gremlin.
    - Nuevos pasos para procesar cadenas, listas y fechas.
    - Nueva sintaxis para especificar la cardinalidad del `mergeV()` paso.
    - `union()`ahora se puede utilizar como paso inicial.
    - Para obtener más información sobre los cambios de la versión 3.7.x, consulte la documentación de TinkerPop actualización.
  - Al actualizar los controladores de lenguaje Gremlin del cliente para Java, tenga en cuenta que las clases de serializadores han sufrido algunos cambios de nombre. Deberá actualizar los nombres de los paquetes y las clases en sus archivos de configuración y en el código, si se especifica.
- `StrictTimeoutValidation`(solo cuando se habilita mediante labmode `StrictTimeoutValidation` mediante la inclusión`StrictTimeoutValidation=enabled`): cuando el `StrictTimeoutValidation` parámetro tiene un valor de`enabled`, un valor de tiempo de espera por consulta especificado como opción de solicitud o sugerencia de consulta no puede superar el valor establecido globalmente en el grupo de parámetros. En tal caso, Neptune lanzará un. `InvalidParameterException` Esta configuración se puede confirmar en una respuesta en el `/status` punto final cuando el valor es`disabled`, y en las versiones 1.3.2.0 y 1.3.2.1 de Neptune, el valor predeterminado de este parámetro es. `Disabled`

Mejoras de openCypher

- Consultas de baja latencia y mejora del rendimiento: mejoras generales del rendimiento de las consultas OpenCypher de baja latencia. La nueva versión también mejora el rendimiento de dichas consultas. Las mejoras son más significativas cuando se utilizan consultas parametrizadas.
- Soporte para Query Plan Cache: cuando se envía una consulta a Neptune, la cadena de consulta se analiza, optimiza y transforma en un plan de consulta, que luego es ejecutado por el motor. Las aplicaciones suelen estar respaldadas por patrones de consulta comunes que se instancian con valores diferentes. La caché del plan de consultas puede reducir la latencia general al almacenar en caché los planes de consulta y, por lo tanto, evitar el análisis y la optimización de dichos patrones repetidos.
- Mejora del rendimiento de las distintas consultas de agregación.
- Mejora del rendimiento de las uniones que incluyen variables anulables.

- Mejora del rendimiento de las consultas que incluyen un predicado distinto de id (nodo/relación).
- Soporte ampliado para la funcionalidad de fecha y hora (solo se habilita en el modo laboratorio al incluir. `DatetimeMillisecond` `DatetimeMillisecond=enabled` Para obtener más información, consulte Soporte temporal en la implementación de Neptune OpenCypher (Neptune Analytics y Neptune Database 1.3.2.0 y versiones posteriores).

Correcciones de defectos incorporadas en la versión 1.3.2.0 del motor

Mejoras generales

- Se actualizó el mensaje de error de NeptuneML al validar el acceso a los depósitos de Graphlytics.

Correcciones de Gremlin

- Se corrigió la falta de información de etiqueta en la traducción de consultas del DFE, en situaciones en las que los pasos que no contribuían a la ruta contenían etiquetas. Por ejemplo:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Se ha `NullPointerException` corregido un error en la traducción de las consultas del DFE, que se producía cuando una consulta se ejecutaba en dos fragmentos del DFE y el primer fragmento se optimizaba para convertirla en un nodo insatisfactorio. Por ejemplo:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Se corrigió un error por el que Neptune podía lanzar un `InternalFailureException` cuando una consulta contenía el modulador by () ValueTraversal interno y su entrada era Map. Por ejemplo:

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

Correcciones de openCypher

- Se han mejorado las operaciones UNWIND (por ejemplo, ampliar una lista de valores en valores individuales) para evitar situaciones de falta de memoria (OOM). Por ejemplo:

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Se ha corregido la optimización de la identificación personalizada en el caso de múltiples operaciones de combinación en las que la identificación se inyectaba mediante UNWIND. Por ejemplo:

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Se corrigió el problema de la pérdida de memoria al planificar consultas complejas con acceso a propiedades y saltos múltiples con relaciones bidireccionales. Por ejemplo:

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
      (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
      (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
      (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
      (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
```

```
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Consultas de agregación fijas con valores nulos agrupados por variables. Por ejemplo:

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Correcciones de SPARQL

- Se corrigió el analizador SPARQL para mejorar el tiempo de análisis de consultas grandes, como INSERT DATA, que contienen muchos triples y símbolos grandes.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.3.2.1, asegúrese de que su proyecto sea compatible con las siguientes versiones del lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.7.1`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.3.2.1 del motor

Puede actualizar a esta versión desde la versión 1.2.0.0 o superior del motor.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de una versión principal, se requiere el allow-major-version-upgrade parámetro. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tienes alguna pregunta o duda, el equipo de AWS Soporte está disponible en los foros de la comunidad y a través del Soporte AWS Premium.

Amazon Neptune Engine versión 1.3.2.0 (10/06/2020)

A partir del 10 de junio de 2022, la versión 1.3.2.0 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> En la versión 1.3.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.3.0.0 a una versión de motor 1.3.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.3`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1` o `neptune1.2`, y esos grupos de parámetros no funcionan con la versión 1.3.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

> **Warning**
>
> Hemos detectado un problema en la memoria caché del plan de consultas cuando se utiliza `skip` o se utiliza en una cláusula interna y `limit` está parametrizada. `WITH` Para evitar este problema, añade la sugerencia de consulta `QUERY:PLANCACHE "disabled"` cuando envíes una consulta que incluya una subcláusula parametrizada para omitir o limitar. Como alternativa, puede codificar los valores en la consulta. Para más información, consulte Se ha mitigado el problema de la caché del plan de consultas.

Mejoras en esta versión del motor

Mejoras generales

- Support para la versión 1.3 de TLS, incluidos los conjuntos de cifrado TLS_AES_128_GCM_SHA256 y TLS_AES_256_GCM_SHA384. El TLS 1.3 es una opción; el TLS 1.2 sigue siendo el mínimo.

Mejoras en Gremlin

- TinkerPop Actualización a la versión 3.7.x
  - Proporciona una gran expansión del lenguaje Gremlin.
    - Nuevos pasos para procesar cadenas, listas y fechas.
    - Nueva sintaxis para especificar la cardinalidad del `mergeV()` paso.
    - `union()`ahora se puede utilizar como paso inicial.
    - Para obtener más información sobre los cambios de la versión 3.7.x, consulte la documentación de TinkerPop actualización.
  - Al actualizar los controladores de lenguaje Gremlin del cliente para Java, tenga en cuenta que las clases de serializadores han sufrido algunos cambios de nombre. Deberá actualizar los nombres de los paquetes y las clases en sus archivos de configuración y en el código, si se especifica.
- `StrictTimeoutValidation`(solo cuando se habilita mediante labmode `StrictTimeoutValidation` mediante la inclusión`StrictTimeoutValidation=enabled`): cuando el `StrictTimeoutValidation` parámetro tiene un valor de`enabled`, un valor de tiempo de espera por consulta especificado como opción de solicitud o sugerencia de consulta no puede superar el valor establecido globalmente en el grupo de parámetros. En tal caso, Neptune lanzará un. `InvalidParameterException` Esta configuración se puede confirmar en una respuesta en el `/status` punto final cuando el valor es`disabled`, y en la versión 1.3.2.0 de Neptune, el valor predeterminado de este parámetro es. `Disabled`

Mejoras de openCypher

- Consultas de baja latencia y mejora del rendimiento: mejoras generales del rendimiento de las consultas OpenCypher de baja latencia. La nueva versión también mejora el rendimiento de dichas consultas. Las mejoras son más significativas cuando se utilizan consultas parametrizadas.
- Soporte para Query Plan Cache: cuando se envía una consulta a Neptune, la cadena de consulta se analiza, optimiza y transforma en un plan de consulta, que luego es ejecutado por el motor. Las aplicaciones suelen estar respaldadas por patrones de consulta comunes que se instancian con valores diferentes. La caché del plan de consultas puede reducir la latencia general al almacenar en caché los planes de consulta y, por lo tanto, evitar el análisis y la optimización de dichos patrones repetidos.
- Mejora del rendimiento de las distintas consultas de agregación.
- Mejora del rendimiento de las uniones que incluyen variables anulables.

- Mejora del rendimiento de las consultas que incluyen un predicado distinto de id (nodo/relación).
- Soporte ampliado para la funcionalidad de fecha y hora (solo se habilita en el modo laboratorio al incluir. `DatetimeMillisecond` `DatetimeMillisecond=enabled` Para obtener más información, consulte Soporte temporal en la implementación de Neptune OpenCypher (Neptune Analytics y Neptune Database 1.3.2.0 y versiones posteriores).

Defectos corregidos en esta versión del motor

Mejoras generales

- Se actualizó el mensaje de error de NeptuneML al validar el acceso a los depósitos de Graphlytics.

Correcciones de Gremlin

- Se corrigió la falta de información de etiqueta en la traducción de consultas del DFE, en situaciones en las que los pasos que no contribuían a la ruta contenían etiquetas. Por ejemplo:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'marko').
  has("name", TextP.regex("mark.*")).as("p1").
  not(out().has("name", P.within("peter"))).
  out().as('p2').
  dedup('p1', 'p2')
```

- Se ha `NullPointerException` corregido un error en la traducción de las consultas del DFE, que se producía cuando una consulta se ejecutaba en dos fragmentos del DFE y el primer fragmento se optimizaba para convertirla en un nodo insatisfactorio. Por ejemplo:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Se corrigió un error por el que Neptune podía lanzar un `InternalFailureException` cuando una consulta contenía el modulador by () ValueTraversal interno y su entrada era Map. Por ejemplo:

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

Correcciones de openCypher

- Se han mejorado las operaciones UNWIND (por ejemplo, ampliar una lista de valores para convertirla en valores individuales) para evitar situaciones de falta de memoria (OOM). Por ejemplo:

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Se ha corregido la optimización de la identificación personalizada en el caso de múltiples operaciones de combinación en las que la identificación se inyectaba mediante UNWIND. Por ejemplo:

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Se corrigió el problema de la pérdida de memoria al planificar consultas complejas con acceso a propiedades y saltos múltiples con relaciones bidireccionales. Por ejemplo:

```
MATCH (person1:person)-[:likes]->(res)-[:partOf]->(group)-[:knows]-(:entity {name:
 'foo'}),
       (person1)-[:knows]->(person2)-[:likes]-(res2), (comment)-[:presentIn]->(:Group
 {name: 'barGroup'}),
       (person1)-[:commented]->(comment2:comment)-[:partOf]->(post:Post), (comment2)-
[:presentIn]->(:Group {name: 'fooGroup'}),
       (comment)-[:contains]->(info:Details)-[:CommentType]->(:CommentType {name:
 'Positive'}),
```

```
       (comment2)-[:contains]->(info2:Details)-[:CommentType]->(:CommentType {name:
 'Positive'})
WHERE datetime('2020-01-01T00:00') <= person1.addedAfter <=
 datetime('2023-01-01T23:59') AND comment.approvedBy = comment2.approvedBy
MATCH (comment)-[:contains]->(info3:Details)-[:CommentType]->(:CommentType {name:
 'Neutral'})
RETURN person1, group.name, info1.value,  post.ranking, info3.value
```

- Consultas de agregación fijas con valores nulos agrupados por variables. Por ejemplo:

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Correcciones de SPARQL

- Se corrigió el analizador SPARQL para mejorar el tiempo de análisis de consultas grandes, como INSERT DATA, que contienen muchos triples y símbolos grandes.

Se ha mitigado el problema de la caché del plan de consultas

En la versión 1.3.2.0, hemos detectado un problema en la memoria caché del plan de consultas cuando se utiliza `skip` o `limit` se utiliza en una `WITH` cláusula interna y está parametrizada. Por ejemplo:

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip $skip LIMIT $limit
RETURN n.name, n.age

parameters={"age": 21, "skip": 2, "limit": 3}
```

En este caso, los valores de los parámetros de omisión y limitación del primer plan también se aplicarán a las consultas posteriores, lo que generará resultados inesperados.

Mitigación

Para evitar este problema, añada la sugerencia de consulta `QUERY:PLANCACHE "disabled"` cuando envíe una consulta que incluya una subcláusula parametrizada para omitir o limitar. Como alternativa, puede codificar los valores en la consulta.

Opción 1: usar la sugerencia de consulta para deshabilitar la memoria caché del plan:

```
Using QUERY:PLANCACHE "disabled"
MATCH (n:Person) WHERE n.age > $age
WITH n skip $skip LIMIT $limit
RETURN n.name, n.age

parameters={"age": 21, "skip": 2, "limit": 3}
```

Opción 2: usar valores codificados de forma rígida para omitir y limitar:

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip 2 LIMIT 3
RETURN n.name, n.age

parameters={"age": 21}
```

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.3.2.0, asegúrese de que su proyecto sea compatible con las siguientes versiones del lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.7.1`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.3.2.0 del motor

Puede actualizar a esta versión desde la versión 1.2.0.0 o superior del motor.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de una versión principal, se requiere el allow-major-version-upgrade parámetro. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tienes alguna pregunta o duda, el equipo de AWS Soporte está disponible en los foros de la comunidad y a través del Soporte AWS Premium.

Amazon Neptune Engine versión 1.3.1.0 (06/03/2022)

A partir del 3 de marzo de 2022, la versión 1.3.1.0 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> En la versión 1.3.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.3.0.0 a una versión de motor 1.3.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.3`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1` o `neptune1.2`, y esos grupos de parámetros no funcionan con la versión 1.3.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

Mejoras en esta versión del motor

Mejoras generales

- Neptune ha mejorado la advertencia que se muestra en el perfil/explain.
- Se eliminaron las curvas EC del NIST obsoletas de los grupos con nombres predeterminados utilizados durante la negociación de TLS. Las curvas eliminadas son sect409k1, sect409r1 y sect571k1.

Mejoras en Gremlin

- Se mejoró el cálculo de las estadísticas del DFE para evitar que las instancias sin servidor fueran muy altas en las NCUs.
- Mejora del rendimiento de Gremlin para WITHIN.

Defectos corregidos en esta versión del motor

Correcciones de Gremlin

- Mejoras diversas en los planes de consulta del DFE de Gremlin.
- Se corrigió un error para las consultas de Gremlin con un recorrido opcional, por ejemplo, para las consultas con el formato `G.v () .hasLabel ('person') .group () .by (id ()) .by (__.in ('friend') .id () .fold ()) `, en las que no se agrupaba a ninguna persona sin perfil de amigo.
- Se corrigió un error por el que las consultas de Gremlin que contenían pasos de fusión dentro de los by moduladores provocaban que se devolviera un error si se ejecutaban con el motor DFE.
- Se ha corregido un error que impedía que las consultas de solo lectura que se ejecutaban en una sesión de Gremlin funcionaran cuando se conectaban a una réplica de lectura.
- Se corrigió un error por el que el ARN de IAM no estaba presente en el registro de auditoría para una solicitud de conexión websocket inicial exitosa para Gremlin.
- Combine los pasos e identifique la cobertura de los pasos con el DFE.
- Optimización del conjunto de características para planes de DFE completos.

Correcciones de openCypher

- Se corrigieron errores en la cláusula SET de OpenCypher para permitir la configuración en expresiones no variables (es decir, match (n:Test) set (mayúscula cuando n.prop = 2 y luego n end) .prop = 3 devuelve n.prop.
- Se ha corregido un error que provocaba errores en las consultas de OpenCypher relacionadas con la agregación y el orden por.
- Se ha mejorado la función de desenrollar una gran lista que contiene mapas estáticos.
- Se corrigió un error en la consulta OpenCypher MERGE que utilizaba un identificador personalizado con valores duplicados.

Correcciones de SPARQL

- Se ha corregido un error de SPARQL relacionado con el ámbito de la variable en los patrones opcionales.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.3.1.0, asegúrese de que su proyecto sea compatible con las siguientes versiones del lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.5`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.3.1.0 del motor

Puede actualizar a esta versión desde la versión 1.2.0.0 o superior del motor.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de una versión principal, se requiere el allow-major-version-upgrade parámetro. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado,

así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tienes alguna pregunta o duda, el equipo de AWS Soporte está disponible en los foros de la comunidad y a través del Soporte AWS Premium.

Versión 1.3.0.0 del motor de Amazon Neptune (15/11/2021)

A partir del 15 de noviembre de 2023, se implementará de forma general la versión 1.3.0.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> En la versión 1.3.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.3.0.0 a una versión de motor 1.3.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.3`. En las versiones anteriores, se utilizaba la familia de grupos de

parámetros `neptune1` o `neptune1.2`, y esos grupos de parámetros no funcionan con la versión 1.3.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

Nuevas características de esta versión del motor

- Se ha lanzado la API de datos de Neptune.

  La API de datos de Amazon Neptune proporciona compatibilidad con el SDK para más de 40 operaciones de datos de Neptune, incluidas la carga de datos, la ejecución de consultas, la consulta de datos y machine learning. Es compatible con los tres lenguajes de consulta de Neptune (Gremlin, openCypher y SPARQL) y está disponible en todos los lenguajes del SDK. Firma automáticamente las solicitudes de la API y simplifica considerablemente la integración de Neptune en sus aplicaciones.
- Se agregó soporte para integrar OpenSearchServerless con Neptune.

Mejoras en esta versión del motor

Mejoras en las actualizaciones del motor de Neptune

Neptune ha cambiado la forma en que publica las actualizaciones del motor para que pueda tener más control sobre el proceso de actualización. En lugar de publicar parches para cambios de no separación, Neptune ahora publica versiones secundarias que se pueden controlar mediante el campo de instancia `AutoMinorVersionUpgrade` y sobre las que puede recibir notificaciones suscribiéndose al evento `RDS-EVENT-0156`.

Consulte Mantenimiento del clúster de base de datos de Amazon Neptune para obtener más información sobre estos cambios.

Mejora del cifrado en tránsito

Neptune ya no admite los conjuntos de cifrado siguientes:

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Neptune solo admite los siguientes conjuntos de cifrado seguro con TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

Mejoras de Gremlin

- Se ha añadido compatibilidad con el motor DFE para los siguientes pasos de Gremlin.
  - `FoldStep`
  - `GroupStep`
  - `GroupCountStep`
  - `TraversalMapStep`
  - `UnfoldStep`
  - `LabelStep`
  - `PropertyKeyStep`
  - `PropertyValueStep`
  - `AndStep`
  - `OrStep`
  - `ConstantStep`
  - `CountGlobalStep`
- Se han optimizado los planes de consulta de DFE de Gremlin para evitar escaneos completos de vértices cuando se utiliza la modulación `by()`.
- Se ha mejorado considerablemente el rendimiento de las consultas de cardinalidad baja y latencia baja.
- Se agregó compatibilidad con DFE para TinkerPop `0r` los predicados de filtro.
- Se ha mejorado la compatibilidad con DFE del recorrido para filtros en la misma clave, para consultas como las siguientes:

```
g.withSideEffect("Neptune#useDFE", true)
 .V()
 .has('name', 'marko')
 .and(
   or(
     has('name', eq("marko")),
```

```
    has('name', eq("vardas"))
  )
)
```

- Se ha mejorado el control de errores del paso `fail()`.

Mejoras de openCypher

- Se ha mejorado considerablemente el rendimiento de las consultas de cardinalidad baja y latencia baja.
- Se ha mejorado el rendimiento de planificación de consultas cuando la consulta contiene muchos tipos de nodos.
- Se ha reducido la latencia de todas las consultas de VLP.
- Se ha mejorado el rendimiento al eliminar las uniones de canalización redundantes para consultas de patrones de un único nodo.
- Se ha mejorado el rendimiento para consultas que contienen patrones de varios saltos con ciclos, como este:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

Mejoras de SPARQL

- Se ha introducido un nuevo operador SPARQL: `PipelineHashIndexJoin`
- Se ha mejorado el rendimiento de la validación de URI para consultas SPARQL.
- Se ha mejorado el rendimiento de las consultas de búsqueda de texto completo de SPARQL mediante la resolución por lotes de términos del diccionario.

Defectos corregidos en esta versión del motor

Correcciones de Gremlin

- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones al comprobar el punto de conexión del estado de la consulta de Gremlin de las consultas con predicados en los recorridos secundarios para los pasos que no se procesaban de forma nativa en el motor DFE.

- Se ha corregido un error de Gremlin que provocaba que `valueMap()` no se optimizara en el motor DFE en recorridos `by()`.
- Se ha corregido un error de Gremlin en el que una etiqueta de paso adjunta a `UnionStep` no se propagase al último elemento de la ruta de sus recorridos secundarios.
- Se ha corregido un error de Gremlin que provocaba que una consulta fallara porque contenía demasiados TinkerPop pasos y, por lo tanto, no se limpiaba.
- Se ha corregido un error de Gremlin en el que se generaba una `NullPointerException` en los pasos `mergeV` y `mergeE`
- Se ha corregido un error de Gremlin por el que `order()` no podía ordenar correctamente las salidas de cadena cuando algunas de ellas incluían un carácter de espacio.
- Se ha corregido un problema de corrección de Gremlin que se producía cuando el paso `valueMap` se procesaba en el motor DFE.
- Se ha corregido un problema de corrección de Gremlin que se producía cuando `GroupStep` o `GroupCountStep` se anidaban en un recorrido de clave.

Correcciones de openCypher

- Se ha corregido un error de openCypher que implicaba el control de errores en torno a los caracteres NULL.
- Se ha corregido un error en el control de transacciones de openCypher Bolt.

Correcciones de SPARQL

- Se ha corregido un error de SPARQL que provocaba que los valores dentro de las funciones recursivas no se resolvieran correctamente.
- Se ha corregido un error de SPARQL que provocaba una disminución del rendimiento cuando se inyectaban un gran número de valores mediante la cláusula `VALUES`.
- Se ha corregido un error de SPARQL que el operador `REGEX` no funcionara correctamente cuando se llamaba desde un literal etiquetado en un lenguaje.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.3.0.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.3.0.0 del motor

Puede actualizar a esta versión desde la versión 1.2.0.0 o superior del motor.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización importante de una versión, se requiere el allow-major-version-upgrade parámetro. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tienes alguna pregunta o duda, el equipo de AWS Soporte está disponible en los foros de la comunidad y a través del Soporte AWS Premium.

Amazon Neptune Engine, versión 1.2.1.1 (11 de marzo de 2022)

A partir del 11 de marzo de 2020, la versión 1.2.1.1 del motor se implementará de forma general. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:
>
> - En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. En consecuencia, todos los registros de deshacer creados por una versión anterior del motor deben purgarse y la `UndoLogsListSize` CloudWatch métrica debe reducirse a cero antes de poder iniciar cualquier actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la `UndoLogsListSize` CloudWatch métrica es muy grande, abrir un caso de soporte puede ayudarte a explorar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

Mejoras generales

Neptune ha mejorado la advertencia que se muestra en el perfil/explain.

Mejoras de Gremlin

- Se ha mejorado el cálculo de las estadísticas del DFE para evitar niveles muy altos de NCUs en las instancias sin servidor.
- Mejora del rendimiento de Gremlin para WITHIN.

Defectos corregidos en esta versión del motor

Correcciones de Gremlin

- Se corrigieron errores al solicitar el plan de consultas del motor DFE de Gremlin.
- Se corrigió un error con el error de Gremlin cuando out-of-memory se informó originalmente como. InternalFailureException
- Se corrigió un error por el que el ARN de IAM no estaba presente en el registro de auditoría para una solicitud de conexión inicial de websocket exitosa.
- Se corrigió un error para las consultas de Gremlin con la TinkerPop sesión habilitada cuando las consultas de una sesión fallaban, incluso cuando todas eran de solo lectura y se conectaban a una instancia de lectura.

Correcciones de openCypher

- Se corrigieron errores en la cláusula SET de OpenCypher para permitir la configuración de expresiones no variables (por ejemplo: match (n:Test) set (mayúscula cuando n.prop = 2 y luego n end) .prop = 3 devuelve n.prop.
- Se ha corregido un error que provocaba errores en las consultas de OpenCypher relacionadas con la agregación y el orden por.
- Se ha mejorado la función de desenrollar una gran lista que contiene mapas estáticos.
- Se corrigió un error en la consulta OpenCypher MERGE que utilizaba un identificador personalizado con valores duplicados.

Correcciones de SPARQL

- Se corrigieron errores en el planificador de consultas SPARQL DFE.
- Se ha corregido un error en SPARQL cuando se utilizaba con las palabras clave BIND y OPTIONAL.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.1, asegúrese de que su proyecto sea compatible con las siguientes versiones del lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`

- Compatible con la última versión de Gremlin: `3.6.2`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Actualice las rutas a la versión 1.2.1.1 del motor

Puede actualizar a esta versión desde la versión 1.2.0.0 o superior del motor.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de una versión principal, se requiere el allow-major-version-upgrade parámetro. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tienes alguna pregunta o duda, el equipo de AWS Soporte está disponible en los foros de la comunidad y a través del Soporte AWS Premium.

Versión 1.2.1.0 del motor de Amazon Neptune (08/03/2023)

A partir del 8 de marzo de 2023, se implementará de forma general la versión 1.2.1.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de

CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Versiones de parche posteriores para esta versión

- Versión: 1.2.1.0.R2 (02/05/2023)
- Versión: 1.2.1.0.R3 (13/06/2023)
- Versión: 1.2.1.0.R4 (10/08/2023)
- Versión: 1.2.1.0.R5 (02/09/2023)
- Versión: 1.2.1.0.R6 (12/09/2023)
- Versión: 1.2.1.0.R7 (06/10/2023)

Nuevas características de esta versión del motor

- Se ha añadido soporte para TinkerPop 3.6.2, que añade muchas características nuevas de Gremlin, como los pasos nuevos `mergeV()`, `mergeE()`, `element()` y `fail()`. Los pasos `mergeV()` y `mergeE()` son de especial interés, ya que ofrecen una opción declarativa muy esperada para realizar operaciones tipo actualización o inserción, lo que debería simplificar en

gran medida los patrones de código existentes y facilitar la lectura de Gremlin. La versión 3.6.x también ha añadido predicados de expresiones regulares, lo que supone una nueva sobrecarga para el paso `property()` que requiere una `Map` y una revisión principal del comportamiento de modulación `by()`, que es mucho más coherente en todos los pasos en los que se utiliza.

Consulta el registro cambios de TinkerPop y la página de actualizaciones para obtener información sobre los cambios en la versión 3.6 y los aspectos a tener en cuenta al realizar la actualización.

Si utiliza `fold().coalesce(unfold(), <mutate>)` para las inserciones condicionales, te recomendamos migrar a la nueva sintaxis `mergeV/E()`, que se describe aquí y aquí. Neptune utiliza un patrón de bloqueo más limitado para `Merge` que para `Coalesce`, lo que puede reducir las excepciones de modificación simultánea (CME).

Para obtener más información sobre las nuevas características disponibles en esta versión de TinkerPop, consulte el blog de Stephen Mallette, Exploring new features of Apache TinkerPop 3.6.x in Amazon Neptune (Exploración de las nuevas características de Apache TinkerPop 3.6x en Amazon Neptune).

- Se ha añadido compatibilidad con los tipos de instancias R6i, con tecnología de procesadores Intel Xeon Scalable de 3.ª generación. Son ideales para cargas de trabajo con uso intensivo de memoria y ofrecen hasta un 15 % más de rendimiento informático y hasta un 20 % más de ancho de banda de memoria por vCPU que los tipos de instancias R5 comparables.
- Se han añadido puntos de conexión de la API de resumen de gráficos tanto para los gráficos de propiedades como para los gráficos RDF, que le permiten obtener un informe resumido rápido sobre el gráfico.

  En el caso de los gráficos de propiedades (PG), la API de resumen de gráficos devuelve una lista de solo lectura de etiquetas de nodos y bordes y claves de propiedades, junto con el recuento de nodos, bordes y propiedades. En el caso de los gráficos RDF, proporciona una lista de clases y claves de predicados, junto con recuentos de cuádruples, sujetos y predicados.

  La nueva API de resumen de gráficos ha incluido los siguientes cambios:

  - Se ha añadido una nueva acción del plano de datos GetGraphSummary.
  - Se ha añadido un nuevo punto de conexión `rdf/statistics` para reemplazar el punto de conexión `sparql/statistics`, que ahora está obsoleto.
  - Se ha cambiado el nombre del campo `summary` en la respuesta de estado de las estadísticas a `signatureInfo` para no confundirlo con la información de resumen del gráfico. Las versiones anteriores del motor siguen utilizando `summary` en la respuesta de JSON.

  - Se ha cambiado la precisión del campo `date` en la respuesta de estado de las estadísticas de un minuto a un milisegundo. El formato anterior era `2020-05-07T23:13Z` (precisión en minutos), mientras que el nuevo formato es `2023-01-24T00:47:43.319Z` (precisión en milisegundos). Ambos cumplen con la norma ISO 8601, pero este cambio puede interrumpir el código existente, según cómo se analice la fecha.
  - Se ha añadido un nuevo comando mágico de línea `%statistics` en el entorno de trabajo que permite recuperar las estadísticas del motor DFE.
  - Se ha añadido un nuevo comando mágico de línea `%summary` en el entorno de trabajo que permite recuperar información del resumen de gráficos.
- Se ha añadido el registro de consultas lentas para registrar las consultas que tardan más en ejecutarse que un umbral especificado. Para habilitar y controlar el registro de consultas lentas, se utilizan los dos nuevos parámetros dinámicos, es decir, neptune_enable_slow_query_log y neptune_slow_query_log_threshold.
- Se ha añadido compatibilidad con dos parámetros dinámicos, los nuevos parámetros de clúster, neptune_enable_slow_query_log y neptune_slow_query_log_threshold. Al realizar un cambio en un parámetro dinámico, se aplica inmediatamente, sin necesidad de reiniciar la instancia.
- Se ha añadido una función removeKeyFromMap() de openCypher específica de Neptune que elimina una clave específica de un mapa y devuelve la nueva asignación resultante.

Mejoras en esta versión del motor

- Se ha ampliado la compatibilidad con el DFE de Gremlin a los pasos `limit` con alcance local.
- Se ha añadido soporte de modulación `by()` para el `DedupGlobalStep` del Gremlin en el motor DFE.
- Se ha añadido compatibilidad con DFE para Gremlin `SelectStep` y `SelectOneStep`.
- Mejoras en el rendimiento y correcciones para varios operadores de Gremlin, incluidos `repeat`, `coalesce`, `store` y `aggregate`.
- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `MERGE` y `OPTIONAL MATCH`.
- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `UNWIND` de una lista de mapas de valores literales.
- Se ha mejorado el rendimiento de las consultas de openCypher que tienen un filtro `IN` para `id`. Por ejemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Se ha añadido la posibilidad de especificar el IRI base para las consultas de SPARQL mediante la instrucción BASE (consulte IRI base predeterminado para consultas y actualizaciones).
- Se ha reducido el tiempo de espera para el procesamiento de carga para las cargas masivas de solo bordes de Gremlin y openCypher.
- Se ha logrado que las cargas masivas se reanuden de forma asíncrona cuando Neptune se reinicia para evitar el prolongado tiempo de espera provocado por problemas de conectividad de Amazon S3 antes de que se produzca un error en los intentos de reanudación.
- Se ha mejorado la gestión de las consultas DESCRIBE de SPARQL que tienen la sugerencia de consulta describeMode establecida en `"CBD"` (descripción concisa y limitada) y que implican una gran cantidad de nodos en blanco.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher que provocaba que las consultas devolvieran la cadena `"null"`, en lugar de un valor nulo en Bolt y SPARQL-JSON.
- Se ha corregido un error de openCypher en la comprensión de las listas que producía un valor nulo en lugar de los valores proporcionados para los elementos de la lista.
- Se ha corregido un error de openCypher que provocaba que los valores de los bytes no se serializaran correctamente.
- Se ha corregido un error de Gremlin en `UnionStep` que se producía cuando una entrada era un recorrido de borde hacia un vértice dentro de un recorrido secundario.
- Se ha corregido un error de Gremlin que provocaba que una etiqueta de paso asociada a `UnionStep` no se propagase correctamente hasta el último paso del recorrido secundario.
- Se ha corregido un error de Gremlin en el paso `dedup` con etiquetas que seguían a un paso `repeat`, por el que las etiquetas adjuntas al paso `dedup` no estaban disponibles para utilizarlas en consultas posteriores.
- Se ha corregido un error de Gremlin que provocaba que al traducir el paso `repeat` dentro de un paso `union` se produjera un error interno.
- Se ha corregido un error de corrección de Gremlin en las consultas del DFE con `limit` como recorrido secundario de pasos no unidos a la unión que recurrían a Tinkerpop. Las consultas en un formato como este se ven afectadas:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Se ha corregido un error de SPARQL por el que los patrones `SPARQL GRAPH` no tenían en cuenta el conjunto de datos proporcionado por una cláusula `FROM NAMED`.
- Se ha corregido un error de SPARQL que provocaba que `DESCRIBE` de SPARQL con algunas clausulas `FROM` o `FROM NAMED` no siempre utilizaba correctamente los datos del gráfico predeterminado y, a veces, generaba una excepción. Consulte Comportamiento de DESCRIBE de SPARQL con respecto al gráfico predeterminado.
- Se ha corregido un error de SPARQL para que se devolviera el mensaje de excepción correcto cuando se rechazaban caracteres nulos.
- Se ha corregido un error explain de SPARQL que afectaba a los planes que incluían un operador PipelinedHashIndexJoin.
- Se ha corregido un error que podía provocar un error interno cuando se enviaba una consulta que devolvía un valor constante.
- Se ha corregido un problema con la lógica del detector de bloqueos que, en ocasiones, hacía que el motor dejara de responder.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.2`
- Versión de openCypher: `Neptune-9.0.20190305-1.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.1.0 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a la versión 1.1.0.0 o una versión posterior.

> **Note**
>
> A partir de la versión 1.2.0.0 del motor, todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados que utilizaba con las versiones del motor anteriores a `1.2.0.0` deben volver a crearse mediante la familia de grupos de parámetros `neptune1.2`. Las versiones anteriores utilizaban la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionarán a partir de la versión `1.2.0.0`. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

No se actualizará automáticamente a esta versión principal.

Actualización a esta versión

La versión 1.2.1.0 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.1.0.R7 del motor de Amazon Neptune (06/10/2023)

A partir del 6 de octubre de 2023, se implementará de forma general la versión 1.2.1.0.R7 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño

del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Defectos corregidos en esta versión del motor

- Se ha corregido un error en el que, en algunos casos, una transacción errónea no se cerraba correctamente.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.0.R7, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.2`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Actualización a esta versión

La versión 1.2.1.0.R7 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.1.0.R6 del motor de Amazon Neptune (12/09/2023)

A partir del 12 de septiembre de 2023, se implementará de forma general la versión 1.2.1.0.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:
>
> - En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una

versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Nuevas características de esta versión del motor

- Se ha lanzado la API de datos de Neptune.

  La API de datos de Amazon Neptune proporciona compatibilidad con el SDK para cargar datos, ejecutar consultas, obtener información sobre los datos y ejecutar operaciones de machine learning. Es compatible con los lenguajes de consulta de Gremlin y openCypher en Neptune y

está disponible en todos los lenguajes del SDK. Firma automáticamente las solicitudes de la API y simplifica considerablemente la integración de Neptune en las aplicaciones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error que podía provocar picos de CPU con cargas elevadas al habilitar los registros de consultas lentas.
- Se ha corregido un error de Gremlin que provocaba que se añadiera un borde y sus propiedades seguidas de `inV()`, o que `outV()` generara una `InternalFailureException`.
- Se han corregido varios problemas relacionados con el encadenamiento de roles de IAM que, en algunos casos, provocaban una disminución en el rendimiento del programa de carga masiva.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.0.R6, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.2`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Actualización a esta versión

La versión 1.2.1.0.R6 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
```

```
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea

manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.1.0.R5 del motor de Amazon Neptune (02/09/2023)

A partir del 2 de septiembre de 2023, se implementará de forma general la versión 1.2.1.0.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:
>
> - En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba

la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Nuevas características de esta versión del motor

- Se ha lanzado la API de datos de Neptune.

  La API de datos de Amazon Neptune proporciona compatibilidad con el SDK para cargar datos, ejecutar consultas, obtener información sobre los datos y ejecutar operaciones de machine learning. Es compatible con los lenguajes de consulta de Gremlin y openCypher en Neptune y está disponible en todos los lenguajes del SDK. Firma automáticamente las solicitudes de la API y simplifica considerablemente la integración de Neptune en las aplicaciones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin que provocaba que se añadiera un borde y sus propiedades seguidas de `inV()`, o que `outV()` generara una `InternalFailureException`.
- Se han corregido varios problemas relacionados con el encadenamiento de roles de IAM que, en algunos casos, provocaban una disminución en el rendimiento del programa de carga masiva.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.0.R5, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.2`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Actualización a esta versión

La versión 1.2.1.0.R5 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.1.0.R4 del motor de Amazon Neptune (10/08/2023)

A partir del 10 de agosto de 2023, se implementará de forma general la versión 1.2.1.0.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

En algunos casos, ese posible que los cambios introducidos en esta versión del motor provoquen una disminución en el rendimiento de las cargas masivas. En consecuencia, las actualizaciones a esta versión se han suspendido temporalmente hasta que se resuelva el problema.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando

la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Añada compatibilidad con GraphSON-1.0 para Gremlin. Para usar GraphSON-1.0, pase `Accept header` con un valor de:

```
application/vnd.gremlin-v1.0+json;types=false
```

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones al comprobar el punto de conexión del estado de la consulta de Gremlin de las consultas con predicados en los recorridos secundarios para los pasos que no se procesaban de forma nativa.
- Se ha corregido un error de openCypher en la gestión de transacciones de Bolt.
- Se ha corregido un problema de simultaneidad en la capa de almacenamiento que podía provocar un bloqueo.
- Se ha corregido un error en los registros de consultas lentos para garantizar que no estén activos cuando están deshabilitados.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.0.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.5`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.1.0.R4 del motor

Actualización a esta versión

La versión 1.2.1.0.R4 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado,

así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.1.0.R3 del motor de Amazon Neptune (13/06/2023)

A partir del 13 de junio de 2023, se implementará de forma general la versión 1.2.1.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Important**
>
> En algunos casos, ese posible que los cambios introducidos en esta versión del motor provoquen una disminución en el rendimiento de las cargas masivas. En consecuencia, las actualizaciones a esta versión se han suspendido temporalmente hasta que se resuelva el problema.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Nuevas características de esta versión del motor

- Se ha añadido compatibilidad con la carga masiva entre cuentas mediante el encadenamiento de roles de IAM.

Mejoras en esta versión del motor

- Se ha mejorado el paso `fail()` de Gremlin para diferenciar la excepción que se producía de una `InternalFailureException` genérica y garantizar que cualquier mensaje proporcionado por el usuario que se le proporcionase se propagase de nuevo al intermediario.
- Se han mejorado las optimizaciones del motor de consultas de Gremlin para `store`, `aggregate`, `cap`, `limit` y `hasLabel`.
- Se ha añadido compatibilidad con las funciones trigonométricas de openCypher:
  - `acos()`
  - `asin()`
  - `atan()`
  - `atan2()`
  - `cos()`
  - `cot()`
  - `degrees()`
  - `pi()`
  - `radians()`
  - `sin()`
  - `tan()`
- Se ha añadido compatibilidad con varias funciones de agregación de openCypher:
  - `percentileDisc()`
  - `stDev()`
- Se ha añadido compatibilidad con la función `epochmillis()` de openCypher que convierte `datetime` en `epochmillis`. Por ejemplo:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

- Se ha añadido compatibilidad con el operador (`%`) del módulo de openCypher.

- Se ha añadido compatibilidad con la herramienta Static Debug Explain de openCypher.
- Se ha añadido compatibilidad con la función `randomUUID()` de openCypher.
- Se ha mejorado el rendimiento de openCypher:
  - Se han mejorado el analizador y el planificador de consultas.
  - Se ha mejorado el uso de la CPU en el motor DFE.
  - Se ha mejorado el rendimiento de las consultas que incluyen varias cláusulas de actualización que reutilizan las mismas variables. Algunos ejemplos son:

```
MERGE (n {name: 'John'})
  or
MERGE (m {name: 'Jim'})
  or
MERGE (n)-[:knows {since: 2023}]#(m)
```

  - Planes optimizados de consultas para patrones de consulta de varios saltos, como, por ejemplo:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

  - Se ha mejorado el rendimiento de la inyección de listas y mapas mediante consultas parametrizadas. Por ejemplo:

```
UNWIND $idList as id MATCH (n {`~id`: id})
RETURN n.name
```

  - Se ha mejorado la ejecución de consultas que incluye `WITH` al convertirla en una barrera adecuada.
  - Se ha optimizado para evitar la materialización redundante de valores en `Unfold` y las funciones de agregación.
- Se ha mejorado el rendimiento de las consultas de SPARQL que incluyen un gran número de entradas estáticas en la cláusula `VALUES`, como, por ejemplo:

```
SELECT ?n WHERE { VALUES (?name) { ("John") ("Jim") ... many values ... } ?n a ?
n_type . ?n ?name . }
```

- Se ha mejorado el rendimiento de las consultas CBD de SPARQL.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin que provocaba que las consultas largas con un anidamiento profundo provocaran un uso elevado de la CPU y tiempos de espera de las consultas durante la fase de planificación de las consultas.
- Se ha corregido un error de Gremlin que provocaba que se pudiera lanzar una `NullPointerException` no válida al usar `mergeV` o `mergeE`.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.0.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.2`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.1.0.R3 del motor

Actualización a esta versión

La versión 1.2.1.0.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.1.0.R2 del motor de Amazon Neptune (02/05/2023)

A partir del 2 de mayo de 2023, se implementará de forma general la versión 1.2.1.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:
>
> - En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha añadido un parámetro `enableInterContainerTrafficEncryption` a todas las API de Neptune ML, que puede usar para habilitar y deshabilitar el cifrado de tráfico entre contenedores en trabajos de ajuste de hiperparámetros o de entrenamiento.
- Se ha añadido compatibilidad con varias etiquetas para `mergeV()` y `mergeE()` de Gremlin.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher por el que las consultas de actualización y devolución no gestionaban correctamente `orderBy`, `limit` ni `skip`.

- Se ha corregido un error de openCypher que permitía anular los parámetros incluidos en una solicitud por los parámetros de otra solicitud simultánea.
- Se ha corregido un error de openCypher por el que los registros de consultas lentas no incluían los tiempos de consulta correctos.
- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones cuando se enviaba una consulta que incluía `GroupCountStep` en forma de cadena.
- Se ha corregido un error de Gremlin que provocaba que se produjera un error en las consultas de WebSocket cuando se habilitaban los registros de consultas lentas.
- Se ha corregido un error de Gremlin por el que faltaban los registros de depuración de los contadores de almacenamiento en los registros de consultas lentas para las solicitudes de WebSocket.
- Se ha corregido varios errores de Gremlin relacionados con `mergeV()` y `mergeE()`.
- Se ha corregido un error de SPARQL por el que se calculaban mal los costos de las consultas de gráficos con nombre asignado, lo que provocaba que los planes de consulta no fueran óptimos y se produjeran errores de memoria.
- Se ha corregido un error que afectaba a la autorización de las consultas de Gremlin y openCypher en un clúster habilitado para IAM.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.1.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.6.2`
- Compatible con la última versión de Gremlin: `3.6.2`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.1.0.R2 del motor

Actualización a esta versión

La versión 1.2.1.0.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.2 del motor de Amazon Neptune (20/11/2022)

A partir del 20 de noviembre de 2022, se implementará de forma general la versión 1.2.0.2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Versiones de parche posteriores para esta versión

- Versión: 1.2.0.2.R2 (15/12/2022)
- Versión: 1.2.0.2.R3 (27/03/2023)
- Versión: 1.2.0.2.R4 (08/05/2023)
- Versión: 1.2.0.2.R5 (16/08/2023)
- Versión: 1.2.0.2.R6 (12/09/2023)

Nuevas características de esta versión del motor

- Se ha introducido la inferencia inductiva en tiempo real para Gremlin en Neptune ML.
- Se ha introducido una extensión de openCypher que permite especificar valores de ID personalizados para las entidades en lugar de los UUID que Neptune genera. La posibilidad de asignar identificadores personalizados facilita la migración a Neptune desde Neo4j.

  > **Warning**
  >
  > Esta extensión de la especificación de openCypher no es compatible con versiones anteriores, ya que ahora `~id` se considera un nombre de propiedad reservada. Si ya utiliza `~id` como propiedad en sus datos y consultas, debe migrar la propiedad `~id` a una nueva clave de propiedad antes de actualizar a esta versión.

- Se han añadido varios modos nuevos de `DESCRIBE` de SPARQL junto con sugerencias de consulta para configurarlos.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de openCypher, especialmente para las consultas de VLP.
- Se ha mejorado el rendimiento del DFE para las consultas de Gremlin con límites distintos de los terminales, como, por ejemplo:

```
g.withSideEffect('Neptune#useDFE',true).V().hasLabel('Student').limit(5).out('takesCourse')
```

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.2 del motor

Actualización a esta versión

La versión 1.2.0.2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por

lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.2.R6 del motor de Amazon Neptune (12/09/2023)

A partir del 12 de septiembre de 2023, se implementará de forma general la versión 1.2.0.2.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de SPARQL que provocaba que el `REGEX` operador no funcionara correctamente cuando se llamaba desde un literal etiquetado en un lenguaje.
- Se ha corregido un problema que provocaba que empeorara el rendimiento de las cargas masivas.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.2.R6, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.5`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.2.R6 del motor

El clúster de base de datos de Neptune se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.2` del motor.

Actualización a esta versión

La versión 1.2.0.2.R6 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.2.R5 del motor de Amazon Neptune (16/08/2023)

A partir del 16 de agosto de 2023, se implementará de forma general la versión 1.2.0.2.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

En algunos casos, ese posible que los cambios introducidos en esta versión del motor provoquen una disminución en el rendimiento de las cargas masivas. En consecuencia, las actualizaciones a esta versión se han suspendido temporalmente hasta que se resuelva el problema.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin por el que `order()` no podía ordenar correctamente las salidas de cadena cuando algunas de ellas incluían un carácter de espacio.
- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones al comprobar el punto de conexión del estado de la consulta de Gremlin de las consultas con predicados en los recorridos secundarios para los pasos que no se procesaban de forma nativa.
- Se ha corregido un error de openCypher en la gestión de transacciones de Bolt.
- Se ha corregido un problema de simultaneidad en la capa de almacenamiento que podía provocar un bloqueo.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.2.R5, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.5`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`

- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.2.R5 del motor

El clúster de base de datos de Neptune se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.2` del motor.

Actualización a esta versión

La versión 1.2.0.2.R5 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.2.R4 del motor de Amazon Neptune (08/05/2023)

A partir del 8 de mayo de 2023, se implementará de forma general la versión 1.2.0.2.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño

del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de SPARQL que provocaba que se introdujera una gran cantidad de valores a través de una cláusula `VALUES` que podía provocar una disminución del rendimiento.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.2.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.6`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.2.R4 del motor

El clúster de base de datos de Neptune se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.2` del motor.

Actualización a esta versión

La versión 1.2.0.2.R4 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.2.R3 del motor de Amazon Neptune (27/03/2023)

A partir del 27 de marzo de 2023, se implementará de forma general la versión 1.2.0.2.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Para los clústeres de bases de datos sin servidor, se ha cambiado la configuración de capacidad mínima a 1,0 NCU y la configuración máxima válida más baja a 2,5 NCU. Consulte Escalado de capacidad en un clúster de base de datos de Neptune sin servidor
- Se ha añadido un parámetro `enableInterContainerTrafficEncryption` a todas las API de Neptune ML, que puede usar para habilitar y deshabilitar el cifrado de tráfico entre contenedores en trabajos de ajuste de hiperparámetros o de entrenamiento.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin por el que `option(Predicate)` impedía reconocer la sintaxis de Gremlin válida.
- Se ha corregido un error de Gremlin que provocaba que las consultas no se limpiaran correctamente si fallaban porque incluían demasiados pasos.
- Se ha corregido un error de corrección de Gremlin que afectaba a las consultas del DFE con `limit` como recorrido secundario de pasos no unidos a la unión que recurrían a Tinkerpop. Un ejemplo de una consulta de este tipo es:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Se ha corregido una posible fuga de transacciones de Gremlin cuando una consulta enviada como una cadena incluía `GroupCountStep`.
- Se ha corregido un error de openCypher que provocaba que el tipo de valor de parámetro no se infiriera correctamente en una lista o lista de mapas.
- Se ha corregido un error de openCypher por el que las consultas de actualización y devolución no gestionaban correctamente `orderBy`, `limit` ni `skip`.
- Se ha corregido un error de openCypher que permitía anular los parámetros incluidos en una solicitud por los parámetros de otra solicitud simultánea.
- Se ha corregido un error de SPARQL que provocaba que se introdujera una gran cantidad de valores en una cláusula `VALUES` que podía provocar una disminución del rendimiento.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.2.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.6`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.2.R3 del motor

El clúster de base de datos de Neptune se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.2` del motor.

Actualización a esta versión

La versión 1.2.0.2.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.2.R2 del motor de Amazon Neptune (15/12/2022)

A partir del 15 de diciembre de 2022, se implementará de forma general la versión 1.2.0.2.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al

intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `MERGE` y `OPTIONAL MATCH`.
- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `UNWIND` de una lista de mapas de valores literales.
- Se ha mejorado el rendimiento de las consultas de openCypher que tienen un filtro `IN` para `id`. Por ejemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Mejoras en el rendimiento y correcciones para varios operadores de Gremlin, incluidos `repeat`, `coalesce`, `store` y `aggregate`.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher que provocaba que las consultas devolvieran la cadena `"null"`, en lugar de un valor nulo en Bolt y SPARQL-JSON.

- Se ha corregido un error de Gremlin que provocaba que una etiqueta de paso adjunta `UnionStep` no se propagase al último elemento de la ruta de sus recorridos secundarios.
- Se ha corregido un error de Gremlin que provocaba que `valueMap()` no se optimizara al realizar un recorrido `by()` en el motor DFE.
- Se ha corregido un error de Gremlin que provocaba que las consultas de lectura ejecutadas como parte de una transacción de Gremlin más larga no bloquearan las filas.
- Se ha corregido un error en el registro de auditoría que provocaba que se registrara información innecesaria y que algunos campos no aparecieran en los registros.
- Se ha corregido un error en el registro de auditoría por el que no se registraba el ARN de IAM de las solicitudes HTTP a un clúster de base de datos habilitado para IAM.
- Se ha corregido un error en la caché de búsqueda para limitar la memoria incremental utilizada para las escrituras en la caché.
- Se ha corregido un error en la caché de búsqueda que implicaba configurar el modo de solo lectura para la caché de búsqueda cuando se producía un error en las escrituras.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.2.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.2.R2 del motor

El clúster de base de datos de Neptune se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.2` del motor.

Actualización a esta versión

La versión 1.2.0.2.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.1 del motor de Amazon Neptune (26/10/2022)

A partir del 26 de octubre de 2022, se implementará de forma general la versión 1.2.0.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Versiones de parche posteriores para esta versión

- Versión de mantenimiento: 1.2.0.1.R2 (13/12/2022)
- Versión de mantenimiento: 1.2.0.1.R3 (27-09-2023)

Nuevas características de esta versión del motor

- Se ha introducido Amazon Neptune sin servidor, una configuración de escalado automático bajo demanda que se ha diseñado para escalar el clúster de base de datos según sea necesario con el fin de hacer frente incluso a los grandes aumentos de la demanda de procesamiento y, luego, volver a reducir verticalmente cuando la demanda disminuya.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de las consultas `order-by` de Gremlin. Las consultas de Gremlin con `order-by` al final de a un `NeptuneGraphQueryStep` ahora utilizan un tamaño de fragmento mayor para obtener un mejor rendimiento. Esto no se aplica a `order-by` en un nodo interno (sin raíz) del plan de consultas.
- Se ha mejorado el rendimiento de las consultas de actualización de Gremlin. Los vértices y los bordes ahora deben bloquearse para que no se eliminen al añadir bordes o propiedades. Este cambio elimina los bloqueos duplicados en una transacción, lo que mejora el rendimiento.
- Se ha mejorado el rendimiento de las consultas de Gremlin que se utilizan `dedup()` dentro de una subconsulta `repeat()` al reducir `dedup` a la capa de ejecución nativa.
- Se han añadido mensajes de error sencillos para los errores de autenticación de IAM. Estos mensajes ahora muestran el ARN de su usuario o rol de IAM, el ARN del recurso y una lista de acciones no autorizadas para la solicitud. La lista de acciones no autorizadas le ayuda a ver lo que podría faltar o denegar explícitamente en la política de IAM que está utilizando.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin que provocaba que, al usar `PartitionStrategy` después de actualizar a TinkerPop 3.5, se producía un error con el mensaje “PartitionStrategy no funciona con recorridos anónimos”, lo que impedía que se ejecutara el recorrido.

- Se ha corregido un error de corrección de Gremlin relacionado con la traducción `WherePredicateStep`, que provocaba que el motor de consultas de Neptune produjera resultados incorrectos en las consultas que utilizaban `where(P.neq('x'))` y sus variantes.
- Se ha corregido un error de openCypher en la cláusula `MERGE` que, en algunos casos, provocaba la creación de nodos y bordes duplicados.
- Se ha corregido un error de SPARQL en la gestión de las consultas que incluyen `(NOT) EXISTS` en una cláusula `OPTIONAL`, por el que, en algunos casos, faltaban los resultados de las consultas.
- Se ha corregido un error de carga masiva que provocaba regresiones en el rendimiento cuando se producían elevadas cargas de inserción.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.1, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.1 del motor

Actualización a esta versión

La versión 1.2.0.1 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
```

```
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea

manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.1.R3 del motor de Amazon Neptune (27/09/2023)

A partir del 27 de septiembre de 2023, se implementará de forma general la versión 1.2.0.1.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:
>
> - En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba

la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha añadido un parámetro `enableInterContainerTrafficEncryption` a todas las API de Neptune ML, que puede usar para habilitar y deshabilitar el cifrado de tráfico entre contenedores en trabajos de ajuste de hiperparámetros o de entrenamiento.
- Para los clústeres de bases de datos sin servidor, se ha cambiado la configuración de capacidad mínima a 1,0 NCU y la configuración máxima válida más baja a 2,5 NCU. Consulte Escalado de capacidad en un clúster de base de datos de Neptune sin servidor *(((antes del lanzamiento, este cambio también debe reflejarse en la página sobre sistemas sin servidor)))*.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher por el que las consultas de actualización y devolución no gestionaban correctamente `orderBy`, `limit` ni `skip`.
- Se ha corregido un error de openCypher que permitía anular los parámetros incluidos en una solicitud por los parámetros de otra solicitud simultánea.
- Se ha corregido un error de openCypher en la gestión de transacciones de Bolt.
- Se ha corregido un error de corrección de Gremlin en las consultas del DFE con `limit` como recorrido secundario de pasos no unidos a la unión que recurrían a Tinkerpop. Por ejemplo, para consultas como las siguientes:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Se ha corregido un error de Gremlin que provocaba que una consulta fallara porque incluía demasiados pasos de TinkerPop y, por lo tanto, no se limpiaba.
- Se ha corregido un error de Gremlin por el que `order()` no podía ordenar correctamente las salidas de cadena cuando algunas de ellas incluían un carácter de espacio.
- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones cuando se enviaba una consulta como una cadena e incluía `GroupCountStep`.
- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones al comprobar el punto de conexión del estado de la consulta de Gremlin de las consultas con predicados en los recorridos secundarios para los pasos que no se procesaban de forma nativa.
- Se ha corregido un error de Gremlin que provocaba que se añadiera un borde y sus propiedades seguidas de `inV()`, o que `outV()` generara una `InternalFailureException`.
- Se ha corregido un problema de simultaneidad en la capa de almacenamiento.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.1.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`

- Compatible con la última versión de Gremlin: `3.5.6`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.1.R3 del motor

El clúster de base de datos de Neptune se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión 1.2.0.1 del motor.

Actualización a esta versión

La versión 1.2.0.1.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.1.R2 del motor de Amazon Neptune (13/12/2022)

A partir del 13 de diciembre de 2022, se implementará de forma general la versión 1.2.0.1.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede

reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `UNWIND` de una lista de mapas de valores literales.
- Mejoras en el rendimiento y correcciones para varios operadores de Gremlin, incluidos `repeat`, `coalesce`, `store` y `aggregate`.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher que provocaba que las consultas devolvieran la cadena `"null"`, en lugar de un valor nulo en Bolt y SPARQL-JSON.
- Se ha corregido un error en el registro de auditoría que provocaba que se registrara información innecesaria y que algunos campos no aparecieran en los registros.
- Se ha corregido un error en la caché de búsqueda para limitar la memoria incremental utilizada para las escrituras en la caché.
- Se ha corregido un error en la caché de búsqueda que implicaba configurar el modo de solo lectura para la caché de búsqueda cuando se producía un error en las escrituras.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.1.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.1.R2 del motor

El clúster de base de datos de Neptune se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión 1.2.0.1 del motor.

Actualización a esta versión

La versión 1.2.0.1.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.0 del motor de Amazon Neptune (21/07/2022)

A partir del 21 de julio de 2022, se implementará de forma general la versión 1.2.0.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede

reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.

- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Versiones de parche posteriores para esta versión

- Versión: 1.2.0.0.R2 (14/10/2022)
- Versión: 1.2.0.0.R3 (15/12/2022)
- Versión: 1.2.0.0.R4 (29/09/2023)

Nuevas características de esta versión del motor

- Se ha añadido compatibilidad con bases de datos globales. Una base de datos global de Neptune abarca varias Regiones de AWS y tiene un clúster de base de datos principal en una región y hasta cinco clústeres de base de datos secundarios en otras regiones.
- Se ha añadido soporte para un control de acceso más detallado en las políticas de IAM de Neptune que el que estaba disponible anteriormente, en función de las acciones del plano de datos. Se trata de un cambio radical, ya que las políticas de IAM existentes que se basan en la acción obsoleta `connect` deben ajustarse para utilizar las acciones más detalladas del plano de datos. Consulte Tipos de políticas de IAM.
- Se ha mejorado la disponibilidad de las instancias de lector. Anteriormente, cuando se reiniciaba una instancia de escritor, también se reiniciaban automáticamente todas las instancias de lector de un clúster de Neptune. A partir de la versión 1.2.0.0 del motor, las instancias de lector permanecen activas tras reiniciar el escritor, lo que mejora la disponibilidad del lector. Las instancias de lector se pueden reiniciar por separado para detectar los cambios en los grupos de parámetros. Consulte Reinicio de una instancia de base de datos en Amazon Neptune.

- Se ha añadido un nuevo parámetro de clúster de base de datos neptune_streams_expiry_days que le permite establecer el número de días que los registros de las transmisiones permanecen en el servidor antes de eliminarlos. El rango va de 1 a 90 y el valor predeterminado es 7.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de serialización de Gremlin para las consultas de ByteCode.
- Neptune ahora procesa los predicados de texto mediante el motor DFE para mejorar el rendimiento.
- Neptune ahora procesa los pasos `limit()` de Gremlin con el motor DFE, incluidos los límites de recorrido no terminales y secundarios.
- Se ha modificado la gestión del paso `union()` de Gremlin en el DFE para que funcione con otras nuevas características, lo que significa que los nodos de referencia se muestran en los perfiles de consulta según lo previsto.
- Gracias a la paralelización de algunas operaciones de unión costosas dentro del DFE, se ha mejorado el rendimiento hasta un factor de cinco.
- Se ha añadido soporte de modulación `by()` para `OrderGlobalStep order(global)` para el motor DFE de Gremlin.
- Se ha añadido la visualización de los valores estáticos inyectados en los detalles de explicación del DFE.
- Se ha mejorado el rendimiento al recortar patrones duplicados.
- Se ha añadido la compatibilidad con la conservación del orden en el motor DFE de Gremlin.
- Se ha mejorado el rendimiento de las consultas de Gremlin que tienen filtros vacíos, como los siguientes:

```
g.V().hasId(P.within([]))
```

```
g.V().hasId([])
```

- Se han mejorado los mensajes de error cuando una consulta de SPARQL utiliza un valor numérico demasiado grande para que Neptune lo represente internamente.
- Se ha mejorado el rendimiento al colocar los vértices con los bordes asociados al reducir las búsquedas en los índices cuando las transmisiones están deshabilitadas.

- Se ha ampliado la compatibilidad con el DFE a más variantes del paso `has()`, en particular a `hasKey()` y `hasLabel()`, y a predicados de rango para cadenas/URI dentro de `has()`. Esto afecta a consultas como las siguientes:

```
// hasKey() on properties
g.V().properties().hasKey("name")
g.V().properties().has(T.key, TextP.startingWith("a"))
g.E().properties().hasKey("weight")
g.E().properties().hasKey(TextP.containing("t"))

// hasLabel() on vertex properties
g.V().properties().hasLabel("name")

// range predicates on ID and Label fields
g.V().has(T.label, gt("person"))
g.E().has(T.id, lte("(an ID value)"))
```

- Se ha añadido una función `join()` de openCypher específica de Neptune que concatena las cadenas de una lista en una sola cadena.
- Se han actualizado las políticas administradas de Neptune para incluir permisos de acceso a los datos y permisos para las nuevas API de bases de datos globales.

Defectos corregidos en esta versión del motor

- Se ha corregido un error que provocaba que una solicitud HTTP sin un tipo de contenido especificado fallara automáticamente.
- Se ha corregido un error de SPARQL en el optimizador de consultas que impedía el uso de una llamada de servicio dentro de una consulta.
- Se ha corregido un error de SPARQL en el analizador RDF de Turtle que provocaba un error debido a una combinación concreta de datos Unicode.
- Se ha corregido un error de SPARQL por el que una determinada combinación de cláusulas `GRAPH` y `SELECT` generaba resultados de consulta incorrectos.
- Se ha corregido un error de Gremlin que provocaba un problema de corrección en las consultas que utilizaban cualquier paso de filtro dentro de un paso de unión, como el siguiente:

```
g.V("1").union(hasLabel("person"), out())
```

- Se ha corregido un error de Gremlin que provocaba que `count()` de `both().simplePath()` duplicara el número real de resultados devueltos sin `count()`.
- Se ha corregido un error de openCypher que provocaba que el servidor generara una excepción de discordancia de firmas defectuosa para las solicitudes de Bolt a clústeres con la autenticación de IAM habilitada.
- Se ha corregido un error de openCypher por el que una consulta que utilizaba keep-alive de HTTP podía cerrarse incorrectamente si se enviaba después de una solicitud errónea.
- Se ha corregido un error de openCypher que podía provocar un error interno cuando se enviaba una consulta que devolvía un valor constante.
- Se ha corregido un error en los detalles de explicación para que `Time(ms)` de la subconsulta del DFE ahora sume correctamente los tiempos de CPU de los operadores de la subconsulta del DFE. Tenga en cuenta el siguiente fragmento del resultado de explicación como ejemplo:

```
subQuery1
#######################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                           #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
  ...
#######################################################################################
# 1  # 2      # -      # DFEChunkLocalSubQuery # subQuery=...graph#336e.../graph_1 #
 -    # 1        # 1         # 1.00  # 0.38      #
#    #        #        #                       # coordinationTime(ms)=0.026        #
     #          #           #       #           #
#######################################################################################
  ...
subQuery=...graph#336e.../graph_1
#######################################################################################
# ID # Out #1 # Out #2 # Name                  # Arguments                           #
 Mode # Units In # Units Out # Ratio # Time (ms) #
#######################################################################################
# 0  # 1      # -      # DFESolutionInjection  # solutions=[?100 -> [-10^^<LONG>]] #
 -    # 0        # 1         # 0.00  # 0.04      #
#    #        #        #                       # outSchema=[?100]                  #
     #          #           #       #           #
#######################################################################################
# 1  # 3      # -      # DFERelationalJoin     # joinVars=[]                         #
 -    # 2        # 1         # 0.50  # 0.29      #
#######################################################################################
```

