aws

使用者指南

Amazon Neptune

Amazon Neptune: 使用者指南

Table of Contents

什麼是 Amazon Neptune？

Amazon Neptune 是快速、可靠、全受管的圖形資料庫服務，可讓您輕鬆建置和執行搭配高度連線資料集使用的應用程式。Neptune 的核心是專門打造的高效能圖形資料庫引擎。此引擎的專屬設計可儲存數十億筆關係，且查詢圖形時只會有數毫秒的延遲。Neptune 支援熱門的屬性圖查詢語言 Apache TinkerPop Gremlin 和 Neo4j 的 openCypher，以及 W3C 的 RDF 查詢語言 SPARQL。這可讓您建置查詢，有效率地導覽高度連線的資料集。Neptune 亦提供多種圖形使用案例，例如推薦引擎、詐欺偵測、知識圖譜、藥物研發，以及網路安全。

Neptune 資料庫不但具備高度可用性、亦推出僅供讀取複本、時間點復原、連續備份至 Amazon S3 等功能，還能夠跨可用區域進行複寫。除此之外，Neptune 更提供了支援靜態與傳輸中加密的資料安全性功能。Neptune 屬於全受管服務，因此您不需再煩惱硬體佈建、軟體修補、設定、組態、備份等資料庫管理任務。

Neptune Analytics 是與 Neptune 資料庫相輔相成的分析資料庫引擎，可以快速分析記憶體中的大量圖形資料，以取得深入見解，並找出趨勢。Neptune Analoaytics 是用於快速分析現有圖形資料庫或儲存在資料湖中的圖形資料集的解決方案。該解決方案使用受歡迎的圖形分析演算法和低延遲分析查詢。

若要進一步了解如何使用 Amazon Neptune，我們建議您從下列各節開始著手：

- Amazon Neptune 入門
- Amazon Neptune 功能概觀

如果您不熟悉圖形，或尚未準備好投資於完整的 Neptune 生產環境，請造訪我們的 入門 主題，了解如何使用 Neptune Jupyter 筆記本進行學習和開發，而不會產生成本。

此外，在您開始設計資料庫之前，我們也建議您參考 GitHub 儲存庫使用圖形資料庫的 AWS 參考架構，您可以在其中通知您的圖形資料模型和查詢語言選擇，以及瀏覽參考部署架構範例。

關鍵服務元件

- 主要資料庫執行個體 – 支援讀寫操作，並對叢集磁碟區執行所有資料修改。每個 Neptune 資料庫叢集都有一個主要資料庫執行個體，負責寫入 (即載入或修改) 圖形資料庫內容。
- Neptune 複本 - 連線到與主要資料庫執行個體相同的儲存磁碟區，僅支援讀取操作。除了主要資料庫執行個體之外，每個 Neptune 資料庫叢集最多可以擁有 15 個 Neptune 複本。這可透過將 Neptune 複本置放在不同可用區及來自讀取用戶端的分散負載，來改善可用性。

- 叢集磁碟區 - Neptune 資料會儲存在叢集磁碟區中，該磁碟區是針對可靠性及高可用性所設計。叢集磁碟區包含跨單一 AWS 區域中多個可用區域的資料複本。因為您的資料會自動跨可用區域複寫，因此不僅耐用性高，且資料遺失的可能性也非常低。

支援開放式圖形 API

Amazon Neptune 支援這兩個屬性圖 (Gremlin 和 openCypher) 和 RDF 圖形 (SPARQL) 的開放圖形 API。它提供這兩種圖形模型及其查詢語言的高效能。您可以選擇屬性圖 (PG)模型，並使用 OpenCypher 查詢語言和/或 Gremlin 查詢語言存取相同的圖形。如果您使用 W3C 標準資源描述架構 (RDF) 模型，則可以使用標準 SPARQL 查詢語言存取您的圖形。

高度安全

Neptune 為您的資料庫提供多層安全性。安全功能包括使用 Amazon VPC 的網路隔離，以及使用您透過 AWS Key Management Service (AWS KMS) 建立並控制之金鑰進行的靜態加密。在加密的 Neptune 執行個體上，系統不僅會加密基礎儲存體中的資料，亦會一併加密同一個叢集中的自動備份、快照和複本。

全受管

隨著 Amazon Neptune 的推出，您不用再煩惱硬體佈建、軟體修補、設定、組態、備份等資料庫管理任務。

您可以使用 Neptune 來建立精密的互動式圖形應用程式，即可在幾毫秒內查詢數十億筆關係。高度連線資料的 SQL 查詢通常很複雜，而且不易調整效能。搭配 Neptune，您可以使用熱門圖形查詢語言 Gremlin、openCypher 和 SPARQL，來執行功能強大的查詢，這些查詢編寫容易，又能在連線的資料完好執行。此功能可大幅降低程式碼的複雜性，讓您可以快速建立處理關係的應用程式。

Neptune 旨在實現高於 99.99% 的可用性。該服務將資料庫引擎與為資料庫工作負載而建置的 SSD 支援型虛擬化儲存層緊密整合，藉此提高資料庫的效能和可用性。Neptune 儲存體具備容錯和自我修復能力。磁碟故障會在背景中修復，而不會遺失資料庫可用性。Neptune 會自動偵測資料庫損毀情況，無需進行損毀復原或重新建立資料庫快取，即可重新啟動。如果整個執行個體失敗，Neptune 會自動容錯移轉至其中一個僅供讀取複本 (最多可有 15 個) 的其中一個。

Amazon Neptune 的變更和更新

下表描述 Amazon Neptune 的重要變更。

變更	描述	日期
引擎版本	截至二零二零年六月二十日，引擎版本 1.3.2.1 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。如需有關此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.3.2.1。	2024年6月20日
引擎版本	截至 2024 年 6 月 10 日，引擎版本 1.3.2.0 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。如需有關此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.3.2.0。	2024年6月10日
引擎版本	截至 2024 年 3 月 11 日，引擎版本 1.2.1.1 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。如需有關此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.1.1。	2024年3月11日
引擎版本	截至 2024 年 3 月 06 日，引擎版本 1.3.1.0 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。如需有關此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.3.1.0。	2024年3月6日
更新為 AWS 受管理策略權限	`NeptuneReadOnlyAccess` 和受`NeptuneFu`	2024年1月22日

	llAccess 管理的政策現在會在原則陳述式中包含 Sid (陳述式 ID) 做為識別碼。	
Neptune 現在有提供 I/O 優化儲存	若使用 I/O 優化儲存，您需要為使用的儲存量和執行個體付費。儲存成本高於標準儲存，但您無需為使用的任何 I/O 付費。	2023 年 12 月 13 日
Neptune 的 IAM 受管政策變更	NeptuneConsoleFullAccessIAM 受管政策已更新，以授予與 Neptune Analytics 圖形交互所需的權限；新增了新的NeptuneGraphReadOnly存取政策以提供 Neptune Analytics 圖形資源的唯讀存取權，並且已新增新AWSServiceRoleForNeptuneGraphPolicy 政策讓 Neptune Analytics 圖形發佈 CloudWatch 操作和使用指標和記錄。	2023 年 11 月 29 日
引擎版本 1.3.0.0	截至 2023-11-15，引擎 1.3.0.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.3.0.0 版。	2023 年 11 月 15 日
Neptune 已在以色列 (特拉維夫) 區域推出	Amazon Neptune 現正於以色列 (特拉維夫) (il-central-1) 區域供應中。	2023 年 11 月 13 日

關於在 Neptune 屬性圖 time-to-live 中實現的博客文章	請參閱在 Amazon Neptune 中實作存留時間，第 1 部分：屬性圖，作者為 Melissa Kwok、Mike Havey 和 Kevin Phillips。	2023 年 10 月 27 日
引擎版本 1.2.1.0.R7	截至 2023-10-06，引擎版本 1.2.1.0.R7 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.1.0.R7。	2023 年 10 月 6 日
引擎版本 1.2.0.0.R4	截至 2023 年 9 月 29 日，引擎版本 1.2.0.0.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.0.R4。	2023 年 9 月 29 日
引擎版本 1.2.0.1.R3	截至 2023 年 9 月 27 日，引擎版本 1.2.0.1.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.1.R3。	2023 年 9 月 27 日
引擎版本 1.2.1.0.R6	截至 2023 年 9 月 12 日，引擎版本 1.2.1.0.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.1.0.R6。	2023 年 9 月 12 日

引擎版本 1.2.0.2.R6	截至 2023 年 9 月 12 日，引擎版本 1.2.0.2.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.2.R6。	2023 年 9 月 12 日
有關使用藍/綠部署策略進行 Neptune 引擎升級的部落格文章	請參閱在使用藍/綠部署進行引擎升級期間改善 Amazon Neptune 的可用性，作者為 Ankit Gupta 和 Abhishek Mishra。	2023 年 9 月 11 日
引擎版本 1.2.1.0.R5	截至 2023 年 9 月 2 日，引擎版本 1.2.1.0.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.1.0.R5。	2023 年 9 月 2 日
引擎版本 1.2.0.2.R5	截至 2023 年 8 月 16 日，引擎版本 1.2.0.2.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.2.R5。	2023 年 8 月 16 日
引擎版本 1.2.1.0.R4	截至 2023 年 8 月 10 日，引擎版本 1.2.1.0.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.1.0.R4。	2023 年 8 月 10 日

有關 Neptune 引擎 1.2.1.0 版的部落格文章	請參閱探索 Amazon Neptune 1.2.1.0 版包裝的功能，作者為 Joy Wang、Kevin Phillips、Andrea Nassisi 和 Navtanay Sinha。	2023 年 8 月 4 日
有關使用 Neptune 建置多模態資料庫解決方案的部落格文章	請參閱使用 Amazon Neptune 設計使用案例驅動、可高度擴展的多模態資料庫解決方案，作者為 Mike Havey。	2023 年 7 月 18 日
有關 Neptune Serverless 使用案例和最佳實務的部落格文章	請參閱使用 Amazon Neptune Serverless 來最佳化成本和效能的使用案例和最佳實務，作者為 Kevin Phillips 和 Ankit Gupta。	2023 年 6 月 28 日
引擎版本 1.2.1.0.R3	截至 2023 年 6 月 13 日，引擎版本 1.2.1.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.1.0.R3。	2023 年 6 月 13 日
有關即時產生閒暇建議的部落格文章	請參閱使用 Amazon Neptune 即時產生閒暇活動的建議，作者為 Michael Meidlinger 和 Nils Müller。	2023 年 6 月 6 日
有關使用 Neptune 和 RDKit 進行分子模型的部落格文章	請參閱使用 Amazon Neptune 和 RDKit 建立分子 SMILL 資料模型，作者為 Graham Kutchek。	2023 年 6 月 1 日

有關使用快取加速 Neptune 效能的部落格文章 (第 3 部分)	請參閱使用 Amazon Neptune 中的快取加速圖形查詢效能，第 3 部分：泰勒·里根、阿布舍克·米什拉、梅麗莎·郭和凱爾文·勞倫斯提供的 Amazon ElastiCache 全 Neptune 叢集快取架構。	2023 年 5 月 26 日
有關使用快取加速 Neptune 效能的部落格文章 (第 2 部分)	請參閱使用 Amazon Neptune 中的快取加速圖形查詢效能，第 2 部分：其他 Neptune 快取功能，作者為 Taylor Riggan、Abhishek Mishra、Melissa Kwok 和 Kelvin Lawrence。	2023 年 5 月 26 日
有關使用快取加速 Neptune 效能的部落格文章 (第 1 部分)	請參閱使用 Amazon Neptune 中的快取加速圖形查詢效能，第 1 部分：查詢和緩衝集區快取，作者為 Taylor Riggan、Abhishek Mishra、Melissa Kwok 和 Kelvin Lawrence。	2023 年 5 月 26 日
有關使用 Neptune 進行供應鏈分析的部落格文章	請參閱使用 Amazon Neptune 和 Neptune 工作台進行供應鏈資料分析和視覺化，作者為 Dhiraj Thakur 和 Rajdip Chaudhur。	2023 年 5 月 10 日
引擎版本 1.2.0.2.R4	截至 2023 年 5 月 8 日，引擎版本 1.2.0.2.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.2.R4。	2023 年 5 月 8 日

Neptune 已在中東 (阿拉伯聯合大公國) 區域推出	Amazon Neptune 現已在中東 (阿拉伯聯合大公國) (me-central-1) 區域提供。	2023 年 5 月 2 日
引擎版本 1.2.1.0.R2	截至 2023 年 5 月 2 日，引擎版本 1.2.1.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.1.0.R2。	2023 年 5 月 2 日
有關使用 Media2Cloud 搭配 AI 驅動的視訊分析，在 Neptune 上建置知識圖譜的部落格文章	請參閱使用 Media2Cloud 搭配 AI 驅動的視訊分析，在 Amazon Neptune 上建置知識圖譜，作者為 Mike Havey。	2023 年 5 月 2 日
關於如何使用 Neptune DevOcean 建置弱點修復平台的部落格文章	請參閱如何使用 Amazon Neptune 的雲端原生應用程式 DevOcean 建置弱點修復管理平台 (Gil Makmel) 和查爾斯·伊維。	2023 年 4 月 25 日
有關 Getir 如何使用 Neptune 建置詐騙偵測系統的部落格文章	請參閱 Getir 如何使用 Amazon Neptune 和 Amazon DynamoDB 建置全方位詐騙偵測系統，作者為 Berkay Berkman、Mahmut Turan、Mutlu Polatcan、Umut Cemal Kıraç、Yağız Yanıkoğlu 和 Esra Kayabali。	2023 年 4 月 6 日
有關 Wiz 如何使用 Neptune 重新構想雲端安全性的部落格文章	請參閱世界是一種圖形：Wiz 如何在 Amazon Neptune 中使用圖形來重新構想雲端安全性，作者為 Ami Luttwak 和 Brad Bebee。	2023 年 3 月 31 日

引擎版本 1.2.0.2.R3	截至 2023 年 3 月 27 日，引擎版本 1.2.0.2.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.2.R3。	2023 年 3 月 27 日
有關 CSC Generation 如何使用 Neptune 為產品探索提供支援的部落格文章	請參閱 CSC Generation 如何使用 Amazon Neptune 搭配知識圖譜來為產品探索提供支援，作者為 Bobber Cheng、Ronit Rudra 和 Melissa Kwok。	2023 年 3 月 21 日
引擎 1.2.1.0 版	截至 2023 年 3 月 8 日，引擎 1.2.1.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.2.1.0 版。	2023 年 3 月 8 日
關於探索 Neptune TinkerPop 3.6.x 新功能的博客文章	請參閱由斯蒂芬·馬萊特在亞馬遜海王星探索 Apache TinkerPop 3.6.x 的新功能。	2023 年 3 月 8 日
有關使用語義推理從 RDF 圖中推斷新事實的部落格文章	請參閱透過整合 RDFox 與 Amazon Neptune，使用語義推理從 RDF 圖中推斷新事實，作者為 Charles Ivie 和 Diana Marks。	2023 年 2 月 20 日
有關使用 Neptune 分析醫療保健 FHIR 資料的部落格文章	請參閱使用 Amazon Neptune 分析醫療保健 FHIR 資料，作者為 Alena Schmickl。	2023 年 2 月 13 日

有關使用 Neptune ML 建置即時詐騙偵測解決方案的部落格文章	請參閱使用 Amazon Neptune ML 建置即時詐騙偵測解決方案，作者為 Hua Shu 和 Soji Adeshina。	2023 年 2 月 8 日
引擎版本 1.1.1.0.R7	截至 2023 年 1 月 23 日，引擎版本 1.1.1.0.R7 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.1.0.R7。	2023 年 1 月 23 日
圖形總管已發行	圖形總管是一種開放原始碼前端 Web 應用程式工具，用於將圖形資料視覺化。請參閱 https://github.com/aws/graph-explorer。	2023 年 1 月 3 日
維護發行版本 1.1.0.0.R3	截至 2022 年 12 月 23 日，引擎版本 1.1.0.0.R3 維護版本已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.0.0.R3。	2022 年 12 月 23 日
Neptune 工作台現在在 Amazon Linux 2 和 JupyterLab 3 上運行。	Neptune 圖形筆記本現在在具有 JupyterLab 3 的 Amazon Linux 2 環境中運行。如需有關如何移至此新環境的詳細資訊，請參閱將 Neptune 筆記本從 Jupyter 遷移到 JupyterLab 3。	2022 年 12 月 21 日

有關使用 IMDb 知識圖譜提供建議和搜尋的部落格文章 (第 3 部分)	請參閱使用 IMDb 知識圖譜提供建議和搜尋 - 第 3 部分，作者為 Divya Bhargavi、Soji Adeshina、Gaurav Rele、Karan Sindwani、Vidya Sagar Ravipati 和 Matthew Rhodes。	2022 年 12 月 20 日
有關使用 IMDb 知識圖譜提供建議和搜尋的部落格文章 (第 2 部分)	請參閱使用 IMDb 知識圖譜提供建議和搜尋 - 第 2 部分，作者為 Matthew Rhodes、Soji Adeshina、Divya Bhargavi、Gaurav Rele、Karan Sindwani 和 Vidya Sagar Ravipati。	2022 年 12 月 20 日
有關使用 IMDb 知識圖譜提供建議和搜尋的部落格文章 (第 1 部分)	請參閱使用 IMDb 知識圖譜提供建議和搜尋 - 第 1 部分，作者為 Gaurav Rele、Soji Adeshina、Divya Bhargavi、Karan Sindwani、Vidya Sagar Ravipati 和 Matthew Rhodes。	2022 年 12 月 20 日
Neptune 無伺服器現已在新 AWS 區域推出	截至 2022 年 12 月 16 日，Neptune Serverless 已在下列新的 AWS 區域推出：加拿大 (中部)、歐洲 (斯德哥爾摩)、歐洲 (法蘭克福)、亞太區域 (新加坡) 和亞太區域 (雪梨)。如需 Neptune Serverless 可用的所有區域，請參閱 Amazon Neptune Serverless 限制。	2022 年 12 月 16 日

引擎版本 1.2.0.2.R2	截至 2022 年 12 月 15 日，引擎版本 1.2.0.2.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.2.R2。	2022 年 12 月 15 日
引擎版本 1.2.0.0.R3	截至 2022 年 12 月 15 日，引擎版本 1.2.0.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.0.R3。	2022 年 12 月 15 日
引擎版本 1.2.0.1.R2	截至 2022 年 12 月 13 日，引擎版本 1.2.0.1.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.1.R2。	2022 年 12 月 13 日
關於設計教育性大數據分析架構的部落格文章 AWS	請參閱使用 AWS設計教育性大數據分析架構，作者為 Lavanya Sood。	2022 年 12 月 13 日
有關圖形資料庫如何加強學習的部落格文章	請參閱圖形資料庫如何加強學習，作者為 Lavanya Sood。	2022 年 12 月 8 日
關於使 AWS 用 Glue 將 RDF 資料載入 Neptune 的部落格文章	請參閱使用麥克·哈維和法布里奇奧·納波利塔諾的 AWS Glue 將 RDF 數據加載到 Amazon Neptune。	2022 年 11 月 23 日

引擎 1.2.0.2 版	截至 2022 年 11 月 20 日，引擎 1.2.0.2 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.2.0.2 版。	2022 年 11 月 20 日
引擎 1.2.0.1 版	截至 2022 年 10 月 26 日，引擎 1.2.0.1 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.2.0.1 版。	2022 年 10 月 26 日
有關使用 Neptune 進行詐騙偵測的部落格文章	請參閱使用 Amazon Neptune 增強 Delivery Hero 的詐騙偵測能力，作者為 Wilson Tang、Amr Elnaggar、Matias Pons、Mohammad Azzam、Saurabh Deshpande 和 Luis Rodrigues Soares。	2022 年 10 月 26 日
有關 Neptune Serverless 的部落格文章	請參閱 Amazon Neptune Serverless 簡介 - 可為工作負載調整容量的全受管圖形資料庫，作者為 Danilo Poccia。	2022 年 10 月 26 日
有關使用 Lambda 和 SPARQL UPDATE LOAD 將事件驅動的 RDF 匯入至 Neptune 的部落格文章	請參閱恩智浦如何使用約翰·沃克，昂諾·貝伊斯，查爾斯·伊維和爪哇德國王使用 AWS Lambda 和 SPARQL 更新負載執行事件驅動的 RDF 導入到亞馬遜海王星。	2022 年 10 月 20 日

引擎版本 1.2.0.0.R2	截至 2022 年 10 月 14 日，引擎版本 1.2.0.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.2.0.0.R2。	2022 年 10 月 14 日
有關在 Neptune 中編碼多語言文字屬性的部落格文章	請參閱在 Amazon Neptune 中編碼多語言文字屬性以訓練預測模型，作者為 Jiani Zhang。	2022 年 10 月 14 日
有關自動測試 Neptune 資料存取的部落格文章	請參閱格雷格·比格利用 Apache TinkerPop Grimlin 對 Amazon Neptune 資料存取進行自動化測試。	2022 年 9 月 28 日
引擎版本 1.1.1.0.R6	截至 2022 年 9 月 23 日，引擎版本 1.1.1.0.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.1.0.R6。	2022 年 9 月 23 日
有關 Informatica® 如何使用 Neptune 的部落格文章	請參閱 Informatica® Cloud Data Governance and Catalog 如何將 Amazon Neptune 用於知識圖譜，作者為 Tiju Titus John、Deepak Ram 和 Farooq Ashraf。	2022 年 9 月 20 日

有關使用 Neptune 和 Tom Sawyer Perspectives 發掘金融詐騙的部落格文章	請參閱使用 Amazon Neptune 和 Tom Sawyer Perspectives 發掘金融詐騙，作者為 Janet M. Six, Senior Product Manager at Tom Sawyer Software。	2022 年 8 月 30 日
有關使用 Neptune 和 DGL 構建基於 GNN 的實時欺詐檢測解決方 SageMaker案的博客文章	請參閱使用 Amazon SageMaker、Amazon Neptune 和張健、王浩珠和朱孟心的深度圖庫建立以 GNN 為基礎的即時詐騙偵測解決方案。	2022 年 8 月 11 日
有關使用資源標籤來停止和啟動 Neptune 環境資源的部落格文章	請參閱使用資源標籤自動停止和啟動 Amazon Neptune 環境資源，作者為 Kevin Phillips。	2022 年 8 月 1 日
關於為 Apache 做出貢獻的藝術家的博客文章 TinkerPop	看到超越代碼：誰為阿帕奇做出貢獻的藝術家斯蒂芬 TinkerPop·馬萊特和凱特里納·湯普森。	2022 年 8 月 1 日
有關 Neptune 資料平面動作之精細存取控制的部落格文章	請參閱 Amazon Neptune 資料平面動作的精細存取控制，作者為 Abhishek Mishra 和 Ankit Gupta。	2022 年 7 月 29 日
有關 Neptune 全球資料庫的部落格文章	請參閱 Amazon Neptune 全球資料庫簡介，作者為 Navtanay Sinha。	2022 年 7 月 27 日
引擎 1.2.0.0 版	截至 2022 年 7 月 21 日，引擎 1.2.0.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.2.0.0 版。	2022 年 7 月 21 日

引擎版本 1.1.1.0.R5	截至 2022 年 7 月 21 日，引擎版本 1.1.1.0.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.1.0.R5。	2022 年 7 月 21 日
引擎版本 1.1.1.0.R4	截至 2022 年 6 月 23 日，引擎版本 1.1.1.0.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.1.0.R4。	2022 年 6 月 23 日
透過 Python 整合簡化的圖形分析和機器學習工作流程	您現在可以使用簡化資料科學和機器學習工作流程的開放原始碼 Python 整合，在 Amazon Neptune 中儲存的圖形資料上執行圖形分析和機器學習任務。請參閱適用於 Neptune 的 AWS Data Wrangler 文件。	2022 年 6 月 7 日
引擎版本 1.1.1.0.R3	截至 2022 年 6 月 7 日，引擎版本 1.1.1.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.1.0.R3。	2022 年 6 月 7 日
有關偵測假新聞的部落格文章	請參閱使用圖形機器學習搭配 Amazon Neptune ML 來偵測社交媒體假新聞，作者為 Hasan Shojaei 和 Sarita Joshi。	2022 年 5 月 19 日

有關使用 SQL Server Integration Services (SSIS) 搭配 Neptune 的部落格文章	請參閱使用 SQL Server Integration Services (SSIS) 和 Amazon Neptune，從 您的資料發現新見解，作者為 Mesgana Gormley 和 Melissa Kwok。	2022 年 5 月 18 日
維護發行版本 1.1.1.0.R2	截至 2022 年 5 月 16 日，引擎版本 1.1.1.0.R2 維護版本已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.1.0.R2。	2022 年 5 月 16 日
維護發行版本 1.1.0.0.R2	截至 2022 年 5 月 16 日，引擎版本 1.1.0.0.R2 維護版本已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.1.0.0.R2。	2022 年 5 月 16 日
維護發行版本 1.0.5.1.R4	截至 2022 年 5 月 16 日，引擎版本 1.0.5.1.R4 維護版本已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.5.1.R4。	2022 年 5 月 16 日

維護發行版本 1.0.5.0.R5	截至 2022 年 5 月 16 日，引擎版本 1.0.5.0.R5 維護版本已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.5.0.R5。	2022 年 5 月 16 日
有關 Neptune 中 OpenCypher 一般可用性的部落格文章	請參閱宣佈 Amazon Neptune 支援 OpenCypher 的一般可用性，作者為 Navtanay Sinha 和 Dave Bechberger。	2022 年 4 月 22 日
引擎 1.1.1.0 版	截至 2022 年 4 月 19 日，引擎 1.1.1.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.1.1.0 版。	2022 年 4 月 19 日
有關資料歷程的部落格文章	請參閱使用 G AWS lue、Amazon Neptune 和雲形線建立資料湖的資料歷程，由 Khoa Nguyen、克里希瓦桑巴拉蘇布拉曼尼亞和 Rahul Shaurya 撰寫。	2022 年 4 月 1 日
升級至引擎 1.1.0.0 版已重新啟用	從 2022 年 2 月 21 日開始，升級至引擎 1.1.0.0 版已暫時停用。它們現在已重新啟用。	2022 年 3 月 22 日
有關設計公用事業可靠性的部落格	請參閱使用雲端型、資料啟發的電力系統模型來設計公用事業可靠性，作者為 Abhineet Parchure。	2022 年 3 月 22 日

Neptune 已在非洲 (開普敦) 推出	Amazon Neptune 現可在非洲 (開普敦) (`af-south-1`) 使用。不過，在該區域的 Neptune 主控台上暫時停用 Neptune 工作台筆記本支援。	2022 年 2 月 24 日
有關使用 OWL 之模型驅動圖形的部落格文章	請參閱在 Amazon Neptune 中使用 OWL 的模型驅動圖形，作者為 Mike Havey。	2022 年 2 月 23 日
有關使用 Rhizomer 探索語義知識圖譜的部落格文章	請參閱使用 Amazon Neptune 搭配 Rhizomer 來探索沒有 SPARQL 的語義知識圖譜，作者為 Roberto García。	2022 年 2 月 22 日
有關繪製公用電網圖形的部落格文章	見圖形上的實用電網 AWS, 由鮑比·威爾遜和約瑟夫·啤酒.	2022 年 2 月 18 日
新的 Neptune ML 文字功能編碼選項	Neptune 現在支持 FastText 和句子 BERT 文本編碼進行培訓。請參閱 Neptune ML 中的 FastText功能和句子 BERT 功能在 Neptune ML 中。	2022 年 2 月 15 日
關於 OpenSearch 與 Neptune 一起使用的地理空間查詢	請參閱結合 Amazon Neptune 和 Amazon OpenSearch 服務以獲取地理空間查詢，羅斯·加貝和阿布拉什維諾德。	2022 年 2 月 1 日
有關使用 Amazon EKS 和 Neptune 發現金融犯罪的部落格文章	請參閱使用 Amazon EKS 和圖形資料庫發現金融犯罪，作者為 Severin Gassauer-Fleissner 和 Zahi Ben Shabat。	2022 年 2 月 1 日