```
# 2  # 1      # -      # DFESolutionInjection  # outSchema=[]                    #
 -    # 0        # 1         # 0.00  # 0.01      #
#########################################################################################
# 3  # -      # -      # DFEDrain              # -                               #
 -    # 1        # 0         # 0.00  # 0.02      #
#########################################################################################
```

Los tiempos de subconsulta de la última columna de la tabla inferior suman 0,36 ms (`.04 + .29 + .01 + .02 = .36`). Al añadir el tiempo de coordinación de esa subconsulta (`.36 + .026 = .386`), se obtiene un resultado cercano al tiempo de la subconsulta registrado en la última columna de la tabla superior, es decir, `0.38` ms.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.0 del motor

Dado que se trata de una versión principal del motor, no hay ninguna actualización automática.

Solo se puede actualizar de forma manual a una versión `1.2.0.0`, a partir del último parche de la versión 1.1.1.0 del motor. Las versiones anteriores del motor deben actualizarse primero a la versión más reciente de `1.1.1.0` antes de poder actualizarse a `1.2.0.0`.

Por lo tanto, antes de intentar actualizar a esta versión, confirme que está utilizando la última versión de parche de la versión `1.1.1.0`. Si no es así, empiece por actualizar a la última versión de parche de `1.1.1.0`.

Antes de realizar la actualización, también debe volver a crear cualquier grupo de parámetros de clúster de base de datos personalizado que haya utilizado con la versión anterior, mediante el uso de la familia de grupos de parámetros `neptune1.2`. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

Si va a actualizar primero a la versión `1.1.1.0` y, después, inmediatamente a la versión `1.2.0.0`, puede que aparezca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior (consulte Mantenimiento del clúster de base de datos de Amazon Neptune).

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de versión principal, es necesario el parámetro allow-major-version-upgrade.

Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea

manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.0.R4 del motor de Amazon Neptune (29/09/2023)

A partir del 29 de septiembre de 2023, se implementará de forma general la versión 1.2.0.0.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Note**
>
> Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:
>
> - En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba

la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.

- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha añadido un parámetro `enableInterContainerTrafficEncryption` a todas las API de Neptune ML, que puede usar para habilitar y deshabilitar el cifrado de tráfico entre contenedores en trabajos de ajuste de hiperparámetros o de entrenamiento.
- Para los clústeres de bases de datos sin servidor, se ha cambiado la configuración de capacidad mínima a 1,0 NCU y la configuración máxima válida más baja a 2,5 NCU. Consulte Escalado de capacidad en un clúster de base de datos de Neptune sin servidor *(((antes del lanzamiento, este cambio también debe reflejarse en la página sobre sistemas sin servidor)))*.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher por el que las consultas de actualización y devolución no gestionaban correctamente `orderBy`, `limit` ni `skip`.
- Se ha corregido un error de openCypher que permitía anular los parámetros incluidos en una solicitud por los parámetros de otra solicitud simultánea.
- Se ha corregido un error de openCypher en la gestión de transacciones de Bolt.
- Se ha corregido un error de corrección de Gremlin en las consultas del DFE con `limit` como recorrido secundario de pasos no unidos a la unión que recurrían a Tinkerpop. Por ejemplo, para consultas como las siguientes:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Se ha corregido un error de Gremlin que provocaba que una consulta fallara porque incluía demasiados pasos de TinkerPop y, por lo tanto, no se limpiaba.
- Se ha corregido un error de Gremlin por el que `order()` no podía ordenar correctamente las salidas de cadena cuando algunas de ellas incluían un carácter de espacio.
- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones cuando se enviaba una consulta como una cadena e incluía `GroupCountStep`.
- Se ha corregido un error de Gremlin que provocaba que se produjera una fuga de transacciones al comprobar el punto de conexión del estado de la consulta de Gremlin de las consultas con predicados en los recorridos secundarios para los pasos que no se procesaban de forma nativa.
- Se ha corregido un error de Gremlin que provocaba que se añadiera un borde y sus propiedades seguidas de `inV()`, o que `outV()` generara una `InternalFailureException`.
- Se ha corregido un problema de simultaneidad en la capa de almacenamiento.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.0.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`

- Compatible con la última versión de Gremlin: `3.5.6`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.0.R4 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.0` del motor.

Solo se puede actualizar de forma manual a una versión `1.2.0.0`, a partir del último parche de la versión 1.1.1.0 del motor. Las versiones anteriores del motor deben actualizarse primero a la versión más reciente de `1.1.1.0` antes de poder actualizarse a `1.2.0.0`.

Si va a actualizar primero a la versión `1.1.1.0` y, después, inmediatamente a la versión `1.2.0.0`, puede que aparezca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
```

```
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de versión principal, es necesario el parámetro allow-major-version-upgrade. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.0.R3 del motor de Amazon Neptune (15/12/2022)

A partir del 15 de diciembre de 2022, se implementará de forma general la versión 1.2.0.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `MERGE` y `OPTIONAL MATCH`.
- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `UNWIND` de una lista de mapas de valores literales.
- Se ha mejorado el rendimiento de las consultas de openCypher que tienen un filtro `IN` para `id`. Por ejemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Mejoras en el rendimiento y correcciones para varios operadores de Gremlin, incluidos `repeat`, `coalesce`, `store` y `aggregate`.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher que provocaba que las consultas devolvieran la cadena `"null"`, en lugar de un valor nulo en Bolt y SPARQL-JSON.
- Se ha corregido un error de openCypher para poder interpretar correctamente el tipo de parámetro cuando el valor sea una lista o una lista de mapas.
- Se ha corregido un error en el registro de auditoría que provocaba que se registrara información innecesaria y que algunos campos no aparecieran en los registros.
- Se ha corregido un error en el registro de auditoría por el que no se registraba el ARN de IAM de las solicitudes HTTP a un clúster de base de datos habilitado para IAM.
- Se ha corregido un error en la caché de búsqueda para limitar la memoria incremental utilizada para las escrituras en la caché.
- Se ha corregido un error en la caché de búsqueda que implicaba configurar el modo de solo lectura para la caché de búsqueda cuando se producía un error en las escrituras.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.0.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`

- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.0.R3 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.0` del motor.

Solo se puede actualizar de forma manual a una versión `1.2.0.0`, a partir del último parche de la versión 1.1.1.0 del motor. Las versiones anteriores del motor deben actualizarse primero a la versión más reciente de `1.1.1.0` antes de poder actualizarse a `1.2.0.0`.

Si va a actualizar primero a la versión `1.1.1.0` y, después, inmediatamente a la versión `1.2.0.0`, puede que aparezca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
```

```
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de versión principal, es necesario el parámetro allow-major-version-upgrade. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.2.0.0.R2 del motor de Amazon Neptune (14/10/2022)

A partir del 14 de octubre de 2022, se implementará de forma general la versión 1.2.0.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Note

Si se realiza la actualización desde una versión del motor anterior a la 1.2.0.0:

- En la versión 1.2.0.0 del motor, se introdujo un nuevo formato para los grupos de parámetros personalizados y los grupos de parámetros de clústeres personalizados. En consecuencia, si va a actualizar una versión de motor anterior a la 1.2.0.0 a una versión de motor 1.2.0.0 o posterior, debe volver a crear todos los grupos de parámetros personalizados y los grupos de parámetros de clúster personalizados existentes utilizando la familia de grupos de parámetros `neptune1.2`. En las versiones anteriores, se utilizaba la familia de grupos de parámetros `neptune1`, y esos grupos de parámetros no funcionan con la versión 1.2.0.0 y las versiones posteriores. Para obtener más información, consulte Grupos de parámetros de Amazon Neptune.
- En la versión 1.2.0.0 del motor, también se introdujo un nuevo formato para los registros de deshacer. Como resultado, se deben purgar todos los registros de deshacer que se hayan creado con una versión anterior del motor y la métrica `UndoLogsListSize` de CloudWatch debe ser cero antes de que se pueda iniciar una actualización desde una versión anterior a la 1.2.0.0. Si hay demasiados registros de deshacer (200 000 o más), al intentar iniciar una actualización, es posible que se agote el tiempo de espera de la purga de los registros de deshacer.

  Para acelerar la velocidad de depuración, actualice la instancia del escritor del clúster, que es donde se realiza la depuración. Si lo hace antes de intentar la actualización, puede reducir el número de registros de deshacer antes de empezar. Si se aumenta el tamaño del escritor a un tipo de instancia 24XL, se puede aumentar la velocidad de purga a más de un millón de registros por hora.

  Si la métrica `UndoLogsListSize` de CloudWatch es muy grande, abra un caso de soporte para ayudarle a analizar estrategias adicionales para reducirla.
- Por último, se ha producido un cambio importante en la versión 1.2.0.0 que afecta al código anterior que utilizaba el protocolo Bolt con autenticación de IAM. A partir de la versión 1.2.0.0, Bolt necesita una ruta de recursos para la firma de IAM. En Java, la ruta del recurso podría tener este aspecto: `request.setResourcePath("/openCypher"));`. En otros lenguajes, `/openCypher` se puede agregar al URI del punto de conexión. Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de las consultas `order-by` de Gremlin. Las consultas de Gremlin con `order-by` al final de a un `NeptuneGraphQueryStep` ahora utilizan un tamaño de fragmento mayor para obtener un mejor rendimiento. Esto no se aplica a `order-by` en un nodo interno (sin raíz) del plan de consultas.
- Se ha mejorado el rendimiento de las consultas de actualización de Gremlin. Los vértices y los bordes ahora deben bloquearse para que no se eliminen al añadir bordes o propiedades. Este cambio elimina los bloqueos duplicados en una transacción, lo que mejora el rendimiento.
- Se ha mejorado el rendimiento de las consultas de Gremlin que se utilizan `dedup()` dentro de una subconsulta `repeat()` al reducir `dedup` a la capa de ejecución nativa.
- Se ha añadido la sugerencia de consulta de Gremlin `Neptune#cardinalityEstimates`. Si se establece en `false`, deshabilitas las estimaciones de cardinalidad.
- Se han añadido mensajes de error sencillos para los errores de autenticación de IAM. Estos mensajes ahora muestran el ARN de su usuario o rol de IAM, el ARN del recurso y una lista de acciones no autorizadas para la solicitud. La lista de acciones no autorizadas le ayuda a ver lo que podría faltar o denegar explícitamente en la política de IAM que está utilizando.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de corrección de Gremlin relacionado con la traducción `WherePredicateStep`, que provocaba que el motor de consultas de Neptune produjera resultados incorrectos en las consultas que utilizaban `where(P.neq('x'))` y sus variantes.
- Se ha corregido un error de Gremlin que provocaba que, al usar `PartitionStrategy` después de actualizar a TinkerPop 3.5, se producía un error con el mensaje “PartitionStrategy no funciona con recorridos anónimos”, lo que impedía que se ejecutara el recorrido.
- Se han corregido varios errores de Gremlin relacionados con la `joinTime` de una unión final y con las estadísticas de los subgrupos `Project.ASK`.
- Se ha corregido un error de openCypher en la cláusula `MERGE` que, en algunos casos, provocaba la creación de nodos y bordes duplicados.
- Se ha corregido un error en las transacciones que provocaba que una sesión pudiera insertar datos de gráficos y confirmarlos incluso cuando se anulaban las correspondientes inserciones simultáneas en el diccionario.
- Se ha corregido un error de carga masiva que provocaba regresiones en el rendimiento cuando se producían elevadas cargas de inserción.

- Se ha corregido un error de SPARQL en la gestión de las consultas que incluyen `(NOT) EXISTS` en una cláusula `OPTIONAL`, por el que, en algunos casos, faltaban los resultados de las consultas.
- Se ha corregido un error que provocaba que los controladores se quedaran bloqueados en los casos en que las solicitudes se cancelaban debido a un tiempo de espera previo al inicio de la evaluación. Era posible entrar en este estado si se consumían todos los subprocesos de procesamiento de consultas del servidor mientras se agotaban los tiempos de espera de los elementos de la cola de solicitudes. Dado que los tiempos de espera de la cola de solicitudes no enviaban inmediatamente mensajes, las respuestas aparecían al cliente como pendientes.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.2.0.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.2.0.0.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.2.0.0` del motor.

Solo se puede actualizar de forma manual a una versión `1.2.0.0`, a partir del último parche de la versión 1.1.1.0 del motor. Las versiones anteriores del motor deben actualizarse primero a la versión más reciente de `1.1.1.0` antes de poder actualizarse a `1.2.0.0`.

Si va a actualizar primero a la versión `1.1.1.0` y, después, inmediatamente a la versión `1.2.0.0`, puede que aparezca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de versión principal, es necesario el parámetro allow-major-version-upgrade. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
> ```

```
proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.1.1.0 del motor de Amazon Neptune (19/04/2022)

A partir del 19 de abril de 2022, se implementará de forma general la versión 1.1.1.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema

operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización. Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Versiones de parche posteriores para esta versión

- Versión de mantenimiento: 1.1.1.0.R2 (16/05/2022)
- Versión: 1.1.1.0.R3 (07/06/2022)
- Versión: 1.1.1.0.R4 (23/06/2022)
- Versión: 1.1.1.0.R5 (21/07/2022)
- Versión: 1.1.1.0.R6 (23/09/2022)
- Versión: 1.1.1.0.R7 (23/01/2023)

Nuevas características de esta versión del motor

- El lenguaje de consulta de openCypher ya se encuentra disponible con carácter general para su uso con fines de producción.

> **Warning**
>
> Hay un cambio importante en esta versión para el código que usa openCypher con autenticación de IAM. En la vista previa de Neptune para openCypher, la cadena de

host de la firma de IAM incluía el protocolo, como, por ejemplo `bolt://`, de la siguiente manera:

```
"Host":"bolt://(host URL):(port)"
```

A partir de esta versión del motor, se debe omitir el protocolo:

```
"Host":"(host URL):(port)"
```

Para ver ejemplos, consulte Uso del protocolo Bolt.

- Se ha añadido compatibilidad con TinkerPop `3.5.2`. Entre los cambios de esta versión se encuentran el soporte para transacciones remotas y el soporte de bytecode para las sesiones (con `g.tx`), y la adición de la función `datetime()` al lenguaje de Gremlin.

  Warning

  Se han introducido varios cambios importantes en TinkerPop 3.5.0, 3.5.1 y 3.5.2 que pueden afectar al código de Gremlin. Por ejemplo, usar recorridos generados por un GraphTraversalSource como secundarios de esta manera ya no funcionará: `g.V().union(identity(), g.V())`.
  Ahora en su lugar, utilice un recorrido anónimo como este: `g.V().union(identity(), __.V())`.

- Se ha añadido compatibilidad con claves de condición globales de AWS que puede usar en las políticas de acceso a datos de IAM que controlan el acceso a los datos almacenados en un clúster de base de datos de Neptune.
- El motor de consultas DFE de Neptune ya está disponible con carácter general para su uso en producción con el lenguaje de consultas de openCypher, pero todavía no para las consultas de Gremlin y SPARQL. Ahora lo habilita con su propio parámetro de instancia neptune_dfe_query_engine en lugar del parámetro lab-mode.

Mejoras en esta versión del motor

- Se han añadido nuevas características a openCypher, como la compatibilidad con consultas parametrizadas, el almacenamiento en caché del árbol de sintaxis abstracta (AST) para las consultas parametrizadas, las mejoras en las rutas de longitud variable (VLP) y nuevos operadores

y cláusulas. Consulte Conformidad con las especificaciones de OpenCypher en Amazon Neptune para obtener información sobre el nivel actual del soporte de lenguajes.

- Se han realizado mejoras importantes en el rendimiento de openCypher para cargas de trabajo sencillas de lectura y escritura, lo que se tradujo en un mayor rendimiento en comparación con la versión 1.1.0.0.
- Se han eliminado las limitaciones bidireccionales y de profundidad de openCypher al gestionar las rutas de longitud variable.
- Se ha completado el soporte del motor DFE para los predicados `within` y `without` de Gremlin, incluidos los casos en los que se combinan con otros operadores de predicados. Por ejemplo:

```
g.V().has('age', within(12, 15, 18).or(gt(30)))
```

- Se ha ampliado el soporte del motor DFE para el paso `order` de Gremlin cuando el ámbito es global (es decir, no `order(local)`) y cuando no se utilizan moduladores `by()`. Por ejemplo, esta consulta ahora sería compatible con el DFE:

```
g.V().values("age").order()
```

- Se ha añadido un campo `isLastOp` al formato de respuesta del registro de cambios de las transmisiones de Neptune para indicar que un registro es la última operación de su transacción.
- Se ha mejorado considerablemente el rendimiento del registro de auditoría y se ha reducido la latencia cuando se ha habilitado el registro de auditoría.
- Se ha convertido el código de bytes y las consultas HTTP de WebSocket de Gremlin a un formato legible por el usuario en los registros de auditoría. Las consultas ahora se pueden copiar directamente de los registros de auditoría para ejecutarlas en los cuadernos de Neptune y en otros lugares. Tenga en cuenta que este cambio en el formato actual del registro de auditoría constituye un cambio radical.

Defectos corregidos en esta versión del motor

- Se ha corregido un error raro de Gremlin que provocaba que no se devolvieran resultados cuando se utilizaban los pasos anidados `filter()` y `count()` combinados, como en la siguiente consulta:

```
g.V("1").filter(out("knows")
        .filter(in("knows")
        .hasId("notExists"))
```

```
        .count())
```

- Se ha corregido un error de Gremlin que provocaba que se devolviera un error al utilizar un vértice almacenado por un paso agregado en uno de los recorridos `to()` o `from()` recorridos junto con un paso `addE`. Un ejemplo de una consulta de este tipo es:

```
g.V("id").aggregate("v").out().addE().to(select("v").unfold()))
```

- Se ha corregido un error de Gremlin que provocaba que el paso `not` fallara en casos de borde al utilizar el motor DFE. Por ejemplo:

```
g.V().not(V())
```

- Se ha corregido un error de Gremlin que provocaba que los valores `sideEffect` no estuvieran disponibles dentro de los recorridos `to()` y `from()`.
- Se ha corregido un error que, en ocasiones, provocaba que un restablecimiento rápido desencadenara una conmutación por error de la instancia.
- Se ha corregido un error de carga masiva que provocaba que una transacción errónea no se cerrara antes de comenzar la siguiente tarea de carga.
- Se ha corregido un error de carga masiva que provocaba que el sistema se bloqueara debido a una falta de memoria.
- Se ha añadido un reintento para corregir un error de carga masiva que provocaba que el programa de carga no esperara lo suficiente para que las credenciales de IAM estuvieran disponibles tras una conmutación por error.
- Se ha corregido un error que provocaba que la caché interna de credenciales no se borrara correctamente en los puntos de conexión que no eran de consulta, como, por ejemplo, el punto de conexión `status`.
- Se ha corregido un error en las transmisiones para garantizar que los números de secuencia de confirmación de las transmisiones estén ordenados correctamente.
- Se ha corregido un error que provocaba que las conexiones de larga duración finalizaran antes de diez días en los clústeres habilitados para IAM.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.1.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.1.0 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión. Tenga en cuenta que las versiones anteriores a la versión principal del motor (1.1.0.0) tardarán más en actualizarse a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.

Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Mensajes de eventos por instancia:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine neptune \
    --engine-version 1.1.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine neptune ^
    --engine-version 1.1.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de versión principal, es necesario el parámetro allow-major-version-upgrade. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>  We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete una actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.1.1.0.R7 del motor de Amazon Neptune (23/01/2023)

A partir del 23 de enero de 2023, se implementará de forma general la versión 1.1.1.0.R7 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Important**
>
> La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.
>
> Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.
Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización. Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `MERGE` y `OPTIONAL MATCH`.
- Se ha mejorado el rendimiento de las consultas de openCypher relacionadas con `UNWIND` de una lista de mapas de valores literales.
- Se ha mejorado el rendimiento de las consultas de openCypher que tienen un filtro `IN` para `id`. Por ejemplo:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Mejoras en el rendimiento y correcciones para varios operadores de Gremlin, incluidos `repeat`, `coalesce`, `store` y `aggregate`.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher por el que una solicitud que utilizaba keep-alive de HTTP podía cerrarse incorrectamente si se enviaba después de una solicitud errónea.
- Se ha corregido un error de openCypher que provocaba que el tipo de parámetro no siempre se interpretara correctamente en una lista o lista de mapas.
- Se ha corregido un error de openCypher que provocaba que las consultas devolvieran la cadena `"null"`, en lugar de un valor nulo en Bolt y SPARQL-JSON.
- Se han corregido los códigos de error y mensajes de error de openCypher por errores de tiempo de espera de consulta y errores de memoria insuficiente.
- Se ha corregido un error de Gremlin que provocaba que `valueMap()` no se optimizara al realizar un recorrido `by()` en el motor DFE.
- Se ha corregido un problema con la lógica del detector de bloqueos que, en ocasiones, hacía que el motor dejara de responder.
- Se ha corregido un error en el registro de auditoría que provocaba que se registrara información innecesaria y que algunos campos no aparecieran en los registros.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.1.0.R7, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.3`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.1.0.R7 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.1.0` del motor.

Actualización a esta versión

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.

Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que

cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.1.1.0.R6 del motor de Amazon Neptune (23/09/2022)

A partir del 23 de septiembre de 2022, se implementará de forma general la versión 1.1.1.0.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.

Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Hay un cambio importante en esta versión para el código que usa openCypher con autenticación de IAM. Hasta ahora, la cadena de host de la firma de IAM incluía el protocolo, como, por ejemplo `bolt://`, de la siguiente manera:
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> A partir de la versión `1.1.1.0` del motor, se debe omitir el protocolo:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de las consultas `order-by` de Gremlin. Las consultas de Gremlin con `order-by` al final de a un `NeptuneGraphQueryStep` ahora utilizan un tamaño de fragmento mayor para obtener un mejor rendimiento. Esto no se aplica a `order-by` en un nodo interno (sin raíz) del plan de consultas.
- Se ha mejorado el rendimiento de las consultas de actualización de Gremlin. Los vértices y los bordes ahora deben bloquearse para que no se eliminen al añadir bordes o propiedades. Este cambio elimina los bloqueos duplicados en una transacción, lo que mejora el rendimiento.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de openCypher en la cláusula `MERGE` que, en algunos casos, provocaba la creación de nodos y bordes duplicados.
- Se ha corregido un error en la gestión de las consultas de SPARQL que incluyen `(NOT) EXISTS` en una cláusula `OPTIONAL`, por el que, en algunos casos, faltaban los resultados de las consultas.
- Se ha corregido un error que retrasaba el reinicio del servidor cuando se estaba realizando una carga masiva.
- Se ha corregido un error que provocaba un error al realizar un recorrido bidireccional con un patrón de longitud variable de openCypher con un filtro en la propiedad de relación. Un ejemplo de este patrón de longitud variable es `(n)-[r*1..2]->(m)`.

- Se ha corregido un error relacionado con la forma en que se devuelven los datos en caché al cliente, que, en algunos casos, provocaba una latencia inesperadamente larga.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.1.0.R6, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.1.0.R6 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.1.0` del motor.

Actualización a esta versión

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante

aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.
Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por

lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
> ```

```
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.1.1.0.R5 del motor de Amazon Neptune (21/07/2022)

A partir del 21 de julio de 2022, se implementará de forma general la versión 1.1.1.0.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos

todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización. Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Hay un cambio importante en esta versión para el código que usa openCypher con autenticación de IAM. Hasta ahora, la cadena de host de la firma de IAM incluía el protocolo, como, por ejemplo `bolt://`, de la siguiente manera:

```
"Host":"bolt://(host URL):(port)"
```

A partir de la versión `1.1.1.0` del motor, se debe omitir el protocolo:

```
"Host":"(host URL):(port)"
```

Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se han realizado mejoras para facilitar la detección de bloqueos.

Defectos corregidos en esta versión del motor

- Se ha corregido un error que impedía cerrar por completo los clústeres de bases de datos en determinadas condiciones.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.1.0.R5, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.1.0.R5 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.1.0` del motor.

Actualización a esta versión

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos

todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.
Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.1.1.0.R4 del motor de Amazon Neptune (23/06/2022)

A partir del 23 de junio de 2022, se implementará de forma general la versión 1.1.1.0.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización. Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Hay un cambio importante en esta versión para el código que usa openCypher con autenticación de IAM. Hasta ahora, la cadena de host de la firma de IAM incluía el protocolo, como, por ejemplo `bolt://`, de la siguiente manera:

```
"Host":"bolt://(host URL):(port)"
```

A partir de la versión `1.1.1.0` del motor, se debe omitir el protocolo:

```
"Host":"(host URL):(port)"
```

Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha actualizado la configuración de instancias para los tipos de instancias `x2g`.

- Se ha mejorado el rendimiento de las caídas de vértices.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin que provocaba que las soluciones no mantuvieran un orden estable para una consulta llamada varias veces o en varios lectores para determinados tipos de uniones de ASK.
- Además, se ha reducido el alcance de un cambio realizado en la versión anterior que provocaba regresiones en el rendimiento de algunos tipos de uniones de ASK en Gremlin.
- Se ha corregido un error de Gremlin en el paso `union()` que se producía cuando había una entrada de borde y un recorrido hacia un vértice dentro de los recorridos secundarios.
- Se ha corregido un error en el perfil de Gremlin que provocaba que algunos pasos no estuvieran optimizados cuando en realidad lo estaban.
- Se ha corregido un error de SPARQL que provocaba que a las variables utilizadas dentro de las expresiones `FILTER` anidadas en las cláusulas `UNION` se les asignara información de alcance no válida.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.1.0.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.1.0.R4 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.1.0` del motor.

Actualización a esta versión

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.

Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que

cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.1.1.0.R3 del motor de Amazon Neptune (07/06/2022)

A partir del 7 de junio de 2022, se implementará de forma general la versión del motor 1.1.1.0.R3. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.
Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.
Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.
Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Hay un cambio importante en esta versión para el código que usa openCypher con autenticación de IAM. Hasta ahora, la cadena de host de la firma de IAM incluía el protocolo, como, por ejemplo `bolt://`, de la siguiente manera:

```
"Host":"bolt://(host URL):(port)"
```

A partir de la versión `1.1.1.0` del motor, se debe omitir el protocolo:

```
"Host":"(host URL):(port)"
```

Para ver ejemplos, consulte Uso del protocolo Bolt.

Mejoras en esta versión del motor

- Se ha añadido compatibilidad con los tipos de instancias `x2g` con tecnología de Graviton2, optimizada para cargas de trabajo que consumen mucha memoria. En un principio, solo están disponibles en cuatro Regiones de AWS:
  - Este de EE. UU. (Norte de Virginia) (`us-east-1`)
  - Este de EE. UU. (Ohio) (`us-east-2`)
  - Oeste de EE. UU. (Oregón) (`us-west-2`)
  - Europa (Irlanda) (`eu-west-1`)

  Para obtener más información, consulte la página de precios de Neptune.
- Se ha mejorado el rendimiento de los pasos de Gremlin en los que se utilizan varios recorridos de bordes o vértices, búsquedas de propiedades o búsquedas de etiquetas.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin al procesar el paso `otherV()` dentro de un recorrido secundario.
- Se ha corregido un error de Gremlin en las consultas con `union` que provocaba que solo los pasos del filtro fueran secundarios. Por ejemplo:

```
g.V().union(has("name"), out("knows")).out()
```

- Se ha corregido un error de SPARQL que provocaba que a las variables utilizadas dentro de las expresiones `FILTER` anidadas en las cláusulas `UNION` se les asignara información de alcance no válida.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.1.0.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`
- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.1.0.R3 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.1.0` del motor.

Actualización a esta versión

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.

Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión de mantenimiento de Amazon Neptune, versión 1.1.1.0.R2 (16/05/2022)

A partir del 16 de mayo de 2022, se implementará de forma general la versión del motor 1.1.1.0.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema

operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización. Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Hay un cambio importante en esta versión para el código que usa openCypher con autenticación de IAM. Hasta ahora, la cadena de host de la firma de IAM incluía el protocolo, como, por ejemplo `bolt://`, de la siguiente manera:

```
"Host":"bolt://(host URL):(port)"
```

A partir de la versión `1.1.1.0` del motor, se debe omitir el protocolo:

```
"Host":"(host URL):(port)"
```

Para ver ejemplos, consulte Uso del protocolo Bolt.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.1.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Compatible con la primera versión de Gremlin: `3.5.2`

- Compatible con la última versión de Gremlin: `3.5.4`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.1.0.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.1.0` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.

Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`

    - Database cluster major version has been upgraded
  - Mensajes de eventos por instancia:
    - Applying off-line patches to DB instance
    - DB instance shutdown
    - Finished applying off-line patches to DB instance
    - DB instance restarted

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.1.0.0 del motor de Amazon Neptune (19/11/2021)

A partir del 19 de noviembre de 2021, se implementará de forma general la versión 1.1.0.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Important

La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.

Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.

Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.

Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.

Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Mensajes de eventos por instancia:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

> **Note**
>
> A partir de esta versión del motor, Neptune ya no admite los tipos de instancias R4. Si utiliza una instancia R4 en el clúster de base de datos, debe sustituirla manualmente por otro tipo de instancia antes de actualizar a esta versión. Si la instancia de escritor es R4, siga estas instrucciones para moverla.

Versiones de parche posteriores para esta versión

- Versión de mantenimiento: 1.1.0.0.R2 (16/05/2022)
- Versión de mantenimiento: 1.1.0.0.R3 (23/12/2022)

Nuevas características de esta versión del motor

- Se han introducido instancias de bases de datos T4g de uso general y R6g optimizadas para memoria con el procesador Graviton2 de AWS. Las instancias basadas en Graviton2 ofrecen una relación precio/rendimiento mucho mejor que las instancias comparables basadas en x86 de la generación actual para una variedad de cargas de trabajo. Las aplicaciones funcionan de forma normal en estos nuevos tipos de instancias y no es necesario transferir el código de la aplicación al actualizarlas.

  Para obtener más información acerca de los precios y la disponibilidad en las regiones, consulte la página de precios de Amazon Neptune.
- Se han introducido modelos personalizados en Neptune ML.

- Se ha añadido compatibilidad con consultas de inferencia de SPARQL en Neptune ML.
- Se ha añadido un nuevo punto de conexión de transmisiones para datos de gráficos de propiedades, en concreto:

```
https://Neptune-DNS:8182/propertygraph/stream
```

  El formato de salida de este punto de conexión, denominado `PG_JSON`, es exactamente el mismo que el formato de salida `GREMLIN_JSON` del anterior `gremlin/stream`.

  El nuevo punto de conexión `propertygraph/stream` amplía el soporte de transmisión de Neptune a openCypher y reemplaza el punto de conexión `gremlin/stream` con su formato de salida `GREMLIN_JSON` asociado.

Mejoras en esta versión del motor

- Se han realizado mejoras en las transmisiones de Neptune:
  - Se ha añadido un campo `commitTimestamp` al objeto `records` en el formato de respuesta del registro de cambios de transmisiones de Neptune, para proporcionar una marca de tiempo para cada registro de un flujo de registro de cambios.
  - Se ha añadido un valor `LATEST` al parámetro `iteratorType`, que permite recuperar el último eventId válido de las transmisiones. Consulte Llamada a la API de Streams.
- Se ha añadido soporte para obtener la puntuación de confianza de inferencia en las consultas de regresión y clasificación de nodos de Gremlin.
- Se ha añadido compatibilidad con la cláusula `OPTIONAL MATCH` de openCypher.
- Se ha añadido compatibilidad con la cláusula `MERGE` de openCypher.
- Se ha añadido compatibilidad para utilizar `ORDER BY` en las cláusulas `WITH` de openCypher.
- Se ha añadido soporte para la comprensión de patrones en openCypher y se ha ampliado el soporte para la expresión de patrones más allá de la comprobación de la existencia.
- Se ha ampliado la compatibilidad con las cláusulas `DELETE` y `DELETE DETACH` en openCypher, de modo que ahora se pueden utilizar con otras cláusulas de actualización.
- Se ha ampliado la compatibilidad con las clausulas `CREATE` y `UPDATE` utilizadas con `RETURN` en openCypher.
- Se ha añadido compatibilidad con el motor DFE para los pasos `limit`, `range` y `skip` de Gremlin.

- Se ha mejorado la ejecución de consultas en el motor DFE cuando no se solicita `explain` ni `profile`.
- Se ha mejorado la ejecución de consultas en el motor DFE para la expresión `value`.
- Se han mejorado varios patrones de inserción condicional encadenados de Gremlin para evitar excepciones de modificación simultánea y permitir el encadenamiento de patrones de consulta como los siguientes:
  - Inserción condicional de vértices por ID, como, por ejemplo:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1").property(id,ID))
```

  - Inserción condicional de vértices con varias etiquetas, como, por ejemplo:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1::L2").property(id,ID))
```

  - Inserción condicional de bordes por identificador, como, por ejemplo:

```
g.E(ID).fold().coalesce(unfold(), V(from).addE(label).to(V(to)).property(id, ID))
```

  - Inserción condicional de bordes con varias etiquetas, como, por ejemplo:

```
g.E(ID).fold().coalesce(unfold(),
 g.addE(label).from(V(from)).to(V(to)).property(id, ID))
```

  - Inserción condicional seguida de una consulta, como, por ejemplo:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID)).project("myvalues").by(valueMap())
```

  - Inserción condicional con propiedades añadidas, como, por ejemplo:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID).property("name","pumba"))
```

Defectos corregidos en esta versión del motor

- Se ha deshabilitado la característica estadísticas en los tipos de instancias `T3.medium` que no podían admitirla.

- Se ha corregido un error de SPARQL en `explain` con una función `IN` que tomaba valores no constantes.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.0.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.0.0 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --allow-major-version-upgrade ^
```

```
    --apply-immediately
```

En lugar de `--apply-immediately`, puede especificar `--no-apply-immediately`. Para realizar una actualización de versión principal, es necesario el parámetro allow-major-version-upgrade. Además, asegúrese de incluir la versión del motor, ya que es posible que el motor se actualice a otra versión.

Si el clúster utiliza un grupo de parámetros del clúster personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Del mismo modo, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado, asegúrese de incluir este parámetro para especificarlo:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión de mantenimiento de Amazon Neptune, versión 1.1.0.0.R3 (23/12/2022)

A partir del 23 de diciembre de 2022, se implementará de forma general la versión 1.1.0.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Important**
>
> La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante

la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.
Para completar la actualización correctamente, cada subred de cada zona de disponibilidad (AZ) debe tener al menos una dirección IP disponible por instancia de Neptune. Por ejemplo, si hay una instancia de escritor y dos instancias de lector en la subred 1 y dos instancias de lector en la subred 2, la subred 1 debe tener al menos 3 direcciones IP libres y la subred 2 debe tener al menos 2 direcciones IP libres antes de iniciar la actualización.
Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.
Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.
Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Mejoras en esta versión del motor

- Mejoras en el rendimiento y correcciones para varios operadores de Gremlin, incluidos `repeat`, `coalesce`, `store` y `aggregate`.

Defectos corregidos en esta versión del motor

- Se ha corregido un problema de picos en el uso de la CPU.
- Se ha corregido un error de openCypher que provocaba que las consultas devolvieran la cadena `"null"`, en lugar de un valor nulo en Bolt y SPARQL-JSON.
- Se ha corregido un error en el registro de auditoría que provocaba que se registrara información innecesaria y que algunos campos no aparecieran en los registros.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.0.0.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.0.0.R3 del motor

El clúster se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.0.0` del motor.

> **Important**
>
> La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.
>
> Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.
>
> Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos

todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.
Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> A partir de esta versión del motor, Neptune ya no admite los tipos de instancias R4. Si utiliza una instancia R4 en el clúster de base de datos, debe sustituirla manualmente por otro tipo de instancia antes de actualizar a esta versión. Si la instancia de escritor es R4, siga estas instrucciones para moverla.

Actualización a esta versión

La versión 1.1.0.0.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión de mantenimiento de Amazon Neptune, versión 1.1.0.0.R2 (16/05/2022)

A partir del 16 de mayo de 2022, se implementará de forma general la versión del motor 1.1.0.0.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

> **Important**
>
> La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante

la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.
Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.
Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante varios minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.
Este proceso genera los siguientes eventos:

- Mensajes de eventos por clúster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Mensajes de eventos por instancia:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Defectos corregidos en esta versión del motor

- Se ha corregido un error que provocaba que la caché interna de credenciales no se borrara correctamente en los puntos de conexión que no eran de consulta, como, por ejemplo, el punto de conexión status.
- Se ha corregido un error que provocaba que el retraso en la replicación aumentara tras una actualización del motor.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.1.0.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de openCypher: `Neptune-9.0.20190305-1.0`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.1.0.0.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.1.0.0` del motor.

> **Important**
>
> La actualización a esta versión del motor desde una versión anterior a **`1.1.0.0`** también desencadena una actualización del sistema operativo en todas las instancias del clúster de base de datos. Dado que las solicitudes de escritura activas que se produzcan durante la actualización del sistema operativo no se procesarán, debe pausar todas las cargas de trabajo de escritura en el clúster que se va a actualizar, incluidas las cargas masivas de datos, antes de iniciar la actualización.
>
> Al inicio de la actualización, Neptune genera una instantánea con un nombre que se compone de `preupgrade` seguido de un identificador generado automáticamente en función de la información del clúster de base de datos. No se le cobrará por esta instantánea y podrá utilizarla para restaurar el clúster de base de datos si se produce algún problema durante el proceso de actualización.
>
> Cuando se complete la propia actualización del motor, la nueva versión del motor estará disponible brevemente en el sistema operativo anterior, pero en menos de cinco minutos todas las instancias del clúster comenzarán simultáneamente a actualizar el sistema operativo. El clúster de base de datos no estará disponible en este momento durante aproximadamente seis minutos. Puede reanudar cargas de trabajo de escritura una vez finalizada la actualización.
>
> Este proceso genera los siguientes eventos:
>
> - Mensajes de eventos por clúster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Mensajes de eventos por instancia:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

> **Note**
>
> A partir de esta versión del motor, Neptune ya no admite los tipos de instancias R4. Si utiliza una instancia R4 en el clúster de base de datos, debe sustituirla manualmente por otro tipo de instancia antes de actualizar a esta versión. Si la instancia de escritor es R4, siga estas instrucciones para moverla.

Actualización a esta versión

La versión 1.1.0.0.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.5.1 del motor de Amazon Neptune (01/10/2021)

A partir del 1 de octubre de 2021, se implementará de forma general la versión 1.0.5.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.5.1.R2 (26/10/2021)
- Versión: 1.0.5.1.R3 (13/01/2022)
- Versión de mantenimiento: 1.0.5.1.R4 (16/05/2022)

Nuevas características de esta versión del motor

- Se ha añadido una caché de resultados para almacenar en caché los resultados de las consultas especificadas.
- Se ha añadido compatibilidad con fecha y hora en openCypher de Neptune.

- Se ha añadido compatibilidad para que `List` y `Map` tengan acceso los elementos de openCypher de Neptune.

Mejoras en esta versión del motor

- Se ha hecho que los nombres que de punto de conexión de openCypher de Neptune no distingan entre mayúsculas y minúsculas.
- Se ha mejorado la explicación de openCypher.
- Se han mejorado los patrones de consulta de actualización o inserción únicos de Gremlin que terminan con los pasos `iterate()` y `profile()`.
- Se ha mejorado el rendimiento en las funciones `keys()` y `property()` de Gremlin.
- El paso `dedup()` de Gremlin se ejecuta en el DFE cuando se utiliza con un alcance global.
- Los siguientes predicados HAS de Gremlin se ejecutan en el motor DFE cuando este está habilitado:
  - `EQ`
  - `NEQ`
  - `LT`
  - `LTE`
  - `GT`
  - `GTE`
  - `BETWEEN`
  - `INSIDE`
  - `OUTSIDE`
  - `WITHIN`
  - `AND (connectives)`
  - `OR (connectives)`
- Se ha mejorado el rendimiento de las consultas LIMIT.
- Se ha mejorado el rendimiento de las consultas de agregación general de openCypher.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin que permitía conectar un borde a otro borde.

- Se ha corregido un error de Gremlin que provocaba que se eligiera una estrategia de unión poco óptima.
- Se ha corregido un error de Gremlin que provocaba que la serialización de los nodos y las relaciones se interrumpiera cuando había más de 100 propiedades.
- Se ha corregido un error que ralentizaba la planificación de ejecución de consultas para consultas con patrones de gráficos de gran tamaño.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.1, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.1 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

La versión 1.0.5.1 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión de mantenimiento de Amazon Neptune, versión 1.0.5.1.R4 (16/05/2022)

A partir del 16 de mayo de 2022, se implementará de forma general la versión del motor 1.0.5.1.R4. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.1.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.1.R4 del motor

El clúster se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.5.1` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.5.1.R4 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.5.1.R3 del motor de Amazon Neptune (13/01/2022)

A partir del 13 de enero de 2022, se implementará de forma general la versión 1.0.5.1.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error que podía provocar una pérdida de recursos cuando una consulta no lograba adquirir todos los recursos que necesitaba.
- Se ha corregido una pequeña pérdida de memoria durante la ejecución de una consulta provocada por una asignación de memoria no reclamada.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.1.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.1.R3 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.5.1` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.5.1.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.5.1.R2 del motor de Amazon Neptune (26/10/2021)

A partir del 26 de octubre de 2021, se implementará de forma general la versión del motor 1.0.5.1.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error que provocaba el reinicio del servidor cuando se producía un error transitorio al crear una versión anterior de un elemento de gráfico, con aislamiento de lectura repetible. Neptune ahora devuelve un error en su lugar, para que el cliente pueda volver a intentarlo.

- Se ha corregido un error que provocaba el reinicio del servidor cuando se producía un error transitorio durante una actualización de cardinalidad única. Neptune ahora devuelve un error en su lugar, para que el cliente pueda volver a intentarlo.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.1.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.1.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.5.1` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.5.1.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.5.0 del motor de Amazon Neptune (27/07/2021)

A partir del 27 de julio de 2021, se implementará de forma general la versión 1.0.5.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.5.0.R2 (16/08/2021)
- Versión: 1.0.5.0.R3 (15/09/2021)
- Versión de mantenimiento: 1.0.5.0.R5 (16/05/2022)

Nuevas características de esta versión del motor

- Neptune ML se lanzó para su uso en producción con numerosas características nuevas y ya no está en modo lab.
- Se ha añadido compatibilidad inicial con el lenguaje de consultas de openCypher, en el modo lab. openCypher es el estándar de código abierto para el lenguaje de consultas de Cypher. Su sintaxis se especifica en la referencia del lenguaje de consulta de Cypher (versión 9) y el proyecto openCypher la mantiene.

Consulte Acceso al gráfico de Neptune con openCypher para obtener información sobre la implementación del lenguaje en Neptune.

También se admite la compatibilidad con el protocolo Bolt, que los clientes de Neptune utilizan para las consultas de openCypher. Consulte Uso del protocolo Bolt para realizar consultas de openCypher a Neptune.

La compatibilidad con openCypher ahora está habilitada automáticamente, pero depende de Motor DFE de Neptune, que actualmente solo está disponible en el modo lab. La configuración predeterminada de `DFEQueryEngine` en el parámetro del clúster de base de datos de `neptune_lab_mode` es ahora `DFEQueryEngine=viaQueryHint`, lo que significa que el motor está habilitado, pero solo se utiliza para las consultas que tienen la sugerencia de consulta `useDFE` presente y establecida en `true`. Si deshabilita el motor DFE mediante la configuración `DFEQueryEngine=disabled`, no podrá utilizar openCypher.

- Se ha añadido compatibilidad con el protocolo HTTP Graph Store SPARQL 1.1. Consulte Uso del protocolo HTTP de almacén de gráficos (GSP) de SPARQL 1.1 en Amazon Neptune.
- Se ha cambiado la configuración predeterminada de modo lab para Motor DFE de Neptune a `viaQueryHint`, lo que significa que el motor DFE ahora está habilitado de forma predeterminada, pero solo se utiliza para las consultas que tienen la sugerencia de consulta `useDFE` presente y establecida en `true`.
- Se ha añadido una nueva métrica de Amazon CloudWatch, `StatsNumStatementsScanned`, para supervisar el cálculo de las estadísticas del motor DFE de Neptune. Consulte Uso de la `StatsNumStatementsScanned` CloudWatch métrica para supervisar el cálculo de las estadísticas.

Mejoras en esta versión del motor

- Se ha añadido compatibilidad con Apache TinkerPop 3.4.11.

> **Important**
>
> Se ha realizado un cambio en la versión 3.4.11 de TinkerPop que mejora la forma en que se procesan las consultas, pero, por el momento, a veces puede afectar gravemente al rendimiento de las consultas.
> Por ejemplo, una consulta de este tipo podría ejecutarse de una forma mucho más lenta:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

Los vértices después del paso límite se obtienen ahora de una manera que no es la óptima debido al cambio de TinkerPop 3.4.11. Para evitarlo, puede modificar la consulta añadiendo el paso barrier() en cualquier punto después de `order().by()`. Por ejemplo:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

- La sugerencia de consulta `joinOrder` de SPARQL ahora es compatible con el motor de consultas alternativo DFE de Neptune.
- El resultado de la API de estado de Neptune se ha ampliado y reorganizado para ofrecer más claridad sobre la configuración y las características del clúster de base de datos.

  El nuevo resultado tiene un objeto `features` de nivel superior que incluye información de estado sobre las características del clúster de base de datos y un objeto `settings` de nivel superior que incluye información de configuración. Para revisar el nuevo formato, consulte Ejemplo del resultado del comando de estado de instancia.
- Se ha mejorado la gestión de los registros de cambios de la transmisión cuando se solicitan transmisiones `AFTER_SEQUENCE_NUMBER` con el último ID de evento del servidor, cuando ese ID de evento ya ha caducado. El servidor ya no lanza un error de ID de evento caducado si el ID de evento solicitado es el ID de evento purgado más recientemente en el servidor.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin relacionado con el orden de los valores numéricos.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.0 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

La versión 1.0.5.0 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por

lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión de mantenimiento de Amazon Neptune, versión 1.0.5.0.R5 (16/05/2022)

A partir del 16 de mayo de 2022, se implementará de forma general la versión del motor 1.0.5.0.R5. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.0.R5, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.0.R5 del motor

El clúster se actualizará automáticamente a esta versión de parche de mantenimiento durante el siguiente periodo de mantenimiento si está ejecutando la versión 1.0.5.0 del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.5.0.R5 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.5.0.R3 del motor de Amazon Neptune (15/09/2021)

A partir del 15 de septiembre de 2021, se implementará de forma general la versión 1.0.5.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error que provocaba que el motor dejara de responder en una de estas situaciones:
  - Se produce una carga masiva al mismo tiempo que se realiza el cálculo automático de las estadísticas.
  - Se solicitó un cálculo estadístico manualmente a la vez que uno que ya se estaba realizando.
- Se ha corregido un error en la detección de bloqueos y en la adquisición de bloqueos que podía provocar un error del motor.
- Se ha corregido un error de Gremlin que provocaba que el motor generara un error al encontrar datos desconocidos de un punto de conexión remoto de ML en una consulta de inferencia de Gremlin.
- Se han corregido varios errores en las API de administración de modelos de ML relacionados con trabajos de transformación de modelos y recomendaciones de instancias.
- Se ha corregido un error que podía provocar un bloqueo del motor al generar los identificadores de nodo y borde.
- Se ha corregido un error que ralentizaba la generación de planes de consulta para consultas con patrones de gráficos de gran tamaño.
- Se ha corregido un error de openCypher que podía provocar que una consulta se bloqueara al recuperar un nodo que tenía más de 100 propiedades.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.0.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.0.R3 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión 1.0.5.0 del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.5.0.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A

continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.5.0.R2 del motor de Amazon Neptune (16/08/2021)

A partir del 16 de agosto de 2021, se implementará de forma general la versión del motor 1.0.5.0.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha deshabilitado una optimización realizada en la versión del motor `1.0.5.0` que hacía que la caché de búsqueda de Neptune se conservara durante los reinicios del motor en las réplicas. Los reinicios de la réplica ahora borran la caché de búsqueda.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.5.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.11`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.5.0.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.5.0` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.5.0.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado,

así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.4.2 del motor de Amazon Neptune (01/06/2021)

> **Note**
>
> La versión 1.0.4.2.R2 del motor fue la primera versión de la 1.0.4.2 que realmente se lanzó.

Temas

Versión 1.0.4.2.R5 del motor de Amazon Neptune (16/08/2021)

A partir del 16 de agosto de 2021, se implementará de forma general la versión 1.0.4.2.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha deshabilitado una optimización realizada en la versión del motor `1.0.4.2.R4` que hacía que la caché de búsqueda de Neptune se conservara durante los reinicios del motor en las réplicas. Los reinicios de la réplica ahora borran la caché de búsqueda.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.2.R5, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.10`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.2.R5 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.4.2` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Versión 1.0.4.2.R4 del motor de Amazon Neptune (23/07/2021)

A partir del 23 de julio de 2021, se implementará de forma general la versión 1.0.4.2.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Mejoras en esta versión del motor

- Se ha mejorado el comportamiento de la caché de búsqueda para evitar el borrado redundante de la memoria caché después de ejecutar un restablecimiento rápido en una réplica.
- Se ha mejorado la gestión de los registros de cambios de la transmisión cuando se solicitan transmisiones `AFTER_SEQUENCE_NUMBER` con el último ID de evento del servidor, cuando ese ID

de evento ya ha caducado. El servidor ya no lanza un error de ID de evento caducado si el ID de evento solicitado es el ID de evento purgado más recientemente en el servidor.

Defectos corregidos en esta versión del motor

- Se ha corregido un error introducido en la versión 1.0.4.0.R1 por el que las consultas no devolvían la totalidad de los valores de cadena de más de 760 caracteres. Los términos afectados por este error eran los URI y literales de RDF, o los ID, claves y valores de cadena de Gremlin.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.2.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.10`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.2.R4 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.4.2` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Versión 1.0.4.2.R3 del motor de Amazon Neptune (28/06/2021)

A partir del 28 de junio de 2021, se implementará de forma general la versión 1.0.4.2.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Problemas conocidos de esta versión del motor

Problema:

Un error de SPARQL que no respeta el tipo de medio de un encabezado `Accept` si hay espacios.

Por ejemplo, una consulta con `-H "Accept: text/csv; q=1.0, */*; q=0.1"` devuelve una salida JSON en lugar de una salida CSV.

Solución:

Si elimina los espacios de la cláusula `Accept` del encabezado, el motor devuelve la salida en el formato solicitado correcto. Dicho de otro modo, en lugar de `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , utilice:

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Defectos corregidos en esta versión del motor

- Se ha corregido un error al borrar la caché de búsqueda de las réplicas tras un restablecimiento rápido.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.2.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.10`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.2.R3 del motor

Esta versión de parche es opcional, a menos que el clúster de base de datos utilice una o varias instancias `R5d`. Si el clúster tiene instancias `R5d`, se actualizará de forma automática en el siguiente periodo de mantenimiento. De lo contrario, no se actualizará de forma automática a esta versión de parche.

Puede actualizar la versión `1.0.4.2.R2` a esta versión `1.0.4.2.R3` manualmente mediante el comando de la AWS CLI apply-pending-maintenance-action (la API ApplyPendingMaintenanceAction).

Versión 1.0.4.2.R2 del motor de Amazon Neptune (01/06/2021)

A partir del 1 de junio de 2021, se implementará de forma general la versión 1.0.4.2.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.4.2.R3 (28/06/2021)

Problemas conocidos de esta versión del motor

Problema:

Un error de SPARQL que no respeta el tipo de medio de un encabezado `Accept` si hay espacios.

Por ejemplo, una consulta con `-H "Accept: text/csv; q=1.0, */*; q=0.1"` devuelve una salida JSON en lugar de una salida CSV.

Solución:

Si elimina los espacios de la cláusula `Accept` del encabezado, el motor devuelve la salida en el formato solicitado correcto. Dicho de otro modo, en lugar de `-H "Accept: text/csv; q=1.0, */*; q=0.1"` , utilice:

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Nuevas características de esta versión del motor

- Se ha añadido el nuevo tipo de instancia R5d, que incluye una caché de búsqueda para acelerar las lecturas en casos de uso que impliquen un gran volumen de búsquedas de valores de propiedades o literales RDF. Consulte La caché de búsqueda de Neptune puede acelerar las consultas de lectura.
- Se ha añadido un nuevo parámetro en modo lab que permite invocar el motor DFE experimental solo por consulta con la sugerencia de consulta `useDFE`.

Mejoras en esta versión del motor

- Se ha añadido compatibilidad con TinkerPop 3.4.10.
- Se ha añadido compatibilidad para utilizar el paso de configuración `withStrategies( )` al enviar solicitudes de scripts de Gremlin. En concreto, se admiten `SubgraphStrategy`, `PartitionStrategy`, `ReadOnlyStrategy`, `EdgeLabelVerificationStrategy` y `ReservedKeysVerificationStrategy`.

- Se ha añadido optimización para recorridos de `V()` en mitad de una consulta. Anteriormente, estos recorridos no estaban optimizados en Neptune.
- Se ha añadido compatibilidad con los URN de RFC 2141 para utilizarlos como parámetros `baseUri` y `namedGraphUri` en una carga masiva.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin en el analizador que hacía que las consultas incorrectas se consideraran válidas.
- Se ha corregido un error de Gremlin que provocaba que se produjera una excepción al desplegar un efecto secundario de `aggregate()` con `cap().unfold()` en un `valueMap()`.
- Se ha corregido un error de Gremlin que provocaba que fallaran algunos pasos `property()` después de un paso `addV()` y apareciera el error “no se puede convertir en cadena”.
- Se ha corregido un error de Gremlin que impedía que algunos patrones de inserción condicionales generaran excepciones de modificación simultánea.
- Se ha corregido un error de Gremlin por el que el tiempo de espera de la solicitud de consulta ahora no puede superar el tiempo de espera de la sesión.
- Se ha corregido un error de SPARQL por el que las actualizaciones que utilizan LOAD o UNLOAD podían fallar con un código HTTP 500 en lugar del código HTTP 400 cuando el servidor remoto no estaba disponible.
- Se ha corregido un error que provocaba que se produjera un error en las llamadas a la API de transmisión cuando se utilizaban valores `commitNum` o `opNum` superiores al límite de enteros firmados de 32 bits (2147 483 647).

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.2.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.10`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.2.R2 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

La versión 1.0.4.2.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.4.2.R1 del motor de Amazon Neptune (27/05/2021)

La versión del motor 1.0.4.2.R1 nunca se implementó.

Versión 1.0.4.1 del motor de Amazon Neptune (08/12/2020)

A partir del 8 de diciembre de 2020, se implementará de forma general la versión 1.0.4.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.4.1.R1.1 (22/03/2021)
- Versión: 1.0.4.1.R2 (24/02/2021)

> **Important**
>
> En Versión: 1.0.4.0 (12/10/2020), se hizo que TLS 1.2 y HTTPS fueran obligatorios para todas las conexiones a Amazon Neptune. Sin embargo, un error de esta versión ha permitido que las conexiones HTTP o las conexiones TLS obsoletas sigan funcionando para los clientes que anteriormente configuraban un parámetro de clúster de base de datos para impedir el establecimiento de las conexiones HTTPS.
> Este error se ha corregido en las versiones de parche 1.0.4.0.R2 y 1.0.4.1.R2, pero la corrección provocaba errores de conexión inesperados cuando los parches se instalaban automáticamente. Por este motivo, ambos parches se han revertido y solo se pueden instalar manualmente, para que pueda actualizar la configuración de TLS 1.2.
> Tener que usar SSL/TLS para todas las conexiones a Neptune afecta a las conexiones con la consola de Gremlin, el controlador de Gremlin, Gremlin Python, .NET, NodeJS, las API de REST y también a las conexiones del equilibrador de carga. Si ha utilizado HTTP o una versión anterior de TLS para alguna o todas estas conexiones hasta ahora, debe actualizar el cliente y los controladores correspondientes y cambiar el código para que utilice exclusivamente HTTPS antes de actualizar el sistema a los últimos parches.

Nuevas características de esta versión del motor

- Se ha presentado la característica Neptune ML, que incorpora potentes capacidades de machine learning a Amazon Neptune. Consulte Amazon Neptune ML para el machine learning en gráficos.
- Se ha añadido una operación `UNLOAD` de SPARQL personalizada para eliminar los datos recuperados de un origen remoto. Consulte SPARQL UPDATE UNLOAD.

Mejoras en esta versión del motor

- Se han optimizado algunos patrones de inserción condicional de Gremlin para evitar excepciones de modificación simultánea.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin que podía provocar que faltaran resultados en un patrón específico de consultas que utilizaban el paso `as()`.
- Se ha corregido un error de Gremlin que podía provocar errores al utilizar el paso `project()` anidado dentro de otro paso, como, por ejemplo `union()`.
- Se ha corregido un error de Gremlin en el paso `project()`.
- Se ha corregido un error de Gremlin en el recorrido basado en cadenas que provocaba que el paso `none()` no funcionara.
- Se ha corregido un error de Gremlin que provocaba que el recorrido basado en cadenas no fuera compatible con un mapa vacío como argumento del paso `inject()`.
- Se ha corregido un error de Gremlin en la ejecución de recorridos basados cadenas en el motor DFE, que provocaba que un método de terminal como `toList()` no funcionara correctamente.
- Se ha corregido un error de Gremlin que impedía cerrar transacciones que utilizaban el paso `iterate()` en una consulta de cadenas.
- Se ha corregido un error de Gremlin que podía provocar que las consultas que utilizaban el patrón `is(P.gte(0))` generaran una excepción en algunas situaciones.
- Se ha corregido un error de Gremlin que podía provocar que las consultas que utilizaban el patrón `order().by(T.id)` generaran una excepción en algunas situaciones.
- Se ha corregido un error de Gremlin que podía provocar que las consultas que utilizaban el patrón `addV().aggregate()` generara resultados incorrectos en algunas situaciones.

- Se ha corregido un error de Gremlin que podía provocar que las consultas que utilizaban el paso `path()` seguido del patrón del paso `project()` generaran una excepción en algunas situaciones.
- Se ha corregido un error de SPARQL que provocaba que la función `SUBSTR` indicara un error en lugar de devolver una cadena vacía.
- Se ha corregido un error en el motor DFE que podía provocar que las operaciones de unión en los planes de consultas sin bloqueo generaran resultados incorrectos en presencia de variables independientes.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.1, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.1 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.4.1` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.4.1 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea

manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.4.1.R1.1 del motor de Amazon Neptune (22/03/2021)

A partir del 22 de marzo de 2021, se implementará de forma general la versión 1.0.4.1.R1.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha deshabilitado una optimización para los patrones de inserción condicional de Gremlin, que pueden añadirse o anexarse a etiquetas y propiedades existentes.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.1.R1.1, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`

- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.1.R1.1 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.4.1` del motor.

Actualización a esta versión

La versión 1.0.4.1.R1.1 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.4.1.R2 del motor de Amazon Neptune (24/02/2021)

A partir del 24 de febrero de 2021, se implementará de forma general la versión del motor 1.0.4.1.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.4.1.R2.1 (11/03/2021)

Nuevas características de esta versión del motor

- Neptune admite la compresión de archivos individuales en formato `bzip2` para cargas masivas. Consulte Formatos de los datos de carga.

Defectos corregidos en esta versión del motor

- Se ha corregido un error en Versión: 1.0.4.0 (12/10/2020) que permitía las conexiones a Neptune con HTTP o versiones anteriores de TLS, en lugar HTTPS y TLS 1.2.

  > **Important**
  >
  > Tener que utilizar SSL/TLS para todas las conexiones a Neptune puede suponer un cambio radical. Afecta a las conexiones con la consola de Gremlin, el controlador de Gremlin, Gremlin Python, .NET, NodeJS, las API de REST y también a las conexiones del equilibrador de carga. Si ha utilizado HTTP o una versión anterior de TLS para alguna o todas estas conexiones hasta ahora, debe actualizar el cliente y los controladores correspondientes antes de instalar este parche y cambiar el código para que utilice exclusivamente HTTPS.

- Se ha corregido un error de Gremlin por el que se establecía `InternalFailureException` como código de respuesta en determinadas circunstancias cuando se producía una `ConcurrentModificationException`.

- Se ha corregido un error de Gremlin que provocaba que, en determinadas condiciones, al actualizar los bordes o los vértices se produjera una `InternalFailureException` transitoria.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.1.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.1.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.4.1` del motor.

Actualización a esta versión

La versión 1.0.4.1.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
```

```
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.4.1.R2.1 del motor de Amazon Neptune (11/03/2021)

A partir del 11 de marzo de 2021, se implementará de forma general la versión 1.0.4.1.R2.1 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha deshabilitado una optimización para los patrones de inserción condicional de Gremlin, que pueden añadirse o anexarse a etiquetas y propiedades existentes.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.1.R2.1, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.1.R2.1 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.4.1.R2` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.4.1.R2.1 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1.R2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1.R2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.4.0 del motor de Amazon Neptune (12/10/2020)

A partir del 12 de octubre de 2020, se implementará de forma general la versión 1.0.4.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.4.0.R2 (24/02/2021)

Nuevas características de esta versión del motor

- Se ha añadido compresión en el nivel de fotogramas para Gremlin.

Mejoras en esta versión del motor

- Amazon Neptune ahora requiere el uso de Secure Sockets Layer (SSL) con el protocolo TLSv1.2 para todas las conexiones a Neptune en todas las regiones, mediante el uso de estos conjuntos de cifrado seguros:
  - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

  Esto es válido para las conexiones REST y WebSocket a Neptune, e implica que debe usar HTTPS en lugar de HTTP cuando se conecte a Neptune en todas las regiones.

  Dado que las conexiones de cliente mediante HTTP o TLS 1.1 ya no se admitirán en ningún lugar, asegúrese de que los clientes y el código se hayan actualizado para usar TLS 1.2 y HTTPS antes de actualizar a esta versión del motor.

> **Important**
>
> Tener que utilizar SSL/TLS para todas las conexiones a Neptune puede suponer un cambio radical. Afecta a las conexiones con la consola de Gremlin, el controlador de Gremlin, Gremlin Python, .NET, NodeJS, las API de REST y también a las conexiones del equilibrador de carga. Si ha utilizado HTTP para alguna o todas estas aplicaciones, ahora debe actualizar el cliente y los controladores correspondientes y cambiar el código para que utilice HTTPS o se producirá un error en las conexiones.
> Un error de esta versión ha permitido que las conexiones HTTP o las conexiones TLS obsoletas sigan funcionando para los clientes que anteriormente configuraban un parámetro de clúster de base de datos para impedir el establecimiento de las conexiones HTTPS.
> Este error se ha corregido en las versiones de parche 1.0.4.0.R2 y 1.0.4.1.R2, pero la corrección provocaba errores de conexión inesperados cuando los parches se instalaban automáticamente.
> Por este motivo, ambos parches se han revertido y solo se pueden instalar manualmente, para que pueda actualizar la configuración de TLS 1.2.

- Se ha actualizado TinkerPop a la versión 3.4.8. Se trata de una actualización compatible con versiones anteriores. Consulta Registro de cambios de TinkerPop para ver las novedades.
- Se ha mejorado el rendimiento para el paso `properties()` de Gremlin.
- Se han añadido detalles sobre `BindOp` y `MultiplexerOp` en los informes de explicación y perfil.
- Se ha añadido la captura previa de datos para mejorar el rendimiento cuando se pierde la memoria caché.
- Se ha añadido una nueva configuración `allowEmptyStrings` en el parámetro `parserConfiguration` del programa de carga masiva que permite que las cadenas vacías se traten como valores de propiedad válidos en las cargas de CSV (consulte Parámetros de solicitudes del programa de carga de Neptune).
- El programa de carga ahora permite introducir un punto y coma de escape en las columnas de CSV con varios valores.

Defectos corregidos en esta versión del motor

- Se ha corregido una posible pérdida de memoria de Gremlin relacionada con el paso `both()`.

- Se ha corregido un error por el que faltaban las métricas de las solicitudes, ya que un punto de conexión que terminaba en “/” no se gestionaba correctamente.
- Se ha corregido un error que provocaba que las réplicas se retrasaran y se reiniciaran cuando había mucha carga y cuando el motor DFE estaba habilitado en modo lab.
- Se ha corregido un error que impedía que se publicara el mensaje de error correcto cuando se producía un error en una carga masiva debido a un problema de memoria insuficiente.
- Se ha corregido un error de SPARQL que provocaba que la codificación de caracteres se colocara en el encabezado Content-Encoding de las respuestas a las consultas de SPARQL. En su lugar, ahora se coloca `charset` en el encabezado Content-Type, lo que permite a los clientes HTTP reconocer automáticamente el conjunto de caracteres que se está utilizando.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.0 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

La versión 1.0.4.0 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado,

así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.4.0.R2 del motor de Amazon Neptune (24/02/2021)

A partir del 24 de febrero de 2021, se implementará de forma general la versión del motor 1.0.4.0.R2. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error en Versión: 1.0.4.0 (12/10/2020) que permitía las conexiones a Neptune con HTTP o versiones anteriores de TLS, en lugar HTTPS y TLS 1.2.