Neptune 叢集磁碟區的大小現在可成長到 128 TiB	除了中國以外的所有支援區域 GovCloud，Neptune 叢集磁碟區的大小限制現在已從 64 TiB 增加到 128 TiB。這適用於從 1.0.2.2 版開始的所有引擎版本。請參閱 Amazon Neptune 儲存體頁面。	2022 年 2 月 1 日
Neptune 全文檢索搜尋現在整合了所有版本的 OpenSearch.	請參閱使用 Amazon OpenSearch 服務在 Amazon Neptune 進行全文搜索	2022 年 1 月 28 日
有關使用 Docker 容器部署圖形筆記本的部落格文章	請參閱使用 Docker 容器部署圖形筆記本 AWS，由甘納許·索尼和張強。	2022 年 1 月 22 日
引擎版本 1.0.5.1.R3	截至 2022 年 1 月 13 日，引擎版本 1.0.5.1.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.5.1.R3。	2022 年 1 月 13 日
有關使用 Neptune ML 的圖形型建議系統的部落格文章	請參閱使用 Neptune ML 的圖形型建議系統：關於社交網路連結預測挑戰的圖例，作者為 Yanwei Cui 和 Will Badr。	2022 年 1 月 12 日
有關自動擴展 Neptune 的部落格文章	請參閱自動擴展您的 Amazon Neptune 資料庫以符合工作負載需求，作者為 Navtanay Sinha 和 Sudhanshu Gupta。	2021 年 11 月 29 日

有關互動式圖形資料分析和視覺化的部落格文章	請參閱使用 Amazon 海王星、亞馬遜雅典娜聯邦查詢和亞馬遜建立互動式圖表資料分析和視覺化 QuickSight，由桑迪普·韋爾迪和阿布舍克密斯拉製作。	2021 年 11 月 24 日
有關使用圖形型深度學習偵測身分詐騙的部落格文章	請參閱 Careem 如何使用圖形型深度學習和 Amazon Neptune 來偵測身分詐騙，作者為 Kevin O'Brien, Kamran Habib, and Will Badr。	2021 年 11 月 23 日
引擎 1.1.0.0 版	截至 2021 年 11 月 19 日，引擎 1.1.0.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.1.0.0 版。	2021 年 11 月 19 日
有關集中資料保護和合規的部落格文章	請參閱使用 Brian O'Keefe 撰寫的 B AWS ackup 在 Amazon Neptune 中集中資料保護和合規。	2021 年 11 月 8 日
有關打擊詐騙和不當支付的部落格文章	請參閱以聯邦支出規模即時打擊詐騙和不當支付，作者為 Vladi Royzman 和 Spencer Smith。	2021 年 11 月 2 日
有關防止假帳戶註冊的部落格文章	請參閱使用 Amazon Fraud Detector 搭配 AI 即時防止假帳戶註冊，作者為 Anjan Biswas。	2021 年 10 月 29 日

引擎版本 1.0.5.1.R2	截至 2021 年 10 月 26 日，引擎版本 1.0.5.1.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.5.1.R2。	2021 年 10 月 26 日
有關使用深圖庫 HawkEye 360 預測船舶風險的博客文章	見 HawkEye 360 預測船舶風險使用深圖庫和 Amazon Neptune 蒂姆·帕夫利克, 伊恩·阿維萊茲, 丹·福特, 和高拉夫·雷勒.	2021 年 10 月 15 日
引擎 1.0.5.1 版	截至 2021 年 10 月 1 日，引擎 1.0.5.1 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.5.1 版。	2021 年 10 月 1 日
關於為什麼開發人員喜歡博客 TinkerPop	請參閱為什麼開發人員喜歡 Apache TinkerPop，圖形計算的開源框架，由布拉德·貝比，凱爾文勞倫斯和斯蒂芬·馬萊特。	2021 年 9 月 27 日
引擎版本 1.0.5.0.R3	截至 2021 年 9 月 15 日，引擎版本 1.0.5.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.5.0.R3。	2021 年 9 月 15 日

有關使用 Amundsen 和 Neptune 建置資料探索解決方案的部落格文章	請參閱使用 Amundsen 和 Amazon Neptune 建置資料探索解決方案，作者為 Peter Hanssens 和 Don Simpson。	2021 年 9 月 8 日
Neptune 已更新串流輸詢器以支援非字串全文檢索搜尋查詢	此版本中包含許多全文檢索搜尋的改進，包括對非字串的屬性值編製索引的支援。請參閱 Amazon Neptune 中的非字串 OpenSearch 索引。	2021 年 8 月 23 日
引擎版本 1.0.5.0.R2	截至 2021 年 8 月 16 日，引擎版本 1.0.5.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.5.0.R2。	2021 年 8 月 16 日
引擎版本 1.0.4.2.R5	截至 2021 年 8 月 16 日，引擎版本 1.0.4.2.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.2.R5。	2021 年 8 月 16 日
有關 Neptune 中圖形儲存通訊協定支援的部落格文章	請參閱 Amazon Neptune 的圖形儲存通訊協定支援簡介，作者為 Chris Smith。	2021 年 8 月 2 日
有關 Neptune ML 新功能的部落格文章	請參閱使用 Amazon Neptune ML 提供的新功能，在您的圖形中發現更多的見解，作者為 Soji Adeshina。	2021 年 7 月 30 日

有關使用 Neptune ML 更快獲得預測的部落格文章	請參閱使用 Amazon Neptune ML 更快獲得不斷變化之圖形資料的預測，作者為 Soji Adeshina。	2021 年 7 月 30 日
有關使用 Neptune ML 更輕鬆、更快地進行圖形機器學習的部落格文章	請參閱使用 Amazon Neptune ML 更輕鬆、更快地進行圖形機器學習，作者為 Soji Adeshina。	2021 年 7 月 30 日
有關 Neptune openCypher 支援的部落格文章	請參閱宣佈適用於 Amazon Neptune 的 OpenCypher：同時使用 openCypher 和 Gremlin 建置更好的圖形應用程式，作者為 Brad Bebee。	2021 年 7 月 29 日
引擎 1.0.5.0 版	截至 2021 年 7 月 27 日，引擎 1.0.5.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.5.0 版。	2021 年 7 月 27 日
引擎版本 1.0.4.2.R4	截至 2021 年 7 月 23 日，引擎版本 1.0.4.2.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.2.R4。	2021 年 7 月 23 日
Neptune 已在中國 (北京) 推出	Amazon Neptune 現可在中國 (北京) (`cn-north-1`) 使用。	2021 年 7 月 21 日

引擎版本 1.0.4.2.R3	截至 2021 年 6 月 28 日，引擎版本 1.0.4.2.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.2.R3。	2021 年 6 月 28 日
有關 Dream11 如何使用 Neptune 擴展其社交網路的部落格文章	請參閱瞭解全球最大的夢幻運動平台 Dream11 如何透過 Amazon 海王星和亞馬遜擴展其社交網路。 ElastiCache	2021 年 6 月 25 日
關於使用 Neptune 和 PoolParty 語義套件將數據轉化為知識的博客文章	請參閱使用 PoolParty 語義套件和 Amazon Neptune 將數據轉換為知識，由約安娜 Lytra 和阿爾賓·艾哈梅蒂。	2021 年 6 月 16 日
關於使用 Neptune 探索 UniProt 知識庫的部落格文章	請參閱透過 AWS 開放資料和 Amazon Neptune 探索 UniProt 蛋白質知識庫，由 Eric Greene、Rafa Xu 和袁施撰寫。	2021 年 6 月 10 日
有關使用 Neptune 進行資料驅動風險分析的部落格文章	請參閱 Amazon 海王星和亞馬遜 OpenSearch 服務進行數據驅動的風險分析，由阿德里安·德容格和羅希特·薩蒂亞納拉亞納。	2021 年 6 月 10 日
引擎版本 1.0.4.2.R2	截至 2021 年 6 月 1 日，引擎版本 1.0.4.2.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.2.R2。	2021 年 6 月 1 日

關於使用 Neptune 將 AWS 基礎結構視覺化的部落格	請參閱使用 Amazon Neptune 和 AWS Config 將您的 AWS 基礎設施視覺化，由羅漢·雷扎達和 Amey Dhavle 撰寫。	2021 年 5 月 25 日
有關使用 Data Lens 搭配 Neptune 之組態的部落格文章	請參閱羅素沃特森使用資料鏡頭在 Amazon Neptune 中設定 AWS 服務以建立知識圖表。	2021 年 5 月 5 日
有關使用 Data Lens 在 Neptune 建置知識圖譜的部落格文章	請參閱使用 Data Lens 在 Amazon Neptune 中建置知識圖譜，作者為 Russell Waterson。	2021 年 5 月 5 日
引擎 1.0.1.0、1.0.1.1 和 1.0.1.2 版現在已棄用	從現在開始，將不會使用任何這些引擎版本或與其相關的任何修補程式來建立新的資料庫執行個體。	2021 年 4 月 26 日
有關日本經濟產業省使用 Neptune 之案例研究的英文翻譯	請參閱日本經濟產業省透過 AWS為 GBizInfo 公司資訊搜尋資料庫提供支援。	2021 年 3 月 31 日
有關使用 Neptune 搭配 Amazon Comprehend 和 Lex 的部落格文章	請參閱使用 Amazon Neptune、Amazon Comprehend 和 Amazon Lex 增強您的知識圖譜，作者為 Dave Bechberger。	2021 年 3 月 31 日
有關使用 Lambda 函數搭配 Neptune 的部落格文章	請參閱 Ian Robinson 撰寫的搭配 Amazon Neptune 使用 AWS Lambda	2021 年 3 月 26 日

引擎版本 1.0.4.1.R1.1	截至 2021 年 3 月 22 日，引擎版本 1.0.4.1.R1.1 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.1.R1.1。	2021 年 3 月 22 日
引擎版本 1.0.4.1.R2.1	截至 2021 年 3 月 11 日，引擎版本 1.0.4.1.R2.1 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.1.R2.1。	2021 年 3 月 11 日
有關使用 Neptune 的開放原始碼圖形筆記本進行圖形視覺化的部落格文章	請參閱開始使用開放原始碼圖形筆記本進行圖形視覺化，作者為 Joy Wang、Ora Lassila 和 Stephen Mallette。	2021 年 3 月 10 日
有關將 Neptune 與 Amundsen 資料探索和中繼資料引擎整合的教學課程	請參閱如何使用 Amundsen 搭配 Amazon Neptune，作者為 Andrew Ciambrone。	2021 年 3 月 2 日
引擎版本 1.0.4.1.R2	截至 2021 年 2 月 24 日，引擎版本 1.0.4.1.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.1.R2。	2021 年 2 月 24 日

引擎版本 1.0.4.0.R2	截至 2021 年 2 月 24 日，引擎版本 1.0.4.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.4.0.R2。	2021 年 2 月 24 日
引擎版本 1.0.3.0.R3	截至 2021 年 2 月 19 日，引擎版本 1.0.3.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.3.0.R3。	2021 年 2 月 19 日
引擎版本 1.0.2.2.R6	截至 2021 年 2 月 19 日，引擎版本 1.0.2.2.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.2.2.R6。	2021 年 2 月 19 日
有關使用 Amazon Comprehend 事件建置知識圖譜的部落格文章	請參閱使用 Amazon Comprehend 事件在 Amazon Neptune 中建置知識圖譜,作者為 Brian O'Keefe, Graham Horwood, and Navtanay Sinha。	2021 年 1 月 19 日
有關啟用低程式碼圖形資料應用程式的部落格文章	請參閱使用 Amazon Neptune 和 Graphistry 啟用低程式碼圖形資料應用程式，作者為 Leo Meyerovich、Dave Bechberger 和 Taylor Riggan。	2021 年 1 月 18 日

已新增圖形資料入門的筆記本文件。	已新增與 Neptune 工作台整合的區段，其會協助您開始建立圖形資料和開發圖形應用程式，而不必啟動 Neptune 叢集，直到您準備就緒。	2021 年 1 月 15 日
有關在幾秒鐘內重設 Neptune 圖形資料的部落格文章	請參閱在 Amazon Neptune 中幾秒鐘內重設您的圖形資料，作者為 Niraj Jetly 和 Navtanay Sinha。	2020 年 12 月 17 日
關於諾華股份公司如何使用 BERT SageMaker 和 Neptune 的部落格文章	請參閱諾華股份公司使用 Amazon SageMaker 和亞馬遜海王星使用 BERT 構建和豐富知識圖，由奧特馬尼·哈姆扎烏伊，法特瑪阿爾干納齊和維克多·馬列塞維克。	2020 年 12 月 14 日
有關使用主題網路建置知識圖譜的部落格文章	請參閱 在 Amazon Neptune 中使用主題網路建置知識圖譜，作者為 Edward Brown (AI 專案主管)、Eduardo Piairo (架構師)、Marcia Oliveira (首席資料科學家) 和 Jack Hampson (Deeper Insights 執行長)。	2020 年 12 月 14 日
引擎 1.0.4.1 版	截至 2020 年 12 月 8 日，引擎 1.0.4.1 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.4.1 版。	2020 年 12 月 8 日
有關開始使用 Neptune ML 的部落格文章	請參閱如何開始使用 Neptune ML，作者為 George Karypis、Dave Bechberger 和 Karthik Bharathy。	2020 年 12 月 8 日