> **Important**
>
> Tener que utilizar SSL/TLS para todas las conexiones a Neptune puede suponer un cambio radical. Afecta a las conexiones con la consola de Gremlin, el controlador de

Gremlin, Gremlin Python, .NET, NodeJS, las API de REST y también a las conexiones del equilibrador de carga. Si ha utilizado HTTP o una versión anterior de TLS para alguna o todas estas conexiones hasta ahora, debe actualizar el cliente y los controladores correspondientes antes de instalar este parche y cambiar el código para que utilice exclusivamente HTTPS.

- Se ha corregido un error en la carga masiva de archivos CSV que implicaba etiquetas que terminaban en #.
- Se ha corregido un error de Gremlin por el que se establecía `InternalFailureException` como código de respuesta en determinadas circunstancias cuando se producía una `ConcurrentModificationException`.
- Se ha corregido un error de Gremlin que provocaba que, en determinadas condiciones, al actualizar los bordes o los vértices se produjera una `InternalFailureException` transitoria.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.4.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.4.0.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.4.0` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.4.0.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.3.0 del motor de Amazon Neptune (03/08/2020)

A partir del 3 de agosto de 2020, se implementará de forma general la versión 1.0.3.0 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.3.0.R2 (12/10/2020)
- Versión: 1.0.3.0.R3 (19/02/2021)

Nuevas características de esta versión del motor

- Neptune ha introducido un nuevo motor de consultas alternativo (DFE) que puede acelerar considerablemente la ejecución de las consultas. Consulte El motor de consultas alternativo (DFE) de Amazon Neptune (DFE).
- El DFE se basa en estadísticas generadas previamente sobre los datos de un gráfico de Neptune que se administran a través de nuevos puntos de conexión de estadísticas. Consulte Estadísticas del DFE.
- Ahora puede excluir los trabajos de carga en cola de la lista de identificadores de carga devueltos por la API Get-Status del programa de carga estableciendo el nuevo parámetro `includeQueuedLoads` en FALSE. Consulte Parámetros de las solicitudes de obtención de estado del programa de carga de Neptune.
- Neptune ahora admite encabezados finales para las respuestas a consultas de SPARQL que pueden incluir un código y un mensaje de error si se produce un error en una solicitud después de que comience a devolver fragmentos de respuesta. Consulte Encabezados finales HTTP opcionales para las respuestas de SPARQL de varias partes.
- Neptune ahora también le permite habilitar la codificación de respuesta fragmentada para las consultas de Gremlin. Como en el caso de SPARQL, los fragmentos de respuesta tienen encabezados finales que pueden incluir un código de error y un mensaje si se produce un error después de que la consulta haya empezado a devolver fragmentos de respuesta. Consulte Use encabezados finales HTTP opcionales para habilitar las respuestas de Gremlin compuestas por varias partes.

Mejoras en esta versión del motor

- Ahora puede proporcionar el tamaño de las solicitudes por lotes a ElasticSearch para búsquedas de texto completo en Gremlin.
- Se ha mejorado el uso de memoria para las consultas GROUP BY de SPARQL.
- Se ha añadido un nuevo optimizador de consultas de Gremlin para eliminar algunos filtros independientes.
- Se ha aumentado el tiempo máximo que puede permanecer abierta una conexión WebSocket autenticada mediante IAM, de 36 horas a 10 días.

Defectos corregidos en esta versión del motor

- Se ha corregido un error por el que, si enviabas un parámetro de URL no codificado en una solicitud POST, Neptune devolvía un código de estado HTTP de 500 y una `InternalServerErrorException`. Ahora Neptune devuelve un código de estado HTTP de 400 y un `BadRequestException`, con el mensaje: `Failure to process the POST request parameters`.
- Se ha corregido un error de Gremlin por el que no se informaba correctamente de un error de conexión a WebSocket.
- Se ha corregido un error de Gremlin que provocaba la desaparición de sideEffects.
- Se ha corregido un error de Gremlin que provocaba que el parámetro `batchsize` de búsqueda de texto completo no se admitiera correctamente.
- Se ha corregido un error de Gremlin para gestionar `toV` y `fromV` de forma individual para cada dirección en `bothE`.
- Se ha corregido un error de Gremlin relacionado con el valor `Edge pathType` en el paso `hasLabel`.
- Se ha corregido un error de SPARQL que provocaba que la reordenación de las uniones con enlaces estáticos no funcionara correctamente.
- Se ha corregido un error UPDATE LOAD de SPARQL que provocaba que no se informara correctamente de un bucket de Amazon S3 no disponible.
- Se ha corregido un error de SPARQL que provocaba que no se informara correctamente de un problema con un nodo SERVICE de una subconsulta.
- Se ha corregido un error de SPARQL por el que no se evaluaban correctamente las consultas que incluían las condiciones anidadas FILTER EXISTS o FILTER NOT EXISTS.
- Se ha corregido un error de SPARQL que impedía gestionar correctamente los enlaces generados por duplicado al llamar a los puntos de conexión del servicio SPARQL mediante consultas de generación.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.3.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.3`

- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.3.0 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Si el clúster tiene el parámetro `AutoMinorVersionUpgrade` establecido en `True`, se actualizará automáticamente a esta versión del motor dos o tres semanas después de la fecha de lanzamiento, durante un período de mantenimiento.

Actualización a esta versión

La versión 1.0.3.0 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:

```
  We're sorry, your request to modify DB cluster (cluster identifier) has
failed.
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.3.0.R3 del motor de Amazon Neptune (19/02/2021)

A partir del 19 de febrero de 2021, se implementará de forma general la versión 1.0.3.0.R3 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error en la carga masiva de archivos CSV que implicaba etiquetas que terminaban en #.
- Se ha corregido un error de Gremlin que podía provocar que faltaran resultados en un patrón específico de consultas que utilizaban el paso `as()`.
- Se ha corregido un error de Gremlin que podía provocar errores al utilizar el paso `project()` anidado dentro de otro paso, como, por ejemplo `union()`.
- Se ha corregido un error de Gremlin en la ejecución del recorrido de cadenas en el motor DFE experimental cuando se utilizaba un método de terminal como `toList()`.
- Se ha corregido un error de Gremlin que impedía cerrar una transacción al utilizar el paso `iterate()` en una consulta de cadenas.
- Se ha corregido un error de Gremlin que podía provocar que las consultas que utilizaban el patrón `is(P.gte(0))` generaran una excepción en determinadas condiciones.
- Se ha corregido un error de Gremlin por el que se establecía `InternalFailureException` como código de respuesta en determinadas circunstancias cuando se producía una `ConcurrentModificationException`.
- Se ha corregido un error de Gremlin que provocaba que, en determinadas condiciones, al actualizar los bordes o los vértices se produjera una `InternalFailureException` transitoria.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.3.0.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.3.0.R3 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.3.0` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.3.0.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.3.0.R2 del motor de Amazon Neptune (12/10/2020)

A partir del 12 de octubre de 2020, se implementará de forma general la versión 1.0.3.0.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento para el paso `properties()` de Gremlin.
- Se han añadido detalles sobre `BindOp` y `MultiplexerOp` en los informes de explicación y perfil.
- En el caso de las respuestas a las consultas de SPARQL, se ha añadido `charset` al encabezado Content-Type, lo que permite a los clientes HTTP reconocer automáticamente el conjunto de caracteres que se está utilizando.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de SPARQL por el que no se gestionaba `CancellationException`.
- Se ha corregido un error de SPARQL por el que las consultas que incluían opciones anidadas no funcionaban correctamente.
- Se ha corregido un error de SPARQL en LOAD por el que `ConcurrentModificationException` podía provocar el bloqueo de una consulta.
- Se ha corregido un error de SPARQL que impedía que las respuestas a las consultas se comprimieran con gzip.
- Se ha corregido un error de Gremlin en el paso `groupBy()`.

- Se ha corregido un error de Gremlin relacionado con el uso de un paso `aggregate()` dentro de otro paso `local()`.
- Se ha corregido un error de Gremlin relacionado con el uso de `bothE()` seguido de un predicado que utilizaba valores de suma.
- Se ha corregido un error de Gremlin relacionado con el uso del paso `bothE()` con el paso `repeat()`.
- Se ha corregido una posible pérdida de memoria de Gremlin relacionada con el paso `both()`.
- Se ha corregido un error por el que faltaban las métricas de las solicitudes, ya que un punto de conexión que terminaba en "/" no se gestionaba correctamente.
- Se ha corregido un error que podía generar una `ThrottlingException`, incluso cuando la cola de solicitudes no estaba llena.
- Se ha corregido un error al obtener el estado de la carga cuando se produce un error en esta por un motivo como `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.3.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.3`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.3.0.R2 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Si el clúster tiene el parámetro `AutoMinorVersionUpgrade` establecido en `True`, se actualizará automáticamente a esta versión del motor dos o tres semanas después de la fecha de lanzamiento, durante un período de mantenimiento.

Actualización a esta versión

La versión 1.0.3.0.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o

mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.2 del motor de Amazon Neptune (09/03/2020)

A partir del 9 de marzo de 2020, se implementará de forma general la versión 1.0.2.2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Versiones de parche posteriores para esta versión

- Versión: 1.0.2.2.R2 (02/04/2020)

- Versión: 1.0.2.2.R3 (22/07/2020)
- Versión: 1.0.2.2.R4 (23/07/2020)
- Versión: 1.0.2.2.R5 (12/10/2020)
- Versión: 1.0.2.2.R6 (19/02/2021)

Mejoras en esta versión del motor

- Se ha agregado información en la API de estado sobre las transacciones que se están revirtiendo. Consulte Estado de la instancia.
- Apache TinkerPop se ha actualizado a la versión 3.4.3.

  La versión 3.4.3 es compatible con la versión anterior admitida por Neptune (3.4.1). Introduce un cambio menor en el comportamiento: Gremlin ya no devuelve un error al intentar cerrar una sesión que no existe (consulte Prevent error when closing sessions that don't exist).
- Se han eliminado los cuellos de botella del rendimiento que se producían al ejecutar los pasos de búsqueda de texto completo de Gremlin.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de SPARQL relacionado con el uso de patrones de gráficos vacíos en las consultas.
- Se ha corregido un error de SPARQL relacionado con el uso de símbolos de puntos y coma sin codificar en consultas codificadas como URL.
- Se ha corregido un error de Gremlin relacionado con el uso de vértices repetidos en el paso `Union`.
- Se ha corregido un error de Gremlin que provocaba que algunas consultas con `.simplePath()` o `.cyclicPath()` dentro de `.repeat()` devolvieran resultados incorrectos.
- Se ha corregido un error de Gremlin que provocaba que `.project()` devolviera resultados incorrectos si su recorrido secundario no devolvía soluciones.
- Se ha corregido un problema de Gremlin en el que los conflictos de lectura-escritura generaban una excepción `InternalFailureException` en lugar de `ConcurrentModificationException`.
- Se ha corregido un problema de Gremlin que provocaba errores `.group().by(...).by(values("property"))`.

- Se han corregido errores de Gremlin en la salida del perfil con los pasos de búsqueda de texto completo.
- Se ha corregido una fuga de recursos en las sesiones de Gremlin.
- Se ha corregido un error que impedía que la API de estado informara de la versión correcta que se podía pedir en algunos casos.
- Se ha corregido un error del programa de carga en bloque que permitía que una URL de una ubicación distinta de Amazon S3 se utilizara en una solicitud de carga en bloque.
- Se ha corregido un error del programa de carga en bloque relacionado con el estado detallado de la carga.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.3`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.2 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Si el clúster tiene el parámetro `AutoMinorVersionUpgrade` establecido en `True`, se actualizará automáticamente a esta versión del motor dos o tres semanas después de la fecha de lanzamiento, durante un período de mantenimiento.

Actualización a esta versión

La versión 1.0.2.2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.2.R6 del motor de Amazon Neptune (19/02/2021)

A partir del 19 de febrero de 2021, se implementará de forma general la versión 1.0.2.2.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin por el que se establecía `InternalFailureException` como código de respuesta en determinadas circunstancias cuando se producía una `ConcurrentModificationException`.
- Se ha corregido un error de Gremlin que provocaba que, en determinadas condiciones, al actualizar los bordes o los vértices se produjera una `InternalFailureException` transitoria.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.2.R6, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.8`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.2.R6 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.2.2` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.2.2.R6 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
> ```

```
  Cannot modify engine version because instance (instance identifier) is
  running on an old configuration. Apply any pending maintenance actions on the
instance before
  proceeding with the upgrade.
```

Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.2.R5 del motor de Amazon Neptune (12/10/2020)

A partir del 12 de octubre de 2020, se implementará de forma general la versión 1.0.2.2.R5 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento para el paso `properties()` de Gremlin.
- Se han añadido detalles sobre `BindOp` y `MultiplexerOp` en los informes de explicación y perfil.
- En el caso de las respuestas a las consultas de SPARQL, se ha añadido `charset` al encabezado Content-Type, lo que permite a los clientes HTTP reconocer automáticamente el conjunto de caracteres que se está utilizando.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de SPARQL por el que no se gestionaba `CancellationException`.
- Se ha corregido un error de SPARQL por el que las consultas que incluían opciones anidadas no funcionaban correctamente.
- Se ha corregido un error de SPARQL en LOAD por el que `ConcurrentModificationException` podía provocar el bloqueo de una consulta.
- Se ha corregido un error de SPARQL que impedía que las respuestas a las consultas se comprimieran con gzip.
- Se ha corregido un error de Gremlin en el paso `groupBy()`.

- Se ha corregido un error de Gremlin relacionado con el uso de un paso `aggregate()` dentro de otro paso `local()`.
- Se ha corregido un error de Gremlin relacionado con el uso de `bothE()` seguido de un predicado que utilizaba valores de suma.
- Se ha corregido un error de Gremlin relacionado con el uso del paso `bothE()` con el paso `repeat()`.
- Se ha corregido una posible pérdida de memoria de Gremlin relacionada con el paso `both()`.
- Se ha corregido un error por el que faltaban las métricas de las solicitudes, ya que un punto de conexión que terminaba en "/" no se gestionaba correctamente.
- Se ha corregido un error que podía generar una `ThrottlingException`, incluso cuando la cola de solicitudes no estaba llena.
- Se ha corregido un error al obtener el estado de la carga cuando se produce un error en esta por un motivo como `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE`.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.2.R5, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.3`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.2.R5 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.2.2` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.2.2.R5 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o

mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.2.R4 del motor de Amazon Neptune (23/07/2020)

A partir del 23 de julio de 2020, se implementará de forma general la versión 1.0.2.2.R4 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Mejoras en esta versión del motor

- Se ha mejorado el uso de la memoria al devolver la memoria no utilizada al sistema operativo con mayor frecuencia.

- También se ha mejorado el uso de memoria para las consultas GROUP BY de SPARQL.
- Se ha aumentado el tiempo máximo que puede permanecer abierta una conexión WebSocket autenticada mediante IAM, de 36 horas a 10 días.
- Se ha añadido la métrica `BufferCacheHitRatio` de CloudWatch, que puede resultar útil para diagnosticar la latencia de las consultas y ajustar los tipos de instancias. Consulte Métricas de Neptune.

Defectos corregidos en esta versión del motor

- Se ha corregido un error al cerrar las conexiones WebSocket de IAM inactivas o caducadas. Neptune envía ahora una trama de cierre antes de cerrar la conexión.
- Se ha corregido un error de SPARQL en la evaluación de las consultas que incluían las condiciones anidadas FILTER EXISTS o FILTER NOT EXISTS.
- Se ha corregido un error de finalización de consultas de SPARQL que provocaba el bloqueo de subprocesos en el servidor en determinadas condiciones extremas.
- Se ha corregido un error de Gremlin relacionado con el valor Edge pathType en el paso `hasLabel`.
- Se ha corregido un error de Gremlin para gestionar `toV` y `fromV` de forma individual para cada dirección en `bothE`.
- Se ha corregido un error de Gremlin que provocaba la desaparición de sideEffects.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.2.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.3`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.2.R4 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.2.2` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.2.2.R4 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A

continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.2.R3 (22/07/2020) del motor de Amazon Neptune

La versión del motor 1.0.2.2.R3 se incorporó a la versión del motor 1.0.2.2.R4.

Versión 1.0.2.2.R2 del motor de Amazon Neptune (02/04/2020)

A partir del 2 de abril de 2020, se implementará de forma general la versión 1.0.2.2.R2 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Mejoras en esta versión del motor

- Ahora puede poner en cola hasta 64 trabajos de carga en bloque, en lugar de tener que esperar a que finalice uno antes de iniciar el siguiente. También puede hacer que la ejecución de una solicitud de carga en cola dependa de la finalización correcta de uno o más trabajos de carga puestos anteriormente en cola con el parámetro `dependencies` del comando `load`. Consulte Comando del programa de carga de Neptune.
- La salida de búsqueda de texto completo ahora se puede ordenar (consulte Parámetros de búsqueda de texto completo).
- Ahora hay un parámetro de clúster de base de datos para invocar flujos de Neptune y la característica se ha movido fuera del modo lab. Consulte Habilitación de flujos de Neptune.

Defectos corregidos en esta versión del motor

- Se ha corregido un error estocástico en el inicio del servidor que retrasaba la creación de instancias.
- Se ha corregido un problema del optimizador por el que las instrucciones `BIND` de la consulta hacían que el optimizador comenzara con patrones no selectivos en la planificación del orden de combinación.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.2.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.3`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.2.R2 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.2.2` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.2.2.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.1 del motor de Amazon Neptune (22/11/2019)

Versiones de parche posteriores para esta versión

- Versión: 1.0.2.1.R6 (22/04/2020)
- Versión: 1.0.2.1.R5 (22/04/2020) Esta versión de parche no se implementó.
- Versión: 1.0.2.1.R4 (20/12/2019)
- Versión: 1.0.2.1.R3 (12/12/2019)
- Versión: 1.0.2.1.R2 (25/11/2019)

Nuevas características de esta versión del motor

- Se han añadido capacidades de búsqueda de texto completo mediante la integración con Amazon OpenSearch Service. Consulte Búsqueda de texto completo de Neptune
- Se ha añadido la opción de uso del modo lab para crear un cuarto índice (un índice OSGP) para un gran número de predicados. Consulte Índice OSGP.
- Se ha añadido un modo de detalles a SPARQL Explain. Consulte Uso de `explain` de SPARQ y Salida del modo de detalles para obtener más información.
- Se ha añadido información del modo lab al informe de estado del motor. Para obtener más información, consulte Estado de la instancia.
- Las instantáneas de clúster de base de datos ahora se pueden copiar en todas las regiones de AWS. Consulte Copia de una instantánea.

Mejoras en esta versión del motor

- Rendimiento mejorado al gestionar un gran número de predicados.
- Optimización de consultas mejorada. Aunque debería ser totalmente transparente para los clientes, le recomendamos que pruebe sus aplicaciones antes de realizar la actualización para asegurarse de que se comportan como se espera.
- Mejoras secundarias en el informe de errores.

- Se han añadido optimizaciones para los pasos `.project()` y `.identity()` de Gremlin.
- Se han añadido optimizaciones para casos `.union()` de Gremlin no terminales.
- Se ha añadido compatibilidad nativa con los recorridos `.path().by()` de Gremlin.
- Se ha añadido compatibilidad nativa para `.coalesce()` de Gremlin.
- Optimización adicional de la escritura en bloque.
- Ahora requerimos que las conexiones HTTPS usen al menos la versión 1.2 de TLS o superior, para evitar que se utilicen cifrados obsoletos o no seguros.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de gestión del recorrido interno `addE()` de Gremlin.
- Se ha corregido un error de Gremlin producido por las anotaciones AST que se filtraban desde los recorridos secundarios al principal.
- Se ha corregido un error que se producía en Gremlin cuando se llamaba a `.otherV()` después de `select()`.
- Se ha corregido un error de Gremlin que provocaba que algunos pasos de `.hasLabel()` fallaran si aparecían después de un paso de `bothE()`.
- Se han efectuado correcciones secundarias a .sum() y.project() de Gremlin.
- Se ha corregido un error en el procesamiento de consultas SPARQL que no tenían llave de cierre.
- Se han corregido algunos errores secundarios en SPARQL Explain.
- Se ha corregido un error en la gestión de solicitudes de obtención de estado de carga simultáneas.
- Reducción de la memoria usada para ejecutar algunos recorridos de Gremlin con pasos `.project()`.
- Se han corregido las comparaciones numéricas de valores especiales en SPARQL. Consulte Conformidad con los estándares.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.1, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.1 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

La versión 1.0.2.1 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.1.R6 del motor de Amazon Neptune (22/04/2020)

A partir del 22 de abril de 2020, se implementará de forma general la versión 1.0.2.1.R6 del motor. Tenga en cuenta que las versiones nuevas tardan unos días en estar disponibles en todas las regiones.

Defectos corregidos en esta versión del motor

- Se ha corregido un error donde `ConcurrentModificationConflictException` y `TransactionException` no se convirtieron en una `NeptuneGremlinException`, lo que provocaba que se devolviera `InternalFailureException` a los clientes.
- Se ha corregido un error en el que Neptune indicaba que tenía buen estado antes de que el servidor estuviera completamente listo.
- Se ha corregido un error en el que las confirmaciones de transacciones de diccionario y usuario estaban fuera de servicio cuando se insertaban dos asignaciones `value->id` simultáneamente.
- Se ha corregido un error en la serialización del estado de carga.
- Se ha corregido un error de sesiones de Gremlin.
- Se ha corregido un error en el que Neptune no podía lanzar una excepción cuando el servidor no se iniciaba.
- Se ha corregido un error por el que Neptune no podía enviar una trama de cierre de WebSocket antes de cerrar el canal.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.1.R6, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.1.R6 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.2.1` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.2.1.R6 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.1.R5 del motor de Amazon Neptune (22/04/2020)

La versión del motor 1.0.2.1.R5 nunca se implementó.

Versión 1.0.2.1.R4 (20/12/2019) del motor de Amazon Neptune

Mejoras en esta versión del motor

- Neptune ahora intenta siempre colocar cualquier llamada de búsqueda de texto completo primero en la canalización de la ejecución. Esto reduce el volumen de llamadas a OpenSearch, lo que puede mejorar considerablemente el rendimiento. Consulte Ejecución de consultas de búsqueda de texto completo.
- Neptune ahora genera una `IllegalArgumentException` si intenta acceder a una propiedad, vértice o borde inexistente. Anteriormente, Neptune generaba una `UnsupportedOperationException` en esa situación.

  Por ejemplo, si intenta agregar un borde que haga referencia a un vértice inexistente, generará una `IllegalArgumentException`.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin en el que un recorrido `union` dentro de un `project-by` no devuelve resultados o devuelve resultados incorrectos.
- Se ha corregido un error de Gremlin que provocaba que los pasos `.project().by()` anidados devolvieran resultados incorrectos.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.1.R4, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.1.R4 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Sin embargo, no se admite la actualización automática de esta versión.

Actualización a esta versión

La versión 1.0.2.1.R4 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.1.R3 del motor de Amazon Neptune (12/12/2019)

Defectos corregidos en esta versión del motor

- Se ha corregido un error en el que el índice `OSGP` estaba deshabilitado aunque la función estaba correctamente habilitada usando enModo lab usando el valor `ObjectIndex` en el parámetro `neptune_lab_mode`.
- Se ha corregido un error que afectaba a las consultas de Gremlin con un `.fold()` dentro de un paso de `.project().by()`. Por ejemplo, provocó que la siguiente consulta devolviese resultados incompletos:

```
g.V().project("a").by(valueMap().fold())
```

- Se ha corregido un cuello de botella de rendimiento en cargas masivas de datos de RDF.
- Se ha corregido un error que provocaba un bloqueo en las réplicas cuando se habilitaban las transmisiones y se reiniciaba la réplica antes del principal.
- Se ha corregido un error que provocaba que los certificados SSL rotados en las instancias no se recogieran sin reiniciar la instancia.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.1.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.1.R3 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Sin embargo, no se admite la actualización automática de esta versión.

Actualización a esta versión

La versión 1.0.2.1.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A

continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.1.R2 del motor de Amazon Neptune (25/11/2019)

Defectos corregidos en esta versión del motor

- Se ha corregido un error que afectaba a todas las consultas `project().by()` con subrecorridos sin turno rotativo y subrecorridos sin `path()`.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.1.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.1.R2 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Sin embargo, no se admite la actualización automática de esta versión.

Actualización a esta versión

La versión 1.0.2.1.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.0 del motor de Amazon Neptune (08/11/2019)

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 19 de mayo de 2020, no se crearán nuevas instancias con esta versión del motor.

Esta versión del motor ahora se ha sustituido por la versión 1.0.2.1, que incluye todas las correcciones de errores de esta versión, así como características adicionales como la integración de la búsqueda de texto completo, compatibilidad con índices OSGP y copia del clúster de instantáneas de bases de datos en las regiones de AWS.

A partir del 1 de junio de 2020, Neptune actualizará automáticamente cualquier clúster que ejecute esta versión del motor al último parche de la versión 1.0.2.1 durante el siguiente periodo de mantenimiento. Puede actualizar manualmente antes de ese momento, como se describe aquí.

Si tiene algún problema con la actualización, contacte con nosotros a través de AWS Support o de los foros para desarrolladores de AWS.

Versiones de parche posteriores para esta versión

- Versión: 1.0.2.0.R3 (05/05/2020)
- Versión: 1.0.2.0.R2 (21/11/2019)

Nuevas características de esta versión del motor

Además de las actualizaciones de mantenimiento, esta versión añade nuevas funcionalidades para admitir más de una versión de motor a la vez (consulte Mantenimiento del clúster de base de datos de Amazon Neptune).

Como resultado, la numeración de las versiones de motor ha cambiado (consulte Numeración de versiones antes de la versión 1.3.0.0 del motor).

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.0, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.0 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

No se actualizará automáticamente a esta versión.

Actualización a esta versión

La versión 1.0.2.0 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
> We're sorry, your request to modify DB cluster (cluster identifier) has failed.
> Cannot modify engine version because instance (instance identifier) is
> running on an old configuration. Apply any pending maintenance actions on the instance before
> proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.0.R3 del motor de Amazon Neptune (05/05/2020)

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 19 de mayo de 2020, no se crearán nuevas instancias con esta versión del motor.

Esta versión del motor ahora se ha sustituido por la versión 1.0.2.1, que incluye todas las correcciones de errores de esta versión, así como características adicionales como la integración de la búsqueda de texto completo, compatibilidad con índices OSGP y copia del clúster de instantáneas de bases de datos en las regiones de AWS.

A partir del 1 de junio de 2020, Neptune actualizará automáticamente cualquier clúster que ejecute esta versión del motor al último parche de la versión 1.0.2.1 durante el siguiente periodo de mantenimiento. Puede actualizar manualmente antes de ese momento, como se describe aquí.

Si tiene algún problema con la actualización, contacte con nosotros a través de AWS Support o de los foros para desarrolladores de AWS.

Defectos corregidos en esta versión del motor

- Se ha corregido un error en el que `ConcurrentModificationConflictException` y `TransactionException` se notificaban como `InternalFailureExceptions` genéricas.
- Se han corregido errores en las comprobaciones de estado que provocaban reinicios frecuentes del servidor durante el inicio.
- Se ha corregido un error en el que los datos no eran visibles en las réplicas porque las confirmaciones estaban fuera de servicio bajo ciertas condiciones.
- Se ha corregido un error en la serialización del estado de carga que provocaba un error de carga debido a la falta de permisos de acceso de Amazon S3.
- Se ha corregido una fuga de recursos en las sesiones de Gremlin.
- Se ha corregido un error en la comprobación de estado que ocultaba el mal estado en el inicio de los componentes que administraban la autenticación de IAM.
- Se ha corregido un error por el que Neptune no podía enviar una trama de cierre de WebSocket antes de cerrar el canal.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.0.R3, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.0.R3 del motor

El clúster se actualizará automáticamente a esta versión de parche durante el siguiente periodo de mantenimiento si está ejecutando la versión `1.0.2.0` del motor.

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Actualización a esta versión

La versión 1.0.2.0.R3 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>  We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>  Cannot modify engine version because instance (instance identifier) is
>  running on an old configuration. Apply any pending maintenance actions on the
> instance before
>  proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.2.0.R2 del motor de Amazon Neptune (21/11/2019)

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 19 de mayo de 2020, no se crearán nuevas instancias con esta versión del motor.

Esta versión del motor ahora se ha sustituido por la versión 1.0.2.1, que incluye todas las correcciones de errores de esta versión, así como características adicionales como la integración de la búsqueda de texto completo, compatibilidad con índices OSGP y copia del clúster de instantáneas de bases de datos en las regiones de AWS.

A partir del 1 de junio de 2020, Neptune actualizará automáticamente cualquier clúster que ejecute esta versión del motor al último parche de la versión 1.0.2.1 durante el siguiente periodo de mantenimiento. Puede actualizar manualmente antes de ese momento, como se describe aquí.

Si tiene algún problema con la actualización, contacte con nosotros a través de AWS Support o de los foros para desarrolladores de AWS.

Defectos corregidos en esta versión del motor

- Mejorada la estrategia de almacenamiento en caché de las páginas sin validar en el servidor para que `FreeableMemory` se recupere más rápido cuando el servidor cambia a un estado de poca memoria.
- Se ha corregido un error que podía producir una condición de carrera y un bloqueo cuando se procesan en el servidor muchas solicitudes de estado de carga o de carga de inicio.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.2.0.R2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Rutas de actualización a la versión 1.0.2.0.R2 del motor

Puede actualizar manualmente cualquier versión anterior del motor de Neptune a esta versión.

Sin embargo, no se admite la actualización automática de esta versión.

Actualización a esta versión

La versión 1.0.2.0.R2 de Amazon Neptune ya está disponible con carácter general.

Si un clúster de base de datos ejecuta una versión de motor desde la que existe una ruta de actualización a esta versión, puede actualizarse ahora. Puede actualizar cualquier clúster que

cumpla los requisitos mediante las operaciones del clúster de base de datos de la consola o mediante el SDK. El siguiente comando de la CLI actualizará inmediatamente un clúster que cumpla los requisitos:

Para Linux, OS X o Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Para Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en esas instancias, por lo que se experimentará un tiempo de inactividad que oscila entre 20-30 segundos y varios minutos, tras el cual se puede reanudar el uso del clúster de base de datos.

Realice siempre una prueba antes de realizar la actualización

Cuando se publique una nueva versión principal o secundaria del motor de Neptune, pruebe siempre las aplicaciones de Neptune en ella antes de actualizar. Incluso en una actualización secundaria podría haber nuevas características o comportamientos que podrían afectar al código.

Comience por comparar las páginas de notas de la versión actual con las de la versión de destino para ver si hay cambios en las versiones del lenguaje de consulta u otros cambios importantes.

La mejor forma de probar una nueva versión antes de actualizar el clúster de base de datos de producción es clonar el clúster de producción para que el clon ejecute la nueva versión del motor. A continuación, puede ejecutar consultas en el clon sin que eso afecte al clúster de base de datos de producción.

Cree siempre una instantánea manual antes de realizar la actualización

Antes de realizar una actualización, se recomienda crear siempre una instantánea manual del clúster de base de datos. Una instantánea automática solo ofrece protección a corto plazo, mientras que una instantánea manual está disponible hasta que la elimine explícitamente.

En algunos casos, Neptune crea una instantánea manual para usted como parte del proceso de actualización, pero no debe confiar en eso y crear su propia instantánea manual.

Cuando tenga la seguridad de que no necesitará revertir el clúster de base de datos al estado anterior a la actualización, puede eliminar de forma explícita la instantánea manual que ha creado, así como la instantánea manual que Neptune podría haber creado. Si Neptune crea una instantánea manual, tendrá un nombre que empieza por `preupgrade`, seguido del nombre del clúster de base de datos, la versión del motor de origen, la versión del motor de destino y la fecha.

> **Note**
>
> Si intenta realizar la actualización mientras hay una acción pendiente en proceso, es posible que se produzca un error como el siguiente:
>
> ```
>   We're sorry, your request to modify DB cluster (cluster identifier) has
> failed.
>   Cannot modify engine version because instance (instance identifier) is
>   running on an old configuration. Apply any pending maintenance actions on the
> instance before
>   proceeding with the upgrade.
> ```
>
> Si se produce este error, espere a que finalice la acción pendiente o active inmediatamente un periodo de mantenimiento para que se complete la actualización anterior.

Para obtener más información sobre la actualización de la versión del motor, consulte Mantenimiento del clúster de base de datos de Amazon Neptune. Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Versión 1.0.1.2 del motor de Amazon Neptune (10/06/2020)

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Mejoras en esta versión del motor

- Neptune ahora genera una `IllegalArgumentException` si intenta acceder a una propiedad, vértice o borde inexistente. Anteriormente, Neptune generaba una `UnsupportedOperationException` en esa situación.

  Por ejemplo, si intenta agregar un borde que haga referencia a un vértice inexistente, generará una `IllegalArgumentException`.

Defectos corregidos en esta versión del motor

- Se ha corregido un error en el que las confirmaciones de transacciones de diccionario y usuario estaban fuera de servicio cuando se insertaban dos asignaciones `value->id` simultáneamente.
- Se ha corregido un error en la serialización del estado de carga.
- Se ha corregido un error estocástico en el inicio del servidor que retrasaba la creación de instancias.
- Se ha corregido una pérdida de cursor.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.1.2, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.4.1`
- Versión de SPARQL: `1.1`

Versión 1.0.1.1 del motor de Amazon Neptune (26/06/2020)

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Defectos corregidos en esta versión del motor

- Se ha corregido un error por el que las confirmaciones estaban fuera de servicio cuando se insertaban simultáneamente.

- Se ha corregido un error en la serialización del estado de carga.
- Se ha corregido un error estocástico en el inicio del servidor que retrasaba la creación de instancias.
- Se ha corregido una fuga de memoria.

Versiones de lenguaje de consulta admitidas en esta versión

Antes de actualizar un clúster de base de datos a la versión 1.0.1.1, asegúrese de que el proyecto sea compatible con estas versiones de lenguaje de consulta:

- Versión de Gremlin: `3.3.2`
- Versión de SPARQL: `1.1`

Versión 1.0.1.0 del motor de Amazon Neptune (02/07/2019)

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Actualizaciones del motor de Amazon Neptune del 31/10/2019

Versión: 1.0.1.0.200502.0

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin en la serialización de la respuesta del paso `tree()` cuando los clientes se conectan a Neptune utilizando `traversal().withRemote(...)` (en otras palabras, usando el código de bytes GLV).

  Esta versión soluciona un problema por el que los clientes que se conectaban a Neptune utilizando `traversal().withRemote(...)` recibían una respuesta no válida a las consultas de Gremlin que contenían el paso `tree()`.

- Se ha corregido un error de SPARQL relacionado con las consultas `DELETE WHERE LIMIT`, por el que el proceso de terminación de la consulta se bloqueaba debido por una condición de carrera, lo que hacía que el tiempo de la consulta se agotara.

Actualizaciones del motor de Amazon Neptune del 15/10/2019

Versión: 1.0.1.0.200463.0

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Nuevas características de esta versión del motor

- Se ha añadido una característica de explicación/perfil de Gremlin (consulte Análisis de la ejecución de las consultas de Neptune con `explain` de Gremlin).
- Se ha añadido Compatibilidad con sesiones basadas en scripts de Gremlin para habilitar la ejecución de varios recorridos de Gremlin en una única transacción.
- Se ha añadido compatibilidad con la extensión de consulta federada de SPARQL en Neptune (consulte Consulta federada de SPARQL 1.1 y Consultas federadas de SPARQL en Neptune mediante la extensión `SERVICE`).
- Se ha añadido una característica que le permite insertar su propio `queryId` en una consulta de Gremlin o SPARQL, ya sea a través de un parámetro de URL HTTP o a través de una sugerencia de consulta `queryId` SPARQL (consulte Inserte un identificador personalizado en una consulta de Neptune Gremlin o SPARQL).
- Se ha añadido una característica Modo lab a Neptune que puede permitirle probar las siguientes características que aún no están listas para utilizarse en producción.
- Se ha añadido una característica Flujos de Neptune próxima que registra de forma fiable todos los cambios realizados en la base de datos en una secuencia que persiste durante una semana. Esta característica solo está disponible en el modo lab.
- Se ha actualizado la semántica formal para transacciones simultáneas (consulte Semántica de transacciones en Neptune). Esta característica proporciona garantías estándares del sector en torno a la simultaneidad.

  De forma predeterminada, esta semántica de transacción está habilitada. En algunos casos, esta característica puede cambiar el comportamiento de carga actual y reducir el rendimiento de la carga. Puede utilizar el parámetro `neptune_lab_mode` del clúster de base de datos para volver

a la semántica anterior incluyendo `ReadWriteConflictDetection=disabled` en el valor del parámetro.

Mejoras en esta versión del motor

- Se ha mejorado la API Estado de la instancia informando de qué versión de TinkerPop y qué versión de SPARQL utiliza el motor.
- Se ha mejorado el rendimiento del operador de subgráfico de Gremlin.
- Se ha mejorado el rendimiento de la serialización de respuestas de Gremlin.
- Se ha mejorado el rendimiento en el paso de Gremlin Union.
- Se ha mejorado la latencia de las consultas de SPARQL sencillas.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de Gremlin por el que el tiempo de espera se devolvía incorrectamente como error interno.
- Se ha corregido un error SPARQL en el que ORDER BY sobre un conjunto parcial de variables provocaba un error interno del servidor.

Actualizaciones del motor de Amazon Neptune del 19/09/2019

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Versión: 1.0.1.0.200457.0

La versión 1.0.1.0.200457.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200457.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar un clúster de base de datos:

```
aws neptune apply-pending-maintenance-action \
```

```
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Defectos corregidos en esta versión del motor

- Se ha corregido un problema de corrección de Gremlin introducido en la versión anterior del motor (1.0.1.0.200369.0) eliminando la mejora del rendimiento de la gestión de predicados conjuntivos que la provocaba.
- Se ha corregido un error SPARQL que provocaba consultas con `DISTINCT` y un único patrón encapsulado en `OPTIONAL` para generar un `InternalServerError`.

Actualizaciones del motor de Amazon Neptune del 13/08/2019

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Nuevas características de esta versión del motor

- Se ha añadido una opción `OVERSUBSCRIBE` al parámetro `parallelism` de Comando del programa de carga de Neptune, lo que provoca que el programa de carga masiva de Neptune utilice todos los subprocesos y recursos disponibles.

Mejoras en esta versión del motor

- Se ha mejorado el rendimiento de los filtros de SPARQL que contienen expresiones lógicas OR sencillas.
- Se ha mejorado el rendimiento de Gremlin en la administración de predicados conjuntivos.

Defectos corregidos en esta versión del motor

- Se ha corregido un error de SPARQL que impedía restar `xsd:duration` a `xsd:date`.
- Se ha corregido un error de SPARQL que provocaba resultados incompletos de incorporación estática en la presencia de `UNION`.
- Se ha corregido un error de SPARQL en la cancelación de consultas.
- Se ha corregido un error de Gremlin que provocaba un desbordamiento durante la promoción de tipos.
- Se ha corregido un error de Gremlin en la administración de elementos de vértice en los pasos de `addE().from().to()`.
- Se ha corregido un error de Gremlin (publicado el 26-07-2019 en la versión 1.0.1.0.0.200366.0 del motor) que implicaba la administración de valores dobles y flotantes NaN en inserciones de cardinalidad única.
- Se ha corregido un error en la generación de planes de consulta que implicaban búsquedas basadas en propiedades.

Actualizaciones del motor de Amazon Neptune del 26/07/2019

Versión: 1.0.1.0.200366.0

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Nuevas características de esta versión del motor

- Se ha actualizado a TinkerPop 3.4.1 (consulte Información de actualización de TinkerPop y Registro de cambios de TinkerPop 3.4.1).

  Para los clientes de Neptune, estos cambios proporcionan nuevas funcionalidades y mejoras, tales como:

  - `GraphBinary` ahora está disponible en formato de serialización.
  - Se ha corregido un error de mantenimiento de conexión que provocaba fugas de memoria en el controlador Java de TinkerPop, por lo que ya no es necesario solucionar el problema.

  Sin embargo, en algunos casos, pueden afectar al código Gremlin existente en Neptune. Por ejemplo:

- `valueMap()` ahora devuelve `Map<Object,Object>` en lugar de `Map<String,Object>`.
- Se ha corregido el comportamiento incoherente del paso `within()` de modo que funcione de un modo coherente con otros pasos. Anteriormente, los tipos tenían que coincidir para que funcionaran las comparaciones. Ahora, los números de los diferentes tipos se pueden comparar con precisión. Por ejemplo, `33` ahora se compara igual que `33L`, lo que no sucedía antes.
- Se ha corregido un error de `ReducingBarrierStep`, por lo que ahora no devuelve ningún valor si no hay elementos disponibles para la salida.
- Se ha cambiado el orden de los ámbitos `select()` (el orden ahora es `maps`, `side-effects` y `paths`). Esto cambia los resultados de las contadas consultas que combinan `side-effects` y `select` con el mismo nombre de clave tanto para `side-effects` como para `select`.
- `bulkSet()` ahora forma parte del protocolo GraphSON. Las consultas que terminen con `toBulkSet()` no funcionan con clientes anteriores.
- Se ha eliminado una parametrización del paso `Submit()` del cliente 3.4.

Se han incorporado muchos cambios en TinkerPop 3.4 que no afectan el comportamiento actual de Neptune. Por ejemplo, se ha añadido `io()` de Gremlin como un paso para `Traversal` y ahora está obsoleto en `Graph`, pero nunca se ha activado en Neptune.

- Se ha añadido compatibilidad para las propiedades de vértice de cardinalidad única al programa de carga masiva de Gremlin, para cargar datos de gráficos de propiedades.
- Se ha añadido una opción para sobrescribir los valores existentes para una propiedad de cardinalidad única en el programa de carga masiva.
- Se ha añadido la posibilidad de recuperar el estado de una consulta de Gremlin y de cancelar una consulta de Gremlin.
- Se ha añadido una sugerencia de consulta para los tiempos de espera de consulta SPARQL.
- Se ha añadido la capacidad de ver la función de la instancia en la API de estado (consulte Estado de la instancia).
- Se ha añadido compatibilidad con la clonación de bases de datos (consulte Clonación de bases de datos en Neptune).

Mejoras en esta versión del motor

- Se ha mejorado la explicación de consulta SPARQL para mostrar variables de gráficos de cláusulas FROM.

- Se ha mejorado el rendimiento para SPARQL en filtros, filtros de igual, cláusulas VALUES y recuentos de rango.
- Se ha mejorado el rendimiento para la ordenación de pasos de Gremlin.
- Se ha mejorado el rendimiento para los recorridos `.repeat.dedup` de Gremlin.
- Se ha mejorado el rendimiento de `valueMap()` y `path().by()` de Gremlin.

Defectos corregidos en esta versión del motor

- Se han corregido varios problemas con las rutas de propiedad de SPARQL incluido el funcionamiento con gráficos denominados.
- Se ha corregido un problema con consultas de SPARQL CONSTRUCT que causa problemas de memoria.
- Se ha corregido un problema con el analizador RDF Turtle y los nombres locales.
- Se ha corregido un problema para corregir mensajes de error que se muestran a los usuarios.
- Se ha corregido un problema con recorridos `repeat()...drop()` de Gremlin.
- Se ha corregido un problema con el paso `drop()` de Gremlin.
- Se ha corregido un problema con filtros de etiqueta de Gremlin.
- Se ha corregido un problema con los tiempos de espera de consultas de Gremlin.

Actualizaciones del motor de Amazon Neptune del 02/07/2019

IMPORTANTE: ESTA VERSIÓN DEL MOTOR AHORA ESTÁ OBSOLETA

A partir del 27 de abril de 2021, no se crearán nuevas instancias con esta versión del motor.

Defectos corregidos en esta versión del motor

- Se ha corregido un error que provocaba que determinados patrones con un nombre y valor de propiedad no se optimizaran.

Versiones anteriores del motor de Neptune

Temas

Actualizaciones del motor de Amazon Neptune del 12/06/2019

Versión: 1.0.1.0.200310.0

La versión 1.0.1.0.200310.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200310.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200310.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Corrige un error en el que insertar y soltar un borde de forma simultánea puede dar lugar a varios bordes con el mismo identificador.
- Otras pequeñas correcciones y mejoras.

Actualizaciones del motor de Amazon Neptune del 01/05/2019

Versión: 1.0.1.0.200296.0

La versión 1.0.1.0.200296.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200296.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200296.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Se ha añadido la característica nueva `explain` a las consultas SPARQL de Neptune para ayudarle a visualizar el plan de consulta y realizar los pasos necesarios para optimizarlo si fuera necesario. Para obtener información, consulte `explain` de SPARQL.
- Se ha mejorado el rendimiento de SPARQL y la generación de informes de varias formas.
- Se ha mejorado el rendimiento y el comportamiento de Gremlin de varias formas.
- Se ha mejorado el agotamiento del tiempo de espera de las consultas `drop( )` de ejecución prolongada.
- Se ha mejorado el rendimiento de las consultas `otherV( )`.
- Se han añadido dos campos a la información devuelta al consultar el estado de Neptune de un clúster de base de datos o de una instancia, en concreto el número de versión del motor y el clúster o la hora de inicio de la instancia. Consulte Estado de la instancia.
- La API `Get-Status` del programa de carga de Neptune devuelve ahora un campo `startTime` que registra el momento en el que se inició una tarea de carga.
- El comando del programa de carga toma ahora un parámetro `parallelism` opcional que le permite restringir el número de subprocesos que utiliza el programa de carga.

Actualizaciones del motor de Amazon Neptune del 21/01/2019

Versión: 1.0.1.0.200267.0

La versión 1.0.1.0.200267.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200267.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200267.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Neptune espera más tiempo (dentro del tiempo de espera de consulta especificado) para que se resuelvan los conflictos. De este modo se reduce el número de excepciones de modificación simultáneas que debe administrar el cliente (consulte Errores de consulta).

- Se ha corregido un problema por el que la aplicación de cardinalidad de Gremlin en ocasiones provocaba que el motor se reiniciara.
- Rendimiento mejorado de Gremlin para consultas repetidas de `emit.times`.
- Se ha corregido un problema de Gremlin por el que `repeat.until` permitía el paso de soluciones de `.emit` que se debían haber filtrado.
- Se ha mejorado la administración de errores en Gremlin.

Actualizaciones del motor de Amazon Neptune del 19/11/2018

Versión: 1.0.1.0.200264.0

La versión 1.0.1.0.200264.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200264.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200264.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad.

A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Se agregó compatibilidad con the section called “Sugerencias de consulta.”.
- Mensajes de error mejorados para la autenticación de IAM. Para obtener más información, consulte the section called “Errores de IAM”.
- Mejora del rendimiento de las consultas SPARQL con un elevado número de predicados.
- Rendimiento mejorado de la ruta de propiedades SPARQL.
- Rendimiento de Gremlin mejorado para mutaciones condicionales, como el patrón `fold().coalesce(unfold(), …)`, cuando se usa con los pasos `addV()`, `addE()` y `property()`.
- Rendimiento de Gremlin mejorado para las modulaciones `by()` y `sack()`.
- Rendimiento de Gremlin mejorado para los pasos `group()` y `groupCount()`.
- Rendimiento de Gremlin mejorado para los pasos `store()`, `sideEffect()` y `cap().unfold()`.
- Compatibilidad mejorada para las restricciones de propiedades de cardinalidad única de Gremlin.
  - Mejora de la aplicación de la cardinalidad única para las propiedades de borde y las propiedades de vértice marcadas como propiedades de cardinalidad única.
  - Se producía un error si se especificaban valores de propiedad adicionales para una propiedad de borde existente durante los trabajos de carga de Neptune.

Actualizaciones del motor de Amazon Neptune del 08/11/2018

Versión: 1.0.1.0.200258.0

La versión 1.0.1.0.200258.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200258.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200258.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Se agregó compatibilidad con Sugerencias de consulta SPARQL.
- Rendimiento mejorado para consultas SPARQL FILTER (NOT) Exists.
- Rendimiento mejorado para consultas SPARQL DESCRIBE.
- Rendimiento mejorado para la repetición hasta el patrón en Gremlin.
- Rendimiento mejorado para añadir bordes en Gremlin.
- Se ha corregido un problema por el que las consultas SPARQL Update DELETE podían tener errores en algunos casos.
- Se ha corregido un problema para gestionar los tiempos de espera con el servidor de WebSocket de Gremlin.

Actualizaciones del motor de Amazon Neptune del 29/10/2018

Versión: 1.0.1.0.200255.0

La versión 1.0.1.0.200255.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200255.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200255.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Se ha añadido la información de la autenticación de IAM a los registros de auditoría.

- Se ha añadido compatibilidad con credenciales temporales usando roles de IAM y perfiles de instancia.
- Se ha añadido la terminación de la conexión WebSocket para la autenticación de IAM cuando se revoca el permiso o si se elimina el usuario o el rol de IAM.
- Se ha limitado el número máximo de conexiones WebSocket a 60.000 por instancia.
- Se ha mejorado el rendimiento de carga masiva para tipos de instancia más pequeños.
- Se ha mejorado el rendimiento de las consultas que incluyen los operadores `and()`, `or()`, `not()`, `drop()` en Gremlin.
- El analizador de NTriples ahora rechaza los URI no válidos, como los URI que contienen espacios en blanco.

Actualizaciones del motor de Amazon Neptune del 06/09/2018

Versión: 1.0.1.0.200237.0

La versión 1.0.1.0.200237.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200237.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200237.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Se ha corregido un problema por el que se producían errores en algunas consultas `SPARQL COUNT(DISTINCT)`.
- Se ha corregido un problema por el que se quedaban sin memoria las consultas `COUNT`, `SUM` y `MIN` con una cláusula `DISTINCT`.
- Se ha corregido un problema por el que los datos de tipo `BLOB` podían provocar un error en un trabajo del programa de carga de Neptune.
- Se ha corregido un problema por el que las inserciones duplicadas provocaban errores en las transacciones.
- Se ha corregido un problema por el que no se podían cancelar las consultas `DROP ALL`.
- Se ha corregido un problema por el que los clientes Gremlin podían bloquearse de forma intermitente.
- Se han actualizado todos los códigos de error de las cargas mayores de 150 M para que sean `HTTP 400`.
- Se ha mejorado el desempeño y la precisión de las consultas `COUNT()` de patrón único-triple.
- Se ha mejorado el desempeño de las consultas `SPARQL UNION` con cláusulas `BIND`.

Actualizaciones del motor de Amazon Neptune del 24/07/2018

Versión: 1.0.1.0.200236.0

La versión 1.0.1.0.200236.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200236.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200236.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Se ha actualizado la serialización SPARQL para el tipo de datos `xsd:string`. xsd:string ya no está incluido en la serialización JSON, que ahora es coherente con otros formatos de salida.
- Se ha corregido el tratamiento de infinito `xsd:double`/`xsd:float`. Los valores `-INF`, `NaN` e `INF` se reconocen y se gestionan ahora correctamente en todos los formatos de cargador de datos SPARQL, SPARQL 1.1 UPDATE y SPARQL 1.1 Query.
- Se ha corregido un problema donde una consulta de Gremlin con valores de cadena vacía produce un error de forma inesperada.

- Se ha corregido un problema donde `aggregate()` y `cap()` de Gremlin en un gráfico vacío produce un error de forma inesperada.
- Se ha corregido un problema en el que se devolvían respuestas de error incorrectas para Gremlin cuando la especificación de cardinalidad no es válida, por ejemplo, `.property(set,id,'10')` y `.property(single,id,'10')`.
- Se ha corregido un problema por el que se devolvía una sintaxis de Gremlin no válida como InternalFailureException.
- Se ha corregido la ortografía en `TimeLimitExceeededException` a `TimeLimitExceededException`, en los mensajes de error.
- Se han cambiado la respuesta de los puntos de enlace SPARQL y GREMLIN de forma coherente cuando no se suministra ningún script.
- Se han aclarado los mensajes de error para demasiadas solicitudes simultáneas.

Actualizaciones del motor de Amazon Neptune del 22/06/2018

Versión: 1.0.1.0.200233.0

La versión 1.0.1.0.200233.0 de Amazon Neptune está disponible con carácter general. Todos los clústeres de base de datos nuevos de Neptune, incluidos los restaurados a partir de instantáneas, se crearán en Neptune 1.0.1.0.200233.0 después de que haya finalizado la actualización del motor para esa región.

Los clústeres existentes pueden actualizarse a esta versión inmediatamente utilizando las operaciones de clúster de base de datos en la consola o mediante el SDK. Puede utilizar el siguiente comando de la CLI para actualizar inmediatamente un clúster de base de datos a esta versión:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Los clústeres de base de datos de Neptune se actualizarán automáticamente a la versión 1.0.1.0.200233.0 del motor durante los periodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de base de datos, así como del tipo de actualización.

> **Note**
>
> El periodo de mantenimiento de las instancias no se aplica a las actualizaciones del motor.

Las actualizaciones se aplican a todas las instancias en un clúster de base de datos simultáneamente. Una actualización requiere un reinicio de la base de datos en todas las instancias de un clúster de base de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. A continuación, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la consola de Neptune.

Si tiene alguna duda, el equipo de AWS Support está disponible en los foros de la comunidad y a través de AWS Premium Support.

Mejoras

- Se ha corregido un problema por el que la emisión en rápida sucesión de un gran número de solicitudes de carga masiva da como resultado un error.
- Se ha corregido un problema dependiente de los datos por el que una consulta podría generar un InternalServerError. En el siguiente ejemplo, se muestra el tipo de consulta afectado.

```
g.V("my-id123").as("start").outE("knows").has("edgePropertyKey1",
 P.gt(0)).as("myedge").inV()
               .as("end").select("start", "end", "myedge").by("vertexPropertyKey1")
               .by("vertexPropertyKey1").by("edgePropertyKey1")
```

- Se ha corregido un problema por el que un cliente Java de Gremlin no puede conectarse con el servidor usando la misma conexión WebSocket una vez agotado el tiempo de espera de una consulta de ejecución prolongada.
- Se ha corregido un problema por el que las secuencias de escape incluidas como parte de la consulta de Gremlin a través de HTTP o las consultas basadas en cadenas sobre la conexión WebSocket no se procesaron correctamente.

Introducción al uso de las API de Amazon Neptune

Las API de administración de Amazon Neptune ofrecen compatibilidad con SDK para crear, administrar y eliminar instancias y clústeres de bases de datos de Neptune, mientras que las API de datos de Neptune ofrecen compatibilidad con SDK para cargar datos en el gráfico, ejecutar consultas, obtener información sobre los datos del gráfico y ejecutar operaciones de machine learning. Estas API están disponibles en todos los lenguajes del SDK. Al firmar automáticamente las solicitudes de la API, simplifican considerablemente la integración de Neptune en las aplicaciones.

En esta página se proporciona información sobre cómo usar estas API.

Acciones de IAM con nombres diferentes a los de sus homólogos del SDK de la API de datos de Neptune

Al llamar a un método de la API de Neptune en un clúster que tiene habilitada la autenticación de IAM, debe tener una política de IAM asociada al usuario o rol que realiza las llamadas y que proporcione permisos para las acciones que desea realizar. Los permisos se establecen en la política mediante las acciones de IAM correspondientes. También puede restringir las acciones que se pueden realizar mediante las claves de condición de IAM.

La mayoría de las acciones de IAM tienen el mismo nombre que los métodos de la API a los que corresponden, pero algunos métodos de la API de datos tienen nombres diferentes, ya que algunos los comparten más de un método. En la siguiente tabla se enumeran los métodos de datos y sus correspondientes acciones de IAM:

Nombre de la operación de la API de datos	Correspondencias con IAM
CancelGremlinQuery (cancel_gremlin_query)	Acción: `neptune-db: CancelQuery`
CancelLoaderJob (cancel_loader_job)	Acción: `neptune-db:CancelLoaderJob`
CancelMLDataProcessingJob (cancel_ml_data_processing_job)	Acción: `neptune-db:CancelMLDataProcessingJob`

Nombre de la operación de la API de datos	Correspondencias con IAM
CancelMLModelTrainingJob (cancel_ml_model_training_job)	Acción: `neptune-db:CancelMLModelTrainingJob`
CancelOpenCypherQuery (cancel_open_cypher_query)	Acción: `neptune-db:` **`CancelQuery`**
CreateMLEndpoint (create_ml_endpoint)	Acción: `neptune-db:CreateMLEndpoint`
DeleteMLEndpoint (delete_ml_endpoint)	Acción: `neptune-db:DeleteMLEndpoint`
DeletePropertygraphStatistics (delete_propertygraph_statistics)	Acción: `neptune-db:` **`DeleteStatistics`**
DeleteSparqlStatistics (delete_sparql_statistics)	Acción: `neptune-db:` **`DeleteStatistics`**
ExecuteFastReset execute_fast_reset()	Acción: `neptune-db:` **`ResetDatabase`**
ExecuteGremlinExplainQuery (execute_gremlin_explain_query)	Actions: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Clave de condición: `neptune-db:QueryLanguage:Gremlin`

Nombre de la operación de la API de datos	Correspondencias con IAM	
ExecuteGremlinProfileQuery (execute_gremlin_profile_query)	Acción: `neptune-db: ReadDataViaQuery` Clave de condición: `neptune-db:QueryLanguage:Gremlin`	
ExecuteGremlinQuery (execute_gremlin_query)	Actions: • `neptune-db: ReadDataViaQuery` • `neptune-db: WriteDataViaQuery` • `neptune-db: DeleteDataViaQuery` Clave de condición: `neptune-db:QueryLanguage:Gremlin`	
ExecuteOpenCypherExplainQuery (execute_open_cypher_explain_query)	Acción: `neptune-db: ReadDataViaQuery` Clave de condición: `neptune-db:QueryLanguage:OpenCypher`	

Nombre de la operación de la API de datos	Correspondencias con IAM	
ExecuteOpenCypherQuery (execute_open_cypher_query)	Actions: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Clave de condición: `neptune-db:QueryLanguage:OpenCypher`	
GetEngineStatus (get_engine_status)	Acción: `neptune-db:GetEngineStatus`	
GetGremlinQueryStatus (get_gremlin_query_status)	Acción: `neptune-db:` **`:GetQueryStatus`** Clave de condición: `neptune-db:QueryLanguage:Gremlin`	
GetLoaderJobStatus (get_loader_job_status)	Acción: `neptune-db:GetLoaderJobStatus`	
GetMLDataProcessingJob (get_ml_data_processing_job)	Acción: `neptune-db:` **`GetMLDataProcessingJobStatus`**	
GetMLEndpoint (get_ml_endpoint)	Acción: `neptune-db:` **`GetMLEndpointStatus`**	

Nombre de la operación de la API de datos	Correspondencias con IAM	
GetMLModelTrainingJob (get_ml_model_training_job)	Acción: `neptune-db:` **`GetMLModelTrainingJobStatus`**	
GetMLModelTransformJob (get_ml_model_transform_job)	Acción: `neptune-db:` **`GetMLModelTransformJobStatus`**	
GetOpenCypherQueryStatus (get_open_cypher_query_status)	Acción: `neptune-db:` **`:GetQueryStatus`** Clave de condición: `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphStatistics (get_propertygraph_statistics)	Acción: `neptune-db:` **`GetStatisticsStatus`**	
GetPropertygraphStream (get_propertygraph_stream)	Acción: `neptune-db:` **`GetStreamRecords`** Claves de condición: • `neptune-db:QueryLanguage:Gremlin` • `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphSummary (get_propertygraph_summary)	Acción: `neptune-db:` **`GetGraphSummary`**	
GetRDFGraphSummary (get_rdf_graph_summary)	Acción: `neptune-db:` **`GetGraphSummary`**	

Nombre de la operación de la API de datos	Correspondencias con IAM	
GetSparqlStatistics (get_sparql_statistics)	Acción: `neptune-db: GetStatisticsStatus`	
GetSparqlStream (get_sparql_stream)	Acción: `neptune-db: GetStreamRecords` Clave de condición: `neptune-db:QueryLanguage:Sparql`	
ListGremlinQueries (list_gremlin_queries)	Acción: `neptune-db: GetQueryStatus` Clave de condición: `neptune-db:QueryLanguage:Gremlin`	
ListMLEndpoints (list_ml_endpoints)	Acción: `neptune-db:ListMLEndpoints`	
ListMLModelTrainingJobs (list_ml_model_training_jobs)	Acción: `neptune-db:ListMLModelTrainingJobs`	
ListMLModelTransformJobs (list_ml_model_transform_jobs)	Acción: `neptune-db:ListMLModelTransformJobs`	
ListOpenCypherQueries (list_open_cypher_queries)	Acción: `neptune-db: GetQueryStatus` Clave de condición: `neptune-db:QueryLanguage:OpenCypher`	

Nombre de la operación de la API de datos	Correspondencias con IAM
ManagePropertygraphStatistics (manage_propertygraph_statistics)	Acción: `neptune-db: ManageStatistics`
ManageSparqlStatistics (manage_sparql_statistics)	Acción: `neptune-db: ManageStatistics`
StartLoaderJob (start_loader_job)	Acción: `neptune-db:StartLoaderJob`
StartMLModelDataProcessingJob (start_ml_data_processing_job)	Acción: `neptune-db:StartMLModelDataProcessingJob`
StartMLModelTrainingJob (start_ml_model_training_job)	Acción: `neptune-db:StartMLModelTrainingJob`
StartMLModelTransformJob (start_ml_model_transform_job)	Acción: `neptune-db:StartMLModelTransformJob`

Referencia de la API de administración de Amazon Neptune

En este capítulo se documentan las operaciones de las API de Neptune que puede utilizar para administrar y mantener el clúster de base de datos de Neptune.

Neptune funciona en clústeres de servidores de base de datos que están conectados en una topología de replicación. Por lo tanto, la administración de Neptune suele requerir la implementación de cambios en varios servidores y asegurarse de que todas las réplicas de Neptune mantengan el ritmo del servidor principal.

Dado que Neptune escala de manera transparente el almacenamiento subyacente a medida que crecen sus datos, la administración de Neptune requiere relativamente poco esfuerzo administrativo del almacenamiento en disco. Del mismo modo, dado que Neptune realiza automáticamente copias de seguridad continuas, el clúster de Neptune no requiere una planificación extensa ni tiempo de inactividad a la hora de realizarlas.