Neptune 現在具有快速重設 API	使用快速重設 API，您可以快速且輕鬆地刪除資料庫叢集中的所有資料。請參閱快速重設 API。	2020 年 12 月 4 日
有關在 Pendulum 建置生物知識圖譜的部落格文章	請參閱使用 Amazon Neptune 在 Pendulum 建置生物知識圖譜，作者為 Connor Skennerton。	2020 年 11 月 26 日
關於 TinkerPop 3.4.8 Neptune 新功能的博客文章	見探索阿帕奇 TinkerPop 3.4.8 在亞馬 Amazon Neptune 的新功能，由斯蒂芬·馬萊特。	2020 年 11 月 18 日
有關使用 Amazon Kendra 搜尋服務搭配 Neptune 的部落格文章	請參閱將您的企業知識圖譜納入 Amazon Kendra，作者為 Yazdan Shirvany、Mohit Mehta 和 Dipto Chakravarty。	2020 年 11 月 17 日
現在可使用事件通知	Neptune 現在支援事件通知，您可以使用這些通知，更輕鬆地監控資料庫叢集。請參閱使用 Neptune 事件通知。	2020 年 10 月 29 日
現在可使用自訂端點	Neptune 現在支援自訂端點，可在連線至資料庫執行個體時取得更大的控制。請參閱連線至 Amazon Neptune 端點。	2020 年 10 月 29 日
有關使用 AWS Database Migration Service (DMS) 填入 Neptune 圖形的部落格文章	請參閱使用 Database Migration Service (DMS) 從關聯式資料 AWS 庫在 Amazon Neptune 中填入圖形 — 第 4 部分：將它們放在一起 (Chris Smith) 撰寫。	2020 年 10 月 22 日

有關使用 AWS Database Migration Service (DMS) 填入 Neptune 圖形的部落格文章	請參閱使用 Database Migration Service (DMS) 從關聯式資 AWS 料庫在 Amazon Neptune 中填入圖形 — 第 3 部分：設計 RDF 模型 (Chris Smith)。	2020 年 10 月 22 日
有關使用 AWS Database Migration Service (DMS) 填入 Neptune 圖形的部落格文章	請參閱使用 Database Migration Service (DMS) 從關聯式資 AWS 料庫在 Amazon Neptune 中填入圖形 — 第 2 部分：設計屬性圖模型 (Chris Smith)。	2020 年 10 月 22 日
有關使用 AWS Database Migration Service (DMS) 填入 Neptune 圖形的部落格文章	請參閱使用資料庫移轉服務 (DMS) 從關聯式資料 AWS 庫在 Amazon Neptune 中填入圖形 — 第 1 部分：設定階段 (Chris Smith) 撰寫。	2020 年 10 月 22 日
引擎 1.0.4.0 版	截至 2020 年 10 月 12 日，引擎 1.0.4.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.4.0 版。	2020 年 10 月 12 日
引擎版本 1.0.3.0.R2	截至 2020 年 10 月 12 日，引擎版本 1.0.3.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.3.0.R2。	2020 年 10 月 12 日

引擎版本 1.0.2.2.R5	截至 2020 年 10 月 12 日，引擎版本 1.0.2.2.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.2.2.R5。	2020 年 10 月 12 日
有關針對 SPARQL 聯合查詢設定 VPC 的部落格文章	請參閱使用 Amazon Neptune 針對 SPARQL 1.1 聯合查詢設定 Amazon VPC，作者為 Charles Ivie。	2020 年 10 月 12 日
有關撰寫 SPARQL 階層式刪除的部落格文章	請參閱以 SPARQL 撰寫階層式刪除，作者為 Ora Lassila。	2020 年 10 月 5 日
關於使用 Neptune 繪製 AWS 資源的博客文章	請參閱圖表您的 AWS 資源與 Amazon Neptune, 由戴夫·貝奇貝格.	2020 年 9 月 28 日
有關使用 Neptune 為藥物主動監視和不良事件報告建置 Meddra 術語對應的部落格文章	請參閱為藥物主動監視和不良事件報告建置 Amazon Neptune 型 Meddra 術語對應，作者為 Vaijayanti Joshi、Deven Atnoor 博士和 Sudhanshu Malhotra。	2020 年 9 月 24 日
有關使用 Neptune 從資料倉儲建置知識圖譜以補充商業智慧的部落格文章	請參閱使用 Amazon Neptune 從資料倉儲建置知識圖譜來補充商業智慧作者為 Shahria Hossain 和 Mikael Graindorge。	2020 年 9 月 23 日
有關使用 Neptune Gemlin 用戶端進行負載平衡的部落格文章	請參閱使用 Amazon Neptune Grimlin 用戶端對圖形查詢進行負載平衡，作者為 Ian Robinson。	2020 年 9 月 16 日

有關在 Cox Automotive 使用身分圖進行數位個人化的部落格文章	請參閱 Cox Automotive 使用由 Amazon Neptune 提供支援的身分圖來擴展數位個人化，作者為 Carlos Rendon 和 Niraj Jetly。	2020 年 9 月 16 日
有關對 Yelp 資料進行協作篩選的部落格文章	請參閱在 Yelp 資料上使用協作篩選，以在 Amazon Neptune 中建立建議系統，作者為 Chad Tindel。	2020 年 9 月 8 日
有關 Amazon Neptune 中查詢結果視覺化的部落格文章	請參閱使用 Amazon Neptune 工作台將查詢結果視覺化，作者為 Kelvin Lawrence。	2020 年 9 月 2 日
Neptune 已發行圖形視覺化	Amazon Neptune 現在於 Neptune 工作台的 Jupyter 筆記本中提供廣泛的圖形視覺化功能，以及許多讓筆記本更容易使用的新功能。請參閱圖形視覺化。	2020 年 8 月 12 日
Neptune 已在南美洲 (聖保羅) 推出	Amazon Neptune 現可在 南美洲 (聖保羅) (`sa-east-1`) 使用。	2020 年 8 月 6 日
Neptune 已在亞太區域 (香港) 推出	Amazon Neptune 現可在亞太區域 (香港) (`ap-east-1`) 使用。	2020 年 8 月 6 日
引擎 1.0.3.0 版	截至 2020 年 8 月 3 日，引擎 1.0.3.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.3.0 版。	2020 年 8 月 3 日

引擎版本 1.0.2.2.R4	截至 2020 年 7 月 23 日，引擎版本 1.0.2.2.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.2.2.R4。	2020 年 7 月 23 日
有關 ZeroBase 使用 Amazon Neptune 自動追蹤聯絡人的部落格文章	請參閱 ZeroBase 使用 Amazon Neptune 建立私有、安全和自動化的聯絡人追蹤，作者為 David Harris 和 Aron Szanto。	2020 年 7 月 13 日
Neptune 已在美國西部 (加利佛尼亞北部) 推出	Amazon Neptune 現可在美國西部 (加利佛尼亞北部) (`us-west-1`) 使用。	2020 年 7 月 9 日
Amazon Neptune 支援標籤型存取控制	您現在可以在 IAM 政策中使用 AWS 標籤來控制對 Neptune 資料庫的存取。請參閱 Amazon Neptune 中的標籤型存取控制。	2020 年 7 月 7 日
Java 串流輪詢器現在可供使用	Amazon Neptune 現在支援 Java 版 Lambda 串流輪詢器用於 Neptune 串流以及 Python 串流。請參閱新增您要建立之 Neptune 串流消費者堆疊的詳細資訊。	2020 年 7 月 6 日
關於 AWS COVID-19 知識圖表的部落格文章	請參閱建立和查詢 AWS COVID-19 知識圖，由尼納德·庫爾卡尼，科爾比智者，喬治·普萊斯和米格爾·羅梅羅。	2020 年 7 月 1 日

引擎 1.0.1.1 版	截至 2020 年 6 月 26 日，引擎 1.0.1.1 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.1.1 版。	2020 年 6 月 26 日
有關從 Blazegraph 遷移到 Amazon Neptune 的部落格文章	請參閱 Dave Bechberger 撰寫的 Moving to the cloud: Migrating Blazegraph to Amazon Neptune。	2020 年 6 月 25 日
將資料擷取從 Neo4j 變更為 Amazon Neptune 的部落格文章	請參閱 Sanjeet Sahay 撰寫的 Change data capture from Neo4j to Amazon Neptune using Amazon Managed Streaming for Apache Kafka。	2020 年 6 月 22 日
有關 Waves 如何使用 Amazon Neptune 的部落格文章	請參閱 Pavel Vasilyev 撰寫的 How Waves runs user queries and recommendations at scale with Amazon Neptune。	2020 年 6 月 16 日
引擎 1.0.1.2 版	截至 2020 年 6 月 10 日，引擎 1.0.1.2 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.1.2 版。	2020 年 6 月 10 日
有關建置客戶知識儲存庫的部落格文章	請參閱 Ram Bhandarkar 撰寫的 Building a customer 360 knowledge repository with Amazon Neptune and Amazon Redshift。	2020 年 6 月 9 日

有關 Gunosy 如何使用 Amazon Neptune 的部落格文章	請參閱 Yosuke Uchiyama 撰寫的 How Gunosy built a comment feature in News Pass using Amazon Neptune。	2020 年 6 月 8 日
關於 AWS COVID-19 知識圖表的部落格文章	請參閱由尼納德庫爾卡尼，科爾比智者，喬治普萊斯和米格爾·羅梅羅構建和查詢 AWS COVID-19 知識圖。	2020 年 6 月 2 日
有關使用 Amazon Neptune 探索 COVID-19 研究的部落格文章	請參閱 George Price、Colby Wise、Miguel Romero 和 Ninad Kulkarni 製作的 Exploring scientific research on COVID-19 with Amazon Neptune、Amazon Comprehend Medical 和 Tom Sawyer Graph Database Browser。	2020 年 6 月 2 日
您現在可以使用將數據加載到 Neptune AWS DMS	請參閱使用 AWS Database Migration Service 將資料從不同的資料存放區載入 Amazon Neptune。	2020 年 6 月 1 日
引擎 1.0.2.0 版即將棄用	Amazon Neptune 引擎 1.0.2.0 版現已棄用。在此引擎版本上執行的叢集會在 2020 年 6 月 1 日之後的第一個維護時段自動升級至 1.0.2.1 版。	2020 年 5 月 19 日
有關使用 Neptune 建置客戶身分圖的部落格文章	請參閱 Rajesh Wunnava 和 Taylor Riggan 所著的 Building a customer identity graph with Amazon Neptune。	2020 年 5 月 12 日

引擎版本 1.0.2.0.R3	截至 2020 年 5 月 5 日，引擎版本 1.0.2.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.2.0.R3。	2020 年 5 月 5 日
引擎版本 1.0.2.1.R6	截至 2020 年 4 月 22 日，引擎版本 1.0.2.1.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.2.1.R6。	2020 年 4 月 22 日
有關將資料從 Neo4j 遷移到 Neptune 的部落格文章	請參閱 Sanjeet Sahay 的將 Neo4j 圖形資料庫遷移到有全自動公用程式的 Amazon Neptune。	2020 年 4 月 13 日
有關使用 Neptune 降低建置圖形應用程式成本的部落格文章	請參閱 Karthik Bharathy and Brad Bebee 的使用 Amazon Neptune T3 執行個體，將建置圖形應用程式的成本降低 76%。	2020 年 4 月 9 日
Neptune 提供了 T3 高載的執行個體類別	您現在可以針對符合成本效益的開發和測試目的，建立 Amazon Neptune T3 高載執行個體。請參閱 Neptune T3 高載執行個體類別。	2020 年 4 月 8 日

引擎版本 1.0.2.2.R2	截至 2020 年 4 月 2 日，引擎版本 1.0.2.2.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎版本 1.0.2.2.R2。	2020 年 4 月 2 日
有關在 EDGAR 繪製投資相依性圖形的部落格文章	請參閱 Lawrence Verdi 的用 Amazon Neptune 繪製投資相依性。	2020 年 3 月 17 日
Neptune 已在歐洲 (巴黎) 推出	Amazon Neptune 現可在歐洲 (巴黎) (eu-west-3) 使用。	2020 年 3 月 11 日
引擎 1.0.2.2 版	截至 2020 年 3 月 9 日，引擎 1.0.2.2 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。如需此引擎版本的詳細資訊，請參閱 Neptune 引擎 1.0.2.2 版。	2020 年 3 月 9 日
停止和重新啟動資料庫叢集	您現在可以使用 Neptune 主控台停止資料庫叢集 7 天，稍後在再次需要時重新啟動它。資料庫叢集停用時，只需支付叢集儲存、手動快照和自動備份儲存的費用，但不需支付任何資料庫執行個體時數的費用。請參閱停止和啟動 Amazon Neptune 資料庫叢集。	2020 年 2 月 19 日

有關 Nike 社交圖的影片	聽聽托德·埃斯卡洛納（Todd Escalona）與耐克高級工程經理馬克·旺恩海姆（Marc Wangenheim）的 AWS 談話，了解該公司如何通過在 Amazon Neptune 上構建的社交圖為許多應用程序提供支持請參閱 Nike：使用 Amazon Neptune 的大型社交圖。	2020 年 2 月 11 日
Neptune 叢集現在可以設定為需要 SSL 連線	在仍支援 HTTP 連線的區域中，現在所有新參數群組都預設為開啟 SSL。現有的參數群組沒有變更，但您可以將 `neptune_enforce_ssl` 參數變更為 1，強制用戶端使用 SSL。請參閱傳輸中加密：使用 SSL/HTTPS 連接到 Neptune，以取得如何在仍支援 HTTP 連線的區域中為叢集啟用 HTTP 連線的相關資訊。如需叢集和執行個體參數的描述，請參閱您可以用來設定 Amazon Neptune 的參數。	2020 年 2 月 10 日
您現在可以在 Neptune 的 CloudFormation 範本中指定引擎版本和刪除保護	Amazon Neptune 已更新 CloudFormation 範本，以包含可讓您為新資料庫叢集指定特定引擎版本的`AWS::Neptune::DBCluster.DeletionProtection` 參數，以及可讓您為其啟用刪除保護的參數。`AWS::Neptune::DBCluster.EngineVersion`	2020 年 2 月 9 日