Contenido

- CreateDBClusterParameterGroup (acción)
- DeleteDBParameterGroup (acción)
- DeleteDBClusterParameterGroup (acción)
- ModifyDBParameterGroup (acción)
- ModifyDBClusterParameterGroup (acción)
- ResetDBParameterGroup (acción)
- ResetDBClusterParameterGroup (acción)
- DescribeDBParameters (acción)
- DescribeDBParameterGroups (acción)
- DescribeDBClusterParameters (acción)
- DescribeDBClusterParameterGroups (acción)
- DescribeEngineDefaultParameters (acción)
- DescribeEngineDefaultClusterParameters (acción)
- Estructuras:
- Parameter (estructura)
- DBParameterGroup (estructura)
- DBClusterParameterGroup (estructura)
- DBParameterGroupStatus (estructura)

- API de subred de Neptune
  - CreateDBSubnetGroup (acción)
  - DeleteDBSubnetGroup (acción)
  - ModifyDBSubnetGroup (acción)
  - DescribeDBSubnetGroups (acción)
  - Estructuras:
  - Subred (estructura)
  - DBSubnetGroup (estructura)
- API de instantáneas de Neptune
  - CreateDBClusterSnapshot (acción)
  - DeleteDBClusterSnapshot (acción)
  - CopyDBClusterSnapshot (acción)

- ModifyDBClusterSnapshotAttribute (acción)
- RestoreDBClusterFromSnapshot (acción)
- RestoreDBClusterToPointInTime (acción)
- DescribeDBClusterSnapshots (acción)
- DescribeDBClusterSnapshotAttributes (acción)
- Estructuras:
- DBClusterSnapshot (estructura)
- DBClusterSnapshotAttribute (estructura)
- DBClusterSnapshotAttributesResult (estructura)

- API de eventos de Neptune
  - CreateEventSubscription (acción)
  - DeleteEventSubscription (acción)
  - ModifyEventSubscription (acción)
  - DescribeEventSubscriptions (acción)
  - AddSourceIdentifierToSubscription (acción)
  - RemoveSourceIdentifierFromSubscription (acción)
  - DescribeEvents (acción)
  - DescribeEventCategories (acción)
  - Estructuras:
  - Event (estructura)
  - EventCategoriesMap (estructura)
  - EventSubscription (estructura)
- Otras API de Neptune
  - AddTagsToResource (acción)
  - ListTagsForResource (acción)
  - RemoveTagsFromResource (acción)
  - ApplyPendingMaintenanceAction (acción)
  - DescribePendingMaintenanceActions (acción)
  - DescribeDBEngineVersions (acción)
  - Estructuras:

- DBClusterSnapshotAlreadyExistsFault (estructura)
- DBClusterSnapshotNotFoundFault (estructura)
- DBInstanceAlreadyExistsFault (estructura)
- DBInstanceNotFoundFault (estructura)
- DBLogFileNotFoundFault (estructura)
- DBParameterGroupAlreadyExistsFault (estructura)
- DBParameterGroupNotFoundFault (estructura)
- DBParameterGroupQuotaExceededFault (estructura)
- DBSecurityGroupAlreadyExistsFault (estructura)
- DBSecurityGroupNotFoundFault (estructura)
- DBSecurityGroupNotSupportedFault (estructura)
- DBSecurityGroupQuotaExceededFault (estructura)
- DBSnapshotAlreadyExistsFault (estructura)
- DBSnapshotNotFoundFault (estructura)
- DBSubnetGroupAlreadyExistsFault (estructura)
- DBSubnetGroupDoesNotCoverEnoughAZs (estructura)
- DBSubnetGroupNotAllowedFault (estructura)
- DBSubnetGroupNotFoundFault (estructura)
- DBSubnetGroupQuotaExceededFault (estructura)
- DBSubnetQuotaExceededFault (estructura)
- DBUpgradeDependencyFailureFault (estructura)
- DomainNotFoundFault (estructura)
- EventSubscriptionQuotaExceededFault (estructura)
- GlobalClusterAlreadyExistsFault (estructura)
- GlobalClusterNotFoundFault (estructura)
- GlobalClusterQuotaExceededFault (estructura)
- InstanceQuotaExceededFault (estructura)
- InsufficientDBClusterCapacityFault (estructura)
- InsufficientDBInstanceCapacityFault (estructura)
- InsufficientStorageClusterCapacityFault (estructura)

- InvalidDBClusterEndpointStateFault (estructura)
- InvalidDBClusterSnapshotStateFault (estructura)
- InvalidDBClusterStateFault (estructura)
- InvalidDBInstanceStateFault (estructura)
- InvalidDBParameterGroupStateFault (estructura)
- InvalidDBSecurityGroupStateFault (estructura)
- InvalidDBSnapshotStateFault (estructura)
- InvalidDBSubnetGroupFault (estructura)
- InvalidDBSubnetGroupStateFault (estructura)
- InvalidDBSubnetStateFault (estructura)
- InvalidEventSubscriptionStateFault (estructura)
- InvalidGlobalClusterStateFault (estructura)
- InvalidOptionGroupStateFault (estructura)
- InvalidRestoreFault (estructura)
- InvalidSubnet (estructura)
- InvalidVPCNetworkStateFault (estructura)
- KMSKeyNotAccessibleFault (estructura)
- OptionGroupNotFoundFault (estructura)
- PointInTimeRestoreNotEnabledFault (estructura)
- ProvisionedIopsNotAvailableInAZFault (estructura)
- ResourceNotFoundFault (estructura)
- SNSInvalidTopicFault (estructura)
- SNSNoAuthorizationFault (estructura)
- SNSTopicArnNotFoundFault (estructura)
- SharedSnapshotQuotaExceededFault (estructura)
- SnapshotQuotaExceededFault (estructura)
- SourceNotFoundFault (estructura)
- StorageQuotaExceededFault (estructura)
- StorageTypeNotSupportedFault (estructura)
- SubnetAlreadyInUse (estructura)

- SubscriptionAlreadyExistFault (estructura)
- SubscriptionCategoryNotFoundFault (estructura)
- SubscriptionNotFoundFault (estructura)

API de clústeres de base de datos de Neptune

Acciones:

- CreateDBCluster (acción)
- DeleteDBCluster (acción)
- ModifyDBCluster (acción)
- StartDBCluster (acción)
- StopDBCluster (acción)
- AddRoleToDBCluster (acción)
- RemoveRoleFromDBCluster (acción)
- FailoverDBCluster (acción)
- PromoteReadReplicaDBCluster (acción)
- DescribeDBClusters (acción)

Estructuras:

- DBCluster (estructura)
- DBClusterMember (estructura)
- DBClusterRole (estructura)
- CloudwatchLogsExportConfiguration (estructura)
- PendingCloudwatchLogsExports (estructura)
- ClusterPendingModifiedValues (estructura)

CreateDBCluster (acción)

> El nombre de la AWS CLI para esta API es: `create-db-cluster`.

Crea un nuevo clúster de base de datos Amazon Neptune.

Puede utilizar el parámetro `ReplicationSourceIdentifier` para crear el clúster de base de datos como réplica de lectura de otro clúster de base de datos o una instancia de base de datos de Amazon Neptune.

Tenga en cuenta que cuando crea un nuevo clúster mediante `CreateDBCluster` directamente, la protección de eliminación está deshabilitada de forma predeterminada (cuando crea un nuevo clúster de producción en la consola, la protección de eliminación está habilitada de forma predeterminada). Solo puede eliminar un clúster de base de datos si su campo `DeletionProtection` está establecido en `false`.

Solicitud

- AvailabilityZones (en la CLI: `--availability-zones`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de zonas de disponibilidad de EC2 en el que las que se pueden crear instancias en el clúster de base de datos.
- BackupRetentionPeriod (en la CLI: `--backup-retention-period`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de días durante los que se retienen las copias de seguridad automatizadas. Debe especificar un valor mínimo de 1.

  Valor predeterminado: 1

  Restricciones:

  - Debe ser un valor entre 1 y 35.
- CopyTagsToSnapshot (en la CLI: `--copy-tags-to-snapshot`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.
- DatabaseName (en la CLI: `--database-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre para la base de datos de hasta 64 caracteres alfanuméricos. Si no proporciona un nombre, Amazon Neptune no creará una base de datos en el clúster de base de datos que está creando.

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identificador de clúster de base de datos. Este parámetro se almacena como una cadena en minúsculas.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

  Ejemplo: `my-cluster1`
- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros del clúster de base de datos que desea asociar a este clúster de base de datos. Si este argumento se omite, se utiliza el predeterminado.

  Restricciones:

  - Si se suministra, debe coincidir con el nombre de un DBClusterParameterGroup existente.
- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un grupo de subred de base de datos con el que asociar este clúster de base de datos.

  Limitaciones: debe coincidir con el nombre de un DBSubnetGroup existente. No debe ser predeterminado.

  Ejemplo: `mySubnetgroup`
- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si el clúster de base de datos tiene habilitada la protección contra eliminación. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación. La protección contra eliminación está habilitada de forma predeterminada.
- EnableCloudwatchLogsExports (en la CLI: `--enable-cloudwatch-logs-exports`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una lista de tipos de registro para los que este clúster de base de datos debe exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría) y `slowquery` (para publicar registros de consultas lentas). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- EnableIAMDatabaseAuthentication (en la CLI: `--enable-iam-database-authentication`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, habilita la autenticación de Amazon Identity and Access Management (IAM) para todo el clúster de base de datos (no se puede configurar en el nivel de instancia).

  Valor predeterminado: `false`.

- Engine (en la CLI: `--engine`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

  Valores válidos: `neptune`

- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Número de versión del motor de base de datos que se va a usar con el nuevo clúster de base de datos.

  Ejemplo: `1.2.1.0`

- GlobalClusterIdentifier (en la CLI:`--global-cluster-identifier`): un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  El ID de la base de datos global de Neptune a la que se debe añadir este nuevo clúster de base de datos.

- KmsKeyId (en la CLI: `--kms-key-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la clave de Amazon KMS de un clúster de base de datos cifrado.

  El identificador de la clave de KMS es el Nombre de recurso de Amazon (ARN) de la clave de cifrado de KMS. Si está creando un clúster de base de datos con la misma cuenta de Amazon a la

que pertenece la clave de cifrado de KMS utilizada para cifrar el clúster de base de datos nuevo, puede utilizar el alias de la clave de KMS en lugar del ARN para la clave de cifrado de KMS.

Si no se especifica una clave de cifrado en `KmsKeyId`:

- Si `ReplicationSourceIdentifier` identifica un origen de cifrado, Amazon Neptune utilizará la clave de cifrado que se utiliza para cifrar el origen. De lo contrario, Amazon Neptune utilizará su clave de cifrado predeterminada.
- Si el parámetro `StorageEncrypted` es "true" y no se especifica `ReplicationSourceIdentifier`, Amazon Neptune utilizará su clave de cifrado predeterminada.

Amazon KMS crea la clave de cifrado predeterminada para su cuenta de Amazon. Su cuenta de Amazon dispone de una clave de cifrado predeterminada diferente para cada región de Amazon.

Si crea una réplica de lectura de un clúster de base de datos cifrado en otra región de Amazon, debe configurar `KmsKeyId` en un ID de la clave de KMS que sea válido en la región de Amazon de destino. Esta clave se utiliza para cifrar la réplica de lectura en esa región de Amazon.

- Port (en la CLI: `--port`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de puerto en el que las instancias en el clúster de base datos aceptan conexiones.

  Valor predeterminado: `8182`
- PreferredBackupWindow (en la CLI: `--preferred-backup-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El intervalo de tiempo diario durante el que se crean las copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas con el parámetro `BackupRetentionPeriod`.

  El valor predeterminado es un periodo de 30 minutos seleccionado al azar dentro de un bloque de 8 horas para cada región de Amazon. Para ver los bloques de tiempo disponibles, consulte Periodo de mantenimiento de Neptune en la Guía del usuario de Amazon Neptune.

  Restricciones:

  - Tiene que tener el formato `hh24:mi-hh24:mi`.
  - Debe indicarse en Tiempo universal coordinado (UTC).
  - No debe entrar en conflicto con la ventana de mantenimiento preferida.

  - Debe durar al menos 30 minutos.
- PreferredMaintenanceWindow (en la CLI: `--preferred-maintenance-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en tiempo universal coordinado (UTC).

  Formato: `ddd:hh24:mi-ddd:hh24:mi`

  El valor predeterminado es un periodo de 30 minutos seleccionado al azar de un bloque de 8 horas de tiempo para cada región de Amazon, que tiene lugar un día de la semana de forma aleatoria. Para ver los bloques de tiempo disponibles, consulte Periodo de mantenimiento de Neptune en la Guía del usuario de Amazon Neptune.

  Días válidos: lunes, martes, miércoles, jueves, viernes, sábado, domingo.

  Restricciones: plazo mínimo de 30 minutos.
- PreSignedUrl (en la CLI: `--pre-signed-url`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Este parámetro es incompatible en estos momentos.
- ReplicationSourceIdentifier (en la CLI: `--replication-source-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instancia de base de datos de origen o el clúster de base de datos si este clúster de base de datos se crea como réplica de lectura.
- ServerlessV2ScalingConfiguration (en la CLI: `--serverless-v2-scaling-configuration`): un objeto ServerlessV2ScalingConfiguration.

  Incluye la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.
- StorageEncrypted (en la CLI: `--storage-encrypted`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType (en la CLI: `--storage-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento para el nuevo clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) configura un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño. Si se establece en `standard`, el tipo de almacenamiento no aparece en la respuesta.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

    El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.
- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas para asignar al nuevo clúster de base de datos.
- VpcSecurityGroupIds (en la CLI: `--vpc-security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de grupos de seguridad de VPC de EC2 para asociar a este clúster de base de datos.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de

base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.

- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.

- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.

- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.

- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.

- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.

- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.

- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.

- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).

- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.

- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.

- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.
- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.
- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.
- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.
- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.
- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.
- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.
- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.
- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.
- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.
- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.
- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.
- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.
- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.
- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.
- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).
- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.
- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.
- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.
- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.
- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.
- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.
- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

- **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
- **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

  El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterAlreadyExistsFault
- InsufficientStorageClusterCapacityFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBClusterStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- InvalidDBInstanceStateFault
- DBClusterParameterGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterNotFoundFault
- DBInstanceNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DeleteDBCluster (acción)

El nombre de la AWS CLI para esta API es: `delete-db-cluster`.

La acción DeleteDBCluster elimina un clúster de base de datos aprovisionado previamente. Al eliminar un clúster de base de datos, se eliminan todas las copias de seguridad automatizadas para ese clúster de base de datos y no se pueden recuperar. Las instantáneas manuales del clúster de base de datos del clúster de base de datos especificado no se eliminan.

Tenga en cuenta que el clúster de base de datos no se puede eliminar si la protección de eliminación está habilitada. Para eliminarlo, primero debe establecer el campo `DeletionProtection` en `False`.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del clúster de base de datos del clúster de base de datos que se va a eliminar. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Debe coincidir con un DBClusterIdentifier existente.
- FinalDBSnapshotIdentifier (en la CLI: `--final-db-snapshot-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de instantánea de clúster de base de datos de la nueva instantánea del clúster de base de datos creada al configurar `SkipFinalSnapshot` como `false`.

  > **Note**
  >
  > Si se especifica este parámetro y también se establece el parámetro `SkipFinalShapshot` en "true", se producirá un error.

  Restricciones:

  - Debe tener de 1 a 255 letras, números o guiones.
  - El primer carácter debe ser una letra
  - No puede terminar por un guion ni contener dos guiones consecutivos

- SkipFinalSnapshot (en la CLI: `--skip-final-snapshot`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Determina si se crea una instantánea de clúster de base de datos antes de que se elimine el clúster de base de datos. Si se especifica `true`, no se crea ninguna instantánea del clúster de base de datos. Si se especifica `false`, se crea una instantánea de clúster de base de datos antes de que se elimine el clúster de base de datos.

  > **Note**
  >
  > Debe especificar un parámetro `FinalDBSnapshotIdentifier` si `SkipFinalSnapshot` es `false`.

  Valor predeterminado: `false`

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.
- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.

- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.
- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.
- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).
- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.
- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.
- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.
- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.
- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.
- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.
- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.
- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.
- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).
- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- DBClusterSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

ModifyDBCluster (acción)

El nombre de la AWS CLI para esta API es: `modify-db-cluster`.

Modificar una configuración para un clúster de base de datos. Puede cambiar uno o varios parámetros de configuración de la base de datos mediante la especificación de estos parámetros y los nuevos valores en la solicitud.

Solicitud

- AllowMajorVersionUpgrade (en la CLI: `--allow-major-version-upgrade`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si se permiten actualizaciones entre distintas versiones principales.

  Restricciones: debe establecer el indicador allow-major-version-upgrade al proporcionar un parámetro `EngineVersion` que utiliza una versión principal diferente de la versión actual del clúster de base de datos.
- ApplyImmediately (en la CLI: `--apply-immediately`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

Un valor que especifica si las modificaciones de esta solicitud y todas las modificaciones pendientes se asignan de manera asincrónica en cuanto sea posible, independientemente del valor de `PreferredMaintenanceWindow` del clúster de base de datos. Si este parámetro es `false`, los cambios realizados en el clúster de base de datos se aplican durante la siguiente ventana de mantenimiento.

El parámetro `ApplyImmediately` solo afecta a los valores `NewDBClusterIdentifier`. Si establece el valor del parámetro `ApplyImmediately` en False, los cambios en los valores `NewDBClusterIdentifier` se aplicarán durante el siguiente periodo de mantenimiento. Todas los demás cambios se aplican de inmediato, con independencia del valor del parámetro `ApplyImmediately`.

Valor predeterminado: `false`

- BackupRetentionPeriod (en la CLI: `--backup-retention-period`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de días durante los que se retienen las copias de seguridad automatizadas. Debe especificar un valor mínimo de 1.

  Valor predeterminado: 1

  Restricciones:

  - Debe ser un valor entre 1 y 35.
- CloudwatchLogsExportConfiguration (en la CLI: `--cloudwatch-logs-export-configuration`): un objeto CloudwatchLogsExportConfiguration.

  La opción de configuración para los tipos de registro que debe habilitarse para la exportación a CloudWatch Logs para un clúster de base de datos específico. Consulte Uso de la CLI para publicar registros de auditoría de Neptune en los regisgtros de CloudWatch.
- CopyTagsToSnapshot (en la CLI: `--copy-tags-to-snapshot`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.
- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El identificador del clúster de base de datos del clúster que se va a modificar. Este parámetro no distingue entre mayúsculas y minúsculas.

Restricciones:

- Debe coincidir con el identificador de un DBCluster existente.

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros del clúster de base de datos que se va a usar para el clúster de base de datos.

- DBInstanceParameterGroupName (en la CLI: `--db-instance-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de base de datos que se aplicará a todas las instancias del clúster de base de datos.

  > **Note**
  >
  > Al aplicar un grupo de parámetros mediante `DBInstanceParameterGroupName`, los cambios de parámetros no se aplicarán durante el siguiente periodo de mantenimiento, sino que se aplicarán inmediatamente.

  Predeterminado: la configuración de nombre existente

  Restricciones:

  - El grupo de parámetros de base de datos debe estar en la misma familia de grupos de parámetros de base de datos que la versión de clúster de base de datos de destino.
  - El parámetro `DBInstanceParameterGroupName` solo es válido en combinación con el parámetro `AllowMajorVersionUpgrade`.

- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si el clúster de base de datos tiene habilitada la protección contra eliminación. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación. La protección contra eliminación está deshabilitada de forma predeterminada.

- EnableIAMDatabaseAuthentication (en la CLI: `--enable-iam-database-authentication`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  True para habilitar el mapeo de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos; de lo contrario, el valor es False.

  Valor predeterminado: `false`

- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El número de versión del motor de base de datos al que desea realizar la actualización. El cambio de este parámetro produce una interrupción. El cambio se aplicará durante el siguiente período de mantenimiento, salvo que el parámetro `ApplyImmediately` esté establecido en "true".

  Para obtener una lista de las versiones válidas del motor, consulte Versiones del motor de Amazon Neptune o llame a the section called “DescribeDBEngineVersions”.

- NewDBClusterIdentifier (en la CLI: `--new-db-cluster-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nuevo identificador del clúster de base de datos cuando se cambia el nombre de un clúster de base de datos. Este valor se almacena como una cadena en minúsculas.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra
  - No puede terminar por un guion ni contener dos guiones consecutivos

  Ejemplo: `my-cluster2`

- Port (en la CLI: `--port`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de puerto en el que el clúster de base de datos acepta las conexiones.

  Limitaciones: el valor debe ser `1150-65535`

  Valor predeterminado: el mismo puerto que el clúster de base de datos original.

- PreferredBackupWindow (en la CLI: `--preferred-backup-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El intervalo de tiempo diario durante el que se crean las copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas con el parámetro `BackupRetentionPeriod`.

El valor predeterminado es un periodo de 30 minutos seleccionado al azar dentro de un bloque de 8 horas para cada región de Amazon.

Restricciones:

- Tiene que tener el formato `hh24:mi-hh24:mi`.
- Debe indicarse en Tiempo universal coordinado (UTC).
- No debe entrar en conflicto con la ventana de mantenimiento preferida.
- Debe durar al menos 30 minutos.

- PreferredMaintenanceWindow (en la CLI: `--preferred-maintenance-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en tiempo universal coordinado (UTC).

  Formato: `ddd:hh24:mi-ddd:hh24:mi`

  El valor predeterminado es un periodo de 30 minutos seleccionado al azar de un bloque de 8 horas de tiempo para cada región de Amazon, que tiene lugar un día de la semana de forma aleatoria.

  Días válidos: lunes, martes, miércoles, jueves, viernes, sábado, domingo.

  Restricciones: plazo mínimo de 30 minutos.

- ServerlessV2ScalingConfiguration (en la CLI: `--serverless-v2-scaling-configuration`): un objeto ServerlessV2ScalingConfiguration.

  Incluye la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- StorageType (en la CLI: `--storage-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se va a asociar al clúster de base de datos.

Valores válidos:

- **`standard`**: (predeterminado) configura un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
- **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

  El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroupIds (en la CLI: `--vpc-security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.

- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.

- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.

- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.

- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.
- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.
- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).
- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.
- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.
- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.
- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.
- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.
- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.
- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.
- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.
- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).
- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- DBClusterParameterGroupNotFoundFault
- InvalidDBSecurityGroupStateFault
- InvalidDBInstanceStateFault
- DBClusterAlreadyExistsFault
- StorageTypeNotSupportedFault

StartDBCluster (acción)

El nombre de la AWS CLI para esta API es: `start-db-cluster`.

Inicia un clúster de base de datos de Amazon Neptune que se detuvo con la consola de Amazon, el comando de la CLI de Amazon stop-db-cluster o la API StopDBCluster.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El identificador del clúster de base de datos del clúster de base de datos de Neptune que se va a iniciar. Este parámetro se almacena como una cadena en minúsculas.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.
- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.
- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.

- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.

- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.

- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).

- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.

- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.

- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.

- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.

- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.

- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.

- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.

- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

  Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

    El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterNotFoundFault

- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

StopDBCluster (acción)

El nombre de la AWS CLI para esta API es: `stop-db-cluster`.

Detiene un nuevo clúster de base de datos Amazon Neptune. Cuando se detiene un clúster de base de datos, Neptune conserva los metadatos del clúster de base de datos, incluidos sus puntos de enlace y grupos de parámetros de base de datos.

Neptune también conserva los registros de transacciones para que se pueda realizar una restauración a un momento dado si es necesario.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del clúster de base de datos del clúster de base de datos de Neptune que se va a detener. Este parámetro se almacena como una cadena en minúsculas.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called “DescribeDBClusters”.

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.

- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.
- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.
- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.
- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).
- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.
- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.
- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.

- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.
- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.
- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.
- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.
- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.
- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.
- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).
- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que

están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.

- **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

  El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.
- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

AddRoleToDBCluster (acción)

El nombre de la AWS CLI para esta API es: `add-role-to-db-cluster`.

Asocia un rol de Identity and Access Management (IAM) con un clúster de base de datos de Neptune.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del clúster de base de datos con el que asociar el rol de IAM.
- FeatureName (en la CLI: `--feature-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la característica del clúster de base de datos de Neptune con el que asociar el rol de IAM. Para ver la lista de nombres de características admitidos, consulte the section called “DBEngineVersion”.
- RoleArn (en la CLI: `--role-arn`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) del rol de IAM que se va a asociar al clúster de base de datos de Neptune, por ejemplo `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Respuesta

- Sin parámetros de respuesta.

Errores

- DBClusterNotFoundFault
- DBClusterRoleAlreadyExistsFault
- InvalidDBClusterStateFault
- DBClusterRoleQuotaExceededFault

RemoveRoleFromDBCluster (acción)

El nombre de la AWS CLI para esta API es: `remove-role-from-db-cluster`.

Disocia un rol de Identity and Access Management (IAM) de un clúster de base de datos.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del clúster de base de datos del que disociar el rol de IAM.
- FeatureName (en la CLI: `--feature-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la característica del clúster de base de datos del que se va a anular la asociación del rol de IAM. Para ver la lista de nombres de características admitidos, consulte the section called “DescribeDBEngineVersions”.
- RoleArn (en la CLI: `--role-arn`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del rol de IAM que se va a disasociar del clúster de base de datos, por ejemplo `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Respuesta

- Sin parámetros de respuesta.

Errores

- DBClusterNotFoundFault
- DBClusterRoleNotFoundFault
- InvalidDBClusterStateFault

FailoverDBCluster (acción)

El nombre de la AWS CLI para esta API es: `failover-db-cluster`.

Fuerza una conmutación por error para un clúster de base de datos.

Una conmutación por error de un clúster de base de datos promueve una de las réplicas de lectura (instancias de solo lectura) del clúster de base de datos a instancia principal (la instancia de escritura del clúster).

Amazon Neptune conmuta automáticamente a una réplica de lectura, si existe, cuando se produce un error en la instancia principal. Puede forzar una conmutación por error cuando desee simular un error en una instancia principal para realizar pruebas. Como cada instancia de un clúster de base de datos tiene su propia dirección de punto de enlace, tendrá que eliminar y restablecer las conexiones existentes que utilizan dichas direcciones de punto de enlace cuando se haya completado la conmutación por error.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador del clúster de base de datos para forzar una conmutación por error. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Debe coincidir con el identificador de un DBCluster existente.
- TargetDBInstanceIdentifier (en la CLI: `--target-db-instance-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El nombre de la instancia que se va a promover a instancia principal.

Debe especificar el identificador de instancias para una réplica de lectura del clúster de base de datos. Por ejemplo, `mydbcluster-replica1`.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.
- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.
- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.

- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.

- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.

- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).

- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.

- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.

- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.

- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.

- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.

- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.

- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.

- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

  Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

    El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterNotFoundFault

- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

PromoteReadReplicaDBCluster (acción)

El nombre de la AWS CLI para esta API es: `promote-read-replica-db-cluster`.

No admitido.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.
- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.

- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.

- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.

- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.

- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.

- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.

- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).

- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.

- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.

- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.

- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.
- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.
- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.
- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.
- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.
- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).
- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterNotFoundFault
- InvalidDBClusterStateFault

DescribeDBClusters (acción)

El nombre de la AWS CLI para esta API es: `describe-db-clusters`.

Devuelve información sobre clústeres de base de datos aprovisionados y admite la paginación.

> **Note**
>
> Esta operación también puede devolver información para clústeres de Amazon RDS y clústeres de Amazon DocDB.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de clúster de base de datos suministrado por el usuario. Si se especifica este parámetro, se devuelve información solo del clúster de base de datos específico. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Si se suministra, debe coincidir con un DBClusterIdentifier existente.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Un filtro que especifica uno o varios clústeres de bases de datos para describir.

Filtros compatibles:

- `db-cluster-id`: acepta identificadores de clúster de base de datos y Nombres de recursos de Amazon (ARN). La lista de resultados solo incluirá información sobre los clústeres de base de datos identificados por estos ARN.
- `engine` - Acepta un nombre de motor (como `neptune`), y restringe la lista de resultados a clústeres de base de datos creados por ese motor.

Por ejemplo, para invocar esta API desde la CLI de Amazon y filtrar de modo que solo se devuelvan los clústeres de base de datos de Neptune, puede utilizar el siguiente comando:

Example

```
aws neptune describe-db-clusters \
            --filters  Name=engine,Values=neptune
```

- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud the section called "DescribeDBClusters" anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se deben incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Mínimo 20, máximo 100.

Respuesta

- DBClusters: matriz de objetos DBCluster.

  Contiene una lista de los clústeres de base de datos para el usuario.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un token de paginación que se pueda utilizar en una solicitud DescribeDBClusters posterior.

Errores

- DBClusterNotFoundFault

Estructuras:

DBCluster (estructura)

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

Campos

- AllocatedStorage: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: se trata de una matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.
- AutomaticRestartTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.
- AvailabilityZones: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.

- BacktrackConsumedChangeRecords: se trata de un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BacktrackWindow: se trata de un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BackupRetentionPeriod: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- Capacity: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.
- CloneGroupId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.
- ClusterCreateTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).
- CopyTagsToSnapshot: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.
- CrossAccountClone: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.
- DatabaseName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.

- DBClusterArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: se trata de una matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.
- Endpoint: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.
- Engine: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.
- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.
- GlobalClusterIdentifier: se trata de un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.
- HostedZoneId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.
- IAMDatabaseAuthenticationEnabled: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).
- IOOptimizedNextAllowedModificationTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- MultiAZ: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.

- PendingModifiedValues: se trata de un objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.

- PercentProgress: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.

- Port: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.

- PreferredBackupWindow: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- ReaderEndpoint: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ReplicationType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: se trata de un objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El tipo de almacenamiento que se utiliza para el clúster de base de datos.

Valores válidos:

- **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
- **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

  El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: se trata de una matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

`DBCluster` se utiliza como el elemento de respuesta para:

- CreateDBCluster
- DeleteDBCluster
- FailoverDBCluster
- ModifyDBCluster
- PromoteReadReplicaDBCluster
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDBCluster
- StopDBCluster

DBClusterMember (estructura)

Contiene información sobre una instancia que forma parte de un clúster de base de datos.

Campos

- DBClusterParameterGroupStatus: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado del grupo de parámetros del clúster de base de datos para este miembro del clúster de base de datos.
- DBInstanceIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador de la instancia de este miembro del clúster de base de datos.
- IsClusterWriter: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  El valor que es `true` si el miembro del clúster es la instancia principal del clúster de base de datos y `false` en caso contrario.
- PromotionTier: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueve una réplica de lectura a la instancia principal tras un error de la instancia principal existente.

DBClusterRole (estructura)

Describe un rol de Amazon Identity and Access Management (IAM) que está asociado con un clúster de base de datos.

Campos

- FeatureName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la característica asociada al rol de Amazon Identity and Access Management (IAM). Para ver la lista de nombres de características admitidos, consulte the section called “DescribeDBEngineVersions”.
- RoleArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del rol de IAM asociado al clúster de base de datos.
- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Describe el estado de asociación entre el rol de IAM y el clúster de base de datos. La propiedad del estado devuelve uno de los siguientes valores:

- `ACTIVE`: el ARN del rol de IAM se asocia con el clúster de base de datos y se puede utilizar para obtener acceso a otros servicios de Amazon en su nombre.
- `PENDING`: el ARN del rol de IAM se está asociando al clúster de base de datos.
- `INVALID`: el ARN del rol de IAM se asocia con el clúster de base de datos, pero el clúster de base de datos no puede asumir el rol de IAM para acceder a otros servicios de Amazon en su nombre.

CloudwatchLogsExportConfiguration (estructura)

La opción de configuración para los tipos de registro que se debe habilitar para la exportación a CloudWatch Logs para una instancia de base de datos específica o un clúster de base de datos.

Las matrices `EnableLogTypes` y `DisableLogTypes` determinan qué registros se exportarán (o no) a CloudWatch Logs.

Los tipos de registro válidos son: `audit` (para publicar registros de auditoría) y `slowquery` (para publicar registros de consultas lentas). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

Campos

- DisableLogTypes: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La lista de tipos de registros que desea deshabilitar.
- EnableLogTypes: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La lista de tipos de registros que desea habilitar.

PendingCloudwatchLogsExports (estructura)

Una lista de los tipos de registro cuya configuración sigue pendiente. En otras palabras, estos tipos de registro se están activando o desactivando.

Los tipos de registro válidos son: `audit` (para publicar registros de auditoría) y `slowquery` (para publicar registros de consultas lentas). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

Campos

- LogTypesToDisable: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Tipos de registro que están en proceso de habilitarse. Una vez habilitados, estos tipos de registro se exportan a CloudWatch Logs.
- LogTypesToEnable: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Tipos de registro que están en proceso de desactivación. Una vez desactivados, estos tipos de registro no se exportan a CloudWatch Logs.

ClusterPendingModifiedValues (estructura)

Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.

Campos

- AllocatedStorage: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El tamaño de almacenamiento asignado en gibibytes (GiB) para los motores de base de datos. Para Neptune, `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- BackupRetentionPeriod: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de días durante los que se conservan las instantáneas de base de datos automáticas.
- DBClusterIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El valor DBClusterIdentifier del clúster de base de datos.
- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de base de datos.
- IAMDatabaseAuthenticationEnabled: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

Un valor que indica si se habilita el mapeo de cuentas de AWS Identity and Access Management (IAM) a la base de datos.

- Iops: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El valor de las IOPS aprovisionadas (las operaciones de E/S por segundo). Este ajuste es solo para clústeres de base de datos Multi-AZ.

- PendingCloudwatchLogsExports: se trata de un objeto PendingCloudwatchLogsExports.

  Esta estructura `PendingCloudwatchLogsExports` especifica los cambios pendientes en los que los registros de CloudWatch están habilitados y cuáles están deshabilitados.

- StorageType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El cambio pendiente en el tipo de almacenamiento del clúster de base de datos. Valores válidos:

  - **`standard`**: (predeterminado) configura un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

    El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

API de base de datos global de Neptune

Actions:

- CreateGlobalCluster (acción)
- DeleteGlobalCluster (acción)
- ModifyGlobalCluster (acción)
- DescribeGlobalClusters (acción)
- FailoverGlobalCluster (acción)
- RemoveFromGlobalCluster (acción)

Estructuras:

- GlobalCluster (estructura)
- GlobalClusterMember (estructura)

CreateGlobalCluster (acción)

El nombre de la AWS CLI para esta API es: `create-global-cluster`.

Crea una base de datos global de Neptune que abarca varias regiones de Amazon. La base de datos global incluye un único clúster principal con capacidad de lectura-escritura y clústeres secundarios de solo lectura que reciben datos del clúster principal mediante replicación de alta velocidad realizada por el subsistema de almacenamiento de Neptune.

Puede crear una base de datos global que inicialmente esté vacía y, a continuación, añadirle un clúster principal y clústeres secundarios, o puede especificar un clúster de Neptune existente durante la operación de creación para que se convierta en el clúster principal de la base de datos global.

Solicitud

- DatabaseName (en la CLI: `--database-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre para la nueva base de datos global (hasta 64 caracteres alfanuméricos).
- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de protección contra eliminación para la nueva base de datos global. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.
- Engine (en la CLI: `--engine`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del motor de base de datos que se debe utilizar en la base de datos global.

  Valores válidos: `neptune`
- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de Neptune que utilizará la base de datos global.

  Valores válidos: `1.2.0.0` o posterior.

- GlobalClusterIdentifier (en la CLI:`--global-cluster-identifier`): obligatorio un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Identificador de clúster del nuevo clúster de base de datos global.

- SourceDBClusterIdentifier (en la CLI: `--source-db-cluster-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  (Opcional) El nombre de recurso de Amazon (ARN) de un clúster de base de datos de Neptune existente para usarlo como clúster principal de la nueva base de datos global.

- StorageEncrypted (en la CLI: `--storage-encrypted`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de cifrado de almacenamiento para el nuevo clúster de base de datos global.

Respuesta

Incluye los detalles de una base de datos global de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en las acciones the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” y the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de protección contra eliminación para la nueva base de datos global.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El motor de base de datos de Neptune utilizado por la base de datos global (`"neptune"`).

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de Neptune utilizado por la base de datos global.

- GlobalClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la base de datos global.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.
- GlobalClusterMembers: matriz de objetos GlobalClusterMember.

  Una lista de los ARN de clústeres y los ARN de instancias de todos los clústeres de bases de datos que forman parte de la base de datos global.
- GlobalClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador inmutable para la base de datos global que es único en todas las regiones. Este identificador se encuentra en las entradas de registro de CloudTrail siempre que se accede a la clave de KMS para el clúster de base de datos.
- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos global.
- StorageEncrypted: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de cifrado de almacenamiento para la base de datos global.

Errores

- GlobalClusterAlreadyExistsFault
- GlobalClusterQuotaExceededFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault

DeleteGlobalCluster (acción)

> El nombre de la AWS CLI para esta API es: `delete-global-cluster`.

Elimina una de base de datos global. El clúster principal y todos los secundarios ya deben estar desconectados o eliminados.

Solicitud

- GlobalClusterIdentifier (en la CLI:`--global-cluster-identifier`): obligatorio un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  El identificador del clúster de la base de datos global que se está eliminando.

Respuesta

Incluye los detalles de una base de datos global de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en las acciones the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” y the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de protección contra eliminación para la nueva base de datos global.
- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El motor de base de datos de Neptune utilizado por la base de datos global (`"neptune"`).
- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de Neptune utilizado por la base de datos global.
- GlobalClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la base de datos global.
- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.
- GlobalClusterMembers: matriz de objetos GlobalClusterMember.

  Una lista de los ARN de clústeres y los ARN de instancias de todos los clústeres de bases de datos que forman parte de la base de datos global.
- GlobalClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un identificador inmutable para la base de datos global que es único en todas las regiones. Este identificador se encuentra en las entradas de registro de CloudTrail siempre que se accede a la clave de KMS para el clúster de base de datos.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos global.

- StorageEncrypted: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de cifrado de almacenamiento para la base de datos global.

Errores

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

ModifyGlobalCluster (acción)

El nombre de la AWS CLI para esta API es: `modify-global-cluster`.

Modifique una configuración para un clúster global de Amazon Neptune. Puede cambiar uno o varios parámetros de configuración de la base de datos mediante la especificación de estos parámetros y sus nuevos valores en la solicitud.

Solicitud

- AllowMajorVersionUpgrade (en la CLI: `--allow-major-version-upgrade`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Es un valor que indica si se permiten actualizaciones de las versiones principales.

  Restricciones: debe permitir las actualizaciones de versiones principales si especifica un valor para el parámetro `EngineVersion` que es una versión principal diferente de la versión actual del clúster de base de datos.

  Si actualiza la versión principal de una base de datos global, los grupos de parámetros del clúster y de la instancia de base de datos se configuran en los grupos de parámetros predeterminados de la nueva versión, por lo que tendrá que aplicar cualquier grupo de parámetros personalizado después de completar la actualización.

- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la base de datos global tiene habilitada la protección contra eliminación. La base de datos global no se puede eliminar cuando está habilitada la protección contra eliminación.

- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El número de versión del motor de base de datos al que desea realizar la actualización. El cambio de este parámetro provocará una interrupción. El cambio se aplicará durante el siguiente periodo de mantenimiento, salvo que el parámetro `ApplyImmediately` esté habilitado.

  Para ver una lista con todas las versiones del motor de Neptune disponibles, utilice el siguiente comando:

  Example

  ```
  aws neptune describe-db-engine-versions \
            --engine neptune \
            --query '*[]|[?SupportsGlobalDatabases == 'true'].[EngineVersion]'
  ```

- GlobalClusterIdentifier (en la CLI:`--global-cluster-identifier`): obligatorio un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  El identificador del clúster de base de datos del clúster global que se va a modificar. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones: debe coincidir con el identificador de un clúster de base de datos global existente.

- NewGlobalClusterIdentifier (en la CLI:`--new-global-cluster-identifier`): un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Un nuevo identificador de clúster para asignarlo a la base de datos global. Este valor se almacena como una cadena en minúsculas.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.

- No se pueden incluir dos guiones consecutivos ni acabar con guion.

Ejemplo: `my-cluster2`

Respuesta

Incluye los detalles de una base de datos global de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en las acciones the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” y the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de protección contra eliminación para la nueva base de datos global.
- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El motor de base de datos de Neptune utilizado por la base de datos global (`"neptune"`).
- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de Neptune utilizado por la base de datos global.
- GlobalClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la base de datos global.
- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.
- GlobalClusterMembers: matriz de objetos GlobalClusterMember.

  Una lista de los ARN de clústeres y los ARN de instancias de todos los clústeres de bases de datos que forman parte de la base de datos global.
- GlobalClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un identificador inmutable para la base de datos global que es único en todas las regiones. Este identificador se encuentra en las entradas de registro de CloudTrail siempre que se accede a la clave de KMS para el clúster de base de datos.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos global.

- StorageEncrypted: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de cifrado de almacenamiento para la base de datos global.

Errores

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DescribeGlobalClusters (acción)

El nombre de la AWS CLI para esta API es: `describe-global-clusters`.

Devuelve información sobre los clústeres de bases de datos globales de Neptune. Esta API admite la paginación.

Solicitud

- GlobalClusterIdentifier (en la CLI:`--global-cluster-identifier`): un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  El identificador de clúster de base de datos suministrado por el usuario. Si se especifica este parámetro, solo se devuelve información del clúster de base de datos especificado. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones: si se proporciona, debe coincidir con el identificador de un clúster de base de datos existente.

- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

(Opcional) Un token de paginación devuelto por una llamada anterior a `DescribeGlobalClusters`. Si se especifica este parámetro, la respuesta solo incluirá registros más allá del marcador, hasta el número especificado por `MaxRecords`.

- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de marcador paginación que puede utilizar para recuperar el resto de resultados.

  Valor predeterminado: `100`

  Restricciones: Minimum 20, máximo 100.

Respuesta

- GlobalClusters: matriz de objetos GlobalCluster.

  La lista de instancias y clústeres globales devuelta por esta solicitud.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación. Si se devuelve este parámetro en la respuesta, hay más registros disponibles, que se pueden recuperar mediante una o varias llamadas adicionales a `DescribeGlobalClusters`.

Errores

- GlobalClusterNotFoundFault

FailoverGlobalCluster (acción)

El nombre de la AWS CLI para esta API es: `failover-global-cluster`.

Inicia el proceso de conmutación por error para una base de datos global de Neptune.

Una conmutación por error para una base de datos global de Neptune convierte uno de los clústeres de base de datos secundarios de solo lectura en el clúster de base de datos principal y degrada el clúster de base de datos principal a un clúster de base de datos secundario (de solo lectura). Dicho

de otro modo, se cambia el rol del clúster de base de datos principal actual y del clúster de base de datos secundario de destino seleccionado. El clúster de base de datos secundario seleccionado asume todas las capacidades de lectura/escritura de la base de datos global de Neptune.

> **Note**
>
> Esta acción solo se aplica a las bases de datos globales de Neptune. Esta acción solo está pensada para usarse en bases de datos globales de Neptune en buen estado con clústeres de bases de datos de Neptune en buen estado y sin interrupciones en toda la región, para probar escenarios de recuperación de desastres o para volver a configurar la topología de la base de datos global.

Solicitud

- GlobalClusterIdentifier (en la CLI:`--global-cluster-identifier`): obligatorio un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Identificador de la base de datos global de Neptune que debe conmutarse por error. El identificador es la clave única asignada por el usuario cuando se creó la base de datos global de Neptune. Dicho de otro modo, es el nombre de la base de datos global que quiere conmutar por error.

  Restricciones: debe coincidir con el identificador de una base de datos global de Neptune existente.
- TargetDbClusterIdentifier (en la CLI: `--target-db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de recurso de Amazon (ARN) del clúster de base de datos secundario de Neptune que desea promover para que sea el principal de la base de datos global.

Respuesta

Incluye los detalles de una base de datos global de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en las acciones the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” y the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de protección contra eliminación para la nueva base de datos global.
- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El motor de base de datos de Neptune utilizado por la base de datos global (`"neptune"`).
- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de Neptune utilizado por la base de datos global.
- GlobalClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la base de datos global.
- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.
- GlobalClusterMembers: matriz de objetos GlobalClusterMember.

  Una lista de los ARN de clústeres y los ARN de instancias de todos los clústeres de bases de datos que forman parte de la base de datos global.
- GlobalClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador inmutable para la base de datos global que es único en todas las regiones. Este identificador se encuentra en las entradas de registro de CloudTrail siempre que se accede a la clave de KMS para el clúster de base de datos.
- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos global.
- StorageEncrypted: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de cifrado de almacenamiento para la base de datos global.

Errores

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

- InvalidDBClusterStateFault
- DBClusterNotFoundFault

RemoveFromGlobalCluster (acción)

El nombre de la AWS CLI para esta API es: `remove-from-global-cluster`.

Desconecta un clúster de base de datos de Neptune de una base de datos global de Neptune. Un clúster secundario se convierte en un clúster independiente normal con capacidad de lectura y escritura en lugar de ser de solo lectura, y ya no recibe datos del clúster principal.

Solicitud

- DbClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) que identifica el clúster que se va a desconectar del clúster de base de datos global de Neptune.
- GlobalClusterIdentifier (en la CLI:`--global-cluster-identifier`): obligatorio un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  El identificador de la base de datos global de Neptune de la que se va a desconectar el clúster de base de datos Neptune especificado.

Respuesta

Incluye los detalles de una base de datos global de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en las acciones the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” y the section called “RemoveFromGlobalCluster”.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de protección contra eliminación para la nueva base de datos global.
- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El motor de base de datos de Neptune utilizado por la base de datos global (`"neptune"`).

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de Neptune utilizado por la base de datos global.

- GlobalClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la base de datos global.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- GlobalClusterMembers: matriz de objetos GlobalClusterMember.

  Una lista de los ARN de clústeres y los ARN de instancias de todos los clústeres de bases de datos que forman parte de la base de datos global.

- GlobalClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador inmutable para la base de datos global que es único en todas las regiones. Este identificador se encuentra en las entradas de registro de CloudTrail siempre que se accede a la clave de KMS para el clúster de base de datos.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos global.

- StorageEncrypted: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de cifrado de almacenamiento para la base de datos global.

Errores

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- DBClusterNotFoundFault

Estructuras:

GlobalCluster (estructura)

Incluye los detalles de una base de datos global de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en las acciones the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” y the section called “RemoveFromGlobalCluster”.

Campos

- DeletionProtection: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de protección contra eliminación para la nueva base de datos global.
- Engine: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El motor de base de datos de Neptune utilizado por la base de datos global (`"neptune"`).
- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de Neptune utilizado por la base de datos global.
- GlobalClusterArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la base de datos global.
- GlobalClusterIdentifier: se trata de un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.
- GlobalClusterMembers: se trata de una matriz de objetos GlobalClusterMember.

  Una lista de los ARN de clústeres y los ARN de instancias de todos los clústeres de bases de datos que forman parte de la base de datos global.
- GlobalClusterResourceId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un identificador inmutable para la base de datos global que es único en todas las regiones. Este identificador se encuentra en las entradas de registro de CloudTrail siempre que se accede a la clave de KMS para el clúster de base de datos.

- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos global.

- StorageEncrypted: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  La configuración de cifrado de almacenamiento para la base de datos global.

`GlobalCluster` se utiliza como el elemento de respuesta para:

- CreateGlobalCluster
- ModifyGlobalCluster
- DeleteGlobalCluster
- RemoveFromGlobalCluster
- FailoverGlobalCluster

GlobalClusterMember (estructura)

Una estructura de datos con información sobre cualquier clúster principal y secundario asociado a una base de datos global de Neptune.

Campos

- DBClusterArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- IsWriter: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de Neptune es el clúster principal (es decir, tiene capacidad de lectura y escritura) de la base de datos global de Neptune a la que está asociado.

- Readers: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para cada clúster secundario de solo lectura asociado a la base de datos global de Neptune.

API de instancias de Neptune

Acciones:

- CreateDBInstance (acción)
- DeleteDBInstance (acción)
- ModifyDBInstance (acción)
- RebootDBInstance (acción)
- DescribeDBInstances (acción)
- DescribeOrderableDBInstanceOptions (acción)
- DescribeValidDBInstanceModifications (acción)

Estructuras:

- DBInstance (estructura)
- DBInstanceStatusInfo (estructura)
- OrderableDBInstanceOption (estructura)
- PendingModifiedValues (estructura)
- ValidStorageOptions (estructura)
- ValidDBInstanceModificationsMessage (estructura)

CreateDBInstance (acción)

El nombre de la AWS CLI para esta API es: `create-db-instance`.

Crea una nueva instancia de base de datos.

Solicitud

- AutoMinorVersionUpgrade (en la CLI: `--auto-minor-version-upgrade`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

Indica que las actualizaciones de motor secundarias se aplican automáticamente a la instancia de base de datos durante el intervalo de mantenimiento.

Valor predeterminado: `true`

- AvailabilityZone (en la CLI: `--availability-zone`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La zona de disponibilidad de EC2 en la que se crea la instancia de base de datos.

  Valor predeterminado: una zona de disponibilidad aleatoria y elegida por el sistema en la región de Amazon del punto de conexión.

  Ejemplo: `us-east-1d`

  Restricción: el parámetro AvailabilityZone no puede especificarse si el parámetro Multi-AZ se establece en `true`. La zona de disponibilidad especificada debe estar en la misma región de Amazon que el actual punto de conexión.
- BackupRetentionPeriod (en la CLI: `--backup-retention-period`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de días durante los que se retienen las copias de seguridad automatizadas.

  No se usa. El clúster de base de datos administra el periodo de retención de copias de seguridad automatizadas. Para obtener más información, consulte the section called “CreateDBCluster”.

  Valor predeterminado: 1

  Restricciones:

  - Debe ser un valor entre 0 y 35.
  - No se puede establecerse en 0 si la instancia de base de datos es un origen a las réplicas de lectura
- CopyTagsToSnapshot (en la CLI: `--copy-tags-to-snapshot`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  "True" para copiar todas las etiquetas de la instancia de base de datos a instantáneas de la instancia de base de datos, de lo contrario "false". El valor predeterminado es false.
- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El identificador del clúster de base de datos al que pertenecerá la instancia.

Para obtener más información acerca de la creación de un clúster de bases de datos, consulte the section called “CreateDBCluster”.

Tipo: cadena

- DBInstanceClass (en la CLI: `--db-instance-class`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La capacidad de memoria y de cómputo de la instancia de base de datos, por ejemplo, `db.m4.large`. No todas las clases de instancia de base de datos están disponibles en todas las regiones de Amazon.

- DBInstanceIdentifier (en la CLI: `--db-instance-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de instancias de base de datos. Este parámetro se almacena como una cadena en minúsculas.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

  Ejemplo: `mydbinstance`

- DBName (en la CLI: `--db-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.

- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del grupo de parámetros de base de datos que desea asociar a esta instancia de base de datos. Si se omite este argumento, se utiliza el DBParameterGroup predeterminado para el motor especificado.

  Restricciones:

  - Debe tener de 1 a 255 letras, números o guiones.

  - El primer carácter debe ser una letra
  - No puede terminar por un guion ni contener dos guiones consecutivos
- DBSecurityGroups (en la CLI: `--db-security-groups`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Lista de grupos de seguridad de base de datos que se van a asociar a esta instancia de base de datos.

  Valor predeterminado: el grupo de seguridad de base de datos predeterminado para el motor de base de datos.
- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un grupo de subred de base de datos con el que asociar esta instancia de base de datos.

  Si no hay un grupo de subred de base de datos, entonces se trata de una instancia de base de datos no VPC.
- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Es un valor que indica si la instancia de base de datos tiene habilitada la protección contra eliminación. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación. La protección contra eliminación está deshabilitada de forma predeterminada. Consulte Eliminación de una instancia de base de datos.

  Las instancias de base de datos en un clúster de bases de datos se pueden eliminar incluso cuando la protección contra eliminación esté habilitada su clúster de bases de datos principal.
- Domain (en la CLI: `--domain`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifique el dominio de Active Directory en el que crear la instancia.
- DomainIAMRoleName (en la CLI: `--domain-iam-role-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifique el nombre del rol de IAM que se utilizará al realizar llamadas a la API al servicio de directorio.
- EnableCloudwatchLogsExports (en la CLI: `--enable-cloudwatch-logs-exports`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

La lista de tipos de registro que deben habilitarse para exportar a CloudWatch Logs.

- EnableIAMDatabaseAuthentication (en la CLI: `--enable-iam-database-authentication`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  No es compatible con Neptune (se pasa por alto).

- Engine (en la CLI: `--engine`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del motor de base de datos que se va a usar para esta instancia.

  Valores válidos: `neptune`

- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El número de versión del motor de base de datos que se debe usar. En la actualidad, no tiene ningún efecto establecer este parámetro.

- Iops (en la CLI: `--iops`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  La cantidad de IOPS provisionadas (operaciones de entrada/salida por segundo) asignada inicialmente para la instancia de base de datos.

- KmsKeyId (en la CLI: `--kms-key-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la clave de Amazon KMS de una instancia de base de datos cifrada.

  El identificador de la clave de KMS es el Nombre de recurso de Amazon (ARN) de la clave de cifrado de KMS. Si está creando una instancia de base de datos con la misma cuenta de Amazon a la que pertenece la clave de cifrado de KMS utilizada para cifrar la nueva instancia de base de datos, puede utilizar el alias de la clave de KMS en lugar del ARN de la clave de cifrado de KMS.

  No se usa. El clúster de base de datos administra el identificador de claves de KMS. Para obtener más información, consulte the section called “CreateDBCluster”.

  Si el parámetro `StorageEncrypted` es "true", y no especifica un valor para el parámetro `KmsKeyId`, Amazon Neptune utilizará su clave de cifrado predeterminada. Amazon KMS crea la clave de cifrado predeterminada para su cuenta de Amazon. Su cuenta de Amazon dispone de una clave de cifrado predeterminada diferente para cada región de Amazon.

- LicenseModel (en la CLI: `--license-model`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Información del modelo de licencias de esta instancia de base de datos.

  Valores válidos: `license-included`| `bring-your-own-license`| `general-public-license`
- MonitoringInterval (en la CLI: `--monitoring-interval`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El intervalo, en segundos, entre puntos cuando se recopila las métricas de monitorización mejoradas para la instancia de base de datos. Para deshabilitar la recopilación de métricas de monitorización mejorada, especifique 0. El valor predeterminado es 0.

  Si se especifica `MonitoringRoleArn`, también debe establecer `MonitoringInterval` en un valor distinto de 0.

  Valores válidos: `0, 1, 5, 10, 15, 30, 60`
- MonitoringRoleArn (en la CLI: `--monitoring-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del rol de IAM que permite a Neptune enviar métricas de monitorización mejoradas a Amazon CloudWatch Logs. Por ejemplo, `arn:aws:iam:123456789012:role/emaccess`.

  Si se establece `MonitoringInterval` en un valor distinto de 0, debe suministrar un valor para `MonitoringRoleArn`.
- MultiAZ (en la CLI: `--multi-az`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos es una implementación Multi-AZ. No puede establecer el parámetro AvailabilityZone si el parámetro MultiAZ está establecido en "true".
- Port (en la CLI: `--port`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de puerto en el que la base de datos acepta conexiones.

  No se usa. El clúster de base de datos administra el puerto. Para obtener más información, consulte the section called “CreateDBCluster”.

Valor predeterminado: 8182

Tipo: número entero

- PreferredBackupWindow (en la CLI: `--preferred-backup-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El intervalo de tiempo diario durante el que se crean las copias de seguridad automatizadas.

  No se usa. El clúster de base de datos administra el intervalo de tiempo diario para la creación de copias de seguridad automatizadas. Para obtener más información, consulte the section called “CreateDBCluster”.
- PreferredMaintenanceWindow (en la CLI: `--preferred-maintenance-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en tiempo universal coordinado (UTC).

  Formato: `ddd:hh24:mi-ddd:hh24:mi`

  El valor predeterminado es un periodo de 30 minutos seleccionado al azar de un bloque de 8 horas de tiempo para cada región de Amazon, que tiene lugar un día de la semana de forma aleatoria.

  Días válidos: lunes, martes, miércoles, jueves, viernes, sábado, domingo.

  Restricciones: plazo mínimo de 30 minutos.
- PromotionTier (en la CLI: `--promotion-tier`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueven las réplicas de lectura a instancia principal tras un error de la instancia principal existente.

  Valor predeterminado: 1

  Valores válidos: 0 - 15
- PubliclyAccessible (en la CLI: `--publicly-accessible`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Hay que dejar de utilizar este marcador.

- StorageEncrypted (en la CLI: `--storage-encrypted`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos está cifrada.

  No se usa. El cifrado para las instancias de base de datos es gestionado por el clúster de base de datos. Para obtener más información, consulte the section called “CreateDBCluster”.

  Valor predeterminado: false

- StorageType (en la CLI: `--storage-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar a la nueva instancia.

- TdeCredentialArn (en la CLI: `--tde-credential-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del almacén de claves con el que asociar la instancia para el cifrado de TDE.

- TdeCredentialPassword (en la CLI: `--tde-credential-password`): una SensitiveString, del tipo: `string` (una cadena codificada con UTF-8).

  La contraseña del ARN dado del almacén de claves para poder obtener acceso al dispositivo.

- Timezone (en la CLI: `--timezone`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La zona horaria de la instancia de base de datos.

- VpcSecurityGroupIds (en la CLI: `--vpc-security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Lista de grupos de seguridad de VPC de EC2 que se van a asociar a esta instancia de base de datos.

  No se usa. La lista asociada de grupos de seguridad de VPC de EC2 administrada por este clúster de base de datos. Para obtener más información, consulte the section called “CreateDBCluster”.

Valor predeterminado: el grupo de seguridad de VPC de EC2 predeterminado para la VPC del grupo de la subred de base de datos.

Respuesta

Contiene los detalles de una instancia de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBInstances".

- AutoMinorVersionUpgrade: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que hay parches de versión secundaria que se aplican automáticamente.
- AvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la zona de disponibilidad en la que se encuentra la instancia de base de datos.
- BackupRetentionPeriod: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- CACertificateIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del certificado de CA para esta instancia de base de datos.
- CopyTagsToSnapshot: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si las etiquetas se copian de la instancia de base de datos a instantáneas de la instancia de base de datos.
- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instancia de base de datos es miembro de un clúster de base de datos, contiene el nombre del clúster de base de datos del que la instancia de base de datos es miembro.
- DBInstanceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instancia de base de datos.
- DBInstanceClass: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Contiene el nombre de la clase de capacidad de cómputo y memoria de la instancia de base de datos.

- DBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de bases de datos facilitado por el usuario. Este identificador es la clave única que identifica una instancia de base de datos.

- DbInstancePort: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que escucha la instancia de base de datos. Si la instancia de base de datos forma parte de un clúster de base de datos, este puede ser un puerto distinto del puerto del clúster de base de datos.

- DBInstanceStatus: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos.

- DbiResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para la instancia de base de datos. Este identificador se encuentra en entradas de registro de Amazon CloudTrail cuando se accede a la clave de Amazon KMS para la instancia de base de datos.

- DBName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la base de datos.

- DBParameterGroups: matriz de objetos DBParameterGroupStatus.

  Proporciona la lista de grupos de parámetros de base de datos que se aplica a esta instancia de base de datos.

- DBSecurityGroups: matriz de objetos DBSecurityGroupMembership.

  Proporciona la lista de elementos de grupo de seguridad de base de datos que contiene solo los subelementos `DBSecurityGroup.Name` y `DBSecurityGroup.Status`.

- DBSubnetGroup: objeto DBSubnetGroup.

  Especifica información sobre el grupo de subred asociado a la instancia de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

Indica si la instancia de base de datos tiene habilitada la protección contra eliminación. La instancia no se puede eliminar cuando está habilitada la protección contra eliminación. Consulte Eliminación de una instancia de base de datos.

- DomainMemberships: matriz de objetos DomainMembership.

  No compatible

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que esta instancia de base de datos está configurada para exportar a CloudWatch Logs.

- Endpoint: objeto Punto de conexión.

  Especifica el punto de conexión de conexión.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se va a usar para esta instancia de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- EnhancedMonitoringResourceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del flujo de registros de Amazon CloudWatch Logs que recibe los datos de métricas de monitorización mejorada para la instancia de base de datos.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True si la autenticación de Amazon Identity and Access Management (IAM) está habilitada; de lo contrario, el valor es False.

- InstanceCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la fecha y hora en que se creó la instancia de base de datos.

- Iops: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

Especifica el valor de las IOPS provisionadas (las operaciones de E/S por segundo).

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- LicenseModel: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Información del modelo de licencias de esta instancia de base de datos.

- MonitoringInterval: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El intervalo, en segundos, entre puntos cuando se recopila las métricas de monitorización mejoradas para la instancia de base de datos.

- MonitoringRoleArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del rol de IAM que permite a Neptune enviar métricas de monitorización mejoradas a Amazon CloudWatch Logs.

- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos es una implementación Multi-AZ.

- PendingModifiedValues: objeto PendingModifiedValues.

  Especifica que los cambios a la instancia de base de datos están pendientes. Este elemento sólo se incluye cuando los cambios están pendientes. Los cambios específicos se identifican por subelementos.

- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- PromotionTier: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueve una réplica de lectura a la instancia principal tras un error de la instancia principal existente.

- PubliclyAccessible: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Hay que dejar de utilizar este marcador.

- ReadReplicaDBClusterIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de los clústeres de base de datos que son réplicas de lectura de esta instancia de base de datos.

- ReadReplicaDBInstanceIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con esta instancia de base de datos.

- ReadReplicaSourceDBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el identificador de la instancia de base de datos de origen si esta instancia de base de datos es una réplica de lectura.

- SecondaryAvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si está presente, especifica el nombre de la zona de disponibilidad secundaria para una instancia de base de datos compatible con Multi-AZ.

- StatusInfos: matriz de objetos DBInstanceStatusInfo.

  El estado de una réplica de lectura. Si la instancia no es una réplica de lectura, está en blanco.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el tipo de almacenamiento asociado con la instancia de base de datos.

- TdeCredentialArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del almacén de claves con el que se asocia la instancia para el cifrado de TDE.

- Timezone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de elementos de grupos de seguridad de VPC a la que pertenece la instancia de base de datos.

Errores

- DBInstanceAlreadyExistsFault
- InsufficientDBInstanceCapacityFault
- DBParameterGroupNotFoundFault
- DBSecurityGroupNotFoundFault
- InstanceQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidDBClusterStateFault
- InvalidSubnet
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBClusterNotFoundFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- KMSKeyNotAccessibleFault
- DomainNotFoundFault

DeleteDBInstance (acción)

El nombre de la AWS CLI para esta API es: `delete-db-instance`.

La acción DeleteDBInstance elimina una instancia de base de datos aprovisionada anteriormente. Al eliminar una instancia de base de datos, se eliminan todas las copias de seguridad automatizadas para esa instancia y no se pueden recuperar. Las instantáneas de base de datos manuales de la instancia de base de datos que va a eliminar `DeleteDBInstance` no se eliminan.

Si solicita a una instantánea de base de datos final el estado de la instancia de base de datos de Amazon Neptune es `deleting` hasta que se crea la instantánea de base de datos. La acción de la API `DescribeDBInstance` se utiliza para monitorizar el estado de esta operación. La acción no se puede cancelar o revertir una vez enviada.

Tenga en cuenta que cuando una instancia de base de datos se encuentra en un estado de error y tiene un estado de `failed`, `incompatible-restore` o `incompatible-network`, solo puede eliminarlo cuando el parámetro `SkipFinalSnapshot` se establece en `true`.

No puede eliminar una instancia de base de datos si es la única instancia en el clúster de base de datos o si tiene habilitada la protección contra eliminación.

Solicitud

- DBInstanceIdentifier (en la CLI: `--db-instance-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de instancias de bases de datos para la instancia de base de datos que se va a eliminar. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Debe coincidir con el nombre de una instancia de base de datos existente.
- FinalDBSnapshotIdentifier (en la CLI: `--final-db-snapshot-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El DBSnapshotIdentifier del nuevo DBSnapshot se crea cuando SkipFinalSnapshot se establece en `false`.

> **Note**
>
> Si se especifica este parámetro y también se establece el parámetro SkipFinalShapshot en "true", se producirá un error.

Restricciones:

- Debe tener de 1 a 255 letras o números.
- El primer carácter debe ser una letra
- No puede terminar por un guion ni contener dos guiones consecutivos
- No puede especificarse al eliminar una réplica de lectura.

- SkipFinalSnapshot (en la CLI: `--skip-final-snapshot`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Determina si se crea una instantánea de base de datos antes de que se elimine la instancia de base de datos. Si se especifica `true`, no se crea DBSnapshot. Si se especifica `false`, se crea una instantánea de base de datos antes de que se elimine la instancia de base de datos.

  Tenga en cuenta que cuando una instancia de base de datos se encuentra en los estados "failed", "incompatible-restore" o "incompatible-network", solo puede eliminarla cuando el parámetro SkipFinalSnapshot se establece en "true".

  Especifique `true` al eliminar una réplica de lectura.

  > **Note**
  >
  > Deberá especificarse el parámetro FinalDBSnapshotIdentifier si SkipFinalSnapshot es `false`.

  Valor predeterminado: `false`

Respuesta

Contiene los detalles de una instancia de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBInstances".

- AutoMinorVersionUpgrade: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que hay parches de versión secundaria que se aplican automáticamente.
- AvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la zona de disponibilidad en la que se encuentra la instancia de base de datos.
- BackupRetentionPeriod: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- CACertificateIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del certificado de CA para esta instancia de base de datos.
- CopyTagsToSnapshot: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si las etiquetas se copian de la instancia de base de datos a instantáneas de la instancia de base de datos.
- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instancia de base de datos es miembro de un clúster de base de datos, contiene el nombre del clúster de base de datos del que la instancia de base de datos es miembro.
- DBInstanceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instancia de base de datos.
- DBInstanceClass: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la clase de capacidad de cómputo y memoria de la instancia de base de datos.
- DBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de bases de datos facilitado por el usuario. Este identificador es la clave única que identifica una instancia de base de datos.
- DbInstancePort: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

Especifica el puerto que escucha la instancia de base de datos. Si la instancia de base de datos forma parte de un clúster de base de datos, este puede ser un puerto distinto del puerto del clúster de base de datos.

- DBInstanceStatus: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos.

- DbiResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para la instancia de base de datos. Este identificador se encuentra en entradas de registro de Amazon CloudTrail cuando se accede a la clave de Amazon KMS para la instancia de base de datos.

- DBName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la base de datos.

- DBParameterGroups: matriz de objetos DBParameterGroupStatus.

  Proporciona la lista de grupos de parámetros de base de datos que se aplica a esta instancia de base de datos.

- DBSecurityGroups: matriz de objetos DBSecurityGroupMembership.

  Proporciona la lista de elementos de grupo de seguridad de base de datos que contiene solo los subelementos `DBSecurityGroup.Name` y `DBSecurityGroup.Status`.

- DBSubnetGroup: objeto DBSubnetGroup.

  Especifica información sobre el grupo de subred asociado a la instancia de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la instancia de base de datos tiene habilitada la protección contra eliminación. La instancia no se puede eliminar cuando está habilitada la protección contra eliminación. Consulte Eliminación de una instancia de base de datos.

- DomainMemberships: matriz de objetos DomainMembership.

  No compatible

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una lista de tipos de registro para los que esta instancia de base de datos está configurada para exportar a CloudWatch Logs.

- Endpoint: objeto Punto de conexión.

  Especifica el punto de conexión de conexión.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se va a usar para esta instancia de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- EnhancedMonitoringResourceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del flujo de registros de Amazon CloudWatch Logs que recibe los datos de métricas de monitorización mejorada para la instancia de base de datos.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True si la autenticación de Amazon Identity and Access Management (IAM) está habilitada; de lo contrario, el valor es False.

- InstanceCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la fecha y hora en que se creó la instancia de base de datos.

- Iops: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el valor de las IOPS provisionadas (las operaciones de E/S por segundo).

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- LicenseModel: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Información del modelo de licencias de esta instancia de base de datos.

- MonitoringInterval: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El intervalo, en segundos, entre puntos cuando se recopila las métricas de monitorización mejoradas para la instancia de base de datos.

- MonitoringRoleArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del rol de IAM que permite a Neptune enviar métricas de monitorización mejoradas a Amazon CloudWatch Logs.

- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos es una implementación Multi-AZ.

- PendingModifiedValues: objeto PendingModifiedValues.

  Especifica que los cambios a la instancia de base de datos están pendientes. Este elemento sólo se incluye cuando los cambios están pendientes. Los cambios específicos se identifican por subelementos.

- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- PromotionTier: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueve una réplica de lectura a la instancia principal tras un error de la instancia principal existente.

- PubliclyAccessible: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Hay que dejar de utilizar este marcador.

- ReadReplicaDBClusterIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de los clústeres de base de datos que son réplicas de lectura de esta instancia de base de datos.

- ReadReplicaDBInstanceIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con esta instancia de base de datos.

- ReadReplicaSourceDBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el identificador de la instancia de base de datos de origen si esta instancia de base de datos es una réplica de lectura.

- SecondaryAvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si está presente, especifica el nombre de la zona de disponibilidad secundaria para una instancia de base de datos compatible con Multi-AZ.

- StatusInfos: matriz de objetos DBInstanceStatusInfo.

  El estado de una réplica de lectura. Si la instancia no es una réplica de lectura, está en blanco.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de almacenamiento asociado con la instancia de base de datos.

- TdeCredentialArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del almacén de claves con el que se asocia la instancia para el cifrado de TDE.

- Timezone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

Proporciona una lista de elementos de grupos de seguridad de VPC a la que pertenece la instancia de base de datos.

Errores

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault
- DBSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- InvalidDBClusterStateFault

ModifyDBInstance (acción)

El nombre de la AWS CLI para esta API es: `modify-db-instance`.

Modifica la configuración de una instancia de base de datos. Puede cambiar uno o varios parámetros de configuración de la base de datos mediante la especificación de estos parámetros y los nuevos valores en la solicitud. Para obtener información sobre qué modificaciones puede realizar a su instancia de base de datos, llame a the section called “DescribeValidDBInstanceModifications” antes de llamar a the section called “ModifyDBInstance”.

Solicitud

- AllowMajorVersionUpgrade (en la CLI: `--allow-major-version-upgrade`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que se permiten actualizaciones de la versión principal. El cambio de este parámetro no produce una interrupción y el cambio se aplica de forma asíncrona tan pronto como sea posible.
- ApplyImmediately (en la CLI: `--apply-immediately`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si las modificaciones de esta solicitud y todas las modificaciones pendientes se aplican de manera asíncrona en cuanto es posible, independientemente de la configuración de `PreferredMaintenanceWindow` de esta instancia de base de datos.

  Si este parámetro se establece en `false`, los cambios realizados en la instancia de base de datos se aplican durante la siguiente ventana de mantenimiento. Algunos cambios de los

parámetros puede causar una interrupción y se aplican en la siguiente llamada a the section called "RebootDBInstance", o el siguiente reinicio por error.

Valor predeterminado: `false`

- AutoMinorVersionUpgrade (en la CLI: `--auto-minor-version-upgrade`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que las actualizaciones de versión secundarias se aplican automáticamente a la instancia de base de datos durante el periodo de mantenimiento. El cambio de este parámetro no produce una interrupción, salvo en el siguiente caso, y el cambio se aplica de forma asíncrona tan pronto como sea posible. Se producirá una interrupción si este parámetro se establece en `true` durante el periodo de mantenimiento, y hay una versión secundaria más nueva disponible y Neptune ha permitido la aplicación de parches automática para esa versión del motor.

- BackupRetentionPeriod (en la CLI: `--backup-retention-period`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No se usa. El clúster de base de datos administra el periodo de retención de copias de seguridad automatizadas. Para obtener más información, consulte the section called "ModifyDBCluster".

  Valor predeterminado: Utiliza la configuración existente

- CACertificateIdentifier (en la CLI: `--ca-certificate-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica el certificado que debe asociarse a la instancia.

- CloudwatchLogsExportConfiguration (en la CLI: `--cloudwatch-logs-export-configuration`): un objeto CloudwatchLogsExportConfiguration.

  La opción de configuración para los tipos de registro que se debe habilitar para la exportación a CloudWatch Logs para una instancia de base de datos específica o un clúster de base de datos.

- CopyTagsToSnapshot (en la CLI: `--copy-tags-to-snapshot`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  "True" para copiar todas las etiquetas de la instancia de base de datos a instantáneas de la instancia de base de datos, de lo contrario "false". El valor predeterminado es false.

- DBInstanceClass (en la CLI: `--db-instance-class`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

La capacidad nueva de memoria y de cómputo de la instancia de base de datos, por ejemplo, `db.m4.large`. No todas las clases de instancia de base de datos están disponibles en todas las regiones de Amazon.

Si modifica la clase de la instancia de base de datos se produce una interrupción durante el cambio. El cambio se aplica durante la siguiente ventana de mantenimiento, a menos que `ApplyImmediately` se especifique como `true` para esta solicitud.

Valor predeterminado: Utiliza la configuración existente

- DBInstanceIdentifier (en la CLI: `--db-instance-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de instancias de base de datos. Este valor se almacena como una cadena en minúsculas.

  Restricciones:

  - Debe coincidir con el identificador de una DBInstance existente.
- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del grupo de parámetros de base de datos que se va a aplicar a esta instancia de base de datos. Cambiar este ajuste no produce una interrupción. El nombre del grupo de parámetros se modifica inmediatamente, pero los cambios del parámetro no se aplican hasta que se reinicia la instancia sin conmutación por error. La instancia de base de datos NO se reiniciará automáticamente y los cambios de los parámetros NO se aplicarán durante la siguiente ventana de mantenimiento.

  Valor predeterminado: Utiliza la configuración existente

  Restricciones: el grupo de parámetros de base de datos debe estar en la misma familia de grupos de parámetros de base de datos que esta instancia de base de datos.
- DBPortNumber (en la CLI: `--db-port-number`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de puerto en el que la base de datos acepta conexiones.

  El valor del parámetro `DBPortNumber` no debe coincidir con ninguno de los valores de puertos especificados para el grupo de opciones para la instancia de base de datos.

La base de datos se reiniciará si se cambia el valor `DBPortNumber` independientemente del valor del parámetro `ApplyImmediately`.

Valor predeterminado: `8182`

- DBSecurityGroups (en la CLI: `--db-security-groups`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Lista de grupos de seguridad de base de datos que se van a autorizar en esta instancia de base de datos. El cambio de esta configuración no produce una interrupción y el cambio se aplica de forma asíncrona tan pronto como sea posible.

  Restricciones:

  - Si se suministra, debe coincidir con DBSecurityGroups existentes.
- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El grupo de subredes de base de datos nuevo de la instancia de base de datos. Puede utilizar este parámetro para mover la instancia de base de datos a otra VPC.

  Cambiar el grupo de subredes provoca una interrupción durante el cambio. El cambio se aplica durante la siguiente ventana de mantenimiento, a menos que especifique `true` para el parámetro `ApplyImmediately`.

  Restricciones: si se proporciona, debe coincidir con el nombre de un DBSubnetGroup existente.

  Ejemplo: `mySubnetGroup`
- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Es un valor que indica si la instancia de base de datos tiene habilitada la protección contra eliminación. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación. La protección contra eliminación está deshabilitada de forma predeterminada. Consulte Eliminación de una instancia de base de datos.
- Domain (en la CLI: `--domain`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.
- DomainIAMRoleName (en la CLI: `--domain-iam-role-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

No compatible

- EnableIAMDatabaseAuthentication (en la CLI: `--enable-iam-database-authentication`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  True para habilitar el mapeo de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos; de lo contrario, el valor es False.

  Puede utilizar la autenticación de bases de datos de IAM para los siguientes motores de bases de datos

  No se usa. El clúster de base de datos administra el mapeo de cuentas de Amazon IAM a cuentas de base de datos. Para obtener más información, consulte the section called “ModifyDBCluster”.

  Valor predeterminado: `false`

- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Número de versión del motor de base de datos al que se va a actualizar. En la actualidad, no tiene ningún efecto establecer este parámetro. Para actualizar el motor de la base de datos a la versión más reciente, utilice la API the section called “ApplyPendingMaintenanceAction”.

- Iops (en la CLI: `--iops`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El valor nuevo de la IOPS provisionada (operaciones de E/S por segundo) para la instancia.

  El cambio de esta configuración no produce una interrupción y el cambio se aplica durante la siguiente ventana de mantenimiento a menos que el parámetro `ApplyImmediately` se haya establecido en `true` para esta solicitud.

  Valor predeterminado: Utiliza la configuración existente

- MonitoringInterval (en la CLI: `--monitoring-interval`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El intervalo, en segundos, entre puntos cuando se recopila las métricas de monitorización mejoradas para la instancia de base de datos. Para deshabilitar la recopilación de métricas de monitorización mejorada, especifique 0. El valor predeterminado es 0.

Si se especifica `MonitoringRoleArn`, también debe establecer `MonitoringInterval` en un valor distinto de 0.

Valores válidos: `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (en la CLI: `--monitoring-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del rol de IAM que permite a Neptune enviar métricas de monitorización mejoradas a Amazon CloudWatch Logs. Por ejemplo, `arn:aws:iam:123456789012:role/emaccess`.

  Si se establece `MonitoringInterval` en un valor distinto de 0, debe suministrar un valor para `MonitoringRoleArn`.

- MultiAZ (en la CLI: `--multi-az`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos es una implementación Multi-AZ. El cambio de este parámetro no produce una interrupción y el cambio se aplica durante la siguiente ventana de mantenimiento a menos que el parámetro `ApplyImmediately` se haya establecido en `true` para esta solicitud.

- NewDBInstanceIdentifier (en la CLI: `--new-db-instance-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nuevo identificador de instancia de base de datos para la instancia de base de datos cuando se cambia el nombre de una instancia de base de datos. Cuando cambie el identificador de instancia de base de datos, la instancia se reiniciará inmediatamente si establece `Apply Immediately` en "true", o durante la siguiente ventana de mantenimiento si establece `Apply Immediately` en "false". Este valor se almacena como una cadena en minúsculas.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

  Ejemplo: `mydbinstance`

- PreferredBackupWindow (en la CLI: `--preferred-backup-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si están habilitadas.

No se usa. El clúster de base de datos administra el intervalo de tiempo diario para la creación de copias de seguridad automatizadas. Para obtener más información, consulte the section called "ModifyDBCluster".

Restricciones:

- Tiene que tener el formato hh24:mi-hh24:mi
- Debe estar en tiempo universal coordinado (UTC)
- No debe entrar en conflicto con la ventana de mantenimiento preferida.
- Debe durar al menos 30 minutos.

- PreferredMaintenanceWindow (en la CLI: `--preferred-maintenance-window`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El intervalo de tiempo semanal (en UTC) durante el cual puede llevarse a cabo el mantenimiento del sistema, que puede producir una interrupción. El cambio de este parámetro no produce una interrupción, salvo en la siguiente situación, y el cambio se aplica de forma asíncrona tan pronto como sea posible. Si hay acciones pendientes que provocan un reinicio, y el periodo de mantenimiento se cambia para incluir la hora actual, cambiar este parámetro provocará un reinicio de la instancia de base de datos. Si cambiar este periodo a la hora actual, debe haber al menos 30 minutos entre la hora actual y el final del periodo, para asegurarse de que se apliquen los cambios pendientes.

  Valor predeterminado: Utiliza la configuración existente

  Formato: ddd:hh24:mi-ddd:hh24:mi

  Días válidos: lunes | martes | miércoles | jueves | viernes | sábado | domingo

  Restricciones: debe durar al menos 30 minutos.

- PromotionTier (en la CLI: `--promotion-tier`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueve una réplica de lectura a la instancia principal tras un error de la instancia principal existente.

  Valor predeterminado: 1

Valores válidos: 0 - 15

- PubliclyAccessible (en la CLI: `--publicly-accessible`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Hay que dejar de utilizar este marcador.

- StorageType (en la CLI: `--storage-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

- TdeCredentialArn (en la CLI: `--tde-credential-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del almacén de claves con el que asociar la instancia para el cifrado de TDE.

- TdeCredentialPassword (en la CLI: `--tde-credential-password`): una SensitiveString, del tipo: `string` (una cadena codificada con UTF-8).

  La contraseña del ARN dado del almacén de claves para poder obtener acceso al dispositivo.

- VpcSecurityGroupIds (en la CLI: `--vpc-security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Lista de grupos de seguridad de VPC de EC2 que se van a autorizar en esta instancia de base de datos. Este cambio se aplica de forma asíncrona, tan pronto como sea posible.

  No se usa. La lista asociada de grupos de seguridad de VPC de EC2 administrada por este clúster de base de datos. Para obtener más información, consulte the section called “ModifyDBCluster”.

  Restricciones:

  - Si se suministra, debe coincidir con VpcSecurityGroupIds existentes.

Respuesta

Contiene los detalles de una instancia de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called “DescribeDBInstances”.

- AutoMinorVersionUpgrade: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

Indica que hay parches de versión secundaria que se aplican automáticamente.

- AvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la zona de disponibilidad en la que se encuentra la instancia de base de datos.

- BackupRetentionPeriod: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.

- CACertificateIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del certificado de CA para esta instancia de base de datos.

- CopyTagsToSnapshot: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si las etiquetas se copian de la instancia de base de datos a instantáneas de la instancia de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instancia de base de datos es miembro de un clúster de base de datos, contiene el nombre del clúster de base de datos del que la instancia de base de datos es miembro.

- DBInstanceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instancia de base de datos.

- DBInstanceClass: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la clase de capacidad de cómputo y memoria de la instancia de base de datos.

- DBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de bases de datos facilitado por el usuario. Este identificador es la clave única que identifica una instancia de base de datos.

- DbInstancePort: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que escucha la instancia de base de datos. Si la instancia de base de datos forma parte de un clúster de base de datos, este puede ser un puerto distinto del puerto del clúster de base de datos.

- DBInstanceStatus: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el estado actual de esta base de datos.

- DbiResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para la instancia de base de datos. Este identificador se encuentra en entradas de registro de Amazon CloudTrail cuando se accede a la clave de Amazon KMS para la instancia de base de datos.

- DBName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la base de datos.

- DBParameterGroups: matriz de objetos DBParameterGroupStatus.

  Proporciona la lista de grupos de parámetros de base de datos que se aplica a esta instancia de base de datos.

- DBSecurityGroups: matriz de objetos DBSecurityGroupMembership.

  Proporciona la lista de elementos de grupo de seguridad de base de datos que contiene solo los subelementos `DBSecurityGroup.Name` y `DBSecurityGroup.Status`.

- DBSubnetGroup: objeto DBSubnetGroup.

  Especifica información sobre el grupo de subred asociado a la instancia de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la instancia de base de datos tiene habilitada la protección contra eliminación. La instancia no se puede eliminar cuando está habilitada la protección contra eliminación. Consulte Eliminación de una instancia de base de datos.

- DomainMemberships: matriz de objetos DomainMembership.

  No compatible

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que esta instancia de base de datos está configurada para exportar a CloudWatch Logs.

- Endpoint: objeto Punto de conexión.

  Especifica el punto de conexión de conexión.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se va a usar para esta instancia de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- EnhancedMonitoringResourceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del flujo de registros de Amazon CloudWatch Logs que recibe los datos de métricas de monitorización mejorada para la instancia de base de datos.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True si la autenticación de Amazon Identity and Access Management (IAM) está habilitada; de lo contrario, el valor es False.

- InstanceCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la fecha y hora en que se creó la instancia de base de datos.

- Iops: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el valor de las IOPS provisionadas (las operaciones de E/S por segundo).

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- LicenseModel: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Información del modelo de licencias de esta instancia de base de datos.

- MonitoringInterval: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

El intervalo, en segundos, entre puntos cuando se recopila las métricas de monitorización mejoradas para la instancia de base de datos.

- MonitoringRoleArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del rol de IAM que permite a Neptune enviar métricas de monitorización mejoradas a Amazon CloudWatch Logs.
- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos es una implementación Multi-AZ.
- PendingModifiedValues: objeto PendingModifiedValues.

  Especifica que los cambios a la instancia de base de datos están pendientes. Este elemento sólo se incluye cuando los cambios están pendientes. Los cambios específicos se identifican por subelementos.
- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.
- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).
- PromotionTier: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueve una réplica de lectura a la instancia principal tras un error de la instancia principal existente.
- PubliclyAccessible: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Hay que dejar de utilizar este marcador.
- ReadReplicaDBClusterIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de los clústeres de base de datos que son réplicas de lectura de esta instancia de base de datos.
- ReadReplicaDBInstanceIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Contiene uno o varios identificadores de las réplicas de lectura asociados con esta instancia de base de datos.

- ReadReplicaSourceDBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el identificador de la instancia de base de datos de origen si esta instancia de base de datos es una réplica de lectura.

- SecondaryAvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si está presente, especifica el nombre de la zona de disponibilidad secundaria para una instancia de base de datos compatible con Multi-AZ.

- StatusInfos: matriz de objetos DBInstanceStatusInfo.

  El estado de una réplica de lectura. Si la instancia no es una réplica de lectura, está en blanco.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de almacenamiento asociado con la instancia de base de datos.

- TdeCredentialArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del almacén de claves con el que se asocia la instancia para el cifrado de TDE.

- Timezone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de elementos de grupos de seguridad de VPC a la que pertenece la instancia de base de datos.

Errores

- InvalidDBInstanceStateFault
- InvalidDBSecurityGroupStateFault

- DBInstanceAlreadyExistsFault
- DBInstanceNotFoundFault
- DBSecurityGroupNotFoundFault
- DBParameterGroupNotFoundFault
- InsufficientDBInstanceCapacityFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBUpgradeDependencyFailureFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- CertificateNotFoundFault
- DomainNotFoundFault

RebootDBInstance (acción)

El nombre de la AWS CLI para esta API es: `reboot-db-instance`.

Es posible que necesite reiniciar su instancia de base de datos, normalmente por razones de mantenimiento. Por ejemplo, si realiza determinadas modificaciones o si cambia el grupo de parámetros de base de datos asociado a la instancia de base de datos, debe reiniciar la instancia para que los cambios surtan efecto.

Cuando se reinicia una instancia de base de datos, se reinicia el servicio del motor de base de datos. Al reiniciar una instancia de base de datos, se produce una interrupción momentánea, durante la cual su estado se establece en rebooting.

Solicitud

- DBInstanceIdentifier (en la CLI: `--db-instance-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de instancias de base de datos. Este parámetro se almacena como una cadena en minúsculas.

Restricciones:

- Debe coincidir con el identificador de una DBInstance existente.

- ForceFailover (en la CLI: `--force-failover`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Cuando `true`, el reinicio se realiza a través de una conmutación por error Multi-AZ.

  Restricción no se puede especificar `true` si la instancia no se ha configurado para Multi-AZ.

Respuesta

Contiene los detalles de una instancia de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBInstances".

- AutoMinorVersionUpgrade: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que hay parches de versión secundaria que se aplican automáticamente.

- AvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la zona de disponibilidad en la que se encuentra la instancia de base de datos.

- BackupRetentionPeriod: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.

- CACertificateIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del certificado de CA para esta instancia de base de datos.

- CopyTagsToSnapshot: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si las etiquetas se copian de la instancia de base de datos a instantáneas de la instancia de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instancia de base de datos es miembro de un clúster de base de datos, contiene el nombre del clúster de base de datos del que la instancia de base de datos es miembro.

- DBInstanceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instancia de base de datos.

- DBInstanceClass: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la clase de capacidad de cómputo y memoria de la instancia de base de datos.

- DBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de bases de datos facilitado por el usuario. Este identificador es la clave única que identifica una instancia de base de datos.

- DbInstancePort: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que escucha la instancia de base de datos. Si la instancia de base de datos forma parte de un clúster de base de datos, este puede ser un puerto distinto del puerto del clúster de base de datos.

- DBInstanceStatus: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos.

- DbiResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para la instancia de base de datos. Este identificador se encuentra en entradas de registro de Amazon CloudTrail cuando se accede a la clave de Amazon KMS para la instancia de base de datos.

- DBName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la base de datos.

- DBParameterGroups: matriz de objetos DBParameterGroupStatus.

  Proporciona la lista de grupos de parámetros de base de datos que se aplica a esta instancia de base de datos.

- DBSecurityGroups: matriz de objetos DBSecurityGroupMembership.

  Proporciona la lista de elementos de grupo de seguridad de base de datos que contiene solo los subelementos `DBSecurityGroup.Name` y `DBSecurityGroup.Status`.

- DBSubnetGroup: objeto DBSubnetGroup.

Especifica información sobre el grupo de subred asociado a la instancia de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la instancia de base de datos tiene habilitada la protección contra eliminación. La instancia no se puede eliminar cuando está habilitada la protección contra eliminación. Consulte Eliminación de una instancia de base de datos.
- DomainMemberships: matriz de objetos DomainMembership.

  No compatible
- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que esta instancia de base de datos está configurada para exportar a CloudWatch Logs.
- Endpoint: objeto Punto de conexión.

  Especifica el punto de conexión de conexión.
- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se va a usar para esta instancia de base de datos.
- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.
- EnhancedMonitoringResourceArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del flujo de registros de Amazon CloudWatch Logs que recibe los datos de métricas de monitorización mejorada para la instancia de base de datos.
- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True si la autenticación de Amazon Identity and Access Management (IAM) está habilitada; de lo contrario, el valor es False.
- InstanceCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

Proporciona la fecha y hora en que se creó la instancia de base de datos.

- Iops: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el valor de las IOPS provisionadas (las operaciones de E/S por segundo).

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- LicenseModel: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Información del modelo de licencias de esta instancia de base de datos.

- MonitoringInterval: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El intervalo, en segundos, entre puntos cuando se recopila las métricas de monitorización mejoradas para la instancia de base de datos.

- MonitoringRoleArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del rol de IAM que permite a Neptune enviar métricas de monitorización mejoradas a Amazon CloudWatch Logs.

- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos es una implementación Multi-AZ.

- PendingModifiedValues: objeto PendingModifiedValues.

  Especifica que los cambios a la instancia de base de datos están pendientes. Este elemento sólo se incluye cuando los cambios están pendientes. Los cambios específicos se identifican por subelementos.

- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).
- PromotionTier: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueve una réplica de lectura a la instancia principal tras un error de la instancia principal existente.
- PubliclyAccessible: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Hay que dejar de utilizar este marcador.
- ReadReplicaDBClusterIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de los clústeres de base de datos que son réplicas de lectura de esta instancia de base de datos.
- ReadReplicaDBInstanceIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con esta instancia de base de datos.
- ReadReplicaSourceDBInstanceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el identificador de la instancia de base de datos de origen si esta instancia de base de datos es una réplica de lectura.
- SecondaryAvailabilityZone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si está presente, especifica el nombre de la zona de disponibilidad secundaria para una instancia de base de datos compatible con Multi-AZ.
- StatusInfos: matriz de objetos DBInstanceStatusInfo.

  El estado de una réplica de lectura. Si la instancia no es una réplica de lectura, está en blanco.
- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.
- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el tipo de almacenamiento asociado con la instancia de base de datos.

- TdeCredentialArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del almacén de claves con el que se asocia la instancia para el cifrado de TDE.
- Timezone: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.
- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de elementos de grupos de seguridad de VPC a la que pertenece la instancia de base de datos.

Errores

- InvalidDBInstanceStateFault
- DBInstanceNotFoundFault

DescribeDBInstances (acción)

El nombre de la AWS CLI para esta API es: `describe-db-instances`.

Devuelve información sobre las instancias aprovisionadas y admite la paginación.

> **Note**
>
> Esta operación también puede devolver información para las instancias de Amazon RDS y las instancias de Amazon DocDB.

Solicitud

- DBInstanceIdentifier (en la CLI: `--db-instance-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identificador de la instancia proporcionado por el usuario. Si se especifica este parámetro, solo se devuelve información de la instancia de base de datos específica. Este parámetro no distingue entre mayúsculas y minúsculas.

Restricciones:

- Si se proporciona, debe coincidir con el identificador de una DBInstance existente.

- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Un filtro que especifica una o varias instancias de bases de datos para describir.

  Filtros compatibles:

  - `db-cluster-id`: acepta identificadores de clúster de base de datos y Nombres de recursos de Amazon (ARN). La lista de resultados solo incluirá información sobre las instancias de bases de datos asociadas con los clústeres de base de datos identificados por estos ARN.
  - `engine` - Acepta un nombre de motor (como `neptune`) y restringe la lista de resultados a instancias de base de datos creadas por ese motor.

  Por ejemplo, para invocar esta API desde la CLI de Amazon y filtrar de modo que sólo se devuelvan las instancias de base de datos de Neptune, puede utilizar el siguiente comando:

  Example

```
aws neptune describe-db-instances \
            --filters  Name=engine,Values=neptune
```

- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeDBInstances` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se deben incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Mínimo 20, máximo 100.

Respuesta

- DBInstances: matriz de objetos DBInstance.

  Una lista de instancias the section called “DBInstance”.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

Errores

- DBInstanceNotFoundFault

DescribeOrderableDBInstanceOptions (acción)

El nombre de la AWS CLI para esta API es: `describe-orderable-db-instance-options`.

Devuelve una lista de opciones de instancia de base de datos ordenable para el motor especificado.

Solicitud

- DBInstanceClass (en la CLI: `--db-instance-class`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El valor del filtro de la clase de instancia de base de datos. Especifique este parámetro para mostrar solo las ofertas disponibles que coinciden con la clase de instancia de base de datos especificada.
- Engine (en la CLI: `--engine`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del motor para el que recuperar opciones de la instancia de base de datos.
- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Valor del filtro de la versión del motor. Especifique este parámetro para mostrar solo las ofertas disponibles que coinciden con la versión del motor especificado.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.

- LicenseModel (en la CLI: `--license-model`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El valor del filtro del modelo de licencia. Especifique este parámetro para mostrar solo las ofertas disponibles que coinciden con el modelo de licencia especificado.

- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud DescribeOrderableDBInstanceOptions anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se deben incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Mínimo 20, máximo 100.

- Vpc (en la CLI: `--vpc`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  El valor de filtro de VPC. Especifique este parámetro para mostrar solo la disponibilidad de ofertas VPC o no VPC.

Respuesta

- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud OrderableDBInstanceOptions anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- OrderableDBInstanceOptions: matriz de objetos OrderableDBInstanceOption.

  Una estructura the section called “OrderableDBInstanceOption” que contiene información acerca de opciones ordenables para la instancia de base de datos.

DescribeValidDBInstanceModifications (acción)

El nombre de la AWS CLI para esta API es: `describe-valid-db-instance-modifications`.

Puede llamar a the section called “DescribeValidDBInstanceModifications” para obtener información sobre qué modificaciones puede realizar a su instancia de base de datos. Puede utilizar esta información cuando llama a the section called “ModifyDBInstance”.

Solicitud

- DBInstanceIdentifier (en la CLI: `--db-instance-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de cliente o el ARN de la instancia de base de datos.

Respuesta

Información sobre las modificaciones válidas que puede realizar a la instancia de base de datos. Contiene el resultado de una llamada correcta a la acción the section called “DescribeValidDBInstanceModifications”. Puede utilizar esta información cuando llama a the section called “ModifyDBInstance”.

- Storage: matriz de objetos ValidStorageOptions.

  Opciones de almacenamiento válidas para su instancia de base de datos.

Errores

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault

Estructuras:

DBInstance (estructura)

Contiene los detalles de una instancia de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBInstances".

Campos

- AutoMinorVersionUpgrade: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que hay parches de versión secundaria que se aplican automáticamente.
- AvailabilityZone: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la zona de disponibilidad en la que se encuentra la instancia de base de datos.
- BackupRetentionPeriod: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- CACertificateIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del certificado de CA para esta instancia de base de datos.
- CopyTagsToSnapshot: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si las etiquetas se copian de la instancia de base de datos a instantáneas de la instancia de base de datos.
- DBClusterIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instancia de base de datos es miembro de un clúster de base de datos, contiene el nombre del clúster de base de datos del que la instancia de base de datos es miembro.
- DBInstanceArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instancia de base de datos.
- DBInstanceClass: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la clase de capacidad de cómputo y memoria de la instancia de base de datos.

- DBInstanceIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de bases de datos facilitado por el usuario. Este identificador es la clave única que identifica una instancia de base de datos.
- DbInstancePort: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que escucha la instancia de base de datos. Si la instancia de base de datos forma parte de un clúster de base de datos, este puede ser un puerto distinto del puerto del clúster de base de datos.
- DBInstanceStatus: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de esta base de datos.
- DbiResourceId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para la instancia de base de datos. Este identificador se encuentra en entradas de registro de Amazon CloudTrail cuando se accede a la clave de Amazon KMS para la instancia de base de datos.
- DBName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la base de datos.
- DBParameterGroups: se trata de una matriz de objetos DBParameterGroupStatus.

  Proporciona la lista de grupos de parámetros de base de datos que se aplica a esta instancia de base de datos.
- DBSecurityGroups: se trata de una matriz de objetos DBSecurityGroupMembership.

  Proporciona la lista de elementos de grupo de seguridad de base de datos que contiene solo los subelementos `DBSecurityGroup.Name` y `DBSecurityGroup.Status`.
- DBSubnetGroup: se trata de un objeto DBSubnetGroup.

  Especifica información sobre el grupo de subred asociado a la instancia de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.
- DeletionProtection: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la instancia de base de datos tiene habilitada la protección contra eliminación. La instancia no se puede eliminar cuando está habilitada la protección contra eliminación. Consulte Eliminación de una instancia de base de datos.

- DomainMemberships: se trata de una matriz de objetos DomainMembership.

  No compatible

- EnabledCloudwatchLogsExports: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que esta instancia de base de datos está configurada para exportar a CloudWatch Logs.

- Endpoint: se trata de un objeto Punto de conexión.

  Especifica el punto de conexión de conexión.

- Engine: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se va a usar para esta instancia de base de datos.

- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- EnhancedMonitoringResourceArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del flujo de registros de Amazon CloudWatch Logs que recibe los datos de métricas de monitorización mejorada para la instancia de base de datos.

- IAMDatabaseAuthenticationEnabled: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True si la autenticación de Amazon Identity and Access Management (IAM) está habilitada; de lo contrario, el valor es False.

- InstanceCreateTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la fecha y hora en que se creó la instancia de base de datos.

- Iops: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el valor de las IOPS provisionadas (las operaciones de E/S por segundo).

- KmsKeyId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- LatestRestorableTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- LicenseModel: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Información del modelo de licencias de esta instancia de base de datos.

- MonitoringInterval: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El intervalo, en segundos, entre puntos cuando se recopila las métricas de monitorización mejoradas para la instancia de base de datos.

- MonitoringRoleArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del rol de IAM que permite a Neptune enviar métricas de monitorización mejoradas a Amazon CloudWatch Logs.

- MultiAZ: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instancia de base de datos es una implementación Multi-AZ.

- PendingModifiedValues: se trata de un objeto PendingModifiedValues.

  Especifica que los cambios a la instancia de base de datos están pendientes. Este elemento sólo se incluye cuando los cambios están pendientes. Los cambios específicos se identifican por subelementos.

- PreferredBackupWindow: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- PromotionTier: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Un valor que especifica el orden en que se promueve una réplica de lectura a la instancia principal tras un error de la instancia principal existente.
- PubliclyAccessible: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Hay que dejar de utilizar este marcador.
- ReadReplicaDBClusterIdentifiers: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de los clústeres de base de datos que son réplicas de lectura de esta instancia de base de datos.
- ReadReplicaDBInstanceIdentifiers: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con esta instancia de base de datos.
- ReadReplicaSourceDBInstanceIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el identificador de la instancia de base de datos de origen si esta instancia de base de datos es una réplica de lectura.
- SecondaryAvailabilityZone: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si está presente, especifica el nombre de la zona de disponibilidad secundaria para una instancia de base de datos compatible con Multi-AZ.
- StatusInfos: se trata de una matriz de objetos DBInstanceStatusInfo.

  El estado de una réplica de lectura. Si la instancia no es una réplica de lectura, está en blanco.
- StorageEncrypted: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  No es compatible: el cifrado para las instancias de base de datos lo administra el clúster de base de datos.

- StorageType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de almacenamiento asociado con la instancia de base de datos.
- TdeCredentialArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del almacén de claves con el que se asocia la instancia para el cifrado de TDE.
- Timezone: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.
- VpcSecurityGroups: se trata de una matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de elementos de grupos de seguridad de VPC a la que pertenece la instancia de base de datos.

`DBInstance` se utiliza como el elemento de respuesta para:

- CreateDBInstance
- DeleteDBInstance
- ModifyDBInstance
- RebootDBInstance

DBInstanceStatusInfo (estructura)

Proporciona una lista de información de estado para una instancia de base de datos.

Campos

- Message: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Detalles del error si hay un error para la instancia. Si la instancia no se encuentra en un estado de error, este valor aparece en blanco.
- Normal: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Valor booleano que es "true" si la instancia funciona con normalidad o "false" si la instancia se encuentra en un estado de error.
- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Estado de la instancia de base de datos. Para obtener un StatusType de réplica de lectura, los valores pueden ser replicando, error, detenido o terminado.

- StatusType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Este valor es actualmente una "replicación de lectura".

OrderableDBInstanceOption (estructura)

Contiene una lista de las opciones disponibles para una instancia de base de datos.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called “DescribeOrderableDBInstanceOptions”.

Campos

- AvailabilityZones: se trata de una matriz de objetos AvailabilityZone.

  Una lista de zonas de disponibilidad para una instancia de base de datos.
- DBInstanceClass: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clase de instancia de base de datos de la instancia de base de datos.
- Engine: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de motor de una instancia de base de datos.
- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión de motor de una instancia de base de datos.
- LicenseModel: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El modelo de licencia de la instancia de base de datos.
- MaxIopsPerDbInstance: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Máximo total de IOPS provisionadas para una instancia de base de datos.
- MaxIopsPerGib: se trata de un DoubleOptional, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

  Máximo de IOPS provisionadas por GiB para una instancia de base de datos.

- MaxStorageSize: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Tamaño máximo de almacenamiento para una instancia de base de datos
- MinIopsPerDbInstance: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Mínimo total de IOPS provisionadas para una instancia de base de datos.
- MinIopsPerGib: se trata de un DoubleOptional, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

  Mínimo de IOPS provisionadas por GiB para una instancia de base de datos.
- MinStorageSize: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Tamaño mínimo de almacenamiento para una instancia de base de datos
- MultiAZCapable: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si una instancia de base de datos admite Multi-AZ.
- ReadReplicaCapable: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si una instancia de base de datos puede tener una réplica de lectura.
- StorageType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.
- SupportsEnhancedMonitoring: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si una instancia de base de datos admite la monitorización mejorada a intervalos de 1 a 60 segundos.
- SupportsGlobalDatabases: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si puede utilizar las bases de datos globales de Neptune con una combinación específica de otros atributos del motor de base de datos.
- SupportsIAMDatabaseAuthentication: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

Indica si una instancia de base de datos admite autenticación de base de datos de IAM.

- SupportsIops: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si una instancia de base de datos admite IOPS provisionadas.

- SupportsStorageEncryption: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si una instancia de base de datos admite almacenamiento cifrado.

- Vpc: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si una instancia de base de datos se encuentra en una VPC.

PendingModifiedValues (estructura)

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "ModifyDBInstance".

Campos

- AllocatedStorage: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Contiene el tamaño nuevo de `AllocatedStorage` correspondiente a la instancia de base de datos que se aplicará o que se está aplicando.

- BackupRetentionPeriod: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días pendientes durante los que se conservan las copias de seguridad automatizadas.

- CACertificateIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador del certificado de CA para la instancia de base de datos.

- DBInstanceClass: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene la nueva `DBInstanceClass` correspondiente a la instancia de base de datos que se aplicará o que se está aplicando.

- DBInstanceIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene la nueva `DBInstanceIdentifier` correspondiente a la instancia de base de datos que se aplicará o que se está aplicando.
- DBSubnetGroupName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El grupo de subredes de base de datos nuevo de la instancia de base de datos.
- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.
- Iops: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el valor nuevo de IOPS provisionadas correspondiente a la instancia de base de datos que se aplicará o que se está aplicando.
- MultiAZ: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que la instancia de base de datos Single-AZ va a cambiar a una implementación Multi-AZ.
- PendingCloudwatchLogsExports: se trata de un objeto PendingCloudwatchLogsExports.

  Esta estructura `PendingCloudwatchLogsExports` especifica los cambios pendientes en los que los registros de CloudWatch están habilitados y cuáles están deshabilitados.
- Port: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto pendiente para la instancia de base de datos.
- StorageType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

ValidStorageOptions (estructura)

No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

Campos

- IopsToStorageRatio: se trata de una matriz de objetos DoubleRange.

No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

- ProvisionedIops: se trata de una matriz de objetos Rango.

  No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

- StorageSize: se trata de una matriz de objetos Rango.

  No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

- StorageType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No se usa. En Neptune, el tipo de almacenamiento se administra en el nivel de clúster de base de datos.

ValidDBInstanceModificationsMessage (estructura)

Información sobre las modificaciones válidas que puede realizar a la instancia de base de datos. Contiene el resultado de una llamada correcta a la acción the section called "DescribeValidDBInstanceModifications". Puede utilizar esta información cuando llama a the section called "ModifyDBInstance".

Campos

- Storage: se trata de una matriz de objetos ValidStorageOptions.

  Opciones de almacenamiento válidas para su instancia de base de datos.

`ValidDBInstanceModificationsMessage` se utiliza como el elemento de respuesta para:

- DescribeValidDBInstanceModifications

API de parámetros de Neptune

Actions:

- CopyDBParameterGroup (acción)

- CopyDBClusterParameterGroup (acción)
- CreateDBParameterGroup (acción)
- CreateDBClusterParameterGroup (acción)
- DeleteDBParameterGroup (acción)
- DeleteDBClusterParameterGroup (acción)
- ModifyDBParameterGroup (acción)
- ModifyDBClusterParameterGroup (acción)
- ResetDBParameterGroup (acción)
- ResetDBClusterParameterGroup (acción)
- DescribeDBParameters (acción)
- DescribeDBParameterGroups (acción)
- DescribeDBClusterParameters (acción)
- DescribeDBClusterParameterGroups (acción)
- DescribeEngineDefaultParameters (acción)
- DescribeEngineDefaultClusterParameters (acción)

Estructuras:

- Parameter (estructura)
- DBParameterGroup (estructura)
- DBClusterParameterGroup (estructura)
- DBParameterGroupStatus (estructura)

CopyDBParameterGroup (acción)

> El nombre de la AWS CLI para esta API es: `copy-db-parameter-group`.

Copia el grupo de parámetros de base de datos especificado.

Solicitud

- SourceDBParameterGroupIdentifier (en la CLI: `--source-db-parameter-group-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El identificador o ARN para el grupo de parámetros de base de datos de origen. Para obtener más información acerca de cómo crear un ARN, consulte Creación de un nombre de recurso de Amazon (ARN).

Restricciones:

- Debe especificar un grupo de parámetros de base de datos válido.
- Debe especificar un identificador de grupo de parámetros de base de datos válido, por ejemplo `my-db-param-group`, o un ARN válido.

- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar al grupo de parámetros de base de datos copiado.

- TargetDBParameterGroupDescription (en la CLI: `--target-db-parameter-group-description`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Descripción del grupo de parámetros de base de datos copiado.

- TargetDBParameterGroupIdentifier (en la CLI: `--target-db-parameter-group-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador para el grupo de parámetros de base de datos copiado.

  Restricciones:

  - No puede ser nulo ni estar vacío o en blanco.
  - Deben contener de 1 a 255 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

  Ejemplo: `my-db-parameter-group`

Respuesta

Contiene los detalles de un grupo de parámetros de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBParameterGroups".

- DBParameterGroupArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para el grupo de parámetros de base de datos.

- DBParameterGroupFamily: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre de la familia del grupo de parámetros de base de datos con el que este grupo de parámetros de base de datos es compatible.
- DBParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de base de datos.
- Description: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción especificada por el usuario para este grupo de parámetros de base de datos.

Errores

- DBParameterGroupNotFoundFault
- DBParameterGroupAlreadyExistsFault
- DBParameterGroupQuotaExceededFault

CopyDBClusterParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `copy-db-cluster-parameter-group`.

Copia el grupo de parámetros de clúster de base de datos especificado.

Solicitud

- SourceDBClusterParameterGroupIdentifier (en la CLI: `--source-db-cluster-parameter-group-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador o Nombre de recurso de Amazon (ARN) para el grupo de parámetros de clúster de base de datos de origen. Para obtener más información acerca de cómo crear un ARN, consulte Creación de un nombre de recurso de Amazon (ARN).

  Restricciones:

  - Debe especificar un grupo de parámetros de clúster de base de datos válido.

  - Si el grupo de parámetros de clúster de base de datos de origen está en la misma región de Amazon que la copia, especifique un identificador de grupos de parámetros de base de datos válido, por ejemplo `my-db-cluster-param-group`, o un ARN válido.
  - Si el grupo de parámetros de base de datos de origen está en una región de Amazon diferente a la de la copia, especifique un ARN de grupo de parámetros de clúster de base de datos válido, por ejemplo `arn:aws:rds:us-east-1:123456789012:cluster-pg:custom-cluster-group1`.
- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar al grupo de parámetros de clúster de base de datos copiado.
- TargetDBClusterParameterGroupDescription (en la CLI: `--target-db-cluster-parameter-group-description`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Descripción del grupo de parámetros de clúster de base de datos copiado.
- TargetDBClusterParameterGroupIdentifier (en la CLI: `--target-db-cluster-parameter-group-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador para el grupo de parámetros de clúster de base de datos copiado.

  Restricciones:
  - No puede ser nulo ni estar vacío o en blanco
  - Deben contener de 1 a 255 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

  Ejemplo: `my-cluster-param-group1`

Respuesta

Contiene los detalles de un grupo de parámetros de clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBClusterParameterGroups".

- DBClusterParameterGroupArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para el grupo de parámetros de clúster de base de datos.

- DBClusterParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de clúster de base de datos.

- DBParameterGroupFamily: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre de la familia del grupo de parámetros de base de datos con el que este grupo de parámetros de clúster de base de datos es compatible.

- Description: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción especificada por el usuario para este grupo de parámetros de clúster de base de datos.

Errores

- DBParameterGroupNotFoundFault
- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `create-db-parameter-group`.

Crea un nuevo grupo de parámetros de base de datos.

Un grupo de parámetros de base de datos se crea inicialmente con los parámetros predeterminados para el motor de base de datos utilizado por la instancia de base de datos. Para proporcionar valores personalizados para cualquiera de los parámetros, debe modificar el grupo después de haberlo creado mediante ModifyDBParameterGroup. Una vez que haya creado un grupo de parámetros de base de datos, tiene que asociarlo a su instancia de base de datos mediante ModifyDBInstance.
Al asociar un nuevo grupo de parámetros de base de datos con una instancia de base de datos en ejecución, debe reiniciar la instancia de base de datos sin conmutación por error para el nuevo grupo de parámetros de base de datos y la configuración asociada para que surta efecto.

> **Important**
>
> Después de crear un grupo de parámetros de base de datos, debe esperar al menos 5 minutos antes de crear la primera instancia de base de datos que utilice ese grupo de parámetros de base de datos como grupo de parámetros predeterminado. Esto permite a Amazon Neptune finalizar por completo la acción de creación antes de que el grupo de parámetros se use de forma predeterminada para una instancia de base de datos nueva. Esto es especialmente importante para los parámetros que son críticos al crear la base de datos predeterminada de una instancia de base de datos, como el conjunto de caracteres de la base de datos predeterminada, definido por el parámetro `character_set_database`. Puede utilizar la opción Parameter Groups (Grupos de parámetros) de la consola de Amazon Neptune o el comando DescribeDBParameters para comprobar que se ha creado o modificado el grupo de parámetros de base de datos.

Solicitud

- DBParameterGroupFamily (en la CLI: `--db-parameter-group-family`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la familia del grupo de parámetros de base de datos. Un grupo de parámetros de base de datos puede asociarse exclusivamente con una familia de grupos de parámetros de base de datos y solo puede aplicarse a una instancia de base de datos que ejecuta un motor de base de datos y la versión del motor que es compatible con esa familia de grupos de parámetros de base de datos.
- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de base de datos.

  Restricciones:

  - Debe tener de 1 a 255 letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

> **Note**
>
> Este valor se almacena como una cadena en minúsculas.

- Description (en la CLI: `--description`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Descripción del grupo de parámetros de base de datos.
- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar al grupo de parámetros de base de datos nuevo.

Respuesta

Contiene los detalles de un grupo de parámetros de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBParameterGroups".

- DBParameterGroupArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el grupo de parámetros de base de datos.
- DBParameterGroupFamily: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre de la familia del grupo de parámetros de base de datos con el que este grupo de parámetros de base de datos es compatible.
- DBParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de base de datos.
- Description: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción especificada por el usuario para este grupo de parámetros de base de datos.

Errores

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBClusterParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `create-db-cluster-parameter-group`.

Crear un nuevo grupo de parámetros del clúster de base de datos.

Los parámetros de un grupo de parámetros de clúster de base de datos se aplican a todas las instancias de un clúster de base de datos.

Un grupo de parámetros de clúster de base de datos se crea inicialmente con los parámetros predeterminados para el motor de base de datos utilizado por las instancias del clúster de base de datos. Para proporcionar valores personalizados para cualquiera de los parámetros, debe modificar el grupo después de haberlo creado mediante the section called “ModifyDBClusterParameterGroup”. Una vez que haya creado un grupo de parámetros de clúster de base de datos, tiene que asociarlo a su clúster de base de datos mediante the section called “ModifyDBCluster”. Al asociar un nuevo grupo de parámetros de clúster de base de datos con un clúster de base de datos en ejecución, debe reiniciar las instancias de base de datos en el clúster de base de datos sin conmutación por error para el nuevo grupo de parámetros del clúster de base de datos y la configuración asociada para que surta efecto.

Important

Después de crear un grupo de parámetros de clúster de base de datos, debe esperar al menos 5 minutos antes de crear el primer clúster de base de datos que utilice ese grupo de parámetros de clúster de base de datos como grupo de parámetros predeterminado. Esto permite a Amazon Neptune finalizar por completo la acción de creación antes de que el grupo de parámetros de clúster de base de datos se use de forma predeterminada para un clúster de base de datos nuevo. Esto es especialmente importante para los parámetros que son críticos al crear la base de datos predeterminada de un clúster de base de datos, como el conjunto de caracteres de la base de datos predeterminada, definido por el parámetro `character_set_database`. Puede utilizar la opción Parameter Groups (Grupos de parámetros) de la consola de Amazon Neptune o el comando the section called “DescribeDBClusterParameters” para comprobar que se ha creado o modificado el grupo de parámetros de clúster de base de datos.

Solicitud

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de clúster de base de datos.

  Restricciones:

  - Debe coincidir con el nombre de un DBClusterParameterGroup existente.

  > **Note**
  >
  > Este valor se almacena como una cadena en minúsculas.

- DBParameterGroupFamily (en la CLI: `--db-parameter-group-family`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la familia del grupo de parámetros de clúster de base de datos. Un grupo de parámetros de clúster de base de datos puede asociarse exclusivamente con una familia de grupos de parámetros de clúster de base de datos y solo puede aplicarse a un clúster de base de datos que ejecuta un motor de base de datos y la versión del motor que es compatible con esa familia de grupos de parámetros de clúster de base de datos.

- Description (en la CLI: `--description`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Descripción del grupo de parámetros de clúster de base de datos.

- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar al grupo de parámetros de clúster de base de datos nuevo.

Respuesta

Contiene los detalles de un grupo de parámetros de clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBClusterParameterGroups".

- DBClusterParameterGroupArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el grupo de parámetros de clúster de base de datos.

- DBClusterParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de clúster de base de datos.
- DBParameterGroupFamily: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre de la familia del grupo de parámetros de base de datos con el que este grupo de parámetros de clúster de base de datos es compatible.
- Description: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción especificada por el usuario para este grupo de parámetros de clúster de base de datos.

Errores

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

DeleteDBParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `delete-db-parameter-group`.

Elimina un determinado DBParameterGroup. El DBParameterGroup que se va a eliminar no puede asociarse a cualquier instancia de base de datos.

Solicitud

- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de base de datos.

  Restricciones:
  - Debe ser el nombre de un grupo de parámetros de base de datos existente.
  - No puede eliminar un grupo de parámetros de base de datos predeterminado.
  - No se puede asociar con cualquier instancia de base de datos

Respuesta

- Sin parámetros de respuesta.

Errores

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DeleteDBClusterParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `delete-db-cluster-parameter-group`.

Elimina un determinado grupo de parámetros del clúster de base de datos. El grupo de parámetros de clúster de base de datos que se va a eliminar no puede asociarse a cualquier clúster de base de datos.

Solicitud

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de clúster de base de datos.

  Restricciones:

  - Debe ser el nombre de un grupo de parámetros de clúster de base de datos existente.
  - No puede eliminar un grupo de parámetros de clúster de base de datos predeterminado.
  - No se puede asociar con cualquier clúster de base de datos.

Respuesta

- Sin parámetros de respuesta.

Errores

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ModifyDBParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `modify-db-parameter-group`.

Modifica los parámetros de un grupo de parámetros de base de datos. Para modificar más de un parámetro, envíe una lista de los siguientes: `ParameterName`, `ParameterValue` y `ApplyMethod`. Se pueden modificar un máximo de 20 parámetros en una única solicitud.

Note

Los cambios realizados en los parámetros dinámicos se aplican inmediatamente. Los cambios en los parámetros estáticos requieren un reinicio sin conmutación por error en la instancia de base de datos asociada con el grupo de parámetros antes de que el cambio surta efecto.

Important

Después de modificar un grupo de parámetros de base de datos, debe esperar al menos 5 minutos antes de crear la primera instancia de base de datos que utilice ese grupo de parámetros de base de datos como grupo de parámetros predeterminado. Esto permite a Amazon Neptune finalizar por completo la acción de modificación antes de que el grupo de parámetros se use de forma predeterminada para una instancia de base de datos nueva. Esto es especialmente importante para los parámetros que son críticos al crear la base de datos predeterminada de una instancia de base de datos, como el conjunto de caracteres de la base de datos predeterminada, definido por el parámetro `character_set_database`. Puede utilizar la opción Parameter Groups (Grupos de parámetros) de la consola de Amazon Neptune o el comando DescribeDBParameters para comprobar que se ha creado o modificado el grupo de parámetros de base de datos.

Solicitud

- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de base de datos.

  Restricciones:

  - Si se proporciona, debe coincidir con el nombre de un DBParameterGroup existente.
- Parameters (en la CLI: `--parameters`) obligatorio: una matriz de objetos Parámetro.

  Una matriz de los nombres de parámetros, valores y el método de aplicación para la actualización del parámetro. Deben proporcionarse al menos un nombre de parámetro, valor y método de aplicación; los argumentos subsiguientes son opcionales. Se pueden modificar un máximo de 20 parámetros en una única solicitud.

  Valores válidos (para el método de aplicación): `immediate | pending-reboot`

  > **Note**
  >
  > Puede utilizar el valor inmediato solo con parámetros dinámicos. Puede utilizar el valor pendiente de reinicio para parámetros dinámicos y estáticos, los cambios se aplican al reiniciar la instancia de base de datos sin conmutación por error.

Respuesta

- DBParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de base de datos.

Errores

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ModifyDBClusterParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `modify-db-cluster-parameter-group`.

Modifica los parámetros de un grupo de parámetros del clúster de base de datos. Para modificar más de un parámetro, envíe una lista de los siguientes: `ParameterName`, `ParameterValue` y `ApplyMethod`. Se pueden modificar un máximo de 20 parámetros en una única solicitud.

> **Note**
>
> Los cambios realizados en los parámetros dinámicos se aplican inmediatamente. Los cambios en los parámetros estáticos requieren un reinicio sin conmutación por error en el clúster de base de datos asociado con el grupo de parámetros antes de que el cambio surta efecto.

> **Important**
>
> Después de crear un grupo de parámetros de clúster de base de datos, debe esperar al menos 5 minutos antes de crear el primer clúster de base de datos que utilice ese grupo de parámetros de clúster de base de datos como grupo de parámetros predeterminado. Esto permite a Amazon Neptune finalizar por completo la acción de creación antes de que el grupo de parámetros se use de forma predeterminada para un nuevo clúster de base de datos. Esto es especialmente importante para los parámetros que son críticos al crear la base de datos predeterminada de un clúster de base de datos, como el conjunto de caracteres de la base de datos predeterminada, definido por el parámetro `character_set_database`. Puede utilizar la opción Parameter Groups (Grupos de parámetros) de la consola de Amazon Neptune o el comando the section called "DescribeDBClusterParameters" para comprobar que se ha creado o modificado el grupo de parámetros de clúster de base de datos.

Solicitud

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de clúster de base de datos que se va a modificar.
- Parameters (en la CLI: `--parameters`) obligatorio: una matriz de objetos Parámetro.

  Un lista de parámetros en el grupo de parámetros de clúster de base de datos que se va a modificar.

Respuesta

- DBClusterParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de clúster de base de datos.

  Restricciones:

  - Debe tener de 1 a 255 letras o números.
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

  > **Note**
  >
  > Este valor se almacena como una cadena en minúsculas.

Errores

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ResetDBParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `reset-db-parameter-group`.

Modifica los parámetros de un grupo de parámetros de base de datos al valor predeterminado del motor/sistema. Para restablecer parámetros específicos, proporcione una lista de lo siguiente: `ParameterName` y `ApplyMethod`. Para restablecer el grupo de parámetros de base datos completo, especifique el nombre de `DBParameterGroup` y los parámetros de `ResetAllParameters`. Cuando restablece todo el grupo, los parámetros dinámicos se actualizan de forma inmediata y los parámetros estáticos se establecen en `pending-reboot` para su aplicación la próxima vez que se reinicie la instancia de base de datos o en la siguiente solicitud `RebootDBInstance`.

Solicitud

- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El nombre del grupo de parámetros de base de datos.

Restricciones:

- Debe coincidir con el nombre de un DBParameterGroup existente.

- Parameters (en la CLI: `--parameters`): una matriz de objetos Parámetro.

  Para restablecer el grupo de parámetros de base datos completo, especifique el nombre de `DBParameterGroup` y los parámetros de `ResetAllParameters`. Para restablecer parámetros específicos, proporcione una lista de lo siguiente: `ParameterName` y `ApplyMethod`. Se pueden modificar un máximo de 20 parámetros en una única solicitud.

  Valores válidos (para Aplique el método): `pending-reboot`

- ResetAllParameters (en la CLI: `--reset-all-parameters`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si se deben restablecer todos los parámetros (`true`) o no (`false`) del grupo de parámetros de base de datos a sus valores predeterminados.

  Valor predeterminado: `true`

Respuesta

- DBParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de base de datos.

Errores

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ResetDBClusterParameterGroup (acción)

El nombre de la AWS CLI para esta API es: `reset-db-cluster-parameter-group`.

Modifica los parámetros de un grupo de parámetros del clúster de base de datos al valor predeterminado. Para restablecer parámetros específicos, envíe una lista de lo siguiente:

`ParameterName` y `ApplyMethod`. Para restablecer el grupo de parámetros de clúster de base datos completo, especifique el `DBClusterParameterGroupName` y los parámetros de `ResetAllParameters`.

Cuando restablece todo el grupo, los parámetros dinámicos se actualizan de forma inmediata y los parámetros estáticos se establecen en `pending-reboot` para su aplicación la próxima vez que se reinicie la instancia de base de datos o en la siguiente solicitud the section called "RebootDBInstance". Debe llamar a the section called "RebootDBInstance" para cada instancia de base de datos en el clúster de base de datos al que desee aplicar el parámetro estático actualizado.

Solicitud

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de clúster de base de datos que se va a restablecer.

- Parameters (en la CLI: `--parameters`): una matriz de objetos Parámetro.

  Una lista de nombres de parámetros en el grupo de parámetros de clúster de base de datos que se va a restablecer a los valores predeterminados. No puede utilizar este parámetro si el parámetro `ResetAllParameters` está establecido en `true`.

- ResetAllParameters (en la CLI: `--reset-all-parameters`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que se establece en `true` para restablecer todos los parámetros en el grupo de parámetros de clúster de base de datos a sus valores predeterminados, y en `false` en caso contrario. No puede utilizar este parámetro si hay una lista de nombres de parámetros especificados para el parámetro `Parameters`.

Respuesta

- DBClusterParameterGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros de clúster de base de datos.

  Restricciones:

  - Debe tener de 1 a 255 letras o números.

- El primer carácter debe ser una letra.
- No puede terminar por un guion ni contener dos guiones consecutivos.

> **Note**
>
> Este valor se almacena como una cadena en minúsculas.

Errores

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DescribeDBParameters (acción)

El nombre de la AWS CLI para esta API es: `describe-db-parameters`.

Devuelve la lista detallada de parámetros para un grupo de parámetros de base de datos determinado.

Solicitud

- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de un grupo de parámetros de base de datos específico para el que devolver detalles.

  Restricciones:

  - Si se proporciona, debe coincidir con el nombre de un DBParameterGroup existente.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeDBParameters` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.

- Source (en la CLI: `--source`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los tipos de parámetros que se devuelven.

  De forma predeterminada: se devuelven todos los tipos de parámetros

  Valores válidos: `user | system | engine-default`

Respuesta

- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- Parameters: matriz de objetos Parámetro.

  Una lista de valores de the section called “Parámetro”.

Errores

- DBParameterGroupNotFoundFault

DescribeDBParameterGroups (acción)

El nombre de la AWS CLI para esta API es: `describe-db-parameter-groups`.

Devuelve una lista de descripciones de `DBParameterGroup`. Si se especifica un `DBParameterGroupName`, la lista contendrá únicamente la descripción del grupo de parámetros de base de datos especificado.

Solicitud

- DBParameterGroupName (en la CLI: `--db-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de un grupo de parámetros de base de datos específico para el que devolver detalles.

  Restricciones:

  - Si se suministra, debe coincidir con el nombre de un DBClusterParameterGroup existente.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeDBParameterGroups` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.

Respuesta

- DBParameterGroups: matriz de objetos DBParameterGroup.

  Una lista de instancias the section called “DBParameterGroup”.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un token de paginación opcional proporcionado por una solicitud anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

Errores

- DBParameterGroupNotFoundFault

DescribeDBClusterParameters (acción)

El nombre de la AWS CLI para esta API es: `describe-db-cluster-parameters`.

Devuelve la lista detallada de parámetros para un grupo de parámetros de clúster de base de datos en particular.

Solicitud

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de un grupo de parámetros de clúster de base de datos específico para el que devolver detalles de parámetros.

  Restricciones:

  - Si se suministra, debe coincidir con el nombre de un DBClusterParameterGroup existente.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeDBClusterParameters` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

Predeterminado: 100

Restricciones: Minimum 20, máximo 100.

- Source (en la CLI: `--source`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un valor que indica devolver solo parámetros de un origen específico. Los orígenes de parámetros pueden ser `engine`, `service` o `customer`.

Respuesta

- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud DescribeDBClusterParameters anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- Parameters: matriz de objetos Parámetro.

  Proporciona una lista de parámetros para el grupo de parámetros de clúster de base de datos.

Errores

- DBParameterGroupNotFoundFault

DescribeDBClusterParameterGroups (acción)

El nombre de la AWS CLI para esta API es: `describe-db-cluster-parameter-groups`.

Devuelve una lista de descripciones de `DBClusterParameterGroup`. Si se especifica un parámetro `DBClusterParameterGroupName`, la lista contendrá únicamente la descripción del grupo de parámetros de clúster de base de datos especificado.

Solicitud

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de un grupo de parámetros de clúster de base de datos específico para el que devolver detalles.

  Restricciones:

  - Si se suministra, debe coincidir con el nombre de un DBClusterParameterGroup existente.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeDBClusterParameterGroups` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.

Respuesta

- DBClusterParameterGroups: matriz de objetos DBClusterParameterGroup.

  Lista de grupos de parámetros de clúster de base de datos.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeDBClusterParameterGroups` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

Errores

- DBParameterGroupNotFoundFault

DescribeEngineDefaultParameters (acción)

El nombre de la AWS CLI para esta API es: `describe-engine-default-parameters`.

Devuelve la información de parámetros del sistema y del motor predeterminada para el motor de base de datos especificado.

Solicitud

- DBParameterGroupFamily (en la CLI: `--db-parameter-group-family`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la familia del grupo de parámetros de base de datos.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  No se admite actualmente.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeEngineDefaultParameters` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.

Respuesta

Contiene el resultado de una invocación correcta de la acción the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la familia del grupo de parámetros de base de datos a los que se aplican los parámetros predeterminados del motor.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud EngineDefaults anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- Parameters: matriz de objetos Parámetro.

  Contiene una lista de los parámetros predeterminados del motor.

DescribeEngineDefaultClusterParameters (acción)

El nombre de la AWS CLI para esta API es: `describe-engine-default-cluster-parameters`.

Devuelve la información de los parámetros del sistema y del motor predeterminados para el motor de base de datos del clúster.

Solicitud

- DBParameterGroupFamily (en la CLI: `--db-parameter-group-family`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la familia del grupo de parámetros de clúster de base de datos para la que devolver información de los parámetros del motor.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeEngineDefaultClusterParameters` anterior. Si se especifica este parámetro,

la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.

Respuesta

Contiene el resultado de una invocación correcta de la acción the section called "DescribeEngineDefaultParameters".

- DBParameterGroupFamily: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la familia del grupo de parámetros de base de datos a los que se aplican los parámetros predeterminados del motor.

- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud EngineDefaults anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- Parameters: matriz de objetos Parámetro.

  Contiene una lista de los parámetros predeterminados del motor.

Estructuras:

Parameter (estructura)

Especifica un parámetro.

Campos

- AllowedValues: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el rango de valores válido del parámetro.
- ApplyMethod: se trata de un ApplyMethod, del tipo `string`: (una cadena codificada con UTF-8).

  Indica cuándo deben aplicarse actualizaciones de parámetros.
- ApplyType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de parámetros específicos del motor.
- DataType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de datos válidos para el parámetro.
- Description: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona una descripción del parámetro.
- IsModifiable: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el parámetro se puede modificar (`true`) o no (`false`). Algunos parámetros tienen implicaciones de seguridad u operativas que impiden que puedan cambiarse.
- MinimumEngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor más antigua al que se puede aplicar el parámetro.
- ParameterName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del parámetro.
- ParameterValue: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el valor del parámetro.
- Source: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica el origen del valor del parámetro.

DBParameterGroup (estructura)

Contiene los detalles de un grupo de parámetros de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBParameterGroups".