刪除保護	Amazon Neptune 已為資料庫叢集和執行個體提供刪除保護。一旦在資料庫叢集或執行個體上啟用刪除保護，就無法刪除。請參閱如果已啟用刪除保護，您就無法刪除資料庫執行個體。	2020 年 1 月 20 日
Neptune 已在中國 (寧夏) 推出	Amazon Neptune 現可在中國 (寧夏) (cn-northwest-1) 使用。	2020 年 1 月 15 日
引擎版本 1.0.2.1.R4	引擎 1.0.2.1 版的修補程式 R4 已全面推出。如需詳細資訊，請參閱 Neptune 引擎版本 1.0.2.1.R4。	2019 年 12 月 20 日
引擎版本 1.0.2.1.R3	引擎 1.0.2.1 版的修補程式 R3 已全面推出。如需詳細資訊，請參閱 Neptune 引擎版本 1.0.2.1.R3。	2019 年 12 月 12 日
有關使用 Neptune 分析社交媒體摘要的部落格文章	請參閱使用 Amazon Neptune 分析社交媒體摘要。	2019 年 11 月 27 日
引擎版本 1.0.2.1.R2	引擎 1.0.2.1 版的修補程式 R2 已全面推出。如需詳細資訊，請參閱 Neptune 引擎版本 1.0.2.1.R2。	2019 年 11 月 25 日
引擎版本 1.0.2.1.R1	Amazon Neptune 引擎版本 1.0.2.1.R1 正式推出。如需詳細資訊，請參閱 Neptune 引擎 1.0.2.1 版。	2019 年 11 月 22 日
引擎版本 1.0.2.0.R2	引擎 1.0.2.0 版的修補程式 R2 已全面推出。如需詳細資訊，請參閱 Neptune 引擎版本 1.0.2.0.R2。	2019 年 11 月 21 日

有關 re:Invent 2019 舉行的 Neptune 講習會和研討會的部落格文章	在 AWS re: Invent 2019 上查看您的 Amazon Neptune 會議、研討會和粉筆演講指南。	2019 年 11 月 20 日
引擎版本 1.0.2.0.R1	Amazon Neptune 引擎版本 1.0.2.0.R1 正式推出。如需詳細資訊，請參閱 Neptune 引擎 1.0.2.0 版。	2019 年 11 月 8 日
有關使用 Neptune 串流擷取圖形變更的部落格文章	請參閱使用 Neptune 串流擷取圖形變更。	2019 年 11 月 6 日
引擎版本 1.0.1.0.200502.0	Amazon Neptune 引擎版本 1.0.1.0.200502.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200502.0。	2019 年 10 月 31 日
Neptune 已在中東 (巴林) 推出	Amazon Neptune 現可在中東 (巴林) (me-south-1) 使用。	2019 年 10 月 30 日
Neptune 已在加拿大 (中部) 推出	Amazon Neptune 現可在加拿大 (中部) (ca-central-1) 使用。	2019 年 10 月 30 日
有關 Neptune 的新 SPARQL 串流功能和 SPARQL 聯合查詢支援的部落格文章	請參閱 Amazon Neptune 發布圖表的串流、SPARQL 聯合查詢等。	2019 年 10 月 17 日
引擎版本 1.0.1.0.200463.0	Amazon Neptune 引擎版本 1.0.1.0.200463.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200463.0。	2019 年 10 月 15 日
引擎版本 1.0.1.0.200457.0	Amazon Neptune 引擎版本 1.0.1.0.200457.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200457.0。	2019 年 9 月 19 日

有關 Neptune 全新 SPARQL Explain 功能的部落格文章	請參閱使用 SPARQL Explain 以了解 Amazon Neptune 中的查詢執行。	2019 年 9 月 17 日
有關 Neptune 支持 TinkerPop 3.4 的博客文章	請參閱 Amazon Neptune 現在支持 TinkerPop 3.4 功能。	2019 年 9 月 6 日
關於 PyTorch 在 Amazon 上使用 Neptune 的博客文章 SageMaker	查看在 Amazon SageMaker 和亞馬 Amazon Neptune PyTorch 上使用的個性化「按風格購物」體驗。	2019 年 8 月 22 日
關於使用 Neptune AWS AppSync 和 Amazon 彈性疼的博客文章	請參閱整合替代資料來源 AWS AppSync：Amazon Neptune 和 Amazon ElastiCache。	2019 年 8 月 22 日
Neptune 在 AWS GovCloud（美國東部）發射	Amazon Neptune 現已在（美國東部）AWS GovCloud（美國東部）（美國東部 -1）上市。	2019 年 8 月 21 日
Neptune 在 AWS GovCloud（美國西部）發射	Amazon Neptune 現已在（美國西部）AWS GovCloud（美國西部）（美國-西 1）上市。	2019 年 8 月 14 日
引擎版本 1.0.1.0.200369.0	Amazon Neptune 引擎版本 1.0.1.0.200369.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200369.0。	2019 年 8 月 13 日
引擎版本 1.0.1.0.200366.0	Amazon Neptune 引擎版本 1.0.1.0.200366.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200366.0。	2019 年 7 月 26 日

關於 PyTorch 在 Amazon 上使用 Neptune 的博客文章 SageMaker	查看在 Amazon SageMaker 和亞馬 Amazon Neptune PyTorch 上使用的個性化「按風格購物」體驗。	2019 年 7 月 3 日
引擎版本 1.0.1.0.200348.0	Amazon Neptune 引擎版本 1.0.1.0.200348.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200348.0。	2019 年 7 月 2 日
Neptune 已在歐洲 (斯德哥爾摩) 推出	Amazon Neptune 現可在歐洲 (斯德哥爾摩) (eu-north-1) 使用。	2019 年 6 月 27 日
Neptune 現在可以將審核日誌發佈到 CloudWatch 日誌	如需詳細資訊，請參閱將 Neptune 日誌發佈到 Amazon CloudWatch 日誌。	2019 年 6 月 18 日
引擎版本 1.0.1.0.200310.0	Amazon Neptune 引擎版本 1.0.1.0.200310.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200310.0。	2019 年 6 月 12 日
關於 LifeOmic的博客文章 JupiterOne	請參閱如何 LifeOmic使用 Amazon Neptune JupiterOne 簡化安全性和合規操作。	2019 年 5 月 2 日
Neptune 已在亞太區域 (首爾) 推出	Amazon Neptune 現可在亞太區域 (首爾) (ap-northeast-2) 使用。	2019 年 5 月 1 日
引擎版本 1.0.1.0.200296.0	Amazon Neptune 引擎版本 1.0.1.0.200296.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200296.0。	2019 年 5 月 1 日

Neptune 已在亞太區域 (孟買) 推出	Amazon Neptune 現可在亞太區域 (孟買) (ap-south-1) 使用。	2019 年 3 月 6 日
有關 Gremlin 查詢提示的部落格文章	請參閱適用於 Amazon Neptune 的 Gremlin 查詢提示簡介。	2019 年 2 月 26 日
Neptune 已在亞太區域 (東京) 推出	Amazon Neptune 現可在亞太區域 (東京) (ap-northeast-1) 使用。	2019 年 1 月 23 日
AWS CloudFormation 用於創建訪問 Neptune 的 AWS Lambda 函數的模板	已更新入門區段，並新增 AWS CloudFormation 範本以建立 Lambda 函數以搭配 Neptune 使用。如需詳細資訊，請參閱 Neptune 入門。	2019 年 1 月 23 日
引擎版本 1.0.1.0.200267.0	Amazon Neptune 引擎版本 1.0.1.0.200267.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200267.0。	2019 年 1 月 21 日
Neptune 已在亞太區域 (雪梨) 推出	Amazon Neptune 現可在亞太區域 (雪梨) (ap-southeast-2) 使用。	2019 年 1 月 9 日
有關使用 Metaphactory 的部落格文章	請參閱使用 Metaphactory 在 Amazon Neptune 上探索知識圖形。	2019 年 1 月 9 日
Neptune 已在亞太區域 (新加坡) 推出	Amazon Neptune 現可在亞太區域 (新加坡) (ap-southeast-1) 使用。	2018 年 12 月 13 日

引擎版本 1.0.1.0.200264.0	Amazon Neptune 引擎版本 1.0.1.0.200264.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200264.0。	2018 年 11 月 19 日
Amazon Neptune SSL 支援	Neptune 現在支援 SSL 連線。	2018 年 11 月 19 日
綜合錯誤主題	所有錯誤訊息和代碼資訊現在都放在單一主題中。	2018 年 11 月 15 日
更新的入門主題	以其他連結和重新整理過的文件更新入門主題。	2018 年 11 月 14 日
引擎版本 1.0.1.0.200258.0	Amazon Neptune 引擎版本 1.0.1.0.200258.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200258.0。	2018 年 11 月 8 日
Neptune 已在歐洲 (法蘭克福) 推出	Amazon Neptune 現可在 歐洲 (法蘭克福) (eu-central-1) 使用。	2018 年 11 月 7 日
部落格文章系列的第 1 篇	請參閱讓我幫您畫圖 - 第一部分 - Air Routes。	2018 年 11 月 7 日
有關使用 Amazon SageMaker Jupyter 筆記本的博客文章	請參閱使用 Amazon SageMaker Jupyter 筆記本分析亞馬遜海王星圖。	2018 年 11 月 1 日
引擎版本 1.0.1.0.200255.0	Amazon Neptune 引擎版本 1.0.1.0.200255.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200255.0。	2018 年 10 月 29 日
Neptune 已在歐洲 (倫敦) 推出	Amazon Neptune 現可在歐洲 (倫敦) (eu-west-2) 使用。	2018 年 10 月 3 日

引擎版本 1.0.1.0.200237.0	Amazon Neptune 引擎版本 1.0.1.0.200237.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200237.0。	2018 年 9 月 6 日
引擎版本 1.0.1.0.200236.0	Amazon Neptune 引擎版本 1.0.1.0.200236.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200236.0。	2018 年 7 月 24 日
引擎版本 1.0.1.0.200233.0	Amazon Neptune 引擎版本 1.0.1.0.200233.0 正式推出。如需詳細資訊，請參閱更新 1.0.1.0.200233.0。	2018 年 6 月 22 日
新 Neptune 快速入門	更新了快速入門 AWS CloudFormation 和 Gemlin 控制台教程。如需詳細資訊，請參閱 Amazon Neptune 快速入門使用 AWS CloudFormation。	2018 年 6 月 19 日
Amazon Neptune 初始版本	此為《Neptune 使用者指南》的初始版本。另請參閱發行部落格文章 Amazon Neptune 已正式上市。	2018 年 5 月 30 日
Neptune 部落格文章簡介	請參閱 Amazon Neptune - 全受管圖形資料庫服務。	2017 年 11 月 29 日

Amazon Neptune 入門

Amazon Neptune 是一種全受管圖形資料庫服務，其可擴展以處理數十億種關係，並讓您以毫秒的延遲查詢它們，而這類容量的成本很低。

如果您正在尋找有關 Neptune 的更多詳細資訊，請參閱 Amazon Neptune 功能概觀。

如果您已了解圖形，請跳至 使用圖形筆記本。或者，如果您想要立即建立 Neptune 資料庫，請參閱 使用 AWS CloudFormation 堆疊建立 Neptune 資料庫叢集。

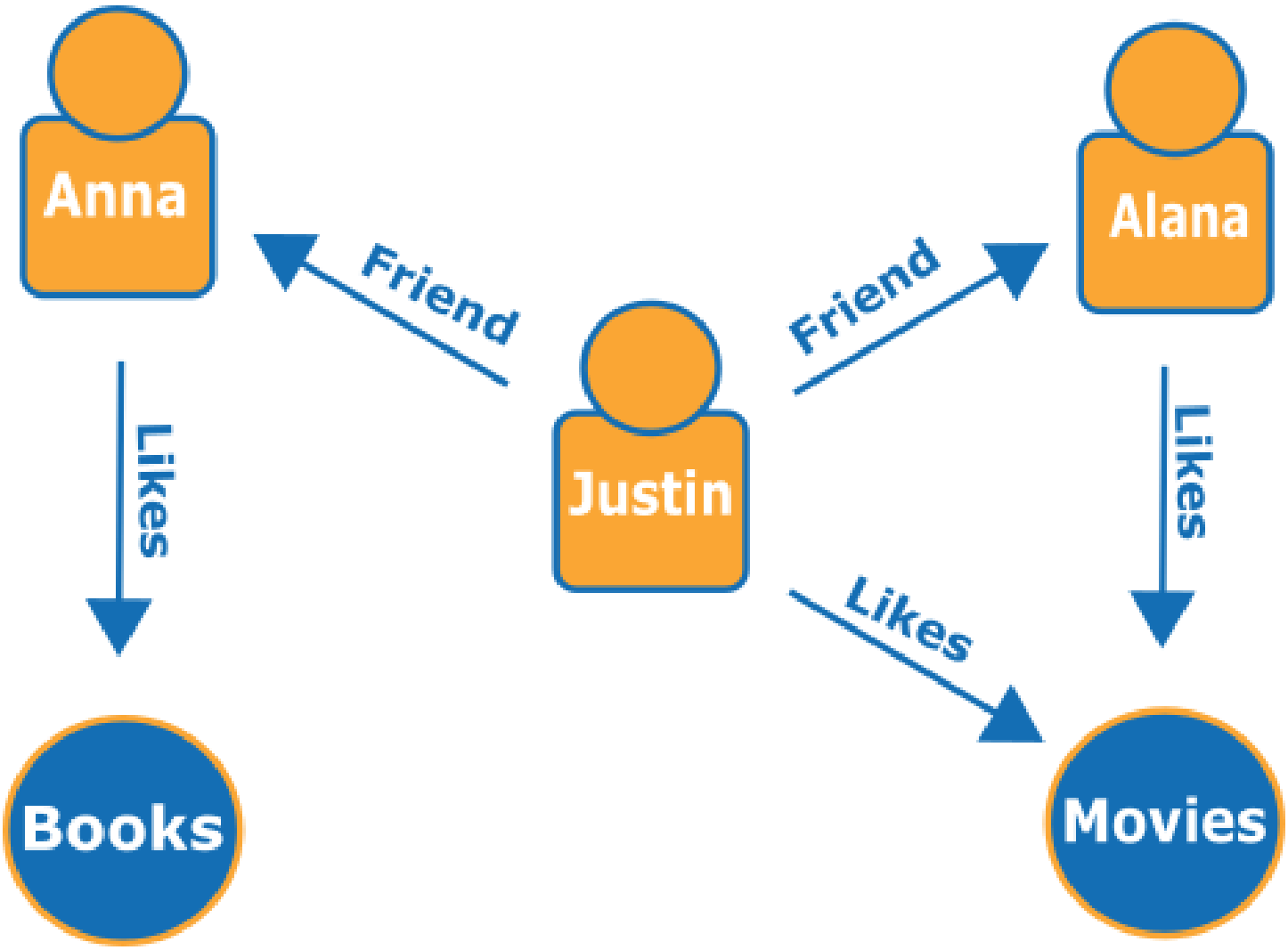
否則，在開始之前，您可能想要稍微了解圖形資料庫。

圖形資料庫到底是什麼？

圖形資料庫會進行最佳化，以儲存和查詢資料項目之間的關係。

它們會將資料項目本身儲存為圖形的頂點，並將它們之間的關係儲存為邊緣。每個邊緣都有一個類型，且會從一個頂點 (起點) 導向到另一個頂點 (終點)。關係可以稱為述詞以及邊緣，而頂點有時也稱為節點。在所謂的屬性圖中，頂點和邊緣也都可以具有與它們相關聯的額外屬性。

以下是代表社交網路中朋友和興趣的小圖：



邊緣顯示為具名箭頭，而頂點代表特定人物及其連結的興趣。

僅需簡單地周遊此圖形，即可了解 Justin 朋友的喜好。

為什麼要使用圖形資料庫？

每當實體間的連線或關係位於您嘗試要建立模型的資料核心時，圖形資料庫自然會成為您的心中首選。

首先，很容易將資料互連建成圖形模型，然後撰寫複雜的查詢，從圖形中擷取真實世界的資訊。

使用關聯式資料庫建置對等應用程式，需要您建立大量具有多個外部索引鍵的資料表，然後撰寫巢狀 SQL 查詢和複雜聯結。從編碼的角度來看，這種方法不僅迅速變得效率低下，而且隨著資料量的增加，其效能也會迅速下降。

相比之下，像 Neptune 這樣的圖形資料庫可以查詢數十億個頂點之間的關係，而不會陷入困境。

可以使用圖形資料庫做什麼？

圖形可以透過許多方式根據行為、所有權、家世、購買選擇、個人關係、家庭關係等表示真實世界實體的相互關係。

以下是一些使用圖形資料庫的最常見領域：

- 知識圖 – 知識圖可讓您組織和查詢各種有關聯的資訊來回答一般問題。使用知識圖，您可以將主題資訊新增至產品目錄，並為 Wikidata 中包含的各種資訊建立模型。

  若要進一步了解知識圖的運作方式以及在何處使用它們，請參閱 AWS 上的知識圖。

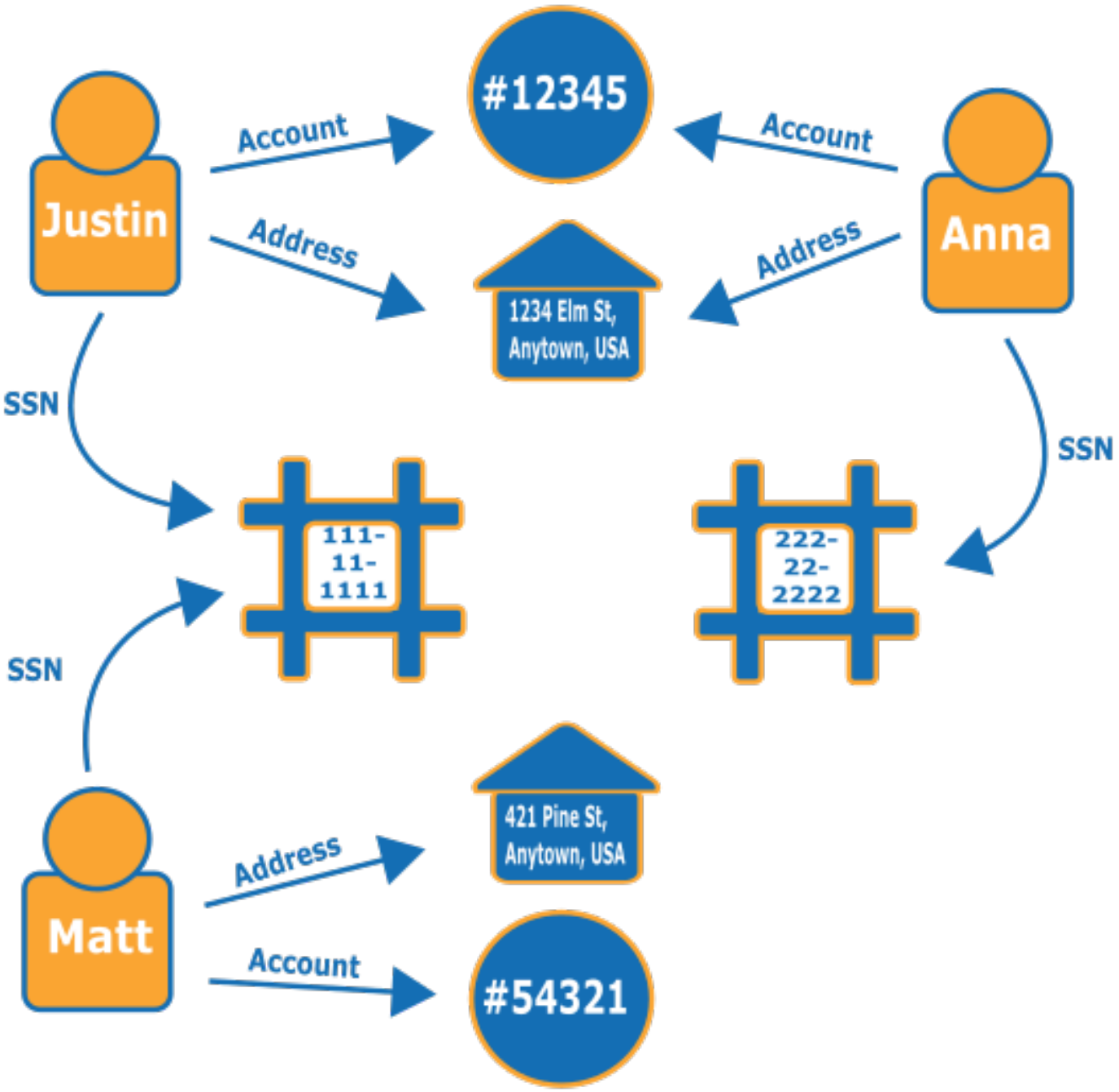
- 身分圖 – 在圖形資料庫中，您可以儲存資訊類別 (例如客戶興趣、朋友和購買歷史記錄) 之間的關係，然後查詢該資料以做出個人化且相關的建議。

  例如，您可以使用圖形資料庫，根據關注相同運動且擁有相似購買歷史記錄的其他人所購買的產品，向使用者推薦產品。或者，您可以找出有共同朋友但還不認識彼此的使用者，藉此推薦朋友人選。

  這種圖形被稱為身分圖，廣泛用於將與使用者的互動個人化。若要深入了解，請參閱 AWS 上的身分圖。若要開始建置自己的身分圖，您可以從使用 Amazon Neptune 的身分圖範例開始。

- 欺詐圖 – 這是圖形資料庫的常見用途。它們可以協助您追蹤信用卡購物和購買地點，以偵測異常使用情況，或偵測購買者是否正在嘗試使用與已知欺詐案件中相同的電子郵件地址和信用卡。它們可讓您檢查是否有多個人與個人電子郵件地址相關聯，或是否在不同實體位置有多個人共用相同 IP 地址。

  請考慮下列圖形。其會顯示三個使用者之間的關係，以及其身分的相關資訊。每位使用者都有一個地址、銀行帳戶與身分證號碼。不過，我們可以看見 Matt 和 Justin 共用相同的身分證號碼，這是非常不尋常的狀況，表示他們其中一個可能進行詐騙。對欺詐圖的查詢可以揭示此類關聯，以便可以對其進行審查。



若要進一步了解欺詐圖以及在何處使用它們，請參閱 AWS 上的欺詐圖。

- 社交網路 – 使用圖形資料庫的其中一個最常見領域，就是社交網路應用程式。

舉例來說，假設您想要在網站中建立社群摘要。您可以在後端輕鬆地使用圖形資料庫，將反映最新更新的結果提供給使用者，而這些更新來自家人、朋友、他們「喜歡」其更新的人，以及住在他們附近的人。

- 行車路線 – 鑒於目前交通情況和典型的交通模式，圖形可以協助您找到從起點到目的地的最佳路線。
- 物流 – 圖形可以協助識別最有效的方式來使用可用的運輸和配銷資源，以滿足客戶的需求。
- 診斷 – 圖形可以代表複雜的診斷樹，您可以查詢這些診斷樹，來識別觀察到的問題和失敗來源。
- 科學研究 – 使用圖形資料庫，您可以建置應用程式，使用靜態加密來儲存和導覽科學資料，甚至是敏感的醫療資訊。例如，您可以儲存疾病和基因互動的模型。您可以搜尋蛋白質化學反應的圖形模式，尋找與疾病相關聯的其他基因。您也能夠建立化合物圖形模型，並在分子結構中查詢模式。您可以將來自不同系統中醫療記錄的患者資料建立關聯。您可以依主題整理研究期刊，快速尋找相關資訊。
- 法規 – 您可以將複雜的法規需求儲存為圖形，並查詢它們，以偵測這些需求可能適用於您日常業務營運的情況。
- 網路拓撲與事件 – 圖形資料庫可以協助您管理和保護 IT 網路。將網路拓撲儲存為圖形時，您也可以在網路上儲存和處理許多不同種類的事件。您可以回答問題，例如有多少主機正在執行特定的應用程式。您可以查詢可能顯示特定主機已遭惡意程式入侵的模式，並查詢連線資料，而這些資料可協助追蹤程式至下載該程式的原始主機。

如何查詢圖形？

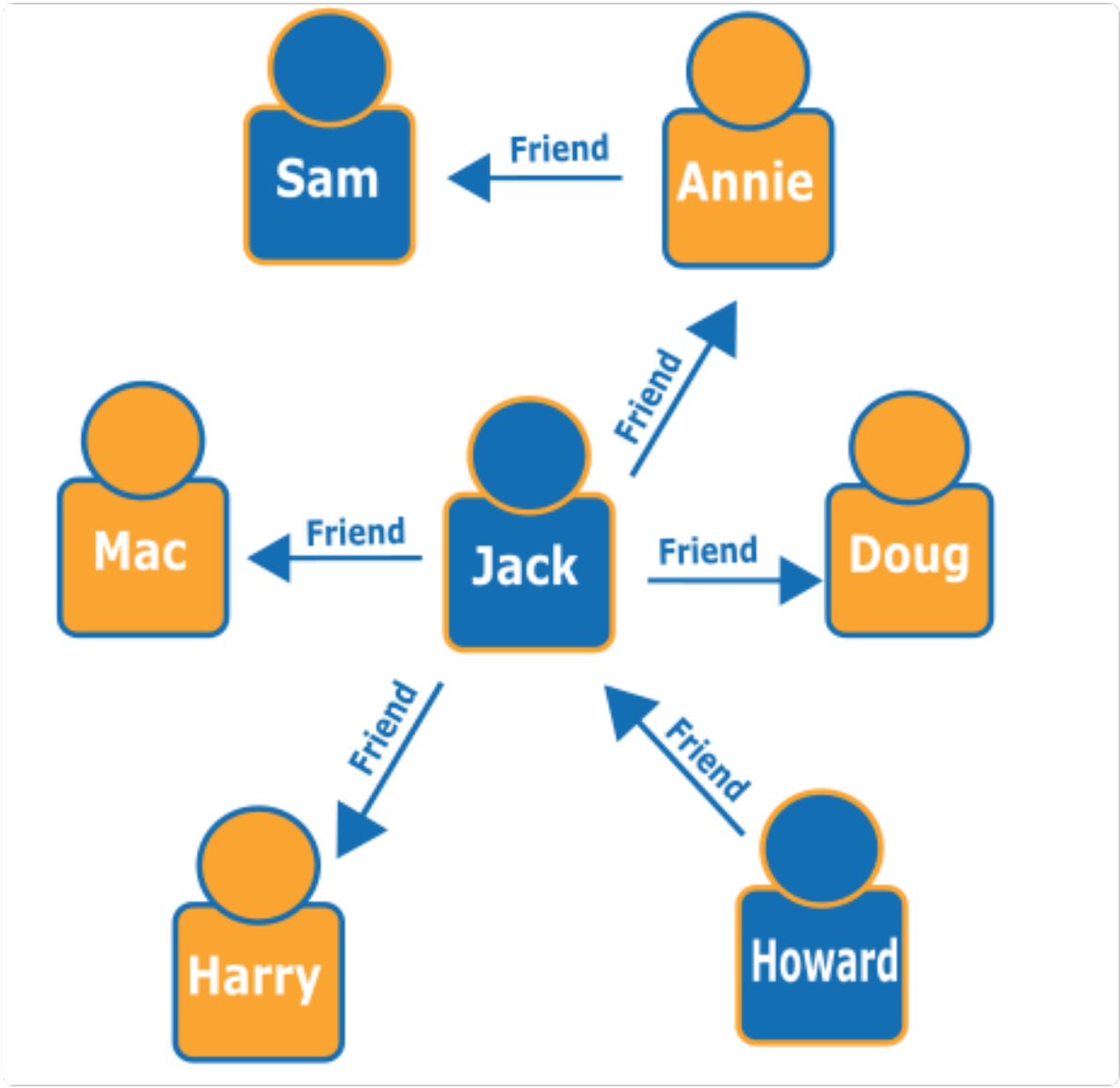
Neptune 支援三種特殊用途的查詢語言，這些語言是專為查詢不同種類的圖形資料而設計。您可以使用下列語言來新增、修改、刪除和查詢 Neptune 圖形資料庫中的資料：