Campos

- DBParameterGroupArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el grupo de parámetros de base de datos.
- DBParameterGroupFamily: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre de la familia del grupo de parámetros de base de datos con el que este grupo de parámetros de base de datos es compatible.
- DBParameterGroupName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de base de datos.
- Description: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción especificada por el usuario para este grupo de parámetros de base de datos.

`DBParameterGroup` se utiliza como el elemento de respuesta para:

- CopyDBParameterGroup
- CreateDBParameterGroup

DBClusterParameterGroup (estructura)

Contiene los detalles de un grupo de parámetros de clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBClusterParameterGroups".

Campos

- DBClusterParameterGroupArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para el grupo de parámetros de clúster de base de datos.

- DBClusterParameterGroupName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del grupo de parámetros de clúster de base de datos.

- DBParameterGroupFamily: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre de la familia del grupo de parámetros de base de datos con el que este grupo de parámetros de clúster de base de datos es compatible.

- Description: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción especificada por el usuario para este grupo de parámetros de clúster de base de datos.

`DBClusterParameterGroup` se utiliza como el elemento de respuesta para:

- CopyDBClusterParameterGroup
- CreateDBClusterParameterGroup

DBParameterGroupStatus (estructura)

El estado del grupo de parámetros de base de datos.

Este tipo de datos se usa como elemento de respuesta en las siguientes acciones:

- the section called “CreateDBInstance”
- the section called “DeleteDBInstance”
- the section called “ModifyDBInstance”
- the section called “RebootDBInstance”

Campos

- DBParameterGroupName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El nombre del grupo de parámetros de base de datos.

- ParameterApplyStatus: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de las actualizaciones de parámetros.

API de subred de Neptune

Actions:

- CreateDBSubnetGroup (acción)
- DeleteDBSubnetGroup (acción)
- ModifyDBSubnetGroup (acción)
- DescribeDBSubnetGroups (acción)

Estructuras:

- Subred (estructura)
- DBSubnetGroup (estructura)

CreateDBSubnetGroup (acción)

El nombre de la AWS CLI para esta API es: `create-db-subnet-group`.

Crea un nuevo grupo de subred de base de datos. Los grupos de subred de base de datos deben incluir al menos una subred en al menos dos zonas de disponibilidad de la región de Amazon.

Solicitud

- DBSubnetGroupDescription (en la CLI: `--db-subnet-group-description`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Descripción para el grupo de subredes de base de datos.

- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El nombre del grupo de subredes de base de datos. Este valor se almacena como una cadena en minúsculas.

Restricciones: debe contener un máximo de 255 letras, números, puntos, guiones bajos, espacios o guiones. No debe ser predeterminado.

Ejemplo: `mySubnetgroup`

- SubnetIds (en la CLI: `--subnet-ids`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  ID de subredes de EC2 para el grupo de subred de base de datos.
- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar al grupo de subred de base de datos nuevo.

Respuesta

Contiene los detalles de un grupo de subred de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBSubnetGroups".

- DBSubnetGroupArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el grupo de subred de base de datos.
- DBSubnetGroupDescription: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción del grupo de subred de base de datos.
- DBSubnetGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del grupo de subred de base de datos.
- SubnetGroupStatus: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el estado del grupo de subred de base de datos.
- Subnets: matriz de objetos Subred.

  Contiene una lista de elementos the section called "Subred".
- VpcId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Proporciona el ID de VPC del grupo de subred de base de datos.

Errores

- DBSubnetGroupAlreadyExistsFault
- DBSubnetGroupQuotaExceededFault
- DBSubnetQuotaExceededFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DeleteDBSubnetGroup (acción)

El nombre de la AWS CLI para esta API es: `delete-db-subnet-group`.

Elimina un grupo de subred de base de datos.

> **Note**
>
> El grupo de subred de base de datos especificado no debe estar asociado a cualquier instancia de base de datos.

Solicitud

- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de subred de base de datos que se va a eliminar.

  > **Note**
  >
  > El grupo de subred predeterminado no se puede eliminar.

  Restricciones:

Limitaciones: debe coincidir con el nombre de un DBSubnetGroup existente. No debe ser predeterminado.

Ejemplo: `mySubnetgroup`

Respuesta

- Sin parámetros de respuesta.

Errores

- InvalidDBSubnetGroupStateFault
- InvalidDBSubnetStateFault
- DBSubnetGroupNotFoundFault

ModifyDBSubnetGroup (acción)

El nombre de la AWS CLI para esta API es: `modify-db-subnet-group`.

Modifica un grupo de subred de base de datos existente. Los grupos de subred de base de datos deben incluir al menos una subred en al menos dos zonas de disponibilidad de la región de Amazon.

Solicitud

- DBSubnetGroupDescription (en la CLI: `--db-subnet-group-description`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Descripción para el grupo de subredes de base de datos.
- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de subredes de base de datos. Este valor se almacena como una cadena en minúsculas. El grupo de subred predeterminado no se puede modificar.

  Limitaciones: debe coincidir con el nombre de un DBSubnetGroup existente. No debe ser predeterminado.

  Ejemplo: `mySubnetgroup`

- SubnetIds (en la CLI: `--subnet-ids`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  ID de subredes de EC2 para el grupo de subred de base de datos.

Respuesta

Contiene los detalles de un grupo de subred de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBSubnetGroups".

- DBSubnetGroupArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el grupo de subred de base de datos.
- DBSubnetGroupDescription: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción del grupo de subred de base de datos.
- DBSubnetGroupName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del grupo de subred de base de datos.
- SubnetGroupStatus: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el estado del grupo de subred de base de datos.
- Subnets: matriz de objetos Subred.

  Contiene una lista de elementos the section called "Subred".
- VpcId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el ID de VPC del grupo de subred de base de datos.

Errores

- DBSubnetGroupNotFoundFault
- DBSubnetQuotaExceededFault
- SubnetAlreadyInUse
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DescribeDBSubnetGroups (acción)

El nombre de la AWS CLI para esta API es: `describe-db-subnet-groups`.

Devuelve una lista de descripciones de DBSubnetGroup. Si se especifica un DBSubnetGroupName, la lista contendrá únicamente las descripciones del DBSubnetGroup especificado.

Para obtener información general de los rangos de CIDR, visite el Tutorial de Wikipedia.

Solicitud

- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de subred de base de datos del que desea consultar los detalles.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud DescribeDBSubnetGroups anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.

Respuesta

- DBSubnetGroups: matriz de objetos DBSubnetGroup.

  Una lista de instancias the section called “DBSubnetGroup”.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un token de paginación opcional proporcionado por una solicitud anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

Errores

- DBSubnetGroupNotFoundFault

Estructuras:

Subred (estructura)

Especifica una subred.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBSubnetGroups".

Campos

- SubnetAvailabilityZone: se trata de un objeto AvailabilityZone.

  Especifica la zona de disponibilidad de EC2 donde se encuentra la subred.
- SubnetIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador de la subred.
- SubnetStatus: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado de la subred.

DBSubnetGroup (estructura)

Contiene los detalles de un grupo de subred de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBSubnetGroups".

Campos

- DBSubnetGroupArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El Nombre de recurso de Amazon (ARN) para el grupo de subred de base de datos.

- DBSubnetGroupDescription: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la descripción del grupo de subred de base de datos.

- DBSubnetGroupName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del grupo de subred de base de datos.

- SubnetGroupStatus: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el estado del grupo de subred de base de datos.

- Subnets: se trata de una matriz de objetos Subred.

  Contiene una lista de elementos the section called “Subred”.

- VpcId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el ID de VPC del grupo de subred de base de datos.

`DBSubnetGroup` se utiliza como el elemento de respuesta para:

- CreateDBSubnetGroup
- ModifyDBSubnetGroup

API de instantáneas de Neptune

Acciones:

- CreateDBClusterSnapshot (acción)
- DeleteDBClusterSnapshot (acción)
- CopyDBClusterSnapshot (acción)
- ModifyDBClusterSnapshotAttribute (acción)
- RestoreDBClusterFromSnapshot (acción)
- RestoreDBClusterToPointInTime (acción)

- DescribeDBClusterSnapshots (acción)
- DescribeDBClusterSnapshotAttributes (acción)

Estructuras:

- DBClusterSnapshot (estructura)
- DBClusterSnapshotAttribute (estructura)
- DBClusterSnapshotAttributesResult (estructura)

CreateDBClusterSnapshot (acción)

El nombre de la AWS CLI para esta API es: `create-db-cluster-snapshot`.

Crea una instantánea de un clúster de base de datos.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la instantánea del clúster de base de datos para el que se va a crear una instantánea. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Debe coincidir con el identificador de un DBCluster existente.

  Ejemplo: `my-cluster1`
- DBClusterSnapshotIdentifier (en la CLI: `--db-cluster-snapshot-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la instantánea del clúster de base de datos. Este parámetro se almacena como una cadena en minúsculas.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

Ejemplo: `my-cluster1-snapshot1`

- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar a la instantánea del clúster de base de datos.

Respuesta

Contiene los detalles para una instantánea de clúster de base de datos de Amazon Neptune

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el tamaño de almacenamiento asignado en gibibytes (GiB).
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 donde se pueden restaurar las instancias de la instantánea del clúster de base de datos.
- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).
- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador del clúster de base de datos del clúster de base de datos a partir del cual se creó esta instantánea del clúster de base de datos.
- DBClusterSnapshotArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instantánea del clúster de base de datos.
- DBClusterSnapshotIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador de una instantánea de clúster de base de datos. Debe coincidir con el identificador de una instantánea existente.

  Después de restaurar un clúster de base de datos con un `DBClusterSnapshotIdentifier`, debe especificar el mismo `DBClusterSnapshotIdentifier` para cualquier actualización futura

del clúster de base de datos. Cuando especifica esta propiedad para una actualización, el clúster de base de datos no se restaura de nuevo a partir de la instantánea de base de datos y los datos de la base de datos no se modifican.

Sin embargo, si no especifica `DBClusterSnapshotIdentifier`, se crea un clúster de base de datos vacío y se elimina el clúster de base de datos original. Si especifica una propiedad diferente de la propiedad de restauración de la instantánea anterior, el clúster de base de datos se restaura a partir de la instantánea especificada por `DBClusterSnapshotIdentifier` y se elimina el clúster de base de datos original.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del motor de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la versión del motor de base de datos para esta instantánea del clúster de base de datos.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true, el identificador de la clave de Amazon KMS para la instantánea de clúster de base de datos cifrada.

- LicenseModel: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la información del modelo de licencia para esta instantánea del clúster de base de datos.

- PercentProgress: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el porcentaje de la estimación de los datos que se han transferido.

- Port: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que el clúster de base de datos estaba escuchando en el momento de la instantánea.

- SnapshotCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la hora a la que se tomó la instantánea, en tiempo universal coordinado (UTC).
- SnapshotType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el tipo de instantánea del clúster de base de datos.
- SourceDBClusterSnapshotArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instantánea del clúster de base de datos se ha copiado de una instantánea de clúster de base de datos de origen, el Nombre de recurso de Amazon (ARN) para la instantánea del clúster de base de datos de origen, de lo contrario, un valor nulo.
- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado de esta instantánea del clúster de base de datos.
- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instantánea del clúster de base de datos está cifrado.
- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento asociado a la instantánea del clúster de base de datos.
- VpcId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el ID de la VPC asociado a la instantánea de clúster de base de datos.

Errores

- DBClusterSnapshotAlreadyExistsFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault
- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

DeleteDBClusterSnapshot (acción)

El nombre de la AWS CLI para esta API es: `delete-db-cluster-snapshot`.

Elimina una instantánea de clúster de base de datos. Si se está copiando la instantánea, se termina la operación de copiado.

> **Note**
>
> La instantánea del clúster de base de datos debe encontrarse en el estado `available` para su eliminación.

Solicitud

- DBClusterSnapshotIdentifier (en la CLI: `--db-cluster-snapshot-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la instantánea del clúster de base de datos que se va a eliminar.

  Restricciones: debe ser el nombre de una instantánea del clúster de base de datos existente en el estado `available`.

Respuesta

Contiene los detalles para una instantánea de clúster de base de datos de Amazon Neptune

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el tamaño de almacenamiento asignado en gibibytes (GiB).
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 donde se pueden restaurar las instancias de la instantánea del clúster de base de datos.
- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).
- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el identificador del clúster de base de datos del clúster de base de datos a partir del cual se creó esta instantánea del clúster de base de datos.

- DBClusterSnapshotArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instantánea del clúster de base de datos.

- DBClusterSnapshotIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador de una instantánea de clúster de base de datos. Debe coincidir con el identificador de una instantánea existente.

  Después de restaurar un clúster de base de datos con un `DBClusterSnapshotIdentifier`, debe especificar el mismo `DBClusterSnapshotIdentifier` para cualquier actualización futura del clúster de base de datos. Cuando especifica esta propiedad para una actualización, el clúster de base de datos no se restaura de nuevo a partir de la instantánea de base de datos y los datos de la base de datos no se modifican.

  Sin embargo, si no especifica `DBClusterSnapshotIdentifier`, se crea un clúster de base de datos vacío y se elimina el clúster de base de datos original. Si especifica una propiedad diferente de la propiedad de restauración de la instantánea anterior, el clúster de base de datos se restaura a partir de la instantánea especificada por `DBClusterSnapshotIdentifier` y se elimina el clúster de base de datos original.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del motor de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la versión del motor de base de datos para esta instantánea del clúster de base de datos.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true, el identificador de la clave de Amazon KMS para la instantánea de clúster de base de datos cifrada.

- LicenseModel: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la información del modelo de licencia para esta instantánea del clúster de base de datos.

- PercentProgress: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el porcentaje de la estimación de los datos que se han transferido.

- Port: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que el clúster de base de datos estaba escuchando en el momento de la instantánea.

- SnapshotCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la hora a la que se tomó la instantánea, en tiempo universal coordinado (UTC).

- SnapshotType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el tipo de instantánea del clúster de base de datos.

- SourceDBClusterSnapshotArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instantánea del clúster de base de datos se ha copiado de una instantánea de clúster de base de datos de origen, el Nombre de recurso de Amazon (ARN) para la instantánea del clúster de base de datos de origen, de lo contrario, un valor nulo.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado de esta instantánea del clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instantánea del clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento asociado a la instantánea del clúster de base de datos.

- VpcId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el ID de la VPC asociado a la instantánea de clúster de base de datos.

Errores

- InvalidDBClusterSnapshotStateFault
- DBClusterSnapshotNotFoundFault

CopyDBClusterSnapshot (acción)

El nombre de la AWS CLI para esta API es: `copy-db-cluster-snapshot`.

Copia una instantánea de un clúster de base de datos.

Para copiar una instantánea del clúster de base de datos de una instantánea manual del clúster de base de datos compartida, `SourceDBClusterSnapshotIdentifier` debe ser el Nombre de recurso de Amazon (ARN) de la instantánea del clúster de base de datos compartida.

Solicitud

- CopyTags (en la CLI: `--copy-tags`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  "True" para copiar todas las etiquetas de la instantánea del clúster de base de datos de origen a la instantánea del clúster de base de datos de destino, de lo contrario "false". El valor predeterminado es false.
- KmsKeyId (en la CLI: `--kms-key-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  ID de la clave de Amazon KMS de una instantánea del clúster de base de datos cifrada. El ID de la clave de KMS es el Nombre de recurso de Amazon (ARN), el identificador de la clave de KMS o el alias de la clave de KMS de la clave de cifrado de KMS.

  Si copia una instantánea del clúster de base de datos cifrada desde la cuenta de Amazon, puede especificar un valor para `KmsKeyId` para cifrar la copia con una nueva clave de cifrado de KMS. Si no especifica ningún valor para `KmsKeyId`, la copia de la instantánea del clúster de base de datos se cifra con la misma clave de KMS que la instantánea del clúster de base de datos de origen.

  Si copia una instantánea del clúster de base de datos cifrada que se ha compartido desde otra cuenta de Amazon, debe especificar un valor para `KmsKeyId`.

  Las claves de cifrado KMS son específicas de la región de Amazon en la que se han creado y no se pueden utilizar las claves de cifrado de una región de Amazon en otra región de Amazon.

No se puede cifrar un snapshot de clúster de base de datos sin cifrar durante el proceso de copia. Si intenta copiar un snapshot de clúster de base de datos sin cifrar y especificar un valor para el parámetro KmsKeyId, se devuelve un error.

- PreSignedUrl (en la CLI: `--pre-signed-url`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No se admite actualmente.

- SourceDBClusterSnapshotIdentifier (en la CLI: `--source-db-cluster-snapshot-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la instantánea del clúster de base de datos que se va a copiar. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Se debe especificar una instantánea del sistema válida cuyo estado sea "available" (disponible).
  - Especifique un identificador de instantánea de base de datos válido.

  Ejemplo: `my-cluster-snapshot1`

- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar a la copia de la instantánea del clúster de base de datos nueva.

- TargetDBClusterSnapshotIdentifier (en la CLI: `--target-db-cluster-snapshot-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la nueva instantánea del clúster de base de datos que se va a crear a partir de la instantánea del clúster de base de datos de origen. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra.
  - No puede terminar por un guion ni contener dos guiones consecutivos.

  Ejemplo: `my-cluster-snapshot2`

Respuesta

Contiene los detalles para una instantánea de clúster de base de datos de Amazon Neptune

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBClusterSnapshots".

- AllocatedStorage: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el tamaño de almacenamiento asignado en gibibytes (GiB).

- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 donde se pueden restaurar las instancias de la instantánea del clúster de base de datos.

- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador del clúster de base de datos del clúster de base de datos a partir del cual se creó esta instantánea del clúster de base de datos.

- DBClusterSnapshotArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instantánea del clúster de base de datos.

- DBClusterSnapshotIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador de una instantánea de clúster de base de datos. Debe coincidir con el identificador de una instantánea existente.

  Después de restaurar un clúster de base de datos con un `DBClusterSnapshotIdentifier`, debe especificar el mismo `DBClusterSnapshotIdentifier` para cualquier actualización futura del clúster de base de datos. Cuando especifica esta propiedad para una actualización, el clúster de base de datos no se restaura de nuevo a partir de la instantánea de base de datos y los datos de la base de datos no se modifican.

  Sin embargo, si no especifica `DBClusterSnapshotIdentifier`, se crea un clúster de base de datos vacío y se elimina el clúster de base de datos original. Si especifica una propiedad diferente de la propiedad de restauración de la instantánea anterior, el clúster de base de datos se restaura

a partir de la instantánea especificada por `DBClusterSnapshotIdentifier` y se elimina el clúster de base de datos original.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del motor de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la versión del motor de base de datos para esta instantánea del clúster de base de datos.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true, el identificador de la clave de Amazon KMS para la instantánea de clúster de base de datos cifrada.

- LicenseModel: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la información del modelo de licencia para esta instantánea del clúster de base de datos.

- PercentProgress: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el porcentaje de la estimación de los datos que se han transferido.

- Port: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que el clúster de base de datos estaba escuchando en el momento de la instantánea.

- SnapshotCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la hora a la que se tomó la instantánea, en tiempo universal coordinado (UTC).

- SnapshotType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el tipo de instantánea del clúster de base de datos.

- SourceDBClusterSnapshotArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Si la instantánea del clúster de base de datos se ha copiado de una instantánea de clúster de base de datos de origen, el Nombre de recurso de Amazon (ARN) para la instantánea del clúster de base de datos de origen, de lo contrario, un valor nulo.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado de esta instantánea del clúster de base de datos.
- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instantánea del clúster de base de datos está cifrado.
- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento asociado a la instantánea del clúster de base de datos.
- VpcId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el ID de la VPC asociado a la instantánea de clúster de base de datos.

Errores

- DBClusterSnapshotAlreadyExistsFault
- DBClusterSnapshotNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBClusterSnapshotStateFault
- SnapshotQuotaExceededFault
- KMSKeyNotAccessibleFault

ModifyDBClusterSnapshotAttribute (acción)

El nombre de la AWS CLI para esta API es: `modify-db-cluster-snapshot-attribute`.

Añade un atributo y valores a una instantánea manual del clúster de base de datos o los elimina.

Para compartir una instantánea manual del clúster de base de datos con otras cuentas de Amazon, especifique `restore` como `AttributeName` y utilice el parámetro `ValuesToAdd` para añadir una lista de ID de las cuentas de Amazon autorizadas a restaurar la instantánea manual del clúster de base de datos. Utilice el valor `all` para convertir la instantánea manual del clúster de base de datos en pública, lo que significa que todas las cuentas de Amazon la pueden copiar o restaurar. No añada

el valor `all` a ninguna instantánea manual del clúster de base de datos que incluya información privada que no quiera que esté disponible para todas las cuentas de Amazon. Si una instantánea manual del clúster de base de datos está cifrada, se podrá compartir, pero solo especificando una lista de ID de cuenta de Amazon autorizados para el parámetro `ValuesToAdd`. No se puede utilizar `all` como valor para ese parámetro en este caso.

Para ver qué cuentas de Amazon tienen acceso a la copia o restauración de una instantánea manual del clúster de base de datos o si una instantánea manual del clúster de base de datos es pública o privada, utilice la acción de la API the section called “DescribeDBClusterSnapshotAttributes”.

Solicitud

- AttributeName (en la CLI: `--attribute-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del atributo de la instantánea del clúster de base de datos que se va a modificar.

  Para administrar la autorización para otras cuentas de Amazon para copiar o restaurar una instantánea manual del clúster de base de datos, establezca este valor en `restore`.
- DBClusterSnapshotIdentifier (en la CLI: `--db-cluster-snapshot-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador para la instantánea del clúster de base de datos para el que se modifican los atributos.
- ValuesToAdd (en la CLI: `--values-to-add`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de atributos de la instantánea del clúster de base de datos que desea añadir al atributo especificado por `AttributeName`.

  Para autorizar que otras cuentas de Amazon copien o restauren una instantánea manual del clúster de base de datos, establezca esta lista para incluir uno o varios identificadores de cuentas de Amazon, o `all` para que cualquier cuenta de Amazon pueda restaurar la instantánea manual del clúster de base de datos. No añada el valor `all` a ninguna instantánea manual del clúster de base de datos que incluya información privada que no quiera que esté disponible para todas las cuentas de Amazon.
- ValuesToRemove (en la CLI: `--values-to-remove`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una lista de atributos de la instantánea del clúster de base de datos que desea eliminar del atributo especificado por `AttributeName`.

Para eliminar la autorización para que otras cuentas de Amazon copien o restauren una instantánea manual del clúster de base de datos, establezca esta lista para incluir uno o varios identificadores de cuentas de Amazon, o `all` para eliminar la autorización para que cualquier cuenta de Amazon pueda copiar o restaurar la instantánea del clúster de base de datos. Si especifica `all`, una cuenta de Amazon cuyo ID de cuenta se añade explícitamente al atributo `restore` puede seguir copiando o restaurando una instantánea manual del clúster de base de datos.

Respuesta

Contiene los resultados de una llamada realizada correctamente a la acción de la API the section called “DescribeDBClusterSnapshotAttributes”.

Los atributos de la instantánea manual del clúster de base de datos se utilizan para autorizar a otras cuentas de Amazon la copia o la restauración de una instantánea de clúster de base de datos manual. Para obtener más información, consulte la acción de la API the section called “ModifyDBClusterSnapshotAttribute”.

- DBClusterSnapshotAttributes: matriz de objetos DBClusterSnapshotAttribute.

  La lista de atributos y valores para la instantánea manual del clúster de base de datos.
- DBClusterSnapshotIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la instantánea manual del clúster de base de datos al que se aplican los atributos.

Errores

- DBClusterSnapshotNotFoundFault
- InvalidDBClusterSnapshotStateFault
- SharedSnapshotQuotaExceededFault

RestoreDBClusterFromSnapshot (acción)

El nombre de la AWS CLI para esta API es: `restore-db-cluster-from-snapshot`.

Crea un nuevo clúster de base de datos desde una instantánea de base de datos o una instantánea del clúster de base de datos.

Si se especifica una instantánea de base de datos, el clúster de base de datos de destino se crea a partir de la instantánea de base de datos de origen con una configuración predeterminada y grupo de seguridad predeterminado.

Si se especifica una instantánea del clúster de base de datos, el clúster de base de datos de destino se crea a partir del clúster de base de datos de origen con la misma configuración que el clúster de base de datos de origen original, salvo que el nuevo clúster de base de datos se crea con el grupo de seguridad predeterminado.

Solicitud

- AvailabilityZones (en la CLI: `--availability-zones`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 donde se pueden crear las instancias del clúster de base de datos.
- CopyTagsToSnapshot (en la CLI: `--copy-tags-to-snapshot`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster restaurado de base de datos que se cree.
- DatabaseName (en la CLI: `--database-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No admitido.
- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del clúster de base de datos que se va a crear a partir de la instantánea de base de datos o de la instantánea del clúster de base de datos. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

- Deben contener de 1 a 63 caracteres (letras, números o guiones).
- El primer carácter debe ser una letra
- No puede terminar por un guion ni contener dos guiones consecutivos

Ejemplo: `my-snapshot-id`

- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros del clúster de base de datos que desea asociar a este nuevo clúster de base de datos.

  Restricciones:

  - Si se suministra, debe coincidir con el nombre de un DBClusterParameterGroup existente.
- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de subredes de base de datos que se va a usar para el clúster de base de datos nuevo.

  Restricciones: si se proporciona, debe coincidir con el nombre de un DBSubnetGroup existente.

  Ejemplo: `mySubnetgroup`
- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si el clúster de base de datos tiene habilitada la protección contra eliminación. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación. La protección contra eliminación está deshabilitada de forma predeterminada.
- EnableCloudwatchLogsExports (en la CLI: `--enable-cloudwatch-logs-exports`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La lista de registros que el clúster de base de datos restaurado debe exportar a Amazon CloudWatch logs.
- EnableIAMDatabaseAuthentication (en la CLI: `--enable-iam-database-authentication`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  True para habilitar el mapeo de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos; de lo contrario, el valor es False.

Valor predeterminado: `false`

- Engine (en la CLI: `--engine`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Motor de base de datos que se va a usar para el clúster de base de datos nuevo.

  Predeterminado: igual que el de origen

  Restricción: debe ser compatible con el motor del origen

- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de base de datos que se va a usar para el clúster de base de datos nuevo.

- KmsKeyId (en la CLI: `--kms-key-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la clave de Amazon KMS que se utiliza para restaurar un clúster de base de datos cifrado a partir de una instantánea de base de datos o de clúster de base de datos.

  El identificador de la clave de KMS es el Nombre de recurso de Amazon (ARN) de la clave de cifrado de KMS. Si está restaurando un clúster de base de datos con la misma cuenta de Amazon a la que pertenece la clave de cifrado de KMS utilizada para cifrar el nuevo clúster de base de datos, puede utilizar el alias de la clave de KMS en lugar del ARN de la clave de cifrado de KMS.

  Si no se especifica un valor para el parámetro `KmsKeyId`, ocurrirá lo siguiente:

  - Si se cifra la instantánea de base de datos o de clúster de base de datos `SnapshotIdentifier`, se cifra el clúster de base de datos restaurado con la clave de KMS que se utilizó para cifrar la instantánea de base de datos o de clúster de base de datos.
  - Si la instantánea de base de datos o del clúster de base de datos de `SnapshotIdentifier` no está cifrada, el clúster de base de datos restaurado no está cifrado.

- Port (en la CLI: `--port`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de puerto en el que el clúster de base de datos nuevo acepta las conexiones.

  Limitaciones: el valor debe ser `1150-65535`

  Valor predeterminado: el mismo puerto que el clúster de base de datos original.

- ServerlessV2ScalingConfiguration (en la CLI: `--serverless-v2-scaling-configuration`): un objeto ServerlessV2ScalingConfiguration.

  Incluye la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.
- SnapshotIdentifier (en la CLI: `--snapshot-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identificador de la instantánea de base de datos o instantánea del clúster de base de datos desde la que se debe realizar la restauración.

  Puede utilizar el nombre o el Nombre de recurso de Amazon (ARN) para especificar una instantánea del clúster de base de datos. Sin embargo, puede utilizar únicamente el ARN para especificar una instantánea de base de datos.

  Restricciones:

  - Debe coincidir con el identificador de una instantánea existente.
- StorageType (en la CLI: `--storage-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de almacenamiento que se va a asociar al clúster de base de datos.

  Valores válidos: `standard`, `iopt1`

  Valor predeterminado: `standard`
- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a asignar al clúster de base de datos restaurado.
- VpcSecurityGroupIds (en la CLI: `--vpc-security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de grupos de seguridad de VPC a los que va a pertenecer el clúster de base de datos nuevo.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.
- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.
- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.
- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.
- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.
- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.
- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.
- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.

- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).

- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en *`true`*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.

- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.

- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.

- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.

- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.

- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.

- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

Especifica el puerto en el que escucha el motor de la base de datos.

- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

  Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

    El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.

- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterAlreadyExistsFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterSnapshotNotFoundFault
- InsufficientDBClusterCapacityFault

- InsufficientStorageClusterCapacityFault
- InvalidDBSnapshotStateFault
- InvalidDBClusterSnapshotStateFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- InvalidRestoreFault
- DBSubnetGroupNotFoundFault
- InvalidSubnet
- OptionGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterParameterGroupNotFoundFault

RestoreDBClusterToPointInTime (acción)

El nombre de la AWS CLI para esta API es: `restore-db-cluster-to-point-in-time`.

Restaura un clúster de base de datos a un punto arbitrario en el tiempo. Los usuarios pueden restaurar a cualquier punto en el tiempo antes de `LatestRestorableTime` durante un máximo de `BackupRetentionPeriod` días. El clúster de base de datos de destino se crea a partir del clúster de base de datos de origen con la misma configuración que el clúster de base de datos de origen original, salvo que el nuevo clúster de base de datos se crea con el grupo de seguridad de base de datos predeterminado.

> **Note**
>
> Esta acción solo restaura el clúster de base de datos, no las instancias de base de datos de ese clúster de base de datos. Debe invocar la acción the section called "CreateDBInstance" para crear instancias de base de datos para el clúster de base de datos restaurado, especificando el identificador de dicho clúster de base de datos restaurado en `DBClusterIdentifier`. Solo puede crear instancias de base de datos después de que se haya completado la acción `RestoreDBClusterToPointInTime` y de que el clúster de base de datos esté disponible.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Nombre del nuevo clúster de base de datos que se va a crear.

  Restricciones:

  - Deben contener de 1 a 63 caracteres (letras, números o guiones).
  - El primer carácter debe ser una letra
  - No puede terminar por un guion ni contener dos guiones consecutivos
- DBClusterParameterGroupName (en la CLI: `--db-cluster-parameter-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de parámetros del clúster de base de datos que desea asociar a este nuevo clúster de base de datos.

  Restricciones:

  - Si se suministra, debe coincidir con el nombre de un DBClusterParameterGroup existente.
- DBSubnetGroupName (en la CLI: `--db-subnet-group-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de subredes de base de datos que se debe utilizar para el clúster de base de datos nuevo.

  Restricciones: si se proporciona, debe coincidir con el nombre de un DBSubnetGroup existente.

  Ejemplo: `mySubnetgroup`
- DeletionProtection (en la CLI: `--deletion-protection`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si el clúster de base de datos tiene habilitada la protección contra eliminación. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación. La protección contra eliminación está deshabilitada de forma predeterminada.
- EnableCloudwatchLogsExports (en la CLI: `--enable-cloudwatch-logs-exports`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La lista de registros que el clúster de base de datos restaurado debe exportar a CloudWatch Logs.
- EnableIAMDatabaseAuthentication (en la CLI: `--enable-iam-database-authentication`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

True para habilitar el mapeo de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos; de lo contrario, el valor es False.

Valor predeterminado: `false`

- KmsKeyId (en la CLI: `--kms-key-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la clave de Amazon KMS que se utiliza para restaurar un clúster de base de datos cifrado a partir de un clúster de base de datos cifrado.

  El identificador de la clave de KMS es el Nombre de recurso de Amazon (ARN) de la clave de cifrado de KMS. Si está restaurando un clúster de base de datos con la misma cuenta de Amazon a la que pertenece la clave de cifrado de KMS utilizada para cifrar el nuevo clúster de base de datos, puede utilizar el alias de la clave de KMS en lugar del ARN de la clave de cifrado de KMS.

  Puede restaurar a un nuevo clúster de base de datos y cifrar el nuevo clúster de base de datos con una clave de KMS que es diferente de la clave de KMS utilizada para cifrar el clúster de base de datos de origen. El nuevo clúster de base de datos se cifra con la clave de KMS identificada por el parámetro `KmsKeyId`.

  Si no se especifica un valor para el parámetro `KmsKeyId`, ocurrirá lo siguiente:

  - Si el clúster de base de datos está cifrado, el clúster de base de datos restaurado se cifra usando la misma clave de KMS que se usó para cifrar el clúster de base de datos de origen.
  - Si el clúster de base de datos no está cifrado, el clúster de base de datos restaurado no está cifrado.

  Si `DBClusterIdentifier` se refiere a un clúster de base de datos que no está cifrada, se rechaza la solicitud de restauración.

- Port (en la CLI: `--port`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de puerto en el que el clúster de base de datos nuevo acepta las conexiones.

  Limitaciones: el valor debe ser `1150-65535`

  Valor predeterminado: el mismo puerto que el clúster de base de datos original.

- RestoreToTime (en la CLI: `--restore-to-time`): un TStamp, del tipo: `timestamp` (un punto en el tiempo, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La fecha y la hora a la que se va a restaurar el clúster de base de datos.

  Valores válidos: el valor debe ser una hora en formato de tiempo universal coordinado (UTC)

  Restricciones:

  - Debe ser anterior a la última hora restaurable de la instancia de base de datos
  - Debe especificarse si no se proporciona el parámetro `UseLatestRestorableTime`
  - No se puede especificar si el parámetro `UseLatestRestorableTime` es "true"
  - No se puede especificar si el parámetro `RestoreType` es `copy-on-write`

  Ejemplo: `2015-03-07T23:45:00Z`
- RestoreType (en la CLI: `--restore-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de restauración que se va a realizar. Puede especificar uno de los siguientes valores:

  - `full-copy`: el nuevo clúster de base de datos se restaura como una copia completa del clúster de la base de datos de origen.
  - `copy-on-write`: el nuevo clúster de base de datos se restaura como un clon del clúster de la base de datos de origen.

  Si no especifica un valor `RestoreType`, el nuevo clúster de base de datos se restaura como una copia completa del clúster de la base de datos de origen.
- ServerlessV2ScalingConfiguration (en la CLI: `--serverless-v2-scaling-configuration`): un objeto ServerlessV2ScalingConfiguration.

  Incluye la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.
- SourceDBClusterIdentifier (en la CLI: `--source-db-cluster-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identificador del clúster de base de datos de origen desde la que se va a restaurar el clúster.

Restricciones:

- Debe coincidir con el identificador de un DBCluster existente.

- StorageType (en la CLI: `--storage-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de almacenamiento que se va a asociar al clúster de base de datos.

  Valores válidos: `standard`, `iopt1`

  Valor predeterminado: `standard`

- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que se van a aplicar al clúster de base de datos restaurado.

- UseLatestRestorableTime (en la CLI: `--use-latest-restorable-time`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que se establece en `true` para restaurar el clúster de base de datos a la hora de la última copia de seguridad restaurable y `false` en caso contrario.

  Valor predeterminado: `false`

  Restricciones: no se pueden especificar si se proporciona el parámetro `RestoreToTime`.

- VpcSecurityGroupIds (en la CLI: `--vpc-security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos nuevo.

Respuesta

Contiene los detalles de un clúster de base de datos de Amazon Neptune.

Este tipo de datos se utiliza como un elemento de respuesta en the section called "DescribeDBClusters".

- AllocatedStorage: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  `AllocatedStorage` siempre devuelve 1, ya que el tamaño de almacenamiento del clúster de base de datos de Neptune no es fijo, sino que se ajusta automáticamente según sea necesario.

- AssociatedRoles: matriz de objetos DBClusterRole.

  Proporciona una lista de los roles de Amazon Identity and Access Management (IAM) que están asociados con el clúster de base de datos. Los roles de IAM que están asociados a un clúster de base de datos conceden permiso para que este obtenga acceso a otros servicios de Amazon en su nombre.

- AutomaticRestartTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Hora en la que el clúster de base de datos se reiniciará automáticamente.

- AvailabilityZones: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 en las que se pueden crear instancias en el clúster de base de datos.

- BacktrackConsumedChangeRecords: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.

- BacktrackWindow: un valor LongOptional, del tipo: `long` (valor entero firmado de 64 bits).

  No es compatible con Neptune.

- BackupRetentionPeriod: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el número de días durante los que se conservan las instantáneas de base de datos automáticas.

- Capacity: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  No es compatible con Neptune.

- CloneGroupId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identifica el grupo de clones al que está asociado el clúster de base de datos.

- ClusterCreateTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).

- CopyTagsToSnapshot: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

Si se establece en *true*, las etiquetas se copian en cualquier instantánea del clúster de base de datos que se cree.

- CrossAccountClone: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, el clúster de base de datos se puede clonar en todas las cuentas.

- DatabaseName: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene el nombre de la base de datos inicial de este clúster de base de datos que se proporcionó en el momento de la creación, si se especificó uno cuando se creó el clúster de base de datos. Este mismo nombre se devuelve durante el tiempo que dura el clúster de base de datos.

- DBClusterArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el clúster de base de datos.

- DBClusterIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene un identificador de clúster de base de datos suministrado por el usuario. Este identificador es la clave única que identifica un clúster de base de datos.

- DBClusterMembers: matriz de objetos DBClusterMember.

  Proporciona la lista de instancias que componen el clúster de base de datos.

- DBClusterParameterGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del grupo de parámetros del clúster de base de datos para el clúster de base de datos.

- DbClusterResourceId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador inmutable único de la región de Amazon para el clúster de base de datos. Este identificador se encuentra en las entradas de registro de Amazon CloudTrail siempre que se accede a la clave de Amazon KMS para el clúster de base de datos.

- DBSubnetGroup: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica información sobre el grupo de subred asociado con el clúster de base de datos, incluido el nombre, la descripción y subredes en el grupo de subred.

- DeletionProtection: un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si el clúster de base de datos tiene habilitada la protección contra eliminación o no. La base de datos no se puede eliminar cuando está habilitada la protección contra eliminación.

- EarliestBacktrackTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  No es compatible con Neptune.

- EarliestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- EnabledCloudwatchLogsExports: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de tipos de registro para los que este clúster de base de datos está configurado para exportar a los registros de CloudWatch. Los tipos de registro válidos son: `audit` (para publicar registros de auditoría en CloudWatch) y slowquery (para publicar registros de consultas lentas en CloudWatch). Consulte Publicación de registros de Neptune en Registros de Amazon CloudWatch.

- Endpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el punto de enlace de conexión para la instancia principal del clúster de base de datos.

- Engine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.

- EngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica la versión del motor de base de datos.

- GlobalClusterIdentifier: un GlobalClusterIdentifier, del tipo: `string` (una cadena codificada con UTF-8), no menos de 1 ni superior a 255 ¿st?s, que coincide con esta expresión regular: `[A-Za-z][0-9A-Za-z-:._]*`.

  Incluye un identificador de clúster de base de datos global facilitado por el usuario. Este identificador es la clave única que identifica una base de datos global.

- HostedZoneId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.

- IAMDatabaseAuthenticationEnabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).

- IOOptimizedNextAllowedModificationTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La próxima vez, podrá modificar el clúster de base de datos para utilizar el tipo de almacenamiento `iopt1`.

- KmsKeyId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true (verdadero), el identificador de la clave de Amazon KMS para el clúster de base de datos cifrado.

- LatestRestorableTime: un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la última hora a la que se puede restaurar una base de datos con la restauración de un punto en el tiempo.

- MultiAZ: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos tiene instancias en varias zonas de disponibilidad.

- PendingModifiedValues: objeto ClusterPendingModifiedValues.

  Este tipo de datos se utiliza como un elemento de respuesta en la operación `ModifyDBCluster` e incluye cambios que se aplicarán durante el siguiente periodo de mantenimiento.

- PercentProgress: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el progreso de la operación como porcentaje.

- Port: un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.

- PreferredBackupWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el intervalo de tiempo diario durante el cual se crean copias de seguridad automatizadas si las copias de seguridad automatizadas están habilitadas, de acuerdo con la propiedad `BackupRetentionPeriod`.

- PreferredMaintenanceWindow: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Especifica el intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en el horario universal coordinado (UTC).

- ReaderEndpoint: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El punto de enlace del lector para el clúster de base de datos. El punto de enlace del lector de un clúster de base de datos equilibra la carga de las conexiones entre las réplicas de lectura que están disponibles en un clúster de base de datos. A medida que los clientes solicitan nuevas conexiones al punto de enlace del lector, Neptune distribuye las solicitudes de conexión entre las réplicas de lectura del clúster de base de datos. Esta funcionalidad puede ayudar a equilibrar la carga de trabajo de lectura entre las distintas réplicas de lectura en el clúster de base de datos.

  Si se produce una conmutación por error y la réplica de lectura a la que está conectado se convierte en la nueva instancia principal, la conexión se interrumpe. Para seguir enviando la carga de trabajo de lectura a otras réplicas de lectura del clúster, puede volver a conectarse al punto de enlace del lector.

- ReadReplicaIdentifiers: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Contiene uno o varios identificadores de las réplicas de lectura asociados con este clúster de base de datos.

- ReplicationSourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ReplicationType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  No es compatible con Neptune.

- ServerlessV2ScalingConfiguration: objeto ServerlessV2ScalingConfigurationInfo.

  Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

  Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado actual de este clúster de base de datos.

- StorageEncrypted: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si el clúster de base de datos está cifrado.

- StorageType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento que se utiliza para el clúster de base de datos.

  Valores válidos:

  - **`standard`**: (predeterminado) proporciona un almacenamiento de bases de datos rentable para aplicaciones con un uso de E/S de moderado a pequeño.
  - **`iopt1`**: habilita el almacenamiento optimizado para E/S que está diseñado para satisfacer las necesidades de las cargas de trabajo de gráficos con uso intensivo de operaciones de E/S que requieren precios predecibles con una latencia de E/S baja y un rendimiento de E/S constante.

    El almacenamiento optimizado para E/S de Neptune solo está disponible a partir de la versión 1.3.0.0 del motor.
- VpcSecurityGroups: matriz de objetos VpcSecurityGroupMembership.

  Proporciona una lista de grupos de seguridad de VPC a los que pertenece el clúster de base de datos.

Errores

- DBClusterAlreadyExistsFault
- DBClusterNotFoundFault
- DBClusterQuotaExceededFault
- DBClusterSnapshotNotFoundFault
- DBSubnetGroupNotFoundFault
- InsufficientDBClusterCapacityFault
- InsufficientStorageClusterCapacityFault
- InvalidDBClusterSnapshotStateFault
- InvalidDBClusterStateFault
- InvalidDBSnapshotStateFault
- InvalidRestoreFault
- InvalidSubnet
- InvalidVPCNetworkStateFault
- KMSKeyNotAccessibleFault

- OptionGroupNotFoundFault
- StorageQuotaExceededFault
- DBClusterParameterGroupNotFoundFault

DescribeDBClusterSnapshots (acción)

El nombre de la AWS CLI para esta API es: `describe-db-cluster-snapshots`.

Devuelve información acerca de instantáneas del clúster de base de datos. Esta acción de la API admite la paginación.

Solicitud

- DBClusterIdentifier (en la CLI: `--db-cluster-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID del clúster de base de datos para recuperar la lista de instantáneas del clúster de base de datos. Este parámetro no puede usarse en combinación con el parámetro `DBClusterSnapshotIdentifier`. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones:

  - Si se proporciona, debe coincidir con el identificador de un DBCluster existente.
- DBClusterSnapshotIdentifier (en la CLI: `--db-cluster-snapshot-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador de instantánea de clúster de base de datos específica para describir. Este parámetro no puede usarse en combinación con el parámetro `DBClusterIdentifier`. Este valor se almacena como una cadena en minúsculas.

  Restricciones:

  - Si se proporciona, debe coincidir con el identificador de una DBClusterSnapshot existente.
  - Si este identificador es para una instantánea automatizada, también se debe especificar el parámetro `SnapshotType`.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- IncludePublic (en la CLI: `--include-public`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

True para incluir instantáneas del clúster de base de datos manuales que son públicas y que cualquier cuenta de Amazon puede copiar o restaurar; de lo contrario, el valor es False. El valor predeterminado es `false`. El valor predeterminado es false.

Con la acción de la API the section called “ModifyDBClusterSnapshotAttribute”, es posible compartir una instantánea manual del clúster de base de datos:

- IncludeShared (en la CLI: `--include-shared`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True para incluir instantáneas del clúster de base de datos manual compartidas de otras cuentas de Amazon a las que esta cuenta de Amazon ha dado permiso para copiar o restaurar; de lo contrario, el valor es False. El valor predeterminado es `false`.

  Puede dar permiso a una cuenta de Amazon para restaurar una instantánea manual del clúster de base de datos desde otra cuenta de Amazon mediante la acción the section called “ModifyDBClusterSnapshotAttribute” de la API.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribeDBClusterSnapshots` anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se deben incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Mínimo 20, máximo 100.
- SnapshotType (en la CLI: `--snapshot-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de instantáneas del clúster de base de datos que se van a devolver. Puede especificar uno de los siguientes valores:

- `automated`: permite devolver todas las instantáneas del clúster de base de datos que ha tomado automáticamente Amazon Neptune para mi cuenta de Amazon.
- `manual`: permite devolver todas las instantáneas del clúster de base de datos tomadas por mi cuenta de Amazon.
- `shared`: permite devolver todas las instantáneas del clúster de base de datos manual que se han compartido a mi cuenta de Amazon.
- `public`: devolver todas las instantáneas del clúster de base de datos que se han marcado como públicas.

Si no se especifica ningún valor para `SnapshotType`, se devuelve las instantáneas del clúster de base de datos automatizadas y manuales. Puede incluir instantáneas del clúster de base de datos compartidas con estos resultados estableciendo el parámetro `IncludeShared` en `true`. Puede incluir instantáneas del clúster de base de datos públicas con estos resultados estableciendo el parámetro `IncludePublic` en `true`.

Los parámetros `IncludeShared` y `IncludePublic` no se aplican para los valores de `SnapshotType` de `manual` o `automated`. El parámetro `IncludePublic` no se aplica cuando se establece `SnapshotType` en `shared`. El parámetro `IncludeShared` no se aplica cuando se establece `SnapshotType` en `public`.

Respuesta

- DBClusterSnapshots: matriz de objetos DBClusterSnapshot.

  Proporciona una lista de instantáneas del clúster de base de datos para el usuario.

- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud the section called "DescribeDBClusterSnapshots" anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

Errores

- DBClusterSnapshotNotFoundFault

DescribeDBClusterSnapshotAttributes (acción)

El nombre de la AWS CLI para esta API es: `describe-db-cluster-snapshot-attributes`.

Devuelve una lista de nombres y valores de atributos de instantáneas de clúster de base de datos y valores para una instantánea manual del clúster de base de datos.

Cuando se comparten instantáneas con otras cuentas de Amazon, `DescribeDBClusterSnapshotAttributes` devuelve el atributo `restore` y una lista de ID de las cuentas de Amazon que tienen autorización para copiar o restaurar la instantánea manual del clúster de base de datos. Si `all` se incluye en la lista de valores para el atributo `restore`, la instantánea manual del clúster de base de datos es pública y todas las cuentas de Amazon pueden copiarla o restaurarla.

Para añadir o eliminar el acceso a una cuenta de Amazon para la copia o restauración de una instantánea manual del clúster de base de datos o para convertir la instantánea manual del clúster de base de datos en pública o privada, utilice la acción de la API the section called “ModifyDBClusterSnapshotAttribute”.

Solicitud

- DBClusterSnapshotIdentifier (en la CLI: `--db-cluster-snapshot-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador para la instantánea del clúster de base de datos para el que describir los atributos.

Respuesta

Contiene los resultados de una llamada realizada correctamente a la acción de la API the section called “DescribeDBClusterSnapshotAttributes”.

Los atributos de la instantánea manual del clúster de base de datos se utilizan para autorizar a otras cuentas de Amazon la copia o la restauración de una instantánea de clúster de base de datos manual. Para obtener más información, consulte la acción de la API the section called “ModifyDBClusterSnapshotAttribute”.

- DBClusterSnapshotAttributes: matriz de objetos DBClusterSnapshotAttribute.

  La lista de atributos y valores para la instantánea manual del clúster de base de datos.

- DBClusterSnapshotIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la instantánea manual del clúster de base de datos al que se aplican los atributos.

Errores

- DBClusterSnapshotNotFoundFault

Estructuras:

DBClusterSnapshot (estructura)

Contiene los detalles para una instantánea de clúster de base de datos de Amazon Neptune

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBClusterSnapshots".

Campos

- AllocatedStorage: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el tamaño de almacenamiento asignado en gibibytes (GiB).
- AvailabilityZones: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la lista de zonas de disponibilidad de EC2 donde se pueden restaurar las instancias de la instantánea del clúster de base de datos.
- ClusterCreateTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la hora a la que se creó el clúster de base de datos, en tiempo universal coordinado (UTC).
- DBClusterIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador del clúster de base de datos del clúster de base de datos a partir del cual se creó esta instantánea del clúster de base de datos.

- DBClusterSnapshotArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) de la instantánea del clúster de base de datos.
- DBClusterSnapshotIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el identificador de una instantánea de clúster de base de datos. Debe coincidir con el identificador de una instantánea existente.

  Después de restaurar un clúster de base de datos con un `DBClusterSnapshotIdentifier`, debe especificar el mismo `DBClusterSnapshotIdentifier` para cualquier actualización futura del clúster de base de datos. Cuando especifica esta propiedad para una actualización, el clúster de base de datos no se restaura de nuevo a partir de la instantánea de base de datos y los datos de la base de datos no se modifican.

  Sin embargo, si no especifica `DBClusterSnapshotIdentifier`, se crea un clúster de base de datos vacío y se elimina el clúster de base de datos original. Si especifica una propiedad diferente de la propiedad de restauración de la instantánea anterior, el clúster de base de datos se restaura a partir de la instantánea especificada por `DBClusterSnapshotIdentifier` y se elimina el clúster de base de datos original.
- Engine: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre del motor de base de datos.
- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona la versión del motor de base de datos para esta instantánea del clúster de base de datos.
- IAMDatabaseAuthenticationEnabled: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  True (Verdadero) si la asignación de cuentas de Amazon Identity and Access Management (IAM) a cuentas de base de datos está habilitada; de lo contrario, el valor es false (falso).
- KmsKeyId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si el valor de `StorageEncrypted` es true, el identificador de la clave de Amazon KMS para la instantánea de clúster de base de datos cifrada.
- LicenseModel: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Proporciona la información del modelo de licencia para esta instantánea del clúster de base de datos.

- PercentProgress: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el porcentaje de la estimación de los datos que se han transferido.

- Port: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto que el clúster de base de datos estaba escuchando en el momento de la instantánea.

- SnapshotCreateTime: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Proporciona la hora a la que se tomó la instantánea, en tiempo universal coordinado (UTC).

- SnapshotType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el tipo de instantánea del clúster de base de datos.

- SourceDBClusterSnapshotArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si la instantánea del clúster de base de datos se ha copiado de una instantánea de clúster de base de datos de origen, el Nombre de recurso de Amazon (ARN) para la instantánea del clúster de base de datos de origen, de lo contrario, un valor nulo.

- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el estado de esta instantánea del clúster de base de datos.

- StorageEncrypted: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Especifica si la instantánea del clúster de base de datos está cifrado.

- StorageType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de almacenamiento asociado a la instantánea del clúster de base de datos.

- VpcId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el ID de la VPC asociado a la instantánea de clúster de base de datos.

`DBClusterSnapshot` se utiliza como el elemento de respuesta para:

- CreateDBClusterSnapshot
- CopyDBClusterSnapshot
- DeleteDBClusterSnapshot

DBClusterSnapshotAttribute (estructura)

Contiene el nombre y los valores de un atributo de instantánea manual del clúster de base de datos.

Los atributos de la instantánea manual del clúster de base de datos se utilizan para autorizar a otras cuentas de Amazon la restauración de una instantánea de clúster de base de datos manual. Para obtener más información, consulte la acción de la API the section called "ModifyDBClusterSnapshotAttribute".

Campos

- AttributeName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del atributo de la instantánea manual del clúster de base de datos.

  El atributo denominado `restore` se refiere a la lista de cuentas de Amazon que tienen permiso para copiar o restaurar la instantánea manual del clúster de base de datos. Para obtener más información, consulte la acción de la API the section called "ModifyDBClusterSnapshotAttribute".
- AttributeValues: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los valores del atributo de la instantánea manual del clúster de base de datos.

  Si el campo `AttributeName` se establece en `restore`, este elemento devuelve una lista de ID de las cuentas de Amazon con autorización para copiar o restaurar la instantánea manual del clúster de base de datos. Si el valor `all` está en la lista, la instantánea manual del clúster de base de datos es pública y está disponible para cualquier cuenta de Amazon para su copia o restauración.

DBClusterSnapshotAttributesResult (estructura)

Contiene los resultados de una llamada realizada correctamente a la acción de la API the section called "DescribeDBClusterSnapshotAttributes".

Los atributos de la instantánea manual del clúster de base de datos se utilizan para autorizar a otras cuentas de Amazon la copia o la restauración de una instantánea de clúster de base de datos manual. Para obtener más información, consulte la acción de la API the section called "ModifyDBClusterSnapshotAttribute".

Campos

- DBClusterSnapshotAttributes: se trata de una matriz de objetos DBClusterSnapshotAttribute.

  La lista de atributos y valores para la instantánea manual del clúster de base de datos.
- DBClusterSnapshotIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de la instantánea manual del clúster de base de datos al que se aplican los atributos.

`DBClusterSnapshotAttributesResult` se utiliza como el elemento de respuesta para:

- DescribeDBClusterSnapshotAttributes
- ModifyDBClusterSnapshotAttribute

API de eventos de Neptune

Acciones:

- CreateEventSubscription (acción)
- DeleteEventSubscription (acción)
- ModifyEventSubscription (acción)
- DescribeEventSubscriptions (acción)
- AddSourceIdentifierToSubscription (acción)
- RemoveSourceIdentifierFromSubscription (acción)
- DescribeEvents (acción)
- DescribeEventCategories (acción)

Estructuras:

- Event (estructura)
- EventCategoriesMap (estructura)
- EventSubscription (estructura)

CreateEventSubscription (acción)

El nombre de la AWS CLI para esta API es: `create-event-subscription`.

Crea una suscripción a notificaciones de eventos. Esta acción requiere un Nombre de recurso de Amazon (ARN) de tema creado por la consola Neptune, la consola de SNS, o la API de SNS. Para obtener un ARN con SNS, debe crear un tema en Amazon SNS y suscribirse al tema. El ARN se muestra en la consola de SNS.

Puede especificar el tipo de origen (SourceType) del que desea recibir notificaciones, proporcione una lista de fuentes Neptune (SourceIds) que desencadena los eventos, y proporcione una lista de las categorías de eventos (EventCategories) de eventos de los que desea recibir notificaciones. Por ejemplo, puede especificar, you can specify SourceType = db-instance, SourceIds = mydbinstance1, mydbinstance2 y EventCategories = Availability, Backup.

Si especifica SourceType y SourceIds, como por ejemplo SourceType = db-instance y SourceIdentifier = myDBInstance1, recibirá notificaciones de todos los eventos db-instance para el origen especificado. Si especifica SourceType, pero no especifica SourceIdentifier, recibirá notificaciones de los eventos de ese tipo de origen para todos sus orígenes de Neptune. Si no especifica ni SourceType ni SourceIdentifier, recibe notificaciones de los eventos generados desde todos los orígenes de Neptune que pertenezcan a su cuenta de cliente.

Solicitud

- Enabled (en la CLI: `--enabled`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano; establecido en true para activar la suscripción, establézcalo en false para crear la suscripción, pero no la active.
- EventCategories (en la CLI: `--event-categories`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de las categorías de eventos para un SourceType a los que desea suscribirse.
  Puede ver una lista de las categorías para un SourceType determinado mediante la acción DescribeEventCategories.

- SnsTopicArn (en la CLI: `--sns-topic-arn`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del tema SNS creado para la notificación de eventos. El ARN es creado por Amazon SNS al crear un tema y suscribirse a él.

- SourceIds (en la CLI: `--source-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La lista de identificadores de los orígenes de eventos para los que se devuelven eventos. Si no se especifica, se incluyen todos los orígenes en la respuesta. Un identificador debe comenzar por una letra y solo deben contener letras ASCII, números y guiones; y no pueden terminar con un guion o contener dos guiones consecutivos.

  Restricciones:

  - Si se proporciona SourceIds, también debe proporcionarse SourceType.
  - Si el tipo de origen es una instancia de base de datos, debe proporcionarse un `DBInstanceIdentifier`.
  - Si el tipo de origen es un grupo de seguridad de base de datos, debe proporcionarse un `DBSecurityGroupName`.
  - Si el tipo de origen es un grupo de parámetros de base de datos, debe proporcionarse un `DBParameterGroupName`.
  - Si el tipo de origen es una instantánea de base de datos, debe proporcionarse un `DBSnapshotIdentifier`.

- SourceType (en la CLI: `--source-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen que está generando los eventos. Por ejemplo, si desea recibir una notificación de eventos generados por una instancia de base de datos, establecería este parámetro para db-instance. Si no se especifica este valor, se devuelven todos los eventos.

  Valores válidos: `db-instance` | `db-cluster` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster-snapshot`

- SubscriptionName (en la CLI: `--subscription-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la suscripción.

Restricciones: el nombre debe tener menos de 255 caracteres.

- Tags (en la CLI: `--tags`): una matriz de objetos Tag.

  Las etiquetas que deben aplicarse a la nueva suscripción de eventos.

Respuesta

Contiene los resultados de una invocación correcta de la acción the section called "DescribeEventSubscriptions".

- CustomerAwsId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cuenta de cliente de Amazon asociada a la suscripción a notificaciones de eventos.
- CustSubscriptionId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de suscripción a notificaciones de eventos.
- Enabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano que indica si la suscripción está habilitada. "True" indica que la suscripción está habilitada.
- EventCategoriesList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de categorías de eventos para la suscripción a notificaciones de eventos.
- EventSubscriptionArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la suscripción de eventos.
- SnsTopicArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de temas de la suscripción a notificaciones de eventos.
- SourceIdsList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de ID de origen para la suscripción a notificaciones de eventos.
- SourceType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen para la suscripción a notificaciones de eventos.
- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la suscripción a notificaciones de eventos de .

Restricciones:

Puede ser uno de los siguientes: creating | modifying | deleting | active | no-permission | topic-not-exist

El estado "no-permission" indica que Neptune ya no tiene permiso para publicar en el tema de SNS. El estado “topic-not-exist” indica que el tema se eliminó después de crear la suscripción.

- SubscriptionCreationTime: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La hora a la que se creó la suscripción a notificaciones de eventos.

Errores

- EventSubscriptionQuotaExceededFault
- SubscriptionAlreadyExistFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault
- SourceNotFoundFault

DeleteEventSubscription (acción)

El nombre de la AWS CLI para esta API es: `delete-event-subscription`.

Elimina una suscripción a notificaciones de eventos.

Solicitud

- SubscriptionName (en la CLI: `--subscription-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la suscripción a notificaciones de eventos que desea eliminar.

Respuesta

Contiene los resultados de una invocación correcta de la acción the section called “DescribeEventSubscriptions”.

- CustomerAwsId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cuenta de cliente de Amazon asociada a la suscripción a notificaciones de eventos.
- CustSubscriptionId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de suscripción a notificaciones de eventos.
- Enabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano que indica si la suscripción está habilitada. "True" indica que la suscripción está habilitada.
- EventCategoriesList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de categorías de eventos para la suscripción a notificaciones de eventos.
- EventSubscriptionArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la suscripción de eventos.
- SnsTopicArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de temas de la suscripción a notificaciones de eventos.
- SourceIdsList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de ID de origen para la suscripción a notificaciones de eventos.
- SourceType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen para la suscripción a notificaciones de eventos.
- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la suscripción a notificaciones de eventos de .

  Restricciones:

  Puede ser uno de los siguientes: creating | modifying | deleting | active | no-permission | topic-not-exist

  El estado "no-permission" indica que Neptune ya no tiene permiso para publicar en el tema de SNS. El estado “topic-not-exist” indica que el tema se eliminó después de crear la suscripción.

- SubscriptionCreationTime: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La hora a la que se creó la suscripción a notificaciones de eventos.

Errores

- SubscriptionNotFoundFault
- InvalidEventSubscriptionStateFault

ModifyEventSubscription (acción)

El nombre de la AWS CLI para esta API es: `modify-event-subscription`.

Modifica una suscripción a notificaciones de eventos existente. Tenga en cuenta que no puede modificar los identificadores de origen mediante esta llamada; para cambiar identificadores de origen de una suscripción, utilice las llamadas the section called “AddSourceIdentifierToSubscription” y the section called “RemoveSourceIdentifierFromSubscription”.

Puede ver una lista de las categorías de eventos para un SourceType determinado mediante la acción DescribeEventCategories.

Solicitud

- Enabled (en la CLI: `--enabled`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano; establecida en true para activar la suscripción.
- EventCategories (en la CLI: `--event-categories`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de las categorías de eventos para un SourceType a los que desea suscribirse.
  Puede ver una lista de las categorías para un SourceType determinado mediante la acción DescribeEventCategories.
- SnsTopicArn (en la CLI: `--sns-topic-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del tema SNS creado para la notificación de eventos. El ARN es creado por Amazon SNS al crear un tema y suscribirse a él.

- SourceType (en la CLI: `--source-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen que está generando los eventos. Por ejemplo, si desea recibir una notificación de eventos generados por una instancia de base de datos, establecería este parámetro para db-instance. Si no se especifica este valor, se devuelven todos los eventos.

  Valores válidos: db-instance | db-parameter-group | db-security-group | db-snapshot
- SubscriptionName (en la CLI: `--subscription-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la suscripción a notificaciones de eventos de .

Respuesta

Contiene los resultados de una invocación correcta de la acción the section called "DescribeEventSubscriptions".

- CustomerAwsId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cuenta de cliente de Amazon asociada a la suscripción a notificaciones de eventos.
- CustSubscriptionId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de suscripción a notificaciones de eventos.
- Enabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano que indica si la suscripción está habilitada. "True" indica que la suscripción está habilitada.
- EventCategoriesList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de categorías de eventos para la suscripción a notificaciones de eventos.
- EventSubscriptionArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la suscripción de eventos.
- SnsTopicArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de temas de la suscripción a notificaciones de eventos.
- SourceIdsList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una lista de ID de origen para la suscripción a notificaciones de eventos.

- SourceType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen para la suscripción a notificaciones de eventos.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la suscripción a notificaciones de eventos de .

  Restricciones:

  Puede ser uno de los siguientes: creating | modifying | deleting | active | no-permission | topic-not-exist

  El estado "no-permission" indica que Neptune ya no tiene permiso para publicar en el tema de SNS. El estado “topic-not-exist” indica que el tema se eliminó después de crear la suscripción.

- SubscriptionCreationTime: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La hora a la que se creó la suscripción a notificaciones de eventos.

Errores

- EventSubscriptionQuotaExceededFault
- SubscriptionNotFoundFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault

DescribeEventSubscriptions (acción)

El nombre de la AWS CLI para esta API es: `describe-event-subscriptions`.

Muestra todas las descripciones de la suscripción para una cuenta de cliente. La descripción de una suscripción incluye SubscriptionName, SNSTopicARN, CustomerID, SourceType, SourceID, CreationTime y Status.

Si especifica un SubscriptionName, muestra la descripción de dicha suscripción.

Solicitud

- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud DescribeOrderableDBInstanceOptions anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.
- SubscriptionName (en la CLI: `--subscription-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la suscripción a notificaciones de eventos que desea describir.

Respuesta

- EventSubscriptionsList: matriz de objetos EventSubscription.

  Una lista de tipos de datos EventSubscriptions.
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud DescribeOrderableDBInstanceOptions anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

Errores

- SubscriptionNotFoundFault

AddSourceIdentifierToSubscription (acción)

El nombre de la AWS CLI para esta API es: `add-source-identifier-to-subscription`.

Agrega un identificador de origen a una suscripción de notificación de eventos existente.

Solicitud

- SourceIdentifier (en la CLI: `--source-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del origen de eventos que se va a añadir.