- Gremlin 是一種適用於屬性圖的圖形周遊語言。Grimlin 中的查詢是由離散步驟組成的周遊，每個步驟都沿著一個邊緣到一個節點。如需詳細資訊，請參閱 Apache TinkerPop3 上的 Gremlin 文件。

  Gremlin 的 Neptune 實作與其他實作有一些不同，尤其是使用 Gremlin-Groovy 時 (以序列化的文字傳送的 Gremlin 查詢)。如需更多詳細資訊，請參閱 Amazon Neptune 中的 Gremlin 標準合規。
- openCypher – OpenCypher 是屬性圖的宣告式查詢語言，最初由 Neo4j 開發，然後在 2015 年成為開放原始碼，並在 Apache 2 開放原始碼授權下投入 OpenCypher 專案。如需語言規格，請參閱 Cypher 查詢語言參考 (第 9 版)，以及如需額外資訊，請參閱 Cypher 樣式指南。
- SPARQL 是 RDF 資料的宣告式查詢語言，以全球資訊網協會 (W3C) 制定的標準圖形模式配對為基礎，於 SPARQL 1.1 概觀和 SPARQL 1.1 查詢語言規格中描述。如需 SPARQL 的 Neptune 實作的特定詳細資訊，請參閱 Amazon Neptune 中的 SPARQL 標準合規。

比對 Gremlin 和 SPARQL 查詢的範例

根據以下的人物 (節點) 與其關係 (邊緣) 圖形，您可以找出特定對象的「朋友的朋友」，例如，Howard 朋友的朋友。



您可以從圖中發現 Howard 有一位朋友 Jack，而 Jack 則有四位朋友：Annie、Harry、Doug 和 Mac。此處採用的簡單範例是搭配簡易圖形使用，但您可以依複雜度、資料集大小與結果大小來擴大這些類型的查詢規模。

以下是進行 Gremlin 周遊查詢後，系統搜尋 Howard 朋友的朋友所傳回的姓名結果：

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

以下是進行 SPARQL 查詢後，系統搜尋 Howard 朋友的朋友所傳回的姓名結果：

```
prefix : <#>

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

> **Note**
>
> 任何資源描述架構 (RDF) 三元組的每個部分都具備與其相關的 URI。在上述範例中，刻意精簡了 URI 字首。

參加使用 Amazon Neptune 的線上課程

如果您喜歡使用影片學習，AWS 會在 AWS Online Tech Talks 中提供線上課程，以協助您開始學習。介紹圖形資料庫的線上課程如下：

使用 Amazon Neptune 的圖形資料庫簡介、深入探討和示範。

深入挖掘圖形參考架構

當您考慮圖形資料庫可以為您解決哪些問題，以及如何處理這些問題時，最好的起點之一就是 Neptune 圖形參考架構 GitHub 專案。

您可以在那裡找到圖形工作負載類型的詳細描述，以及三個章節，協助您設計有效的圖形資料庫：

- 資料模型和查詢語言 – 本節會為您逐步解說 Gremlin 和 SPARQL 之間的差異，以及如何在它們之間進行選擇。
- 圖形資料模型 – 這是如何做出圖形資料建模決策的全面討論，包括使用 Gremlin 進行屬性圖建模的詳細逐步解說，以及使用 SPARQL 進行 RDF 建模。
- 將其他資料模型轉換為圖形模型 – 在這裡您可以了解如何將關係資料模型轉換成圖形模型。

還有三個章節為您逐步解說使用 Neptune 的特定步驟：

- 從 Neptune VPC 外部的用戶端連線至 Amazon Neptune – 本節顯示從資料庫叢集所在的 VPC 外部連線到 Neptune 的幾個選項。
- 從 AWS Lambda 函數存取 Amazon Neptune – 在這裡您將了解如何從 Lambda 函數可靠地連線到 Neptune。
- 從 Amazon Kinesis Data Streams 寫入 Amazon Neptune – 本節可協助您使用 Neptune 處理高寫入輸送量案例。

使用 Neptune 圖形筆記本快速開始

您不必使用 Neptune 圖形筆記本來處理 Neptune 圖形，因此，如果您想要的話，可以繼續使用 AWS CloudFormation 範本立即建立新的 Neptune 資料庫。

同時，無論您是圖形新手且想要學習和實驗，還是有經驗且想要完善查詢，Neptune 工作台都會提供一個互動式開發環境 (IDE)，可在您建置圖形應用程式時提高生產力。

Neptune 在開源 Neptune 圖形筆記本項目和 JupyterLabNeptune 工作台中提供 Jupyter 和筆記本。GitHub這些筆記本會在互動式編碼環境中提供範例應用程式教學課程和程式碼片段，您可以在此了解圖形技術和 Neptune。您可以使用它們搭配不同的查詢語言、不同的資料集，甚至是後端的不同資料庫，來逐步完成設定、填入和查詢圖形。

您可以透過數種不同的方式託管這些筆記本：

- Neptune 工作台可讓您在完全受管的環境中執行 Jupyter 筆記本 (託管於 Amazon) SageMaker，並自動為您載入最新版本的 Neptune 圖形筆記本專案。建立新的 Neptune 資料庫時，可以在 Neptune 主控台中輕鬆設定工作台。

  > **Note**
  >
  > 建立 Neptune 筆記本執行個體時，會提供兩種網路存取選項：透過 Amazon 直接存取 SageMaker (預設值) 和透過 VPC 存取。在任一選項中，筆記本都需要訪問互聯網以獲取安裝 Neptune 工作台的軟件包依賴關係。缺乏互聯網訪問將導致 Neptune 筆記本實例的創建失敗。

- 您也可以在本機安裝 Jupyter。這可讓您從筆記型電腦執行筆記本，而此筆記型電腦已連線至 Neptune 或其中一個開放原始碼圖形資料庫的本機執行個體。在後一種情況下，您可以在花一分錢

之前盡可能多嘗試圖形技術。然後，當您準備就緒時，就可以順利移至 Neptune 提供的受管生產環境。

使用 Neptune 工作台託管 Neptune 筆記本

Neptune 提供 T3 和 T4g 執行個體類型，可讓您以每小時不到 $0.10 開始使用這些執行個體類型。您需要透過 Amazon 支付工作台資源的費用 SageMaker，與您的 Neptune 帳單分開。請參閱 Neptune 定價頁面。在 Neptune 工作台上創建的 Jupyter 和 JupyterLab 筆記本全部使用 Amazon Linux 2 和 JupyterLab 3 環境。如需 JupyterLab 筆記型電腦支援的詳細資訊，請參閱 Amazon SageMaker 文件。

您可以使用以下兩種方式之一，使用 Neptune 工作台建立 Jupyter 或 JupyterLab 筆記本： AWS Management Console

- 建立新的 Neptune 資料庫叢集時，請使用筆記本組態功能表。若要執行此操作，請遵循 使用 AWS Management Console 啟動 Neptune 資料庫叢集 中概述的步驟。
- 在建立了資料庫叢集之後，請使用左側導覽窗格中的筆記本功能表。若要執行此操作，請按以下步驟進行。

使用記事本功能表建立 Jupyter 或 JupyterLab 記事本

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在左側的導覽窗格中，選擇 Notebooks (筆記本)。
3. 選擇建立筆記本。
4. 在叢集清單中，選擇您的 Neptune 資料庫叢集。如果還沒有資料庫叢集，請選擇 Create cluster (建立叢集) 以建立叢集。
5. 選取筆記本執行個體類型。
6. 為您的筆記本提供名稱，並選擇性提供描述。
7. 除非您已經為筆記本建立 AWS Identity and Access Management (IAM) 角色，否則請選擇 [建立 IAM 角色]，然後輸入 IAM 角色名稱。

> **Note**
>
> 如果您確實選擇重複使用已針對先前筆記本建立的 IAM 角色，則角色政策必須包含正確的許可，才能存取您正在使用的 Neptune 資料庫叢集。您可以透過檢查 `neptune-db:*`

動作下資源 ARN 中的元件是否符合該叢集來驗證此情況。當您嘗試執行筆記本魔術命令時，未正確設定的許可會導致連線錯誤。

8. 選擇建立筆記本。建立程序可能需要 5 到 10 分鐘，一切才能準備就緒。
9. 建立記事本之後，請選取它，然後選擇 [開啟 Jupyter] 或 [開啟]。 JupyterLab

主控台可以為您的筆記本建立 AWS Identity and Access Management (IAM) 角色，您也可以自行建立。此角色的政策應包括下列各項：

```
{
  "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)/*"
      ]
    }
  ]
}
```

請注意，上述政策中的第二個聲明會列出一或多個 Neptune 叢集資源 ID。

此外，角色也應該建立下列信任關係：

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

```
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

同樣，準備好一切可能需要 5 到 10 分鐘。

您可以將新筆記本設定為使用 Neptune ML，如 為 Neptune ML 手動設定 Neptune 筆記本 中所述。

使用 Python 將通用 SageMaker 筆記本連接到 Neptune

如果您已經安裝了 Neptune 魔術，將筆記本連接到 Neptune 很容易，但是即使您沒有使用 Neptune SageMaker 筆記本，也可以使用 Python 將筆記本連接到 Neptune。

在 SageMaker 筆記本電腦中連接到 Neptune 的步驟

1. 安裝 Gremlin Python 用戶端：

   ```
   !pip install gremlinpython
   ```

   Neptune 筆記本為您安裝 Gemlin Python 客戶端，因此只有在您使用普通 SageMaker 筆記本時，才需要執行此步驟。

2. 撰寫如下的程式碼來連線並發出 Gemlin 查詢：

   ```
   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**
>
> 如果您碰巧使用的 Gremlin Python 用戶端版本早於 3.5.0，則此行：
>
> ```
> connection = DriverRemoteConnection(endpoint,'g',
>
>  transport_factory=lambda:AiohttpTransport(call_from_event_loop=True))
> ```
>
> 只會是：
>
> ```
> connection = DriverRemoteConnection(endpoint,'g')
> ```

啟用 CloudWatch Neptune 筆記型電腦上

CloudWatch Neptune 筆記本現在預設為啟用記錄。如果您的筆記型電腦沒有產生記 CloudWatch 錄檔，請依照下列步驟手動啟用：

1. 登入 AWS Management Console 並開啟SageMaker 主控台。

2. 在左側的導覽窗格中，選擇筆記本，然後選擇筆記本執行個體。尋找您要為其啟用日誌的 Neptune 筆記本名稱。
3. 選取該筆記本執行個體的名稱，以前往詳細資訊頁面。
4. 如果筆記本執行個體正在執行，請選取筆記本詳細資料頁面右上方的停止按鈕。
5. 在許可和加密下，IAM 角色 ARN 有一個欄位。選取此欄位中的連結，即可前往此筆記本執行個體搭配執行的 IAM 角色。
6. 建立下列政策：

```
{
  "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. 儲存此新政策並將其附加至步驟 4 中找到的 IAM 角色。
8. 按一下 SageMaker 筆記本執行個體詳細資訊頁面右上角的 [開始]。
9. 當日誌開始流動時，您應該會在詳細資料頁面的筆記本執行個體設定區段左下方附近，於標示為生命週期組態的欄位下方看到檢視日誌連結。

如果筆記型電腦無法啟動， SageMaker 主控台的筆記型電腦詳細資料頁面中會出現訊息，指出筆記型電腦執行個體需要 5 分鐘以上才能啟動。 CloudWatch 您可以在此名稱下找到與此問題相關的記錄檔：

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

在您的本機電腦上設定圖形筆記本

圖形筆記本專案具有在本機電腦上設定 Neptune 筆記本的指示：

- 先決條件
- 重複和安裝 JupyterLab
- 連線至圖形資料庫。

您可以將本機筆記本連線到 Neptune 資料庫叢集，或連線至開放原始碼圖形資料庫的本機或遠端執行個體。

使用 Neptune 筆記本搭配 Neptune 叢集

如果您要連接到後端的 Neptune 叢集，則可能需要在 Amazon 中執行筆記本電腦 SageMaker。從連接到 Neptune SageMaker 可以比從本地安裝筆記本電腦更方便，它可以讓您更輕鬆地使用 Neptune ML 進行工作。

如需有關如何在中設定筆記本的指示 SageMaker，請參閱使用 Amazon 啟動圖形筆記本。 SageMaker

如需設定 Neptune 本身的相關指示，請參閱 設定 Neptune。

您也可以將 Neptune 筆記本的本機安裝連線到 Neptune 資料庫叢集。這種情況可能會稍微複雜一些，因為 Amazon Neptune 資料庫叢集只能在 Amazon Virtual Private Cloud (VPC) 中建立，而此 VPC 是專為與外界隔離而設計的。有多種方法可從外部連線到 VPC。其中一種為使用負載平衡器。另一種為使用 VPC 互連 (請參閱 Amazon Virtual Private Cloud 互連指南)。

不過，對於大多數人來說，最方便的方法是連線以在 VPC 內設定 Amazon EC2 代理伺服器，然後使用 SSH 隧道 (也稱為連接埠轉發) 連線至其中。您可以在圖形筆記本專`additional-databases/neptune`案的資料夾中將圖形筆記本本機連接到 Amazon Neptune 中找到有關如何設定的指示。 GitHub

使用 Neptune 筆記本搭配開放原始碼圖形資料庫

若要免費開始使用圖形技術，您也可以使用 Neptune 筆記型電腦搭配後端的各種開放原始碼資料庫。例如 TinkerPop Girmlin 伺服器和火焰圖資料庫。