  Restricciones:

  - Si el tipo de origen es una instancia de base de datos, debe proporcionarse un `DBInstanceIdentifier`.
  - Si el tipo de origen es un grupo de seguridad de base de datos, debe proporcionarse un `DBSecurityGroupName`.
  - Si el tipo de origen es un grupo de parámetros de base de datos, debe proporcionarse un `DBParameterGroupName`.
  - Si el tipo de origen es una instantánea de base de datos, debe proporcionarse un `DBSnapshotIdentifier`.
- SubscriptionName (en la CLI: `--subscription-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la suscripción a notificaciones de eventos al que desea añadir un identificador de origen.

Respuesta

Contiene los resultados de una invocación correcta de la acción the section called “DescribeEventSubscriptions”.

- CustomerAwsId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cuenta de cliente de Amazon asociada a la suscripción a notificaciones de eventos.
- CustSubscriptionId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de suscripción a notificaciones de eventos.

- Enabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano que indica si la suscripción está habilitada. "True" indica que la suscripción está habilitada.

- EventCategoriesList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de categorías de eventos para la suscripción a notificaciones de eventos.

- EventSubscriptionArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la suscripción de eventos.

- SnsTopicArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de temas de la suscripción a notificaciones de eventos.

- SourceIdsList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de ID de origen para la suscripción a notificaciones de eventos.

- SourceType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen para la suscripción a notificaciones de eventos.

- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la suscripción a notificaciones de eventos de .

  Restricciones:

  Puede ser uno de los siguientes: creating | modifying | deleting | active | no-permission | topic-not-exist

  El estado "no-permission" indica que Neptune ya no tiene permiso para publicar en el tema de SNS. El estado “topic-not-exist” indica que el tema se eliminó después de crear la suscripción.

- SubscriptionCreationTime: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La hora a la que se creó la suscripción a notificaciones de eventos.

Errores

- SubscriptionNotFoundFault
- SourceNotFoundFault

RemoveSourceIdentifierFromSubscription (acción)

El nombre de la AWS CLI para esta API es: `remove-source-identifier-from-subscription`.

Elimina un identificador de origen de una suscripción a notificaciones de eventos existente.

Solicitud

- SourceIdentifier (en la CLI: `--source-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de origen que se va a eliminar de la suscripción, como, por ejemplo, el identificador de instancias de bases de datos para una instancia de base de datos o el nombre de un grupo de seguridad.
- SubscriptionName (en la CLI: `--subscription-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la suscripción a notificaciones de eventos del que desea eliminar un identificador de origen.

Respuesta

Contiene los resultados de una invocación correcta de la acción the section called “DescribeEventSubscriptions”.

- CustomerAwsId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cuenta de cliente de Amazon asociada a la suscripción a notificaciones de eventos.
- CustSubscriptionId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de suscripción a notificaciones de eventos.
- Enabled: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano que indica si la suscripción está habilitada. "True" indica que la suscripción está habilitada.
- EventCategoriesList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de categorías de eventos para la suscripción a notificaciones de eventos.

- EventSubscriptionArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la suscripción de eventos.
- SnsTopicArn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de temas de la suscripción a notificaciones de eventos.
- SourceIdsList: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de ID de origen para la suscripción a notificaciones de eventos.
- SourceType: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen para la suscripción a notificaciones de eventos.
- Status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la suscripción a notificaciones de eventos de .

  Restricciones:

  Puede ser uno de los siguientes: creating | modifying | deleting | active | no-permission | topic-not-exist

  El estado "no-permission" indica que Neptune ya no tiene permiso para publicar en el tema de SNS. El estado “topic-not-exist” indica que el tema se eliminó después de crear la suscripción.
- SubscriptionCreationTime: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La hora a la que se creó la suscripción a notificaciones de eventos.

Errores

- SubscriptionNotFoundFault
- SourceNotFoundFault

DescribeEvents (acción)

El nombre de la AWS CLI para esta API es: `describe-events`.

Devuelve eventos relacionados con las instancias de base de datos, grupos de seguridad de base de datos, instantáneas de base de datos y grupos de parámetros de base de datos para los últimos

14 días. Los eventos específicos de una instancia de base de datos concreta, grupo de seguridad de base de datos, instantánea de base de datos o grupo de parámetros de base de datos se pueden obtener proporcionando el nombre como parámetro. De forma predeterminada, se devuelven los eventos de la última hora.

Solicitud

- Duration (en la CLI: `--duration`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de minutos para los que recuperar eventos.

  Predeterminado: 60
- EndTime (en la CLI: `--end-time`): un TStamp, del tipo: `timestamp` (un punto en el tiempo, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  El final del intervalo de tiempo para el que recuperar eventos, especificado en formato ISO 8601. Para obtener más información acerca de ISO 8601, consulte la página página de Wikipedia ISO8601.

  Ejemplo: 2009-07-08T18:00Z
- EventCategories (en la CLI: `--event-categories`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de categorías de eventos que desencadena notificaciones para la suscripción a notificaciones de eventos.
- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud DescribeEvents anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

Predeterminado: 100

Restricciones: Minimum 20, máximo 100.

- SourceIdentifier (en la CLI: `--source-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del origen de eventos para el que se devuelven eventos. Si no se especifica, se incluyen todos los orígenes en la respuesta.

  Restricciones:

  - Si se proporciona SourceIdentifier, también debe proporcionarse SourceType.
  - Si el tipo de origen es `DBInstance`, debe proporcionarse un `DBInstanceIdentifier`.
  - Si el tipo de origen es `DBSecurityGroup`, debe suministrarse `DBSecurityGroupName`.
  - Si el tipo de origen es `DBParameterGroup`, debe suministrarse `DBParameterGroupName`.
  - Si el tipo de origen es `DBSnapshot`, debe suministrarse `DBSnapshotIdentifier`.
  - No puede terminar por un guion ni contener dos guiones consecutivos.
- SourceType (en la CLI: `--source-type`): un SourceType, del tipo: `string` (una cadena codificada con UTF-8).

  El origen del evento para el que recuperar eventos. Si no se especifica ningún valor, se devuelven todos los eventos.
- StartTime (en la CLI: `--start-time`): un TStamp, del tipo: `timestamp` (un punto en el tiempo, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  El principio del intervalo de tiempo para el que recuperar eventos, especificado en formato ISO 8601. Para obtener más información acerca de ISO 8601, consulte la página página de Wikipedia ISO8601.

  Ejemplo: 2009-07-08T18:00Z

Respuesta

- Events: matriz de objetos Evento.

  Una lista de instancias the section called “Evento”.

- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud Events anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

DescribeEventCategories (acción)

El nombre de la AWS CLI para esta API es: `describe-event-categories`.

Muestra una lista de categorías de todos los tipos de origen de eventos o, si se especifica, para un tipo de origen especificado.

Solicitud

- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.

- SourceType (en la CLI: `--source-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen que está generando los eventos.

  Valores válidos: db-instance | db-parameter-group | db-security-group | db-snapshot

Respuesta

- EventCategoriesMapList: matriz de objetos EventCategoriesMap.

  Una lista de tipos de datos EventCategoriesMap.

Estructuras:

Event (estructura)

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeEvents".

Campos

- Date: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  Especifica la fecha y la hora del evento.
- EventCategories: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica la categoría para el evento.
- Message: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el texto de este evento.
- SourceArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para el evento.
- SourceIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Proporciona el identificador del origen del evento.
- SourceType: se trata de un SourceType, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el tipo de origen para este evento.

EventCategoriesMap (estructura)

Contiene los resultados de una invocación correcta de la acción the section called "DescribeEventCategories".

Campos

- EventCategories: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Las categorías de eventos para el tipo de origen especificado

- SourceType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen al que pertenecen las categorías devueltas

EventSubscription (estructura)

Contiene los resultados de una invocación correcta de la acción the section called "DescribeEventSubscriptions".

Campos

- CustomerAwsId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cuenta de cliente de Amazon asociada a la suscripción a notificaciones de eventos.
- CustSubscriptionId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de suscripción a notificaciones de eventos.
- Enabled: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor booleano que indica si la suscripción está habilitada. "True" indica que la suscripción está habilitada.
- EventCategoriesList: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de categorías de eventos para la suscripción a notificaciones de eventos.
- EventSubscriptionArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) para la suscripción de eventos.
- SnsTopicArn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de temas de la suscripción a notificaciones de eventos.
- SourceIdsList: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de ID de origen para la suscripción a notificaciones de eventos.
- SourceType: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de origen para la suscripción a notificaciones de eventos.
- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El estado de la suscripción a notificaciones de eventos de .

Restricciones:

Puede ser uno de los siguientes: creating | modifying | deleting | active | no-permission | topic-not-exist

El estado "no-permission" indica que Neptune ya no tiene permiso para publicar en el tema de SNS. El estado “topic-not-exist” indica que el tema se eliminó después de crear la suscripción.

- SubscriptionCreationTime: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La hora a la que se creó la suscripción a notificaciones de eventos.

`EventSubscription` se utiliza como el elemento de respuesta para:

- CreateEventSubscription
- ModifyEventSubscription
- AddSourceIdentifierToSubscription
- RemoveSourceIdentifierFromSubscription
- DeleteEventSubscription

Otras API de Neptune

Acciones:

- AddTagsToResource (acción)
- ListTagsForResource (acción)
- RemoveTagsFromResource (acción)
- ApplyPendingMaintenanceAction (acción)
- DescribePendingMaintenanceActions (acción)
- DescribeDBEngineVersions (acción)

Estructuras:

- DBEngineVersion (estructura)
- EngineDefaults (estructura)
- PendingMaintenanceAction (estructura)
- ResourcePendingMaintenanceActions (estructura)
- UpgradeTarget (estructura)
- Tag (estructura)

AddTagsToResource (acción)

El nombre de la AWS CLI para esta API es: `add-tags-to-resource`.

Añade etiquetas de metadatos a un recurso de Amazon Neptune. Estas etiquetas también se pueden utilizar con los informes de asignación de costos para realizar un seguimiento de los costos asociados con los recursos de Amazon Neptune o se utilizan en una Declaración de condición en una política de IAM para Amazon Neptune.

Solicitud

- ResourceName (en la CLI: `--resource-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El recurso de Amazon Neptune al que se añaden las etiquetas. Este valor es un Nombre de recurso de Amazon (ARN). Para obtener más información acerca de cómo crear un ARN, consulte Creación de un nombre de recurso de Amazon (ARN).
- Tags (en la CLI: `--tags`) obligatorio: una matriz de objetos Tag.

  Las etiquetas que se van a asignar al recurso de Amazon Neptune.

Respuesta

- Sin parámetros de respuesta.

Errores

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault

- DBClusterNotFoundFault

ListTagsForResource (acción)

El nombre de la AWS CLI para esta API es: `list-tags-for-resource`.

Enumera todas las etiquetas en un recurso de Amazon Neptune.

Solicitud

- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Este parámetro es incompatible en estos momentos.
- ResourceName (en la CLI: `--resource-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El recurso de Amazon Neptune con las etiquetas que se van a incluir en la lista. Este valor es un Nombre de recurso de Amazon (ARN). Para obtener más información acerca de cómo crear un ARN, consulte Creación de un nombre de recurso de Amazon (ARN).

Respuesta

- TagList: matriz de objetos Tag.

  Lista de etiquetas devueltas por la operación ListTagsForResource.

Errores

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

RemoveTagsFromResource (acción)

El nombre de la AWS CLI para esta API es: `remove-tags-from-resource`.

Elimina etiquetas de metadatos de un recurso de Amazon Neptune.

Solicitud

- ResourceName (en la CLI: `--resource-name`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El recurso de Amazon Neptune del que se eliminan las etiquetas. Este valor es un Nombre de recurso de Amazon (ARN). Para obtener más información acerca de cómo crear un ARN, consulte Creación de un nombre de recurso de Amazon (ARN).
- TagKeys (en la CLI: `--tag-keys`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de la etiqueta (nombre) de la etiqueta que se va a eliminar.

Respuesta

- Sin parámetros de respuesta.

Errores

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ApplyPendingMaintenanceAction (acción)

El nombre de la AWS CLI para esta API es: `apply-pending-maintenance-action`.

Aplica una acción de mantenimiento pendiente a un recurso (por ejemplo, a una instancia de base de datos).

Solicitud

- ApplyAction (en la CLI: `--apply-action`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La acción de mantenimiento pendiente que se aplica a este recurso.

  Valores válidos: `system-update`, `db-upgrade`
- OptInType (en la CLI: `--opt-in-type`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un valor que indica el tipo de solicitud de alta o deshace una solicitud de alta. Una solicitud de alta de tipo `immediate` no se puede deshacer.

Valores válidos:

- `immediate`: aplicar inmediatamente la acción de mantenimiento.
- `next-maintenance`: aplicar la acción de mantenimiento durante la siguiente ventana de mantenimiento del recurso.
- `undo-opt-in`: cancelar todas las solicitudes de alta `next-maintenance` existentes.

- ResourceIdentifier (en la CLI: `--resource-identifier`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El Nombre de recurso de Amazon (ARN) del recurso al que se aplica la acción de mantenimiento pendiente. Para obtener más información acerca de cómo crear un ARN, consulte Creación de un nombre de recurso de Amazon (ARN).

Respuesta

Describe las acciones de mantenimiento pendientes para un recurso.

- PendingMaintenanceActionDetails: matriz de objetos PendingMaintenanceAction.

  Una lista que proporciona detalles sobre las acciones de mantenimiento pendientes para el recurso.
- ResourceIdentifier: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del recurso que tiene acciones de mantenimiento pendientes.

Errores

- ResourceNotFoundFault

DescribePendingMaintenanceActions (acción)

El nombre de la AWS CLI para esta API es: `describe-pending-maintenance-actions`.

Devuelve una lista de recursos (por ejemplo, instancias de base de datos) que tienen al menos una acción de mantenimiento pendiente.

Solicitud

- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  Un filtro que especifica uno o más recursos para devolver acciones de mantenimiento pendientes.

  Filtros compatibles:

  - `db-cluster-id`: acepta identificadores de clúster de base de datos y Nombres de recursos de Amazon (ARN). La lista de resultados solo incluirá las acciones de mantenimiento pendientes para los clústeres de base de datos identificados por estos ARN.
  - `db-instance-id`: admite identificadores de instancia de base de datos y ARN de instancia de base de datos. La lista de resultados solo incluirá las acciones de mantenimiento pendientes para las instancias de base de datos identificadas por estos ARN.
- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud `DescribePendingMaintenanceActions` anterior. Si se especifica este parámetro, la respuesta incluye solo los registros más allá del marcador, hasta un número de registros especificado por `MaxRecords`.
- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `MaxRecords` especificado, se incluye en la respuesta un token de paginación que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.
- ResourceIdentifier (en la CLI: `--resource-identifier`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un recuerdo para el que devolver acciones de mantenimiento pendientes.

Respuesta

- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un token de paginación opcional proporcionado por una solicitud `DescribePendingMaintenanceActions` anterior. Si se especifica este parámetro, la respuesta incluye solo los registros más allá del marcador, hasta un número de registros especificado por `MaxRecords`.

- PendingMaintenanceActions: matriz de objetos ResourcePendingMaintenanceActions.

  La lista de acciones de mantenimiento pendientes para el recurso.

Errores

- ResourceNotFoundFault

DescribeDBEngineVersions (acción)

El nombre de la AWS CLI para esta API es: `describe-db-engine-versions`.

Devuelve una lista con los motores de base de datos disponibles.

Solicitud

- DBParameterGroupFamily (en la CLI: `--db-parameter-group-family`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de una familia de grupos de parámetros de base de datos específicos para los que devolver detalles.

  Restricciones:

  - Si se suministran, debe coincidir con una DBParameterGroupFamily existente.
- DefaultOnly (en la CLI: `--default-only`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica que solo se devuelve la versión predeterminada del motor especificado o motor y combinación de la versión principal.
- Engine (en la CLI: `--engine`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El motor de base de datos que se debe devolver.

- EngineVersion (en la CLI: `--engine-version`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de base de datos que se debe devolver.

  Ejemplo: `5.1.49`

- Filters (en la CLI: `--filters`): una matriz de objetos Filtro.

  No se admite actualmente.

- ListSupportedCharacterSets (en la CLI: `--list-supported-character-sets`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se especifica este parámetro y el motor solicitado es compatible con el parámetro `CharacterSetName` para `CreateDBInstance`, la respuesta incluye una lista de conjuntos de caracteres admitidos para cada versión del motor.

- ListSupportedTimezones (en la CLI: `--list-supported-timezones`): un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Si se especifica este parámetro y el motor solicitado es compatible con el parámetro `TimeZone` para `CreateDBInstance`, la respuesta incluye una lista de zonas horarias admitidas para cada versión del motor.

- Marker (en la CLI: `--marker`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

- MaxRecords (en la CLI: `--max-records`): un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número máximo de registros que se debe incluir en la respuesta. Si hay más de un valor `MaxRecords` disponible, se incluye un token de paginación que se conoce como marcador en la respuesta para poder recuperar los siguientes resultados.

  Predeterminado: 100

  Restricciones: Minimum 20, máximo 100.

Respuesta

- DBEngineVersions: matriz de objetos DBEngineVersion.

  Una lista de elementos `DBEngineVersion`
- Marker: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.

Estructuras:

DBEngineVersion (estructura)

Este tipo de datos se utiliza como un elemento de respuesta en la acción the section called "DescribeDBEngineVersions".

Campos

- DBEngineDescription: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La descripción del motor de base de datos.
- DBEngineVersionDescription: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La descripción de la versión del motor de base de datos.
- DBParameterGroupFamily: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la familia de grupos de parámetros de base de datos para el motor de base de datos.
- Engine: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del motor de base de datos.
- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Número de versión del motor de base de datos.
- ExportableLogTypes: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Los tipos de registros que el motor de base de datos tiene disponibles para la exportación a CloudWatch Logs.

- SupportedTimezones: se trata de una matriz de objetos Zona horaria.

  Una lista de las zonas horarias admitidas por este motor para el parámetro `Timezone` de la acción `CreateDBInstance`.

- SupportsGlobalDatabases: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si puede utilizar las bases de datos globales de Aurora con una determinada versión de motor de base de datos.

- SupportsLogExportsToCloudwatchLogs: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si la versión del motor admite la exportación de tipos de registro especificados por ExportableLogTypes a CloudWatch Logs.

- SupportsReadReplica: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la versión del motor de base de datos admite réplicas de lectura.

- ValidUpgradeTarget: se trata de una matriz de objetos UpgradeTarget.

  Una lista de versiones de motor a la que esta versión del motor de base de datos se puede actualizar.

EngineDefaults (estructura)

Contiene el resultado de una invocación correcta de la acción the section called "DescribeEngineDefaultParameters".

Campos

- DBParameterGroupFamily: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el nombre de la familia del grupo de parámetros de base de datos a los que se aplican los parámetros predeterminados del motor.

- Marker: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un token de paginación opcional proporcionado por una solicitud EngineDefaults anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del marcador, hasta el valor especificado por `MaxRecords`.
- Parameters: se trata de una matriz de objetos Parámetro.

  Contiene una lista de los parámetros predeterminados del motor.

`EngineDefaults` se utiliza como el elemento de respuesta para:

- DescribeEngineDefaultParameters
- DescribeEngineDefaultClusterParameters

PendingMaintenanceAction (estructura)

Proporciona información acerca de una acción de mantenimiento pendiente para un recurso.

Campos

- Action: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de acción de mantenimiento pendiente disponible para el recurso.
- AutoAppliedAfterDate: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La fecha del periodo de mantenimiento cuando se aplica la acción. La acción de mantenimiento se aplica al recurso durante su primer periodo de mantenimiento después de esta fecha. Si se especifica esta fecha, se omite cualquier solicitud de alta `next-maintenance`.
- CurrentApplyDate: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La fecha de entrada en vigor en que se aplica la acción de mantenimiento pendiente al recurso. Esta fecha tiene en cuenta las solicitudes de alta recibidas de la API the section called “ApplyPendingMaintenanceAction”, la `AutoAppliedAfterDate` y la `ForcedApplyDate`. Este valor está en blanco si no se ha recibido una solicitud de alta y no se ha especificado nada como `AutoAppliedAfterDate` o `ForcedApplyDate`.
- Description: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Una descripción que proporciona información más detallada sobre la acción de mantenimiento.

- ForcedApplyDate: se trata de un TStamp, del tipo: `timestamp` (un momento específico, generalmente definido como un desplazamiento desde la medianoche del 1 de enero de 1970).

  La fecha en que se aplica automáticamente la acción de mantenimiento. La acción de mantenimiento se aplica al recurso en esta fecha independientemente del periodo de mantenimiento para el recurso. Si se especifica esta fecha, se omite cualquier solicitud de alta `immediate`.
- OptInStatus: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Indica el tipo de solicitud de alta que se ha recibido para el recurso.

ResourcePendingMaintenanceActions (estructura)

Describe las acciones de mantenimiento pendientes para un recurso.

Campos

- PendingMaintenanceActionDetails: se trata de una matriz de objetos PendingMaintenanceAction.

  Una lista que proporciona detalles sobre las acciones de mantenimiento pendientes para el recurso.
- ResourceIdentifier: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del recurso que tiene acciones de mantenimiento pendientes.

`ResourcePendingMaintenanceActions` se utiliza como el elemento de respuesta para:

- ApplyPendingMaintenanceAction

UpgradeTarget (estructura)

La versión del motor de base de datos a la que puede actualizarse una instancia de base de datos.

Campos

- AutoUpgrade: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

Un valor que indica si la versión de destino se aplica a cualquier instancia de base de datos de origen que tiene AutoMinorVersionUpgrade establecido en "true·.

- Description: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del motor de base de datos a la que puede actualizarse una instancia de base de datos.

- Engine: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del motor de base de datos de destino de actualización.

- EngineVersion: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El número de versión del motor de base de datos de destino de actualización.

- IsMajorVersionUpgrade: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si un motor de base de datos se actualiza a una versión principal.

- SupportsGlobalDatabases: se trata de un BooleanOptional, del tipo: `boolean` (un valor booleano [true o false]).

  Un valor que indica si puede utilizar las bases de datos globales de Neptune con una determinada versión de motor de destino.

Tag (estructura)

Metadatos asignados a un recurso de Amazon Neptune que consta de un par clave-valor.

Campos

- Key: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave es el nombre obligatorio de la etiqueta. El valor de la cadena puede tener una longitud de entre 1 y 128 caracteres Unicode y no puede llevar el prefijo `aws:` ni `rds:`. La cadena puede incluir únicamente el conjunto de letras de Unicode, dígitos y espacio en blanco, “_”, “.”, “/”, “=”, “+”, “-” (Java regex: “^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$”).

- Value: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El valor es la parte opcional de la etiqueta. El valor de la cadena puede tener una longitud de entre 1 y 256 caracteres Unicode y no puede llevar el prefijo `aws:` ni `rds:`. La cadena puede

incluir únicamente el conjunto de letras de Unicode, dígitos y espacio en blanco, “_”, “.”, “/”, “=”, “+”, “-” (Java regex: “^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$”).

Tipos de datos comunes de Neptune

Estructuras:

- AvailabilityZone (estructura)
- DBSecurityGroupMembership (estructura)
- DomainMembership (estructura)
- DoubleRange (estructura)
- Endpoint (estructura)
- Filter (estructura)
- Range (estructura)
- ServerlessV2ScalingConfiguration (estructura)
- ServerlessV2ScalingConfigurationInfo (estructura)
- Zona horaria (estructura)
- VpcSecurityGroupMembership (estructura)

AvailabilityZone (estructura)

Especifica una zona de disponibilidad.

Campos

- Name: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la zona de disponibilidad.

DBSecurityGroupMembership (estructura)

Especifica la membresía en un grupo de seguridad de base de datos designado.

Campos

- DBSecurityGroupName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de seguridad de base de datos.
- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado del grupo de seguridad de base de datos.

DomainMembership (estructura)

Un registro de pertenencia de dominio de Active Directory asociado a una instancia de base de datos.

Campos

- Domain: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador del dominio de Active Directory.
- FQDN: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de dominio completo del dominio de Active Directory.
- IAMRoleName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del rol de IAM que se utilizará al realizar llamadas a la API al Directory Service.
- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la pertenencia al dominio de Active Directory de la instancia de base de datos, como, por ejemplo, "joined", "pending-join", "failed", etc.

DoubleRange (estructura)

Un rango de valores dobles.

Campos

- From: se trata de un doble, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

El valor mínimo del rango.

- To: se trata de un doble, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

  El valor máximo del rango.

Endpoint (estructura)

Especifica el punto de enlace de una conexión.

Para ver la estructura de datos que representa los puntos de conexión del clúster de base de datos de Amazon Neptune, consulte `DBClusterEndpoint`.

Campos

- Address: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica la dirección DNS de la instancia de base de datos.
- HostedZoneId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Especifica el ID que Amazon Route 53 asigna al crear una zona alojada.
- Port: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Especifica el puerto en el que escucha el motor de la base de datos.

Filter (estructura)

Este tipo no se admite en la actualidad.

Campos

- Name: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Este parámetro es incompatible en estos momentos.
- Values: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Este parámetro es incompatible en estos momentos.

Range (estructura)

Un rango de valores enteros.

Campos

- From: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El valor mínimo del rango.

- Step: se trata de un IntegerOptional, del tipo: `integer` (un valor entero firmado de 32 bits).

  El valor de incremento para el rango. Por ejemplo, si tiene un rango de 5000 a 10 000, con un valor de incremento de 1000, los valores válidos comienzan a partir de 5000 e incrementan en 1000. Aunque 7500 se encuentra dentro del rango, no es un valor válido para el rango. Los valores válidos son 5000, 6000, 7000, 8000...

- To: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El valor máximo del rango.

ServerlessV2ScalingConfiguration (estructura)

Incluye la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

Campos

- MaxCapacity: se trata de un DoubleOptional, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

  La cantidad máxima de unidades de capacidad de Neptune (NCU) para una instancia de base de datos en un clúster de Neptune sin servidor. Puede especificar los valores de la NCU en incrementos de medio punto, como 40; 40,5; 41; etc.

- MinCapacity: se trata de un DoubleOptional, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

La cantidad mínima de unidades de capacidad de Neptune (NCU) para una instancia de base de datos en un clúster de Neptune sin servidor. Puede especificar los valores de la NCU en incrementos de medio punto, como 8, 8,5, 9, etc.

ServerlessV2ScalingConfigurationInfo (estructura)

Muestra la configuración de escalado de un clúster de base de datos de Neptune sin servidor.

Para obtener más información, consulte Uso de Amazon Neptune sin servidor en la Guía del usuario de Amazon Neptune.

Campos

- MaxCapacity: se trata de un DoubleOptional, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

  La cantidad máxima de unidades de capacidad de Neptune (NCU) para una instancia de base de datos en un clúster de Neptune sin servidor. Puede especificar los valores de la NCU en incrementos de medio punto, como 40; 40,5; 41; etc.
- MinCapacity: se trata de un DoubleOptional, del tipo: `double` (un número en coma flotante IEEE 754 de precisión doble).

  La cantidad mínima de unidades de capacidad de Neptune (NCU) para una instancia de base de datos en un clúster de Neptune sin servidor. Puede especificar los valores de la NCU en incrementos de medio punto, como 8, 8,5, 9, etc.

Zona horaria (estructura)

Una zona horaria asociada a una the section called “DBInstance”.

Campos

- TimezoneName: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la zona horaria.

VpcSecurityGroupMembership (estructura)

Este tipo de datos se utiliza como elemento de respuesta para las consultas de pertenencia al grupo de seguridad de VPC.

Campos

- Status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado del grupo de seguridad de VPC.
- VpcSecurityGroupId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del grupo de seguridad de VPC.

Excepciones de Neptune específicas de API individuales

Excepciones:

- AuthorizationAlreadyExistsFault (estructura)
- AuthorizationNotFoundFault (estructura)
- AuthorizationQuotaExceededFault (estructura)
- CertificateNotFoundFault (estructura)
- DBClusterAlreadyExistsFault (estructura)
- DBClusterNotFoundFault (estructura)
- DBClusterParameterGroupNotFoundFault (estructura)
- DBClusterQuotaExceededFault (estructura)
- DBClusterRoleAlreadyExistsFault (estructura)
- DBClusterRoleNotFoundFault (estructura)
- DBClusterRoleQuotaExceededFault (estructura)
- DBClusterSnapshotAlreadyExistsFault (estructura)
- DBClusterSnapshotNotFoundFault (estructura)
- DBInstanceAlreadyExistsFault (estructura)
- DBInstanceNotFoundFault (estructura)
- DBLogFileNotFoundFault (estructura)

- DBParameterGroupAlreadyExistsFault (estructura)
- DBParameterGroupNotFoundFault (estructura)
- DBParameterGroupQuotaExceededFault (estructura)
- DBSecurityGroupAlreadyExistsFault (estructura)
- DBSecurityGroupNotFoundFault (estructura)
- DBSecurityGroupNotSupportedFault (estructura)
- DBSecurityGroupQuotaExceededFault (estructura)
- DBSnapshotAlreadyExistsFault (estructura)
- DBSnapshotNotFoundFault (estructura)
- DBSubnetGroupAlreadyExistsFault (estructura)
- DBSubnetGroupDoesNotCoverEnoughAZs (estructura)
- DBSubnetGroupNotAllowedFault (estructura)
- DBSubnetGroupNotFoundFault (estructura)
- DBSubnetGroupQuotaExceededFault (estructura)
- DBSubnetQuotaExceededFault (estructura)
- DBUpgradeDependencyFailureFault (estructura)
- DomainNotFoundFault (estructura)
- EventSubscriptionQuotaExceededFault (estructura)
- GlobalClusterAlreadyExistsFault (estructura)
- GlobalClusterNotFoundFault (estructura)
- GlobalClusterQuotaExceededFault (estructura)
- InstanceQuotaExceededFault (estructura)
- InsufficientDBClusterCapacityFault (estructura)
- InsufficientDBInstanceCapacityFault (estructura)
- InsufficientStorageClusterCapacityFault (estructura)
- InvalidDBClusterEndpointStateFault (estructura)
- InvalidDBClusterSnapshotStateFault (estructura)
- InvalidDBClusterStateFault (estructura)
- InvalidDBInstanceStateFault (estructura)

- InvalidDBParameterGroupStateFault (estructura)
- InvalidDBSecurityGroupStateFault (estructura)
- InvalidDBSnapshotStateFault (estructura)
- InvalidDBSubnetGroupFault (estructura)
- InvalidDBSubnetGroupStateFault (estructura)
- InvalidDBSubnetStateFault (estructura)
- InvalidEventSubscriptionStateFault (estructura)
- InvalidGlobalClusterStateFault (estructura)
- InvalidOptionGroupStateFault (estructura)
- InvalidRestoreFault (estructura)
- InvalidSubnet (estructura)
- InvalidVPCNetworkStateFault (estructura)
- KMSKeyNotAccessibleFault (estructura)
- OptionGroupNotFoundFault (estructura)
- PointInTimeRestoreNotEnabledFault (estructura)
- ProvisionedIopsNotAvailableInAZFault (estructura)
- ResourceNotFoundFault (estructura)
- SNSInvalidTopicFault (estructura)
- SNSNoAuthorizationFault (estructura)
- SNSTopicArnNotFoundFault (estructura)
- SharedSnapshotQuotaExceededFault (estructura)
- SnapshotQuotaExceededFault (estructura)
- SourceNotFoundFault (estructura)
- StorageQuotaExceededFault (estructura)
- StorageTypeNotSupportedFault (estructura)
- SubnetAlreadyInUse (estructura)
- SubscriptionAlreadyExistFault (estructura)
- SubscriptionCategoryNotFoundFault (estructura)
- SubscriptionNotFoundFault (estructura)

AuthorizationAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

El CIDRIP especificado o grupo de seguridad de EC2 especificado ya se ha autorizado para el grupo de seguridad de base de datos especificado.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

AuthorizationNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

El CIDRIP especificado o grupo de seguridad de EC2 especificado no está autorizado para el grupo de seguridad de base de datos especificado.

Neptune puede no recibir la autorización mediante IAM para realizar acciones necesarias en su nombre.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

AuthorizationQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

Se ha alcanzado la cuota de autorizaciones del grupo de seguridad de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

CertificateNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

CertificateIdentifier no hace referencia a un certificado existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

El usuario ya tiene un clúster de base de datos con el identificador concreto.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBClusterIdentifier no hace referencia a un clúster de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterParameterGroupNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBClusterParameterGroupName no hace referencia a un grupo de parámetros de clúster de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 403.

El usuario intentó crear un nuevo clúster de base de datos y el usuario ya ha alcanzado la cuota máxima permitida de clústeres de bases de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterRoleAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

El Nombre de recurso de Amazon (ARN) del rol de IAM especificado ya está asociado con el clúster de base de datos especificado.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterRoleNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

El Nombre de recurso de Amazon (ARN) del rol de IAM especificado no está asociado con el clúster de base de datos especificado.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterRoleQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

Ha superado el número máximo de roles de IAM que se pueden asociar con el clúster de base de datos especificado.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterSnapshotAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

El usuario ya tiene una instantánea del clúster de base de datos con el identificador concreto.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBClusterSnapshotNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBClusterSnapshotIdentifier no hace referencia a una instantánea del clúster de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

Mensaje que describe los detalles del problema.

DBInstanceAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

El usuario ya tiene una instancia de base de datos con el identificador concreto.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBInstanceNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBInstanceIdentifier no hace referencia a una instancia de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBLogFileNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

LogFileName no hace referencia a un archivo de registro de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBParameterGroupAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

Existe un grupo de parámetros de base de datos con el mismo nombre.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBParameterGroupNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBParameterGroupName no hace referencia a un grupo de parámetros de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBParameterGroupQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

La solicitud daría lugar a que el usuario superara el número permitido de grupos de parámetros de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSecurityGroupAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

Ya existe un grupo de seguridad de base de datos con el nombre especificado en DBSecurityGroupName.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSecurityGroupNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBSecurityGroupName no hace referencia a un grupo de seguridad de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSecurityGroupNotSupportedFault (estructura)

Código de estado HTTP devuelto: 400.

Un grupo de seguridad de base de datos no está permitido para esta acción.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSecurityGroupQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

La solicitud daría lugar a que el usuario superara el número permitido de grupos de seguridad de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSnapshotAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

DBSnapshotIdentifier ya lo utiliza una instantánea existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSnapshotNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBSnapshotIdentifier no hace referencia a una instantánea de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSubnetGroupAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

DBSubnetGroupName ya lo utiliza un grupo de subred de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSubnetGroupDoesNotCoverEnoughAZs (estructura)

Código de estado HTTP devuelto: 400.

Las subredes del grupo de subredes de base de datos deben incluir al menos dos zonas de disponibilidad a menos que solo haya una zona de disponibilidad.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSubnetGroupNotAllowedFault (estructura)

Código de estado HTTP devuelto: 400.

Indica que no debe especificarse el DBSubnetGroup al crear réplicas de lectura que están en la misma región que la instancia de origen.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSubnetGroupNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

DBSubnetGroupName no hace referencia a un grupo de subred de base de datos existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSubnetGroupQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

La solicitud daría lugar a que el usuario superara el número permitido de grupos de subred de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBSubnetQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

La solicitud daría lugar a que el usuario superara el número permitido de subredes en un grupo de subredes de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DBUpgradeDependencyFailureFault (estructura)

Código de estado HTTP devuelto: 400.

Se ha producido un error en la actualización de la base de datos porque no se ha podido modificar un recurso del que depende la base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

DomainNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

Domain no hace referencia a un dominio de Active Directory existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

EventSubscriptionQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

Ha superado el número de eventos al que puede suscribirse.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

GlobalClusterAlreadyExistsFault (estructura)

Código de estado HTTP devuelto: 400.

El `GlobalClusterIdentifier` ya existe. Elija un nuevo identificador de base de datos global (nombre único) para crear un nuevo clúster de base de datos global.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

GlobalClusterNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

El `GlobalClusterIdentifier` no hace referencia a un clúster de base de datos global existente.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

Mensaje que describe los detalles del problema.

GlobalClusterQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

El número de clústeres de bases de datos globales para esta cuenta ya está en el máximo permitido.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InstanceQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

La solicitud daría lugar a que el usuario superara el número permitido de instancias de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InsufficientDBClusterCapacityFault (estructura)

Código de estado HTTP devuelto: 403.

El clúster de base de datos no tiene capacidad suficiente para la operación actual.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InsufficientDBInstanceCapacityFault (estructura)

Código de estado HTTP devuelto: 400.

La clase de instancia de base de datos especificada no está disponible en la zona de disponibilidad especificada.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InsufficientStorageClusterCapacityFault (estructura)

Código de estado HTTP devuelto: 400.

Existe insuficiente almacenamiento disponible para la acción en curso. Es posible que pueda resolver este error mediante la actualización de su grupo de subredes para utilizar diferentes zonas de disponibilidad que tienen más almacenamiento disponible.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBClusterEndpointStateFault (estructura)

Código de estado HTTP devuelto: 400.

La operación solicitada no se puede realizar en el punto de conexión mientras este se encuentre en este estado.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBClusterSnapshotStateFault (estructura)

Código de estado HTTP devuelto: 400.

El valor suministrado no es un estado válido de instantánea de clúster de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBClusterStateFault (estructura)

Código de estado HTTP devuelto: 400.

El clúster de la base de datos no se encuentra en un estado válido.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBInstanceStateFault (estructura)

Código de estado HTTP devuelto: 400.

La instancia de base de datos especificada no se encuentra en el estado disponible.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBParameterGroupStateFault (estructura)

Código de estado HTTP devuelto: 400.

El grupo de parámetros de base de datos está en uso o está en un estado no válido. Si intenta eliminar el grupo de parámetros, no puede eliminarlo cuando el grupo de parámetros se encuentra en este estado.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBSecurityGroupStateFault (estructura)

Código de estado HTTP devuelto: 400.

El estado del grupo de seguridad de base de datos no permite la eliminación.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBSnapshotStateFault (estructura)

Código de estado HTTP devuelto: 400.

El estado de la instantánea de base de datos no permite la eliminación.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBSubnetGroupFault (estructura)

Código de estado HTTP devuelto: 400.

Indica que el DBSubnetGroup no pertenece a la misma VPC que una réplica de lectura entre regiones existente de la misma instancia de origen.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBSubnetGroupStateFault (estructura)

Código de estado HTTP devuelto: 400.

El grupo de subredes de base de datos no se puede eliminar porque está en uso.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidDBSubnetStateFault (estructura)

Código de estado HTTP devuelto: 400.

La subred de base de datos no se encuentra en el estado disponible.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidEventSubscriptionStateFault (estructura)

Código de estado HTTP devuelto: 400.

La suscripción a eventos se encuentra en un estado no válido.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidGlobalClusterStateFault (estructura)

Código de estado HTTP devuelto: 400.

El estado del clúster global no es válido y no puede realizar la operación solicitada.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidOptionGroupStateFault (estructura)

Código de estado HTTP devuelto: 400.

El grupo de opciones no se encuentra en el estado disponible.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidRestoreFault (estructura)

Código de estado HTTP devuelto: 400.

No se puede restaurar desde la copia de seguridad de VPC a una instancia de base de datos no VPC.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidSubnet (estructura)

Código de estado HTTP devuelto: 400.

La subred solicitada no es válida o se solicitaron varias subredes que no están en la misma VPC.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

InvalidVPCNetworkStateFault (estructura)

Código de estado HTTP devuelto: 400.

El grupo de subredes de base de datos no cubre todas las zonas de disponibilidad después de crearla ya que los usuarios cambian.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

KMSKeyNotAccessibleFault (estructura)

Código de estado HTTP devuelto: 400.

Error al obtener acceso a la clave de KMS.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

OptionGroupNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

No se encuentra el grupo de opciones designadas.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

PointInTimeRestoreNotEnabledFault (estructura)

Código de estado HTTP devuelto: 400.

SourceDBInstanceIdentifier hace referencia a una instancia de base de datos con BackupRetentionPeriod igual a 0.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

ProvisionedIopsNotAvailableInAZFault (estructura)

Código de estado HTTP devuelto: 400.

Las IOPS provisionadas no están disponibles en la zona de disponibilidad especificada.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

ResourceNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

No se ha encontrado el ID del recurso especificado.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

Mensaje que describe los detalles del problema.

SNSInvalidTopicFault (estructura)

Código de estado HTTP devuelto: 400.

El tema de SNS no es válido.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SNSNoAuthorizationFault (estructura)

Código de estado HTTP devuelto: 400.

No hay autorización de SNS.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SNSTopicArnNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

El ARN del tema de SNS no se puede encontrar.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SharedSnapshotQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

Ha superado el número máximo de cuentas con los que puede compartir una instantánea manual de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SnapshotQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

La solicitud daría lugar a que el usuario superara el número permitido de instantáneas de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SourceNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

No se encuentra el origen.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

StorageQuotaExceededFault (estructura)

Código de estado HTTP devuelto: 400.

La solicitud daría lugar a que el usuario superara la cantidad permitida de almacenamiento disponible a través de todas las instancias de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

StorageTypeNotSupportedFault (estructura)

Código de estado HTTP devuelto: 400.

El StorageType especificado no se puede asociar a la instancia de base de datos.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SubnetAlreadyInUse (estructura)

Código de estado HTTP devuelto: 400.

La subred de base de datos ya está en uso en la zona de disponibilidad.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SubscriptionAlreadyExistFault (estructura)

Código de estado HTTP devuelto: 400.

Esta suscripción ya existe.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SubscriptionCategoryNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

No se ha encontrado la categoría de suscripción designada.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

SubscriptionNotFoundFault (estructura)

Código de estado HTTP devuelto: 404.

No se encuentra la suscripción designada.

Campos

- message: se trata de un ExceptionMessage, del tipo: `string` (una cadena codificada con UTF-8).

  Mensaje que describe los detalles del problema.

Referencia de la API de datos de Amazon Neptune

En este capítulo se documentan las operaciones de la API de datos de Neptune que puede utilizar para cargar, acceder y mantener los datos de un gráfico de Neptune.

La API de datos de Neptune proporciona compatibilidad con el SDK para cargar datos, ejecutar consultas, obtener información sobre los datos y ejecutar operaciones de machine learning. Es compatible con los lenguajes de consulta de Gremlin y openCypher en Neptune y está disponible en todos los lenguajes del SDK. Firma automáticamente las solicitudes de la API y simplifica considerablemente la integración de Neptune en las aplicaciones.

Contenido

- BulkLoadIdNotFoundException (estructura)
- CancelledByUserException (estructura)
- ClientTimeoutException (estructura)
- ConcurrentModificationException (estructura)
- ConstraintViolationException (estructura)
- ExpiredStreamException (estructura)
- FailureByQueryException (estructura)
- IllegalArgumentException (estructura)
- InternalFailureException (estructura)
- InvalidArgumentException (estructura)
- InvalidNumericDataException (estructura)
- InvalidParameterException (estructura)
- LoadUrlAccessDeniedException (estructura)
- MalformedQueryException (estructura)
- MemoryLimitExceededException (estructura)
- MethodNotAllowedException (estructura)
- MissingParameterException (estructura)
- MLResourceNotFoundException (estructura)
- ParsingException (estructura)
- PreconditionsFailedException (estructura)
- QueryLimitExceededException (estructura)
- QueryLimitException (estructura)
- QueryTooLargeException (estructura)
- ReadOnlyViolationException (estructura)
- S3Exception (estructura)
- ServerShutdownException (estructura)
- StatisticsNotAvailableException (estructura)
- StreamRecordsNotFoundException (estructura)
- ThrottlingException (estructura)
- TimeLimitExceededException (estructura)

- TooManyRequestsException (estructura)
- UnsupportedOperationException (estructura)
- UnloadUrlAccessDeniedException (estructura)

API de estructura general, restablecimiento rápido y motor de plano de datos de Neptune

Operaciones del motor:

- GetEngineStatus (acción)
- ExecuteFastReset (acción)

Estructuras de funcionamiento del motor:

- QueryLanguageVersion (estructura)
- FastResetToken (estructura)

GetEngineStatus (acción)

> El nombre de la AWS CLI para esta API es: `get-engine-status`.

Recupera el estado de la base de datos de gráficos del host.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetEngineStatus en dicho clúster.

Solicitud

- Ningún parámetro de solicitud.

Respuesta

- dbEngineVersion: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Se establece en la versión del motor de Neptune que se ejecuta en el clúster de base de datos. Si esta versión del motor se ha parcheado de forma manual desde que se lanzó, el número de versión está precedido por `Patch-`.

- dfeQueryEngine: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Se establece en `enabled` si el motor DFE está totalmente habilitado o en `viaQueryHint` (valor predeterminado) si el motor de DFE solo se usa con consultas que tienen la sugerencia de consulta `useDFE` establecida en `true`.
- features: es una matriz de mapas de pares de clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Cada valor es un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  Incluye información de estado sobre las características habilitadas en el clúster de base de datos.
- gremlin: objeto QueryLanguageVersion.

  Incluye información sobre el lenguaje de consultas de Gremlin disponible en el clúster. Específicamente, incluye un campo de versión que especifica la versión actual de TinkerPop que utiliza el motor.
- labMode: es una matriz de mapas de pares de clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Cada valor es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Incluye los ajustes del modo lab que utiliza el motor.
- opencypher: objeto QueryLanguageVersion.

  Incluye información sobre el lenguaje de consultas de openCypher disponible en el clúster. Específicamente, incluye un campo de versión que especifica la versión actual de openCypher que utiliza el motor.
- role: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Se establece en `reader` si la instancia es una réplica de lectura o en `writer` si la instancia es la instancia principal.
- rollingBackTrxCount: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

Si hay transacciones que se están restaurando, este campo se establece en el número de dichas transacciones. Si no hay ninguna transacción, el campo no aparecerá.

- rollingBackTrxEarliestStartTime: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Se establece en la hora de inicio de la primera transacción que se va a restaurar. Si no se está revirtiendo ninguna transacción, este campo no aparecerá.

- settings: es una matriz de mapas de pares de clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Cada valor es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Incluye información sobre la configuración actual del clúster de base de datos. Por ejemplo, incluye el valor de tiempo de espera de la consulta del clúster actual (`clusterQueryTimeoutInMs`).

- sparql: objeto QueryLanguageVersion.

  Incluye información sobre el lenguaje de consultas de SPARQL disponible en el clúster. Específicamente, incluye un campo de versión que especifica la versión actual de SPARQL que utiliza el motor.

- startTime: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Se establece en la hora UTC a la que se inició el proceso actual del servidor.

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Se establece en `healthy` si la instancia no está experimentando problemas. Si la instancia se está recuperando de un bloqueo o se reinicia y hay transacciones activas en ejecución desde el último cierre del servidor, status se establece en `recovery`.

Errores

- UnsupportedOperationException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException

- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

ExecuteFastReset (acción)

El nombre de la AWS CLI para esta API es: `execute-fast-reset`.

La API de REST de restablecimiento rápido le permite restablecer un gráfico de Neptune de forma rápida y sencilla, eliminando todos sus datos.

La restauración rápida de Neptune es un proceso de dos pasos. En primer lugar, llama a `ExecuteFastReset` con la opción `action` establecida en `initiateDatabaseReset`. Esto devuelve un token de UUID que luego debe incluirse cuando se vuelva a llamar a `ExecuteFastReset` con opción `action` establecida en `performDatabaseReset`. Consulte Vaciado de un clúster de base de datos de Amazon Neptune con la API de restablecimiento rápido.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ResetDatabase en dicho clúster.

Solicitud

- action (en la CLI: `--action`): obligatorio: una acción, del tipo: `string` (una cadena codificada con UTF-8).

  La acción de restablecimiento rápido. Uno de los valores siguientes:

  - **`initiateDatabaseReset`**: esta acción genera un token único necesario para realizar realmente el restablecimiento rápido.
  - **`performDatabaseReset`**: esta acción utiliza el token generado por la acción `initiateDatabaseReset` para realizar realmente el restablecimiento rápido.

- token (en la CLI: `--token`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El token de restablecimiento rápido para iniciar el restablecimiento.

Respuesta

- payload: objeto FastResetToken.

La acción `initiateDatabaseReset` solo devuelve `payload`, que incluye el token único que se usará con la acción `performDatabaseReset` para que se realice el restablecimiento.

- status: obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La acción `performDatabaseReset` solo devuelve `status`, que indica si se acepta o no la solicitud de restablecimiento rápido.

Errores

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- ServerShutdownException
- PreconditionsFailedException
- MethodNotAllowedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Estructuras de funcionamiento del motor:

QueryLanguageVersion (estructura)

Estructura para expresar la versión del lenguaje de consulta.

Campos

- version: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión del lenguaje de consulta.

FastResetToken (estructura)

Estructura que incluye el token de restablecimiento rápido que se utiliza para iniciar un restablecimiento rápido.

Campos

- token: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un UUID generado por la base de datos en la acción `initiateDatabaseReset` y luego utilizado por `performDatabaseReset` para restablecer la base de datos.

API de consulta de Neptune

Acciones de consulta de Gremlin:

- ExecuteGremlinQuery (acción)
- ExecuteGremlinExplainQuery (acción)
- ExecuteGremlinProfileQuery (action)
- ListGremlinQueries (acción)
- GetGremlinQueryStatus (acción)
- CancelGremlinQuery (acción)

Acciones de consulta de openCypher:

- ExecuteOpenCypherQuery (acción)
- ExecuteOpenCypherExplainQuery (acción)
- ListOpenCypherQueries (acción)
- GetOpenCypherQueryStatus (acción)
- CancelOpenCypherQuery (acción)

Estructuras de consulta:

- QueryEvalStats (estructura)
- GremlinQueryStatus (estructura)

- GremlinQueryStatusAttributes (estructura)

ExecuteGremlinQuery (acción)

El nombre de la AWS CLI para esta API es: `execute-gremlin-query`.

Este comando ejecuta una consulta de Gremlin. Amazon Neptune es compatible con Apache TinkerPop3 y Gremlin, por lo que puede usar el lenguaje transversal de Gremlin para consultar el gráfico, tal y como se describe en The Graph en la documentación de Apache TinkerPop3. También puede encontrar más información en Acceso al gráfico de Neptune con Gremlin.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita una de las siguientes acciones de IAM en dicho clúster, en función de la consulta:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:Gremlin se puede utilizar en el documento de política para restringir el uso de consultas de Gremlin (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- gremlinQuery (en la CLI: `--gremlin-query`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Con esta API, puede ejecutar consultas de Gremlin en formato de cadena de la misma manera que con el punto de conexión HTTP. La interfaz es compatible con cualquier versión de Gremlin que utilice el clúster de base de datos (consulte la sección cliente TinkerPop para determinar las versiones de Gremlin que admite la versión del motor).
- serializer (en la CLI: `--serializer`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si no es nulo, los resultados de la consulta se devuelven en un mensaje de respuesta serializado en el formato especificado por este parámetro. Consulte la sección GraphSON en la documentación de TinkerPop para ver una lista de los formatos compatibles actualmente.

Respuesta

- meta: un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  Metadatos sobre la consulta de Gremlin.
- requestId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único de la consulta de Gremlin.
- result: un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  El resultado de la consulta de Gremlin del servidor.
- status: objeto GremlinQueryStatusAttributes.

  El estado de la consulta de Gremlin.

Errores

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException

- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinExplainQuery (acción)

El nombre de la AWS CLI para esta API es: `execute-gremlin-explain-query`.

Ejecuta una consulta Explain de Gremlin.

Amazon Neptune ha añadido una característica de Gremlin denominada `explain` que proporciona una herramienta de autoservicio para comprender el enfoque de ejecución que adopta el motor de Neptune para la consulta. Puede invocarla añadiendo un parámetro `explain` a una llamada HTTP que envíe una consulta de Gremlin.

La característica explain proporciona información sobre la estructura lógica de los planes de ejecución de consultas. Puede utilizar esta información para identificar posibles obstáculos en la evaluación y la ejecución y ajustar la consulta, tal y como se explica en Ajuste de consultas de Gremlin. También puede usar sugerencias de consulta para mejorar los planes de ejecución de consultas.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita una de las siguientes acciones de IAM en dicho clúster, en función de la consulta:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:Gremlin se puede utilizar en el documento de política para restringir el uso de consultas de Gremlin (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- gremlinQuery (en la CLI: `--gremlin-query`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cadena de consulta Explain de Gremlin.

Respuesta

- output: un ReportAsText, del tipo: `blob` (un bloque de datos binarios sin interpretar).

  Un blob de texto que incluye el resultado Explain de Gremlin, tal y como se describe en Ajuste de consultas de Gremlin.

Errores

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

ExecuteGremlinProfileQuery (action)

El nombre de la AWS CLI para esta API es: `execute-gremlin-profile-query`.

Ejecuta un consulta Profile de Gremlin, que ejecuta un recorrido especificado, recopila varias métricas sobre la ejecución y genera un informe de perfil como resultado. Consulte API de perfil de Gremlin en Neptune para obtener más información.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ReadDataViaQuery en dicho clúster.

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:Gremlin se puede utilizar en el documento de política para restringir el uso de consultas de Gremlin (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- chop (en la CLI: `--chop`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Si es distinto de cero, hace que la cadena de resultados se trunque con ese número de caracteres. Si se establece en cero, la cadena contiene todos los resultados.
- gremlinQuery (en la CLI: `--gremlin-query`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cadena de consulta de Gremlin para la que generar el perfil.
- indexOps (en la CLI: `--index-ops`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si este indicador está establecido en `TRUE`, muestra un informe detallado de todas las operaciones de índice que se realizaron durante la ejecución y serialización de consultas.
- results (en la CLI: `--results`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si este indicador está establecido en `TRUE`, los resultados de la consulta se recopilan y se muestran como parte del informe de perfil. Si está establecido en `FALSE`, solo se muestra el recuento de resultados.

- serializer (en la CLI: `--serializer`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si no es nulo, los resultados recopilados se devuelven en un mensaje de respuesta serializado en el formato especificado por este parámetro. Consulte API de perfil de Gremlin en Neptune para obtener más información.

Respuesta

- output: un ReportAsText, del tipo: `blob` (un bloque de datos binarios sin interpretar).

  Un blob de texto que incluye el resultado Profile de Gremlin. Consulte API de perfil de Gremlin en Neptune para obtener más información.

Errores

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException

- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListGremlinQueries (acción)

El nombre de la AWS CLI para esta API es: `list-gremlin-queries`.

Muestra las consultas activas de Gremlin. Consulte API del estado de la consulta de Gremlin para obtener más información sobre el resultado.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetQueryStatus en dicho clúster.

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:Gremlin se puede utilizar en el documento de política para restringir el uso de consultas de Gremlin (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- includeWaiting (en la CLI: `--include-waiting`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `TRUE`, la lista devuelta incluye las consultas en espera. El valor predeterminado es `FALSE`;

Respuesta

- acceptedQueryCount: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de consultas que se han aceptado pero que aún no se han completado, incluidas las consultas en la cola.
- queries: matriz de objetos GremlinQueryStatus.

  Una lista de las consultas actuales.
- runningQueryCount: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de consultas de Gremlin que se están ejecutando actualmente.

Errores

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetGremlinQueryStatus (acción)

El nombre de la AWS CLI para esta API es: `get-gremlin-query-status`.

Obtiene el estado de una consulta específica de Gremlin.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetQueryStatus en dicho clúster.

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:Gremlin se puede utilizar en el documento de política para restringir el uso de consultas de Gremlin (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- queryId (en la CLI: `--query-id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único que identifica la consulta de Gremlin.

Respuesta

- queryEvalStats: objeto QueryEvalStats.

  El estado de evaluación de la consulta de Gremlin.
- queryId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la consulta cuyo estado se devuelve.
- queryString: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cadena de consulta de Gremlin.

Errores

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

CancelGremlinQuery (acción)

El nombre de la AWS CLI para esta API es: `cancel-gremlin-query`.

Cancela una consulta de Gremlin. Consulte Cancelación de consultas de Gremlin para obtener más información.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:CancelQuery en dicho clúster.

Solicitud

- queryId (en la CLI: `--query-id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único que identifica la consulta que se va a cancelar.

Respuesta

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la cancelación

Errores

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException

- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Acciones de consulta de openCypher:

ExecuteOpenCypherQuery (acción)

El nombre de la AWS CLI para esta API es: `execute-open-cypher-query`.

Ejecuta una consulta de openCypher. Consulte Acceso al gráfico de Neptune con openCypher para obtener más información.

Neptune permite crear aplicaciones de gráficos mediante openCypher, que actualmente es uno de los lenguajes de consulta más populares entre los desarrolladores que trabajan con bases de datos de gráficos. A los desarrolladores, analistas empresariales y científicos de datos les gusta la sintaxis declarativa de openCypher, inspirada en SQL, ya que proporciona una estructura conocida para consultar gráficos de propiedades.

Neo4j desarrolló originalmente el lenguaje de openCyperh y que pasó a ser de código abierto en 2015. Además, contribuyó al proyecto openCypher en virtud de una licencia de código abierto Apache 2.

Tenga en cuenta que al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita una de las acciones de IAM en dicho clúster, en función de la consulta:

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

Además, tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:OpenCypher se puede utilizar en el documento de política para restringir el uso

de consultas de openCypher (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- openCypherQuery (en la CLI: `--open-cypher-query`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cadena de consulta de openCypher que se va a ejecutar.
- parameters (en la CLI: `--parameters`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los parámetros de consulta de openCypher para la ejecución de consultas. Consulte Ejemplos de consultas parametrizadas de openCypher para obtener más información.

Respuesta

- results: obligatorio: un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  Los resultados de la consulta de openCypherquery.

Errores

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException

- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteOpenCypherExplainQuery (acción)

El nombre de la AWS CLI para esta API es: `execute-open-cypher-explain-query`.

Ejecuta una solicitud `explain` de openCypher. Consulte La característica Explain de openCypher para obtener más información.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ReadDataViaQuery en dicho clúster.

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:OpenCypher se puede utilizar en el documento de política para restringir el uso de consultas de openCypher (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- explainMode (en la CLI: `--explain-mode`): obligatorio: un valor OpenCypherExplainMode, del tipo: `string` (una cadena codificada con UTF-8).

  El modo `explain` de openCypher. Puede ser: `static`, `dynamic` o `details`.
- openCypherQuery (en la CLI: `--open-cypher-query`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

La cadena de consulta de openCypher.

- parameters (en la CLI: `--parameters`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los parámetros de consulta de openCypher.

Respuesta

- results: obligatorio: un blob, del tipo: `blob` (un bloque de datos binarios sin interpretar).

  Un blob de texto que incluye los resultados `explain` de openCypher.

Errores

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException

- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListOpenCypherQueries (acción)

El nombre de la AWS CLI para esta API es: `list-open-cypher-queries`.

Muestra las consultas activas de openCypher. Consulte Punto de conexión de estado de openCypher en Neptune para obtener más información.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetQueryStatus en dicho clúster.

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:OpenCypher se puede utilizar en el documento de política para restringir el uso de consultas de openCypher (consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- includeWaiting (en la CLI: `--include-waiting`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Cuando se establece en `TRUE` y otros parámetros no están presentes, se devuelve la información de estado tanto para las consultas en espera como para las consultas en ejecución.

Respuesta

- acceptedQueryCount: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de consultas que se han aceptado pero que aún no se han completado, incluidas las consultas en la cola.
- queries: matriz de objetos GremlinQueryStatus.

  Una lista de las consultas actuales de openCypher.
- runningQueryCount: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

El número de consultas de openCypher que se están ejecutando actualmente.

Errores

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetOpenCypherQueryStatus (acción)

El nombre de la AWS CLI para esta API es: `get-open-cypher-query-status`.

Recupera el estado de una consulta específica de openCypher.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetQueryStatus en dicho clúster.

Tenga en cuenta que la clave de condición de IAM neptune-db:QueryLanguage:OpenCypher se puede utilizar en el documento de política para restringir el uso de consultas de openCypher

(consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune).

Solicitud

- queryId (en la CLI: `--query-id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único de la consulta de openCypher del que se va a recuperar el estado de la consulta.

Respuesta

- queryEvalStats: objeto QueryEvalStats.

  El estado de evaluación de la consulta de openCypher.
- queryId: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único de la consulta cuyo estado se devuelve.
- queryString: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cadena de consulta de openCypher.

Errores

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException

- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelOpenCypherQuery (acción)

El nombre de la AWS CLI para esta API es: `cancel-open-cypher-query`.

Cancela una consulta específica de openCypher. Consulte Punto de conexión de estado de openCypher en Neptune para obtener más información.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:CancelQuery en dicho clúster.

Solicitud

- queryId (en la CLI: `--query-id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único de la consulta de openCypher que se va a cancelar.
- silent (en la CLI: `--silent`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en`TRUE`, se lleva a cabo de forma silenciosa la cancelación de la consulta de openCypher.

Respuesta

- payload: un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  La carga de cancelación de la consulta de openCypher.
- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de cancelación de la consulta de openCypher.

Errores

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Estructuras de consulta:

QueryEvalStats (estructura)

Estructura para capturar estadísticas de consultas, como el número de consultas en ejecución, aceptadas o en espera y sus detalles.

Campos

- cancelled: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Se establece en TRUE si la consulta se ha cancelado o en FALSE en caso contrario.
- elapsed: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número de milisegundos que la consulta lleva en ejecución.

- subqueries: se trata de un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  El número de subconsultas de esta consulta.
- waited: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Indica cuánto tiempo esperó la consulta, en milisegundos.

GremlinQueryStatus (estructura)

Captura el estado de una consulta de Gremlin (consulta la página API del estado de la consulta de Gremlin).

Campos

- queryEvalStats: se trata de un objeto QueryEvalStats.

  Las estadísticas de la consulta de Gremlin.
- queryId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la consulta de Gremlin.
- queryString: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La cadena de la consulta de Gremlin.

GremlinQueryStatusAttributes (estructura)

Incluye los componentes de estado de una consulta de Gremlin.

Campos

- attributes: se trata de un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  Atributos del estado de la consulta de Gremlin.
- code: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El código de respuesta HTTP devuelto por la solicitud de consulta de Gremlin.
- message: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Mensaje del estado.

API del programa de carga masiva de plano de datos de Neptune

Acciones de carga masiva:

- StartLoaderJob (acción)
- GetLoaderJobStatus (acción)
- ListLoaderJobs (acción)
- CancelLoaderJob (acción)

Estructura de carga masiva:

- LoaderIdResult (estructura)

StartLoaderJob (acción)

El nombre de la AWS CLI para esta API es: `start-loader-job`.

Inicia un trabajo del programa de carga masiva de Neptune para cargar datos de un bucket de Amazon S3 en una instancia de base de datos de Neptune. Consulte Uso del programa de carga masiva de Amazon Neptune para adquirir datos.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:StartLoaderJob en dicho clúster.

Solicitud

- dependencies (en la CLI: `--dependencies`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Se trata de un parámetro opcional que puede supeditar una solicitud de carga en cola a la finalización correcta de uno o varios trabajos anteriores de la cola.

  Neptune puede poner en cola hasta 64 solicitudes de carga a la vez, si sus parámetros `queueRequest` están configurados en `"TRUE"`. El parámetro `dependencies` le permite hacer

que la ejecución de dicha solicitud en cola dependa de la finalización correcta de una o más solicitudes anteriores especificadas en la cola.

Por ejemplo, si las cargas `Job-A` y `Job-B` son independientes entre sí, pero la carga `Job-C` necesita `Job-A` y `Job-B` debe terminar antes de que comience, proceda de la siguiente manera:

1. Envíe `load-job-A` y `load-job-B` uno tras otro en cualquier orden, y guarde sus identificadores de carga.
2. Envíe `load-job-C` con los identificadores de carga de los dos trabajos en su campo `dependencies`:

Example

```
"dependencies" : ["(job_A_load_id)", "(job_B_load_id)"]
```

Debido al parámetro `dependencies`, el programa de carga en bloque no iniciará `Job-C` hasta que `Job-A` y `Job-B` se hayan completado correctamente. Si se produce un error en alguno de ellos, Job-C no se ejecutará y su estado se establecerá en `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Puede configurar varios niveles de dependencia de esta manera, de modo que el error de un trabajo provoque la cancelación de todas las solicitudes que dependen directa o indirectamente de él.

- failOnError (en la CLI: `--fail-on-error`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  `failOnError`: un indicador para activar la detención total al encontrar un error.

  Valores permitidos: `"TRUE"` y `"FALSE"`.

  Valor predeterminado: `"TRUE"`.

  Cuando este parámetro se establece en `"FALSE"`, el programa de carga intenta cargar todos los datos de la ubicación especificada, omitiendo cualquier entrada con errores.

  Cuando este parámetro se establece en `"TRUE"`, el programa de carga se detiene en cuanto encuentra un error. Los datos cargados hasta ese momento persisten.

- format (en la CLI: `--format`): obligatorio: un formato, del tipo: `string` (una cadena codificada con UTF-8).

El formato de los datos. Para obtener más información sobre los formatos de los datos para el comando `Loader` de Neptune, consulte Formatos de los datos de carga.

Valores permitidos

- **`csv`** para el formato de datos CSV de Gremlin.
- **`opencypher`** para el formato de datos CSV de openCypher.
- **`ntriples`** para el formato de datos RDF N-Triples.
- **`nquads`** para el formato de datos RDF N-Quads.
- **`rdfxml`** para el formato de datos RDF RDF\XML.
- **`turtle`** para el formato de datos RDF de Turtle.

- iamRoleArn (en la CLI: `--iam-role-arn`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de recurso de Amazon (ARN) para que la instancia de base de datos de Neptune asuma el rol de IAM para obtener acceso al bucket de S3. El ARN del rol de IAM que se indica aquí debe adjuntarse al clúster de base de datos (consulte Adición del rol de IAM a un clúster de Amazon Neptune).
- mode (en la CLI: `--mode`): un modo, del tipo: `string` (una cadena codificada con UTF-8).

  El modo de tarea de carga.

  Valores permitidos: `RESUME`, `NEW`, `AUTO`.

  Valor predeterminado: `AUTO`.

  - `RESUME`: en el modo RESUME, el programa de carga busca una carga anterior de este origen y, si encuentra una, reanuda ese trabajo de carga. Si no se encuentra ningún trabajo de carga anterior, el programa de carga se detiene.

    El programa de carga evita la recarga de archivos cargados correctamente en un trabajo anterior. Solo intenta procesar los archivos con errores. Si ha eliminado los datos cargados anteriormente del clúster de Neptune, esos datos no se vuelven a cargar en este modo. Si un trabajo de carga anterior ha cargado todos los archivos del mismo origen correctamente, no se vuelve a cargar nada y el programa de carga devuelve una operación correcta.
  - `NEW`: en el modo NEW, crea una solicitud de carga, independientemente de cualquier carga anterior. Puede utilizar este modo para volver a cargar todos los datos de un origen después

de descartar los datos cargados anteriormente desde el clúster de Neptune o bien para cargar nuevos datos disponibles en el mismo origen.

- `AUTO`: en el modo AUTO, el programa de carga busca un trabajo de carga anterior del mismo origen y, si encuentra uno, lo reanuda, igual que en el modo `RESUME`.

  Si el programa de carga no encuentra un trabajo de carga anterior del mismo origen, carga todos los datos del origen, al igual que en el modo `NEW`.

- parallelism (en la CLI: `--parallelism`): un paralelismo, del tipo: `string` (una cadena codificada con UTF-8).

  El parámetro `parallelism` opcional que se puede establecer para reducir el número de subprocesos utilizados por el proceso de carga masiva.

  Valores permitidos:

  - `LOW`: el número de subprocesos utilizados es el número de vCPU disponibles dividido entre 8.
  - `MEDIUM`: el número de subprocesos utilizados es el número de vCPU disponibles dividido entre 2.
  - `HIGH`: el número de subprocesos utilizados es el mismo número de vCPU disponibles.
  - `OVERSUBSCRIBE`: el número de subprocesos utilizados es el número de vCPU disponibles multiplicado por 2. Si se utiliza este valor, el programa de carga masiva absorbe todos los recursos disponibles.

    Sin embargo, esto no significa que el ajuste de `OVERSUBSCRIBE` dé como resultado un uso del 100 % de la CPU. Dado que la operación de carga está vinculada a la E/S, la máxima utilización de la CPU que cabe esperar se sitúa entre el 60 y el 70 %.

  Valor predeterminado: `HIGH`

  En ocasiones, este ajuste de `parallelism` puede provocar un bloqueo entre los subprocesos al cargar datos de openCypher. Cuando esto ocurre, Neptune devuelve el error `LOAD_DATA_DEADLOCK`. Por lo general, puede solucionar el problema configurando `parallelism` en un ajuste inferior y volviendo a intentar ejecutar el comando de carga.

- parserConfiguration (en la CLI: `--parser-configuration`): es una matriz de mapeo de pares de clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Cada valor es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

`parserConfiguration`: objeto opcional con valores de configuración de analizador adicionales. Cada uno de los parámetros secundarios también es opcional:

- **`namedGraphUri`**: el gráfico predeterminado para todos los formatos RDF cuando no se especifica ningún gráfico (para formatos no QUAD y entradas NQUAD sin gráfico).

  El valor predeterminado es `https://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

- **`baseUri`**: el URI base para los formatos RDF/XML y Turtle.

  El valor predeterminado es `https://aws.amazon.com/neptune/default`.

- **`allowEmptyStrings`**: los usuarios de Gremlin deben ser capaces de pasar valores de cadenas vacías (“”) como propiedades de nodo y borde al cargar datos CSV. Si `allowEmptyStrings` se establece en `false` (el valor predeterminado), estas cadenas vacías se tratan como nulas y no se cargan.

  Si `allowEmptyStrings` se establece en `true`, el programa de carga trata las cadenas vacías como valores de propiedad válidos y las carga en consecuencia.

- queueRequest (en la CLI: `--queue-request`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Se trata de un parámetro de indicador opcional que indica si la solicitud de carga se puede poner en cola o no.

  No tiene que esperar a que se complete un trabajo de carga antes de emitir el siguiente, porque Neptune puede poner en cola hasta 64 trabajos a la vez, siempre que sus parámetros `queueRequest` estén configurados en `"TRUE"`. El orden de cola de los trabajos será el primero en entrar es el primero en salir (FIFO).

  Si el parámetro `queueRequest` se omite o se establece en `"FALSE"`, se producirá un error en la solicitud de carga si ya se está ejecutando otro trabajo de carga.

  Valores permitidos: `"TRUE"` y `"FALSE"`.

  Valor predeterminado: `"FALSE"`.

- s3BucketRegion (en la CLI: `--s-3-bucket-region`): obligatorio: una S3BucketRegion, del tipo: `string` (una cadena codificada con UTF-8).

La región de Amazon del bucket S3. Debe coincidir con la región de Amazon del clúster de base de datos.

- source (en la CLI: `--source`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El parámetro `source` acepta un URI de S3 que identifica un solo archivo, varios archivos, una carpeta o varias carpetas. Neptune carga todos los archivos de datos de cualquier carpeta especificada.

  El URI puede tener cualquiera de los siguientes formatos:

  - `s3://(bucket_name)/(object-key-name)`
  - `https://s3.amazonaws.com/(bucket_name)/(object-key-name)`
  - `https://s3.us-east-1.amazonaws.com/(bucket_name)/(object-key-name)`

  El elemento `object-key-name` del URI equivale al parámetro prefix de una llamada a la API ListObjects de S3. Identifica todos los objetos del bucket de S3 especificado cuyos nombres comienzan con ese prefijo. Puede ser un único archivo o carpeta o varios archivos o carpetas.

  La carpeta o carpetas especificadas pueden contener varios archivos de vértice y varios archivos de borde.

- updateSingleCardinalityProperties (en la CLI: `--update-single-cardinality-properties`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  `updateSingleCardinalityProperties` es un parámetro opcional que controla cómo el programa de carga masiva trata un nuevo valor para las propiedades de vértice o borde de cardinalidad única. Esto no se admite para cargar datos de openCypher.

  Valores permitidos: `"TRUE"` y `"FALSE"`.

  Valor predeterminado: `"FALSE"`.

  De forma predeterminada, o cuando `updateSingleCardinalityProperties` está configurado explícitamente en `"FALSE"`, el programa de carga trata un nuevo valor como un error, porque infringe la cardinalidad única.

  Por el contrario, cuando `updateSingleCardinalityProperties` está configurado en `"TRUE"`, el programa de carga en bloque reemplaza el valor existente por el nuevo. Si se proporcionan varios valores de propiedades de vértices de borde o de cardinalidad única en los archivos origen

que se están cargando, el valor final al terminar la carga masiva podría ser cualquiera de esos nuevos valores. El programa de carga solo garantiza que el valor existente se ha reemplazado por uno de los nuevos.

- userProvidedEdgeIds (en la CLI: `--user-provided-edge-ids`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Este parámetro solo es necesario cuando se cargan datos de openCypher que incluyen identificadores de relación. Debe incluirse y configurarse en `True` cuando los identificadores de relación de openCypher se proporcionen de forma explícita en los datos de carga (recomendado).

  Si `userProvidedEdgeIds` está ausente o se establece en `True`, debe haber una columna `:ID` en todos los archivos de relaciones de la carga.

  Cuando `userProvidedEdgeIds` está presente y se establece en `False`, los archivos de relaciones de la carga no deben contener ninguna columna `:ID`. En su lugar, el programa de carga de Neptune genera automáticamente un identificador para cada relación.

  Resulta útil proporcionar los identificadores de relación de forma explícita para que el programa de carga pueda reanudar la carga una vez que se haya corregido un error en los datos CSV, sin tener que volver a cargar ninguna relación que ya se haya cargado. Si los identificadores de relación no se han asignado de forma explícita, el programa de carga no puede reanudar una carga fallida si se ha tenido que corregir algún archivo de relación y, en su lugar, debe volver a cargar todas las relaciones.

Respuesta

- payload: obligatorio: es una matriz de mapeo de pares de clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Cada valor es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Incluye un par de nombre-valor `loadId` que proporciona un identificador para la operación de carga.

- status: obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de devolución HTTP que indica el estado del trabajo de carga.

Errores

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- InternalFailureException
- S3Exception
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetLoaderJobStatus (acción)

El nombre de la AWS CLI para esta API es: `get-loader-job-status`.

Obtiene información del estado de un determinado trabajo de carga. Neptune realiza un seguimiento de los 1024 trabajos de carga masiva más recientes y solo almacena los últimos 10 000 detalles de error por trabajo.

Consulte API de obtención de estado del programa de carga de Neptune para obtener más información.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetLoaderJobStatus en dicho clúster.

Solicitud

- details (en la CLI: `--details`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

Indicador que especifica si se deben incluir o no detalles más allá del estado general (`TRUE` o `FALSE`; el valor predeterminado es `FALSE`).

- errors (en la CLI: `--errors`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indicador que especifica si se debe incluir o no una lista de los errores encontrados (`TRUE` o `FALSE`; el valor predeterminado es `FALSE`).

  Dicha lista está paginada. Los parámetros `page` y `errorsPerPage` le permiten desplazarse por todos los errores.

- errorsPerPage (en la CLI: `--errors-per-page`): un PositiveInteger, del tipo: `integer` (un valor entero firmado de 32 bits), al menos 1 ?st?.

  El número de errores devueltos en cada página (un entero positivo; el valor predeterminado es `10`). Solo es válido si el parámetro `errors` está establecido en `TRUE`.

- loadId (en la CLI: `--load-id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de carga del trabajo de carga del que obtener el estado.

- page (en la CLI: `--page`): un PositiveInteger, del tipo: `integer` (un valor entero firmado de 32 bits), al menos 1 ?st?.

  El número de la página de error (un entero positivo; el valor predeterminado es 1). Solo es válido si el parámetro `errors` está establecido en `TRUE`.

Respuesta

- payload: obligatorio: un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  Información de estado sobre el trabajo de carga, en un diseño que podría tener este aspecto:

  Example

```
{
          "status" : "200 OK",
          "payload" : {
            "feedCount" : [
              {
```

```
            "LOAD_FAILED" : (number)
          }
        ],
        "overallStatus" : {
          "fullUri" : "s3://(bucket)/(key)",
          "runNumber" : (number),
          "retryNumber" : (number),
          "status" : "(string)",
          "totalTimeSpent" : (number),
          "startTime" : (number),
          "totalRecords" : (number),
          "totalDuplicates" : (number),
          "parsingErrors" : (number),
          "datatypeMismatchErrors" : (number),
          "insertErrors" : (number),
        },
        "failedFeeds" : [
          {
            "fullUri" : "s3://(bucket)/(key)",
            "runNumber" : (number),
            "retryNumber" : (number),
            "status" : "(string)",
            "totalTimeSpent" : (number),
            "startTime" : (number),
            "totalRecords" : (number),
            "totalDuplicates" : (number),
            "parsingErrors" : (number),
            "datatypeMismatchErrors" : (number),
            "insertErrors" : (number),
          }
        ],
        "errors" : {
          "startIndex" : (number),
          "endIndex" : (number),
          "loadId" : "(string),
          "errorLogs" : [ ]
        }
      }
    }
```

- status: obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Código de respuesta HTTP de la solicitud.

Errores

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ListLoaderJobs (acción)

El nombre de la AWS CLI para esta API es: `list-loader-jobs`.

Recupera una lista de los `loadIds` de todos los trabajos de carga activos.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ListLoaderJobs en dicho clúster.

Solicitud

- includeQueuedLoads (en la CLI: `--include-queued-loads`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Un parámetro opcional que se puede utilizar para excluir los identificadores de carga de las solicitudes de carga en cola cuando se solicita una lista de identificadores de carga estableciendo el parámetro en `FALSE`. El valor predeterminado es `TRUE`.
- limit (en la CLI: `--limit`): un ListLoaderJobsInputLimitInteger, del tipo: `integer` (un valor entero firmado de 32 bits), no inferior a 1 ni superior a 100 ?st?s.

El número de ID de carga que se van a incluir en la lista. Debe ser un entero positivo mayor que cero y no mayor que `100` (que es el valor predeterminado).

Respuesta

- payload: obligatorio: objeto LoaderIdResult.

  La lista solicitada de ID de trabajo.

- status: obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Devuelve el estado de la solicitud de la lista de trabajos.

Errores

- UnsupportedOperationException
- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

CancelLoaderJob (acción)

El nombre de la AWS CLI para esta API es: `cancel-loader-job`.

Cancela un trabajo de carga especificado. Se trata de una solicitud `DELETE` de HTTP. Consulte API de obtención de estado del programa de carga de Neptune para obtener más información.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:CancelLoaderJob en dicho clúster.

Solicitud

- loadId (en la CLI: `--load-id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID del trabajo de carga que se va a eliminar.

Respuesta

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la cancelación.

Errores

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Estructura de carga masiva:

LoaderIdResult (estructura)

Incluye una lista de ID de carga.

Campos

- loadIds: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de ID de carga.

API de plano de datos de transmisiones de Neptune

Acciones de acceso a las transmisiones:

- GetPropertygraphStream (acción)

Estructuras de datos de transmisiones:

- PropertygraphRecord (estructura)
- PropertygraphData (estructura)

GetPropertygraphStream (acción)

El nombre de la AWS CLI para esta API es: `get-propertygraph-stream`.

Obtiene una transmisión para un gráfico de propiedades.

Con la característica Neptune Streams, puede generar una secuencia completa de entradas de registros de cambios que capten cada cambio realizado en los datos de gráficos a medida que se produzca. `GetPropertygraphStream` le permite recopilar estas entradas para un gráfico de propiedad.

La característica Neptune Streams debe estar habilitada en el clúster de base de datos de Neptune. Para habilitar Streams, establezca el parámetro de clúster de base de datos neptune_streams en 1.

Consulte Captura de cambios de gráficos en tiempo real con Neptune Streams.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetStreamRecords en dicho clúster.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita una de las siguientes acciones de IAM, en función de la consulta:

Tenga en cuenta que puede restringir las consultas de gráficos de propiedades mediante las siguientes claves de contexto de IAM:

- neptune-db:QueryLanguage:Gremlin
- neptune-db:QueryLanguage:OpenCypher

Consulte Claves de condición disponibles en las declaraciones de políticas de acceso a datos de IAM de Neptune.

Solicitud

- commitNum (en la CLI: `--commit-num`): un valor Long, del tipo: `long` (un valor entero firmado de 64 bits).

  El número de confirmación del registro de inicio que se va a leer del flujo de registro de cambios. Este parámetro es obligatorio cuando `iteratorType` es `AT_SEQUENCE_NUMBER` o `AFTER_SEQUENCE_NUMBER` y se ignora cuando `iteratorType` es `TRIM_HORIZON` o `LATEST`.
- encoding (en la CLI: `--encoding`): una codificación, del tipo: `string` (una cadena codificada con UTF-8).

  Si se establece en TRUE, Neptune comprime la respuesta mediante la codificación gzip.
- iteratorType (en la CLI: `--iterator-type`): un IteratorType, del tipo: `string` (una cadena codificada con UTF-8).

  Puede ser uno de los siguientes:

  - `AT_SEQUENCE_NUMBER`: indica que la lectura debe comenzar con el número de secuencia de eventos especificado conjuntamente por los parámetros `commitNum` y `opNum`.
  - `AFTER_SEQUENCE_NUMBER`: indica que la lectura debe comenzar justo después del número de secuencia de eventos especificado conjuntamente por los parámetros `commitNum` y `opNum`.

  - `TRIM_HORIZON`: indica que la lectura debe comenzar en el último registro no recortado del sistema, que es el registro más antiguo que no ha caducado (aún no se ha eliminado) del flujo de registro de cambios.
  - `LATEST`: indica que la lectura debe comenzar en el registro más reciente del sistema, que es el último registro que no ha caducado (aún no se ha eliminado) del flujo de registro de cambios.
- limit (en la CLI:`--limit`): un GetPropertygraphStreamInputLimitLong, del tipo: `long` (un valor entero firmado de 64 bits), no inferior a 1 ni superior a 100 000 ?st?s.

  Especifica el número máximo de registros que se van a devolver. También existe un límite de tamaño de 10 MB en la respuesta que no se puede modificar y que prevalece sobre el número de registros especificado en el parámetro `limit`. La respuesta incluye un registro de incumplimiento de umbral si se alcanzó el límite de 10 MB.

  El rango de `limit` es de 1 a 100 000, con un valor predeterminado de 10.
- opNum (en la CLI: `--op-num`): un valor Long, del tipo: `long` (un valor entero firmado de 64 bits).

  El número secuencial de la operación dentro de la confirmación especificada desde la que empezar a leer en los datos del flujo de registro de cambios. El valor predeterminado es 1.

Respuesta

- format: obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Formato de serialización de los registros de cambios que se devuelven. Actualmente el único valor admitido es `PG_JSON`.
- lastEventId: obligatorio: es una matriz de mapeo de pares de clave-valor donde:

    Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

    Cada valor es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Identificador de secuencia del último cambio en la respuesta de la transmisión.

  Un ID de evento se compone de dos campos: `commitNum`, que identifica una transacción que ha cambiado el gráfico y `opNum`, que identifica una operación específica dentro de dicha transacción:

  Example

```
"eventId": {
```

```
            "commitNum": 12,
            "opNum": 1
        }
```

- lastTrxTimestampInMillis: obligatorio: un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  La hora a la que se solicitó la confirmación de la transacción, en milisegundos a partir de la fecha de inicio de Unix.
- records (obligatorio): una matriz de objetos PropertygraphRecord.

  Una matriz de registros de flujos de registro de cambios serializados incluidos en la respuesta.
- totalRecords: obligatorio: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número total de registros de la respuesta.

Errores

- UnsupportedOperationException
- ExpiredStreamException
- InvalidParameterException
- MemoryLimitExceededException
- StreamRecordsNotFoundException
- ClientTimeoutException
- PreconditionsFailedException
- ThrottlingException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

Estructuras de datos de transmisiones:

PropertygraphRecord (estructura)

Estructura de un registro de gráfico de propiedades.

Campos

- commitTimestampInMillis: esto es obligatorio: un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  La hora a la que se solicitó la confirmación de la transacción, en milisegundos a partir de la fecha de inicio de Unix.
- data: esto es obligatorio: un objeto PropertygraphData.

  El registro de cambios serializados de Gremlin u openCypher.
- eventId: esto es obligatorio: es una matriz de mapeo de pares de clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Cada valor es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador de secuencia del registro de cambios de la transmisión.
- isLastOp: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Solo está presente si esta operación es la última de su transacción. Si está presente, se establece en True. Es útil para garantizar que se realice una transacción completa.
- op: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La operación que creó el cambio.

PropertygraphData (estructura)

Un registro de cambios de Gremlin u openCypher.

Campos

- from: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si se trata de un borde (tipo = `e`), el identificador del vértice `from` o nodo de origen correspondientes.
- id: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID del elemento de Gremlin u openCypher.
- key: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Nombre de la propiedad. Para las etiquetas de elementos, se trata de `label`.

- to: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Si se trata de un borde (tipo = `e`), el identificador del vértice `to` o nodo de destino correspondientes.

- type: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de este elemento de Gremlin u openCypher. Debe ser uno de los siguientes:

  - **v1**: etiqueta de vértice para Gremlin o etiqueta de nodo para openCypher.
  - **vp**: propiedades de vértice para Gremlin o propiedades de nodo para openCypher.
  - **e**: borde y etiqueta de borde para Gremlin, o relación y tipo de relación para openCypher.
  - **ep**: propiedades de borde para Gremlin o propiedades de relación para openCypher.

- value: esto es obligatorio: un documento, del tipo: `document` (un contenido abierto independiente del protocolo representado por un modelo de datos similar a JSON).

  Se trata de un objeto JSON que contiene un campo value para el propio valor y un campo datatype para el tipo de datos JSON de ese valor.

  Example

```
"value": {
          "value": "(the new value"),
          "dataType": "(the JSON datatypenew value")
        }
```

API de resumen de gráficos y estadísticas del plano de datos de Neptune

Acciones estadísticas de gráficos de propiedades:

- GetPropertygraphStatistics (acción)
- ManagePropertygraphStatistics (acción)
- DeletePropertygraphStatistics (acción)
- GetPropertygraphSummary (acción)

Estructuras de estadísticas:

- Statistics (estructura)
- StatisticsSummary (estructura)
- DeleteStatisticsValueMap (estructura)
- RefreshStatisticsIdMap (estructura)
- NodeStructure (estructura)
- EdgeStructure (estructura)
- SubjectStructure (estructura)
- PropertygraphSummaryValueMap (estructura)
- PropertygraphSummary (estructura)

GetPropertygraphStatistics (acción)

El nombre de la AWS CLI para esta API es: `get-propertygraph-statistics`.

Obtiene las estadísticas de los gráficos de propiedades (Gremlin y openCypher).

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetStatisticsStatus en dicho clúster.

Solicitud

- Ningún parámetro de solicitud.

Respuesta

- payload: obligatorio: objeto Estadísticas.

  Estadísticas de los datos de gráficos de propiedades.
- status: obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de devolución HTTP de la solicitud. Si la solicitud se ha realizado correctamente, el código es 200. Consulte Códigos de error comunes para la solicitud de estadísticas del DFE para obtener una lista de los errores más comunes.

Errores

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ManagePropertygraphStatistics (acción)

El nombre de la AWS CLI para esta API es: `manage-propertygraph-statistics`.

Administra la generación y el uso de estadísticas de gráficos de propiedades.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ManageStatistics en dicho clúster.

Solicitud

- mode (en la CLI: `--mode`): un StatisticsAutoGenerationMode, del tipo: `string` (una cadena codificada con UTF-8).

  El modo de generación de estadísticas. Uno de los siguientes: `DISABLE_AUTOCOMPUTE`, `ENABLE_AUTOCOMPUTE` o `REFRESH`, el último de los cuales desencadena manualmente la generación de estadísticas del DFE.

Respuesta

- payload: objeto RefreshStatisticsIdMap.

  Esto solo se devuelve en el modo de actualización.

- status: obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de devolución HTTP de la solicitud. Si la solicitud se ha realizado correctamente, el código es 200.

Errores

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

DeletePropertygraphStatistics (acción)

El nombre de la AWS CLI para esta API es: `delete-propertygraph-statistics`.

Elimina las estadísticas de los datos (gráficos de propiedades) de Gremlin y openCypher.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:DeleteStatistics en dicho clúster.

Solicitud

- Ningún parámetro de solicitud.

Respuesta

- payload: objeto DeleteStatisticsValueMap.

  La carga de eliminación.
- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de cancelación.
- statusCode: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El código de respuesta HTTP: 200 si la eliminación se realizó correctamente o 204 si no había estadísticas que eliminar.

Errores

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetPropertygraphSummary (acción)

El nombre de la AWS CLI para esta API es: `get-propertygraph-summary`.

Obtiene un resumen de gráficos de un gráfico de propiedades.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetGraphSummary en dicho clúster.

Solicitud

- mode (en la CLI: `--mode`): un GraphSummaryType, del tipo: `string` (una cadena codificada con UTF-8).

  El modo puede tomar uno de estos dos valores: `BASIC` (el valor predeterminado) y `DETAILED`.

Respuesta

- payload: objeto PropertygraphSummaryValueMap.

  Carga que incluye la respuesta de resumen de gráficos de propiedades.
- statusCode: un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El código de devolución HTTP de la solicitud. Si la solicitud se ha realizado correctamente, el código es 200.

Errores

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)
- UnsupportedOperationException
- PreconditionsFailedException

- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Estructuras de estadísticas:

Statistics (estructura)

Incluye información de estadísticas. El motor DFE utiliza la información sobre los datos de gráficos de Neptune para realizar compensaciones efectivas a la hora de planificar la ejecución de las consultas. Esta información adopta la forma de estadísticas que incluyen los denominados conjuntos de características y estadísticas de predicados que pueden guiar la planificación de las consultas. Consulte Administración de las estadísticas para que las utilice el DFE de Neptune.

Campos

- active: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la generación de estadísticas del DFE está habilitada o no.
- autoCompute: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Indica si la generación automática de estadísticas está habilitada o no.
- date: se trata de un valor SyntheticTimestamp_date_time, del tipo: `string` (una cadena codificada en UTF-8).

  La hora UTC a la que se generaron las estadísticas del DFE por última vez.
- note: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una nota sobre los problemas en el caso de que las estadísticas no sean válidas.
- signatureInfo: se trata de un objeto StatisticsSummary.

  Una estructura StatisticsSummary que incluye:
  - `signatureCount`: el número total de firmas en todos los conjuntos de características.
  - `instanceCount`: el número total de instancias del conjunto de características.
  - `predicateCount`: el número total de predicados únicos.
- statisticsId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Indica el ID de la ejecución de generación de estadísticas actual. Un valor de -1 indica que no se ha generado ninguna estadística.

StatisticsSummary (estructura)

Información sobre los conjuntos de características generados en las estadísticas.

Campos

- instanceCount: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número total de instancias del conjunto de características.
- predicateCount: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número total de predicados únicos.
- signatureCount: se trata de un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número total de firmas en todos los conjuntos de características.

DeleteStatisticsValueMap (estructura)

La carga de DeleteStatistics.

Campos

- active: se trata de un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  El estado actual de las estadísticas.
- statisticsId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la ejecución de generación de estadísticas que se está produciendo actualmente.

RefreshStatisticsIdMap (estructura)

Estadísticas del modo `REFRESH`.

Campos

- statisticsId: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El ID de la ejecución de generación de estadísticas que se está produciendo actualmente.

NodeStructure (estructura)

Una estructura de nodos.

Campos

- count: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  Número de nodos que tienen esta estructura específica.
- distinctOutgoingEdgeLabels: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de las etiquetas de borde saliente distintas presentes en esta estructura específica.
- nodeProperties: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de propiedades de los nodos presentes en esta estructura específica.

EdgeStructure (estructura)

Una estructura de borde.

Campos

- count: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número de bordes que tiene esta estructura específica.
- edgeProperties: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de propiedades de los bordes presentes en esta estructura específica.

SubjectStructure (estructura)

Una estructura de tema.

Campos

- count: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

Número de apariciones de esta estructura específica.

- predicates: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de predicados presentes en esta estructura específica.

PropertygraphSummaryValueMap (estructura)

Carga de la respuesta de resumen de gráficos de propiedades.

Campos

- graphSummary: se trata de un objeto PropertygraphSummary.

  El resumen del gráfico.
- lastStatisticsComputationTime: se trata de un valor SyntheticTimestamp_date_time, del tipo: `string` (una cadena codificada en UTF-8).

  La marca temporal, en formato ISO 8601, de la hora en que Neptune calculó las estadísticas por última vez.
- version: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La versión de esta respuesta del resumen de gráficos.

PropertygraphSummary (estructura)

La API de resumen de gráficos devuelve una lista de solo lectura de etiquetas de nodos y bordes y claves de propiedades, junto con el recuento de nodos, bordes y propiedades. Consulte Respuesta del resumen de gráficos para un gráfico de propiedades (PG).

Campos

- edgeLabels: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de etiquetas de bordes distintos del gráfico.
- edgeProperties: se trata de objetos LongValuedMap. Se trata de una matriz de mapeo de pares clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Cada valor es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

Una lista de las propiedades de bordes distintos del gráfico, junto con el recuento de bordes en las que se utiliza cada propiedad.

- edgeStructures: se trata de una matriz de objetos EdgeStructure.

  Este campo solo está presente cuando el modo solicitado es `DETAILED`. Incluye una lista de estructuras de bordes.

- nodeLabels: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una lista de etiquetas de nodos distintas del gráfico.

- nodeProperties: se trata de objetos LongValuedMap. Se trata de una matriz de mapeo de pares clave-valor donde:

  Cada clave es una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Cada valor es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número de propiedades de nodos distintas del gráfico.

- nodeStructures: se trata de una matriz de objetos NodeStructure.

  Este campo solo está presente cuando el modo solicitado es `DETAILED`. Incluye una lista de estructuras de nodos.

- numEdgeLabels: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número de etiquetas de bordes distintos del gráfico.

- numEdgeProperties: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número de propiedades de bordes distintos del gráfico.

- numEdges: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número de bordes del gráfico.

- numNodeLabels: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número de etiquetas de nodos distintas del gráfico.

- numNodeProperties: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

Una lista de las propiedades de nodos distintas del gráfico, junto con el recuento de nodos en los que se utiliza cada propiedad.

- numNodes: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número de nodos del gráfico.

- totalEdgePropertyValues: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número total de usos de todas las propiedades de bordes.

- totalNodePropertyValues: esto es un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El número total de usos de todas las propiedades de nodos.

API de procesamiento de datos Neptune ML

Acciones de procesamiento de datos:

- StartMLDataProcessingJob (acción)
- ListMLDataProcessingJobs (acción)
- GetMLDataProcessingJob (acción)
- CancelMLDataProcessingJob (acción)

Estructuras de uso general de ML:

- MlResourceDefinition (estructura)
- MlConfigDefinition (estructura)

StartMLDataProcessingJob (acción)

El nombre de la AWS CLI para esta API es: `start-ml-data-processing-job`.

Crea un nuevo trabajo de procesamiento de datos de Neptune ML para procesar los datos de gráficos exportados desde Neptune para el entrenamiento. Consulte El comando `dataprocessing`.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:StartMLModelDataProcessingJob en dicho clúster.

Solicitud

- configFileName (en la CLI: `--config-file-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un archivo de especificación de datos que describe cómo cargar los datos de gráficos exportados para el entrenamiento. El kit de herramientas de exportación de Neptune genera automáticamente el archivo. El valor predeterminado es `training-data-configuration.json`.
- id (en la CLI: `--id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador único para el nuevo trabajo. El valor predeterminado es un UUID generado automáticamente.
- inputDataS3Location (en la CLI: `--input-data-s3-location`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El URI de la ubicación de Amazon S3 en la que desea que SageMaker descargue los datos necesarios para ejecutar el trabajo de procesamiento de datos.
- modelType (en la CLI: `--model-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Uno de los dos tipos de modelos que Neptune ML admite actualmente: modelos de subgráficos heterogéneos (`heterogeneous`) y gráficos de conocimientos (`kge`). El valor predeterminado es Ninguno. Si no se especifica, Neptune ML elige automáticamente el modelo en función de los datos.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de recurso de Amazon (ARN) de un rol de IAM que SageMaker puede asumir para realizar tareas en su nombre. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.
- previousDataProcessingJobId (en la CLI: `--previous-data-processing-job-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de un trabajo de procesamiento de datos completado que se ejecuta en una versión anterior de los datos.
- processedDataS3Location (en la CLI: `--processed-data-s3-location`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El URI de la ubicación de Amazon S3 en la que desea que SageMaker guarde los resultados de un trabajo de procesamiento de datos.

- processingInstanceType (en la CLI: `--processing-instance-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de instancia de ML que se utiliza durante el procesamiento de datos. Su memoria debe ser lo suficientemente grande como para incluir el conjunto de datos procesado. El valor predeterminado es el tipo ml.r5 de menor tamaño cuya memoria es diez veces mayor que el tamaño de los datos de gráficos exportados en el disco.

- processingInstanceVolumeSizeInGB (en la CLI: `--processing-instance-volume-size-in-gb`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El tamaño del volumen del disco de la instancia de procesamiento. Tanto los datos de entrada como los datos procesados se almacenan en el disco, por lo que el tamaño del volumen debe ser lo suficientemente grande como para incluir ambos conjuntos de datos. El valor predeterminado es 0. Si no se especifica o el valor es 0, Neptune ML elige el tamaño del volumen automáticamente en función del tamaño de los datos.

- processingTimeOutInSeconds (en la CLI: `--processing-time-out-in-seconds`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Tiempo de espera en segundos para el trabajo de procesamiento de datos. El valor predeterminado es 86 400 (1 día).

- s3OutputEncryptionKMSKey (en la CLI: `--s-3-output-encryption-kms-key`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de Amazon Key Management Service (Amazon KMS) que SageMaker utiliza para cifrar la salida del trabajo de procesamiento. El valor predeterminado es Ninguno.

- sagemakerIamRoleArn (en la CLI: `--sagemaker-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM para la ejecución de SageMaker. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- securityGroupIds (en la CLI: `--security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los ID del grupo de seguridad de la VPC. El valor predeterminado es Ninguno.

- subnets (en la CLI: `--subnets`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los ID de las subredes en la VPC de Neptune. El valor predeterminado es Ninguno.

- volumeEncryptionKMSKey (en la CLI: `--volume-encryption-kms-key`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de Amazon Key Management Service (Amazon KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan el trabajo de entrenamiento. El valor predeterminado es Ninguno.

Respuesta

- arn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del trabajo de procesamiento de datos.

- creationTimeInMillis: un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  El tiempo que se tardó en crear el nuevo trabajo de procesamiento, en milisegundos.

- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único del nuevo trabajo de procesamiento de datos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException (Excepción de demasiadas solicitudes)

ListMLDataProcessingJobs (acción)

El nombre de la AWS CLI para esta API es: `list-ml-data-processing-jobs`.

Devuelve una lista de los trabajos de procesamiento de datos de Neptune ML. Consulte Enumeración de trabajos de procesamiento de datos activos mediante el comando dataprocessing de Neptune ML.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ListMLDataProcessingJobs en dicho clúster.

Solicitud

- maxItems (en la CLI: `--max-items`): un ListMLDataProcessingJobsInputMaxItemsInteger, del tipo: `integer` (un valor entero firmado de 32 bits), no inferior a 1 ni superior a 1024 ?st?s.

  El número máximo de elementos que se recuperan (de 1 a 1024; el valor predeterminado es 10).
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- ids: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una página con los identificadores de los trabajos de procesamiento de datos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException

- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

GetMLDataProcessingJob (acción)

El nombre de la AWS CLI para esta API es: `get-ml-data-processing-job`.

Recupera información sobre un trabajo de procesamiento de datos específico. Consulte El comando `dataprocessing`.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:neptune-db:GetMLDataProcessingJobStatus en dicho clúster.

Solicitud

- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del trabajo de procesamiento de datos que se va a recuperar.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único de este trabajo de procesamiento de datos.

- processingJob: objeto MlResourceDefinition.

  Definición del trabajo de procesamiento de datos.

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Estado del trabajo de procesamiento de datos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

CancelMLDataProcessingJob (acción)

> El nombre de la AWS CLI para esta API es: `cancel-ml-data-processing-job`.

Cancela un trabajo de procesamiento de datos de Neptune ML. Consulte El comando `dataprocessing`.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:CancelMLDataProcessingJob en dicho clúster.

Solicitud

- clean (en la CLI: `--clean`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

Si se establece en `TRUE`, este indicador especifica que todos los artefactos de S3 de Neptune ML deben eliminarse cuando se detenga el trabajo. El valor predeterminado es `FALSE`.

- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del trabajo de procesamiento de datos.

- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la solicitud de cancelación.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

Estructuras de uso general de ML:

MlResourceDefinition (estructura)

Define un recurso de Neptune ML.

Campos

- arn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del recurso.
- cloudwatchLogUrl: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La URL del registro de CloudWatch del recurso.
- failureReason: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El motivo del error, en caso de que se produzca un error.
- name: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del recurso.
- outputLocation: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La ubicación de salida.
- status: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado del recurso.

MlConfigDefinition (estructura)

Incluye una configuración de Neptune ML.

Campos

- arn: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de la configuración.
- name: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de la configuración.

API de entrenamiento de modelos de Neptune ML

Acciones de entrenamiento de modelos:

- StartMLModelTrainingJob (acción)
- ListMLModelTrainingJobs (acción)
- GetMLModelTrainingJob (acción)
- CancelMLModelTrainingJob (acción)

Estructuras de entrenamiento de modelos:

- CustomModelTrainingParameters (estructura)

StartMLModelTrainingJob (acción)

El nombre de la AWS CLI para esta API es: `start-ml-model-training-job`.

Crea un trabajo de entrenamiento de modelos de Neptune ML. Consulte Entrenamiento de modelos con el comando `modeltraining`.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:StartMLModelTrainingJob en dicho clúster.

Solicitud

- baseProcessingInstanceType (en la CLI: `--base-processing-instance-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de instancia de ML que se utiliza para preparar y administrar el entrenamiento de modelos de ML. Se trata de una instancia de CPU que se elige en función de los requisitos de memoria para procesar los datos y el modelo de entrenamiento.
- customModelTrainingParameters (en la CLI: `--custom-model-training-parameters`): un objeto CustomModelTrainingParameters.

  La configuración para el entrenamiento de modelos personalizados. Este es un objeto de JSON.
- dataProcessingJobId (en la CLI: `--data-processing-job-id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El ID del trabajo de procesamiento de datos completado que ha creado los datos con los que se trabajará en el entrenamiento.

- enableManagedSpotTraining (en la CLI: `--enable-managed-spot-training`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Optimiza el costo de entrenar modelos de machine learning mediante el uso de instancias de spot de Amazon Elastic Compute Cloud. El valor predeterminado es `False`.

- id (en la CLI: `--id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador único para el nuevo trabajo. El valor predeterminado es un UUID generado automáticamente.

- maxHPONumberOfTrainingJobs (en la CLI: `--max-hpo-number-of-training-jobs`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Número total máximo de trabajos de entrenamiento que se deben iniciar para el trabajo de ajuste de hiperparámetros. El valor predeterminado es 2. Neptune ML ajusta automáticamente los hiperparámetros del modelo de machine learning. Para obtener un modelo que funcione bien, utilice al menos 10 trabajos (dicho de otro modo, establezca `maxHPONumberOfTrainingJobs` en 10). Por lo general, cuantos más ajustes se realicen, mejores serán los resultados.

- maxHPOParallelTrainingJobs (en la CLI: `--max-hpo-parallel-training-jobs`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Número máximo de trabajos de entrenamiento en paralelo que se deben iniciar para el trabajo de ajuste de hiperparámetros. El valor predeterminado es 2. El número de trabajos en paralelo que puede ejecutar está limitado por los recursos disponibles en la instancia de entrenamiento.

- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- previousModelTrainingJobId (en la CLI: `--previous-model-training-job-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID del trabajo de entrenamiento de modelos completado que desee actualizar de forma incremental en función de los datos actualizados.

- s3OutputEncryptionKMSKey (en la CLI: `--s-3-output-encryption-kms-key`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de Amazon Key Management Service (KMS) que SageMaker utiliza para cifrar la salida del trabajo de procesamiento. El valor predeterminado es Ninguno.
- sagemakerIamRoleArn (en la CLI: `--sagemaker-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM para la ejecución de SageMaker. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.
- securityGroupIds (en la CLI: `--security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los ID del grupo de seguridad de la VPC. El valor predeterminado es Ninguno.
- subnets (en la CLI: `--subnets`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los ID de las subredes en la VPC de Neptune. El valor predeterminado es Ninguno.
- trainingInstanceType (en la CLI: `--training-instance-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de instancia de ML que se utiliza para el entrenamiento de modelos. Todos los modelos de Neptune ML admiten el entrenamiento con CPU, GPU y varias GPU. El valor predeterminado es `ml.p3.2xlarge`. La elección del tipo de instancia adecuado para el entrenamiento depende del tipo de tarea, del tamaño del gráfico y del presupuesto.
- trainingInstanceVolumeSizeInGB (en la CLI: `--training-instance-volume-size-in-gb`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El tamaño del volumen del disco de la instancia de entrenamiento. Tanto los datos de entrada como los datos de salida se almacenan en el disco, por lo que el tamaño del volumen debe ser lo suficientemente grande como para incluir ambos conjuntos de datos. El valor predeterminado es 0. Si no se especifica o el valor es 0, Neptune ML selecciona un tamaño de volumen de disco en función de la recomendación generada en el paso de procesamiento de datos.
- trainingTimeOutInSeconds (en la CLI: `--training-time-out-in-seconds`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  Tiempo de espera en segundos para el trabajo de entrenamiento. El valor predeterminado es 86 400 (1 día).

- trainModelS3Location (en la CLI: `--train-model-s3-location`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La ubicación en Amazon S3 donde se van a almacenar los artefactos del modelo.

- volumeEncryptionKMSKey (en la CLI: `--volume-encryption-kms-key`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de Amazon Key Management Service (KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan el trabajo de entrenamiento. El valor predeterminado es Ninguno.

Respuesta

- arn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del nuevo trabajo de entrenamiento de modelos.

- creationTimeInMillis: un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  La hora de creación del trabajo de entrenamiento de modelos, en milisegundos.

- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único del nuevo trabajo de entrenamiento de modelos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException (Excepción de demasiadas solicitudes)

ListMLModelTrainingJobs (acción)

El nombre de la AWS CLI para esta API es: `list-ml-model-training-jobs`.

Enumera los trabajos de entrenamiento de modelos de Neptune ML. Consulte Entrenamiento de modelos con el comando `modeltraining`.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:neptune-db:ListMLModelTrainingJobs en dicho clúster.

Solicitud

- maxItems (en la CLI: `--max-items`): un ListMLModelTrainingJobsInputMaxItemsInteger, del tipo: `integer` (un valor entero firmado de 32 bits), no inferior a 1 ni superior a 1024 ?st?s.

  El número máximo de elementos que se recuperan (de 1 a 1024; el valor predeterminado es 10).
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- ids: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una página de la lista de ID de trabajos de entrenamiento de modelos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException

- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

GetMLModelTrainingJob (acción)

El nombre de la AWS CLI para esta API es: `get-ml-model-training-job`.

Recupera información sobre un trabajo de entrenamiento de modelos de Neptune ML. Consulte Entrenamiento de modelos con el comando `modeltraining`.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetMLModelTrainingJobStatus en dicho clúster.

Solicitud

- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del trabajo de entrenamiento de modelos que se va a recuperar.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- hpoJob: objeto MlResourceDefinition.

  El trabajo de HPO.
- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El identificador único de este trabajo de entrenamiento de modelos.

- mlModels: matriz de objetos MlConfigDefinition.

  Una lista de las configuraciones de los modelos de ML que se utilizan.

- modelTransformJob: objeto MlResourceDefinition.

  El trabajo de transformación de modelos.

- processingJob: objeto MlResourceDefinition.

  El trabajo de procesamiento de datos.

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado del trabajo de entrenamiento de modelos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

CancelMLModelTrainingJob (acción)

El nombre de la AWS CLI para esta API es: `cancel-ml-model-training-job`.

Cancela un trabajo de entrenamiento de modelos de Neptune ML. Consulte Entrenamiento de modelos con el comando `modeltraining`.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:CancelMLModelTrainingJob en dicho clúster.

Solicitud

- clean (en la CLI: `--clean`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `TRUE`, este indicador especifica que todos los artefactos de Amazon S3 deben eliminarse cuando se detenga el trabajo. El valor predeterminado es `FALSE`.
- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del trabajo de entrenamiento de modelos que se va a cancelar.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la cancelación.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

Estructuras de entrenamiento de modelos:

CustomModelTrainingParameters (estructura)

Incluye parámetros de entrenamiento de modelos personalizados. Consulte Modelos personalizados de Neptune ML.

Campos

- sourceS3DirectoryPath: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La ruta a la ubicación de Amazon S3 donde se encuentra el módulo de Python que implementa el modelo. Debe apuntar a una ubicación válida de Amazon S3 existente que incluya, como mínimo, un script de entrenamiento, un script de transformación y un archivo `model-hpo-configuration.json`.
- trainingEntryPointScript: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del punto de entrada en el módulo de un script que realiza el entrenamiento de modelos y utiliza los hiperparámetros como argumentos de la línea de comandos, incluidos los hiperparámetros fijos. El valor predeterminado es `training.py`.
- transformEntryPointScript: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del punto de entrada en el módulo de un script que debe ejecutarse después de identificar el mejor modelo de la búsqueda de hiperparámetros, con el fin de calcular los artefactos de modelos necesarios para su implementación. Debería poder ejecutarse sin argumentos de línea de comandos. El valor predeterminado es `transform.py`.

API de transformación de modelos de Neptune ML

Acciones de transformación de modelos:

- StartMLModelTransformJob (acción)

- ListMLModelTransformJobs (acción)
- GetMLModelTransformJob (acción)
- CancelMLModelTransformJob (acción)

Estructuras de transformación de modelos:

- CustomModelTransformParameters (estructura)

StartMLModelTransformJob (acción)

El nombre de la AWS CLI para esta API es: `start-ml-model-transform-job`.

Crea un nuevo trabajo de transformación de modelos. Consulte Uso de un modelo entrenado para generar nuevos artefactos de modelo.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:StartMLModelTransformJob en dicho clúster.

Solicitud

- baseProcessingInstanceType (en la CLI: `--base-processing-instance-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de instancia de ML que se utiliza para preparar y administrar el entrenamiento de modelos de ML. Se trata de una instancia informática de ML que se elige en función de los requisitos de memoria para procesar los datos y el modelo de entrenamiento.
- baseProcessingInstanceVolumeSizeInGB (en la CLI: `--base-processing-instance-volume-size-in-gb`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El tamaño del volumen del disco de la instancia de entrenamiento en gigabytes. El valor predeterminado es 0. Tanto los datos de entrada como los datos de salida se almacenan en el disco, por lo que el tamaño del volumen debe ser lo suficientemente grande como para incluir ambos conjuntos de datos. Si no se especifica o el valor es 0, Neptune ML selecciona un tamaño de volumen de disco en función de la recomendación generada en el paso de procesamiento de datos.
- customModelTransformParameters (en la CLI: `--custom-model-transform-parameters`): un objeto CustomModelTransformParameters.

Información de configuración para la transformación de un modelo mediante un modelo personalizado. El objeto `customModelTransformParameters` incluye los siguientes campos, que deben tener valores compatibles con los parámetros del modelo guardados del trabajo de entrenamiento:

- dataProcessingJobId (en la CLI: `--data-processing-job-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID del trabajo de un trabajo de procesamiento de datos completado. Debe incluir un `dataProcessingJobId` y un `mlModelTrainingJobId`, o un `trainingJobName`.

- id (en la CLI: `--id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador único para el nuevo trabajo. El valor predeterminado es un UUID generado automáticamente.

- mlModelTrainingJobId (en la CLI: `--ml-model-training-job-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de trabajo de un trabajo de entrenamiento de modelos completado. Debe incluir un `dataProcessingJobId` y un `mlModelTrainingJobId`, o un `trainingJobName`.

- modelTransformOutputS3Location (en la CLI: `--model-transform-output-s3-location`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La ubicación en Amazon S3 donde se van a almacenar los artefactos del modelo.

- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- s3OutputEncryptionKMSKey (en la CLI: `--s-3-output-encryption-kms-key`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de Amazon Key Management Service (KMS) que SageMaker utiliza para cifrar la salida del trabajo de procesamiento. El valor predeterminado es Ninguno.

- sagemakerIamRoleArn (en la CLI: `--sagemaker-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El ARN de un rol de IAM para la ejecución de SageMaker. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- securityGroupIds (en la CLI: `--security-group-ids`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los ID del grupo de seguridad de la VPC. El valor predeterminado es Ninguno.

- subnets (en la CLI: `--subnets`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Los ID de las subredes en la VPC de Neptune. El valor predeterminado es Ninguno.

- trainingJobName (en la CLI: `--training-job-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre de un trabajo de entrenamiento completado de SageMaker. Debe incluir un `dataProcessingJobId` y un `mlModelTrainingJobId`, o un `trainingJobName`.

- volumeEncryptionKMSKey (en la CLI: `--volume-encryption-kms-key`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de Amazon Key Management Service (KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan el trabajo de entrenamiento. El valor predeterminado es Ninguno.

Respuesta

- arn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del trabajo de transformación de modelos.

- creationTimeInMillis: un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  La hora de creación del trabajo de transformación de modelos, en milisegundos.

- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único del nuevo trabajo de transformación de modelos.

Errores

- UnsupportedOperationException

- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

ListMLModelTransformJobs (acción)

El nombre de la AWS CLI para esta API es: `list-ml-model-transform-jobs`.

Devuelve una lista de los ID de los trabajos de transformación de modelos. Consulte Uso de un modelo entrenado para generar nuevos artefactos de modelo.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ListMLModelTransformJobs en dicho clúster.

Solicitud

- maxItems (en la CLI: `--max-items`): un ListMLModelTransformJobsInputMaxItemsInteger, del tipo: `integer` (un valor entero firmado de 32 bits), no inferior a 1 ni superior a 1024 ?st?s.

  El número máximo de elementos que se recuperan (de 1 a 1024; el valor predeterminado es 10).
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- ids: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una página de la lista de ID de transformación de modelos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

GetMLModelTransformJob (acción)

El nombre de la AWS CLI para esta API es: `get-ml-model-transform-job`.

Obtiene información sobre un trabajo específico de transformación de modelos. Consulte Uso de un modelo entrenado para generar nuevos artefactos de modelo.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetMLModelTransformJobStatus en dicho clúster.

Solicitud

- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del trabajo de transformación de modelos que se va recuperar.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- baseProcessingJob: objeto MlResourceDefinition.

  El trabajo de procesamiento de datos base.
- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del trabajo de transformación de modelos que se va a recuperar.
- models: matriz de objetos MlConfigDefinition.

  Una lista de la información de configuración de los modelos que se utilizan.
- remoteModelTransformJob: objeto MlResourceDefinition.

  El trabajo remoto de transformación de modelos.
- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado del trabajo de transformación de modelos.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

CancelMLModelTransformJob (acción)

El nombre de la AWS CLI para esta API es: `cancel-ml-model-transform-job`.

Cancela un trabajo específico de transformación de modelo. Consulte Uso de un modelo entrenado para generar nuevos artefactos de modelo.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:CancelMLModelTransformJob en dicho clúster.

Solicitud

- clean (en la CLI: `--clean`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si este indicador está establecido en `TRUE`, todos los artefactos de S3 de Neptune ML deben eliminarse cuando se detenga el trabajo. El valor predeterminado es `FALSE`.
- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único del trabajo de transformación de modelos que se va a cancelar.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la cancelación.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException

- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

Estructuras de transformación de modelos:

CustomModelTransformParameters (estructura)

Incluye parámetros de transformación de modelos personalizados. Consulte Uso de un modelo entrenado para generar nuevos artefactos de modelo.

Campos

- sourceS3DirectoryPath: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La ruta a la ubicación de Amazon S3 donde se encuentra el módulo de Python que implementa el modelo. Debe apuntar a una ubicación válida de Amazon S3 existente que incluya, como mínimo, un script de entrenamiento, un script de transformación y un archivo `model-hpo-configuration.json`.
- transformEntryPointScript: se trata de una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El nombre del punto de entrada en el módulo de un script que debe ejecutarse después de identificar el mejor modelo de la búsqueda de hiperparámetros, con el fin de calcular los artefactos de modelos necesarios para su implementación. Debería poder ejecutarse sin argumentos de línea de comandos. El valor predeterminado es `transform.py`.

API de punto de conexión de inferencia de Neptune ML

Acciones del punto de conexión de inferencia:

- CreateMLEndpoint (acción)
- ListMLEndpoints (acción)
- GetMLEndpoint (acción)
- DeleteMLEndpoint (acción)

CreateMLEndpoint (acción)

El nombre de la AWS CLI para esta API es: `create-ml-endpoint`.

Crea un nuevo punto de conexión de inferencia de Neptune ML que le permite consultar un modelo específico que se creó durante el proceso de entrenamiento del modelo. Consulte Administración de los puntos de conexión de inferencia mediante el comando endpoints.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:CreateMLEndpoint en dicho clúster.

Solicitud

- id (en la CLI: `--id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un identificador único para el nuevo punto de conexión de inferencia. El valor predeterminado es un nombre con marca temporal generado automáticamente.
- instanceCount (en la CLI: `--instance-count`): un valor entero, del tipo: `integer` (un valor entero firmado de 32 bits).

  El número mínimo de instancias de Amazon EC2 que se deben implementar en un punto de conexión para la predicción. El valor predeterminado es 1.
- instanceType (en la CLI: `--instance-type`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El tipo de instancia de Neptune ML que se utiliza para el mantenimiento en línea. El valor predeterminado es `ml.m5.xlarge`. La selección de la instancia de ML para un punto de conexión de inferencia depende del tipo de tarea, del tamaño del gráfico y del presupuesto.
- mlModelTrainingJobId (en la CLI: `--ml-model-training-job-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El ID del trabajo de entrenamiento de modelos completado que ha creado el modelo al que apuntará el punto de conexión de inferencia. Debe proporcionar el valor `mlModelTrainingJobId` o el valor `mlModelTransformJobId`.

- mlModelTransformJobId (en la CLI: `--ml-model-transform-job-id`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de trabajo de un trabajo de transformación de modelos completado. Debe proporcionar el valor `mlModelTrainingJobId` o el valor `mlModelTransformJobId`.

- modelName (en la CLI: `--model-name`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Tipo de modelo para entrenamiento. De forma predeterminada, el modelo de Neptune ML se basa automáticamente en el `modelType` utilizado en el procesamiento de datos, pero aquí puede especificar otro tipo de modelo. El valor predeterminado es `rgcn` para gráficos heterogéneos y `kge` para gráficos de conocimientos. El único valor válido para gráficos heterogéneos es `rgcn`. Los valores válidos para los gráficos de conocimientos son: `kge`, `transe`, `distmult` y `rotate`.

- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

- update (en la CLI: `--update`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si se establece en `true`, `update` indica que se trata de una solicitud de actualización. El valor predeterminado es `false`. Debe proporcionar el valor `mlModelTrainingJobId` o el valor `mlModelTransformJobId`.

- volumeEncryptionKMSKey (en la CLI: `--volume-encryption-kms-key`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  La clave de Amazon Key Management Service (Amazon KMS) que SageMaker utiliza para cifrar los datos del volumen de almacenamiento asociado con las instancias de computación de ML que ejecutan el trabajo de entrenamiento. El valor predeterminado es Ninguno.

Respuesta

- arn: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN del nuevo punto de conexión de inferencia.
- creationTimeInMillis: un valor Long, del tipo: `long` (valor entero firmado de 64 bits).

  La hora de creación del punto de conexión, en milisegundos.
- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID único del nuevo punto de conexión de inferencia.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

ListMLEndpoints (acción)

El nombre de la AWS CLI para esta API es: `list-ml-endpoints`.

Enumera los puntos de conexión de inferencia existentes. Consulte Administración de los puntos de conexión de inferencia mediante el comando endpoints.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:ListMLEndpoints en dicho clúster.

Solicitud

- maxItems (en la CLI: `--max-items`): un ListMLEndpointsInputMaxItemsInteger, del tipo: `integer` (un valor entero firmado de 32 bits), no inferior a 1 ni superior a 1024 ?st?s.

  El número máximo de elementos que se recuperan (de 1 a 1024; el valor predeterminado es 10).

- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- ids: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Una página de la lista de ID de puntos de conexión de inferencia.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

GetMLEndpoint (acción)

El nombre de la AWS CLI para esta API es: `get-ml-endpoint`.

Recupera detalles sobre un punto de conexión de inferencia. Consulte Administración de los puntos de conexión de inferencia mediante el comando endpoints.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:GetMLEndpointStatus en dicho clúster.

Solicitud

- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del punto de conexión de inferencia.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- endpoint: objeto MlResourceDefinition.

  La definición del punto de conexión.
- endpointConfig: objeto MlConfigDefinition.

  La configuración del punto de conexión.
- id: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del punto de conexión de inferencia.
- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado del punto de conexión de inferencia.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException

- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

DeleteMLEndpoint (acción)

El nombre de la AWS CLI para esta API es: `delete-ml-endpoint`.

Cancela la creación de un punto de conexión de inferencia de Neptune ML. Consulte Administración de los puntos de conexión de inferencia mediante el comando endpoints.

Al invocar esta operación en un clúster de Neptune que tiene habilitada la autenticación de IAM, el usuario o rol de IAM que realiza la solicitud debe tener una política adjunta que permita la acción de IAM neptune-db:DeleteMLEndpoint en dicho clúster.

Solicitud

- clean (en la CLI: `--clean`): un booleano, del tipo: `boolean` (un valor booleano [true o false]).

  Si este indicador está establecido en `TRUE`, todos los artefactos de S3 de Neptune ML deben eliminarse cuando se detenga el trabajo. El valor predeterminado es `FALSE`.
- id (en la CLI: `--id`): obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El identificador único del punto de conexión de inferencia.
- neptuneIamRoleArn (en la CLI: `--neptune-iam-role-arn`): una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ARN de un rol de IAM que proporciona a Neptune acceso a los recursos de SageMaker y Amazon S3. Debe figurar en el grupo de parámetros del clúster de base de datos o se producirá un error.

Respuesta

- status: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El estado de la cancelación.

Errores

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException (Excepción de demasiadas solicitudes)

Excepciones de la API del plano de datos de Neptune

Excepciones:

- AccessDeniedException (estructura)
- BadRequestException (estructura)
- BulkLoadIdNotFoundException (estructura)
- CancelledByUserException (estructura)
- ClientTimeoutException (estructura)
- ConcurrentModificationException (estructura)
- ConstraintViolationException (estructura)
- ExpiredStreamException (estructura)
- FailureByQueryException (estructura)
- IllegalArgumentException (estructura)
- InternalFailureException (estructura)

- InvalidArgumentException (estructura)
- InvalidNumericDataException (estructura)
- InvalidParameterException (estructura)
- LoadUrlAccessDeniedException (estructura)
- MalformedQueryException (estructura)
- MemoryLimitExceededException (estructura)
- MethodNotAllowedException (estructura)
- MissingParameterException (estructura)
- MLResourceNotFoundException (estructura)
- ParsingException (estructura)
- PreconditionsFailedException (estructura)
- QueryLimitExceededException (estructura)
- QueryLimitException (estructura)
- QueryTooLargeException (estructura)
- ReadOnlyViolationException (estructura)
- S3Exception (estructura)
- ServerShutdownException (estructura)
- StatisticsNotAvailableException (estructura)
- StreamRecordsNotFoundException (estructura)
- ThrottlingException (estructura)
- TimeLimitExceededException (estructura)
- TooManyRequestsException (estructura)
- UnsupportedOperationException (estructura)
- UnloadUrlAccessDeniedException (estructura)

AccessDeniedException (estructura)

Se genera en caso de un error de autenticación o autorización.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El código de estado HTTP devuelto con la excepción.

- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

BadRequestException (estructura)

Se genera cuando se envía una solicitud que no se puede procesar.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud incorrecta.

BulkLoadIdNotFoundException (estructura)

Se genera cuando no se encuentra un ID específico de trabajo de carga masiva.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un mensaje detallado que describe el problema.

- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID del trabajo de carga masiva que no se pudo encontrar.

CancelledByUserException (estructura)

Se genera cuando un usuario cancela una solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

ClientTimeoutException (estructura)

Se genera cuando se agota el tiempo de espera de una solicitud en el cliente.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

ConcurrentModificationException (estructura)

Se genera cuando una solicitud intenta modificar datos que otro proceso está modificando simultáneamente.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

ConstraintViolationException (estructura)

Se genera cuando un valor de un campo de solicitud no cumplía con las restricciones establecidas.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

ExpiredStreamException (estructura)

Se genera cuando una solicitud intenta acceder a una transmisión que ha caducado.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

FailureByQueryException (estructura)

Se genera cuando se produce un error en una solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

IllegalArgumentException (estructura)

Se genera cuando no se admite un argumento en una solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.

- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

InternalFailureException (estructura)

Se genera cuando se produce un error inesperado en el procesamiento de la solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

InvalidArgumentException (estructura)

Se genera cuando un argumento de una solicitud tiene un valor no válido.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El ID de la solicitud en cuestión.

InvalidNumericDataException (estructura)

Se genera cuando se encuentran datos numéricos no válidos al atender una solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

InvalidParameterException (estructura)

Se genera cuando el valor de un parámetro no es válido.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud que incluye un parámetro no válido.

LoadUrlAccessDeniedException (estructura)

Se genera cuando se deniega el acceso a una URL de carga específica.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

MalformedQueryException (estructura)

Se genera cuando se envía una consulta que es sintácticamente incorrecta o no pasa una validación adicional.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud de consulta con un formato incorrecto.

MemoryLimitExceededException (estructura)

Se genera cuando se produce un error en una solicitud por falta de recursos de memoria. Se puede volver a intentar la solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud que devolvió un error.

MethodNotAllowedException (estructura)

Se genera cuando el método HTTP utilizado por una solicitud no es compatible con el punto de conexión que se utiliza.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

MissingParameterException (estructura)

Se genera cuando falta un parámetro obligatorio.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El código de estado HTTP devuelto con la excepción.

- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en la que falta el parámetro.

MLResourceNotFoundException (estructura)

Se genera cuando no se ha podido encontrar un recurso específico de machine learning.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

ParsingException (estructura)

Se genera cuando se encuentra un problema de análisis.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

Un mensaje detallado que describe el problema.

- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

PreconditionsFailedException (estructura)

Se genera cuando no se cumple una condición previa para procesar una solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

QueryLimitExceededException (estructura)

Se genera cuando el número de consultas activas supera lo que el servidor puede procesar. La consulta en cuestión se puede volver a intentar cuando el sistema esté menos ocupado.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

El ID de la solicitud que ha superado el límite.

QueryLimitException (estructura)

Se genera cuando el tamaño de una consulta supera el límite del sistema.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud que ha superado el límite.

QueryTooLargeException (estructura)

Se genera cuando el cuerpo de una consulta es demasiado grande.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud que es demasiado grande.

ReadOnlyViolationException (estructura)

Se genera cuando una solicitud intenta escribir en un recurso de solo lectura.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en la que falta el parámetro.

S3Exception (estructura)

Se genera cuando hay un problema para acceder a Amazon S3.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

ServerShutdownException (estructura)

Se genera cuando el servidor se apaga mientras se procesa una solicitud.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

StatisticsNotAvailableException (estructura)

Se genera cuando las estadísticas necesarias para hacer frente a una solicitud no están disponibles.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

StreamRecordsNotFoundException (estructura)

Se genera cuando no se encuentran los registros de transmisión solicitados por una consulta.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.

- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

ThrottlingException (estructura)

Se genera cuando la tasa de solicitudes supera el rendimiento máximo. Las solicitudes se pueden volver a intentar después de que se produzca esta excepción.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud que no se ha podido procesar por este motivo.

TimeLimitExceededException (estructura)

Se genera cuando una operación supera el límite de tiempo permitido para ella.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.

- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud que no se ha podido procesar por este motivo.

TooManyRequestsException (estructura)

Se genera cuando el número de solicitudes que se están procesando supera el límite.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud que no se ha podido procesar por este motivo.

UnsupportedOperationException (estructura)

Se genera cuando una solicitud intenta iniciar una operación que no es compatible.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

UnloadUrlAccessDeniedException (estructura)

Se genera cuando se deniega el acceso a una URL que es un destino de descarga.

Campos

- code: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El código de estado HTTP devuelto con la excepción.
- detailedMessage: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  Un mensaje detallado que describe el problema.
- requestId: esto es obligatorio: una cadena, del tipo: `string` (una cadena codificada con UTF-8).

  El ID de la solicitud en cuestión.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.