若要使用 Girmlin 伺服器作為後端資料庫，請遵循下列資料夾中的指示：

- 將圖形筆記本連接到 Gemlin 服務器文件夾。 GitHub
- 圖形筆記本電腦的小鬼配置文件夾。 GitHub

若要使用 Blazegraph 的本機執行個體做為後端資料庫，請遵循下列指示：

- Blazegraph 快速入門指示
- 圖形筆記本電腦的配置文件夾。 GitHub

將您的 Neptune 筆記本從 Jupyter 遷移到 3 JupyterLab

在 2022 年 12 月 21 日之前建立的 Neptune 筆記本使用 Amazon Linux 1 環境。您可以透過執行以下 AWS 部落格文章中所述的步驟，將在該日期之前建立的舊版 Jupyter 筆記本移轉到新的 Amazon Linux 2 環境 (使 JupyterLab 用 Amazon Linux 2) 將您的工作移轉到 Amazon SageMaker 筆記本執行個體 (使用 Amazon Linux 2)。

此外，還有一些專門適用於將 Neptune 筆記本遷移到新環境的步驟：

Neptune 特定的先決條件

在來源 Neptune 筆記本的 IAM 角色中，新增下列所有許可：

```
{
  "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": [
```

```
    "*"
  ]
}
```

請務必針對將用於備份的 S3 儲存貯體指定正確的 ARN。

Neptune 特定的生命週期組態

如部落格文章所述，建立第二個生命週期組態指令碼來還原備份 (從 `on-create.sh`) 時，生命週期名稱必須遵循 `aws-neptune-*` 格式，例如 `aws-neptune-sync-from-s3`。這會確保在 Neptune 主控台中建立筆記本期間可以選取 LCC。

從快照到新執行個體的 Neptune 特定同步

在部落格文章中所述從快照同步至新執行個體的步驟中，以下是 Neptune 特定的變更：

- 在步驟 4 上，選擇 notebook-al2-v2。
- 在步驟 5 上，重複使用來源 Neptune 筆記本中的 IAM 角色。
- 在步驟 7 與 8 之間：
  - 在筆記本執行個體設定中，設定使用 `aws-neptune-*` 格式的名稱。
  - 開啟網路設定摺疊式功能表，然後選取與來源筆記本中相同的 VPC、子網路和安全群組。

在建立了新筆記本之後的 Neptune 特定步驟

1. 為筆記本選取開啟 Jupyter 按鈕。一旦 `SYNC_COMPLETE` 檔案顯示在主目錄中，就會繼續執行下一個步驟。
2. 前往主控台中的筆記本執行個體頁 SageMaker 面。
3. 停止筆記本。
4. 選擇 Edit (編輯)。
5. 在筆記本執行個體設定中，選取來源 Neptune 筆記本的原始生命週期，以編輯生命週期組態欄位。請注意，這不是 EBS 備份生命週期。
6. 選擇更新筆記本設定。
7. 重新啟動筆記本。

透過部落格文章中所述步驟的修改，您的圖形筆記本現在應該移轉到使用 Amazon Linux 2 和 JupyterLab 3 環境的新 Neptune 筆記本執行個體上。它們會在中的 Neptune 頁面上顯示以供存取和管

理 AWS Management Console，您現在可以選取 [開啟 Jupyter] 或 [開啟]，繼續從中斷的地方繼續工作。 JupyterLab

在筆記本中使用 Neptune 工作台魔法

Neptune 工作台會在筆記本中提供多個所謂的「魔法」命令，這些命令可以節省大量的時間和精力。它們分為兩類：「行魔法」和「儲存格魔法」。

「行魔法」是前面有單一的百分比符號 (%) 的命令。它們只接受行輸入，而不是來自儲存格本文其餘部分的輸入。Neptune 工作台提供以下行魔法：

- %seed
- %load
- %load_ids
- %load_status
- %cancel_load
- %status
- %gremlin_status
- %opencypher_status 或 %oc_status
- %stream_viewer
- %sparql_status
- %graph_notebook_config
- %graph_notebook_host
- %graph_notebook_version
- %graph_notebook_vis_options
- %statistics
- %summary

「儲存格魔法」前面有兩個百分比符號 (%%) 而不是一個，並使用儲存格內容作為輸入，然而它們也可以接受行內容作為輸入。Neptune 工作台提供以下儲存格魔法：

- %%sparql
- %%gremlin

- %%opencypher, or %%oc
- %%graph_notebook_config
- %%graph_notebook_vis_options

還有兩種魔法 (一種行魔法和一種儲存格魔法)，用於使用 Neptune 機器學習：

- %neptune_ml
- %%neptune_ml

> **Note**
>
> 使用 Neptune 魔法時，您通常可以使用 `--help` 或 `-h` 參數來取得說明文字。使用儲存格魔法，本文不能是空的，因此在取得說明時，將填充文字甚至單一字元放在本文中。例如：
>
> ```
> %%gremlin --help
>   x
> ```

儲存格或行魔法中的變數注入

可在筆記本中的任何儲存格或行魔法內使用下列格式，來參考筆記本中定義的變數：`${VAR_NAME}`。

例如，假設您定義了這些變數：

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

然後，儲存格魔法中的這個 Gremlin 查詢：

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

相當於下列查詢：

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

使用所有查詢語言的查詢引數

下列查詢引數會在 Neptune 工作台中使用 %%gremlin、%%opencypher 和 %%sparql 魔法：

常見查詢引數

- **--store-to** (或 **-s**) – 指定要在其中儲存查詢結果的變數名稱。
- **--silent** – 如果存在，查詢完成後不會顯示任何輸出。

- **--group-by** (或 **-g**) – 指定用來將節點分組的屬性 (例如 code 或 T.region)。頂點會根據其指派的群組著色。
- **--ignore-groups** – 如果存在，則忽略所有分組選項。
- **--display-property** (或 **-d**) – 指定應為每個頂點顯示其值的屬性。

  每種查詢語言的預設值如下：

  - 對於 Gremlin：T.label。
  - 對於 openCypher：~labels。
  - 對於 SPARQL：type。
- **--edge-display-property** (或 **-t**) – 指定應為每個邊緣顯示其值的屬性。

  每種查詢語言的預設值如下：

  - 對於 Gremlin：T.label。
  - 對於 openCypher：~labels。
  - 對於 SPARQL：type。
- **--tooltip-property** (或 **-de**) – 指定其值應顯示為每個節點之工具提示的屬性。

  每種查詢語言的預設值如下：

  - 對於 Gremlin：T.label。
  - 對於 openCypher：~labels。
  - 對於 SPARQL：type。
- **--edge-tooltip-property** (或 **-te**) – 指定其值應顯示為每個邊緣之工具提示的屬性。

  每種查詢語言的預設值如下：

  - 對於 Gremlin：T.label。
  - 對於 openCypher：~labels。

  - 對於 SPARQL：`type`。
- **`--label-max-length`** (或 **`-l`**) – 指定任何頂點標籤的字元長度上限。預設為 10。
- **`--edge-label-max-length`** (或 **`-le`**) – 指定任何邊緣標籤的字元長度上限。預設為 10。

  僅在 openCypher 的情況下，這會改為 `--rel-label-max-length` 或 `-rel`。
- **`--simulation-duration`** (或 **`-sd`**) – 指定視覺化物理模擬的持續時間上限。預設為 1500 毫秒。
- **`--stop-physics`** (或 **`-sp`**) – 在初始模擬穩定之後停用視覺化物理。

這些引數的屬性值可以由單一屬性索引鍵組成，或由可以為每個標籤類型指定不同屬性的 JSON 字串組成。只能使用變數注入來指定 JSON 字串。

`%seed` 行魔法

`%seed` 行魔法是將資料新增至 Neptune 端點的便利方式，您可以使用此方式，來探索和嘗試 Gremlin、OpenCypher 或 SPARQL 查詢。它會提供一個表單，您可以在其中選擇要瀏覽的資料模型 (屬性圖或 RDF)，然後從 Neptune 提供的許多不同範例資料集進行選擇。

`%load` 行魔法

`%load` 行魔法會產生一個表單，您可以使用此表單，將大量載入請求提交至 Neptune (請參閱 Neptune 載入器命令)。此來源必須是 Amazon S3 路徑，與 Neptune 叢集位在同一個區域中。

`%load_ids` 行魔法

`%load_ids` 行魔法會擷取已提交至筆記本主機端點的載入 ID (請參閱 Neptune 載入器 Get-Status 請求參數)。此請求採用以下形式：

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

`%load_status` 行魔法

`%load_status` 行魔法會擷取已提交至筆記本主機端點之特定載入工作的載入狀態，此狀態是由行輸入指定 (請參閱 Neptune 載入器 Get-Status 請求參數)。此請求採用以下形式：

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

行魔法看起來像這樣：

```
%load_status load id
```

%cancel_load 行魔法

%cancel_load 行魔法會取消特定的載入操作 (請參閱 Neptune 載入器取消工作)。此請求採用以下形式：

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

行魔法看起來像這樣：

```
%cancel_load load id
```

%status 行魔法

從筆記本的主機端點擷取狀態資訊 (%graph_notebook_config 會顯示主機端點)。

%gremlin_status 行魔法

擷取 Gremlin 查詢狀態資訊。

%opencypher_status 行魔法 (也是 %oc_status)

擷取 opencypher 查詢的查詢狀態。此行魔法會採用以下選用引數：

- **--queryId** 或 **-q** – 指定要為其顯示狀態之特定執行中查詢的 ID。
- **--cancel_query** 或 **-c** – 取消執行中查詢。不需要一個值。
- **--silent** 或 **-s** – 如果取消查詢時將 --silent 設為 true，則會取消執行中查詢，HTTP 回應代碼為 200。否則，HTTP 回應代碼將是 500。
- **--store-to** – 指定要在其中儲存查詢結果的變數名稱。

%sparql_status 行魔法

擷取 SPARQL 查詢狀態資訊。

%stream_viewer 行魔法

如果在 Neptune 叢集上啟用串流，%stream_viewer 行魔法會顯示一個介面，允許以互動方式探索 Neptune 串流中記錄的項目。其會接受以下選用引數：

- **language** – 串流資料的查詢語言：gremlin 或 sparql。如果您未提供此引數，則預設值為 gremlin。
- **--limit** – 指定每頁要顯示的串流項目數量上限。如果您未提供此引數，則預設值為 10。

> **Note**
>
> 只在引擎 1.0.5.1 版及更舊版本上才會完全支援 %stream_viewer 行魔法。

%graph_notebook_config 行魔法

此行魔法會顯示 JSON 物件，其中包含筆記本用來與 Neptune 通訊的組態。組態包含：

- host：要連線至其中並向其發出命令的端點。
- port：向 Neptune 發出命令時使用的連接埠。預設值為 8182。
- auth_mode：向 Neptune 發出命令時使用的身分驗證模式。如果連線至已啟用 IAM 身分驗證的叢集，則必須是 IAM，否則為 DEFAULT。
- load_from_s3_arn：為要使用的 %load 魔法指定 Amazon S3 ARN。如果此值為空的，則必須在 %load 命令中指定 ARN。
- ssl：指示是否要使用 TLS 連線到 Neptune 的布林值。預設值為 true。
- aws_region：部署此筆記本的區域。此資訊用於 IAM 身分驗證和 %load 請求。

您可以透過將 %graph_notebook_config 輸出複製到新的儲存格來變更組態，然後在該處對其進行變更。然後，如果您在新儲存格上執行 %%graph_notebook_config 儲存格魔法，則組態將相應地變更。

%graph_notebook_host 行魔法

將行輸入設為筆記本的主機。

%graph_notebook_version 行魔法

%graph_notebook_version 行魔法會傳回 Neptune 工作台筆記本版本編號。例如，已在版本 1.27 中引入圖形視覺化。

%graph_notebook_vis_options 行魔法

%graph_notebook_vis_options 行魔法會顯示筆記本目前使用的視覺化設定。這些選項會在 vis.js 文件中加以說明。

您可以修改這些設定，方法是將輸出複製到新的儲存格、進行所需的變更，然後在儲存格上執行 %%graph_notebook_vis_options 儲存格魔法。

若要將視覺化設定還原為其預設值，您可以搭配 reset 參數執行 %graph_notebook_vis_options 行魔法。這會重設所有視覺化設定：

```
%graph_notebook_vis_options reset
```

%statistics 行魔法

%statistics 行魔法用來擷取或管理 DFE 引擎統計資料 (請參閱 管理要供 Neptune DFE 使用的統計資料)。這種魔法也可以用來擷取圖形摘要。

它接受下列參數：

- **--language** – 統計資料端點的查詢語言：propertygraph (或 pg) 或 rdf。

  如果未提供，則預設值為 propertygraph。
- **--mode** (或 **-m**) – 指定要提交的請求或動作類型：其中一個 status、disableAutoCompute、enableAutoCompute、refresh、delete、detailed 或 basic。

  如果未提供，則預設值為 status，除非指定 --summary，在此情況下，預設值為 basic。
- **--summary** – 從所選語言的統計資料摘要端點擷取圖形摘要。
- **--silent** – 如果存在，查詢完成後不會顯示任何輸出。
- **--store-to** – 用來指定要儲存查詢結果的變數。

%summary 行魔法

%summary 行魔法用來擷取圖形摘要資訊。從 Neptune 引擎版本 1.2.1.0 開始，可以使用它。

它接受下列參數：

- **--language** – 統計資料端點的查詢語言：propertygraph (或 pg) 或 rdf。

  如果未提供，則預設值為 propertygraph。
- **--detailed** – 在輸出中開啟或關閉結構欄位的顯示。

  如果未提供，預設為 basic 摘要顯示模式。
- **--silent** – 如果存在，查詢完成後不會顯示任何輸出。
- **--store-to** – 用來指定要儲存查詢結果的變數。

%%graph_notebook_config 儲存格魔法

%%graph_notebook_config 儲存格魔法會使用包含組態資訊的 JSON 物件，來修改筆記本用來與 Neptune 通訊的設定，如果可能的話。組態會採用 %graph_notebook_config 行魔法傳回的相同形式。

例如：

```
%%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"
}
```

%%sparql 儲存格魔法

%%sparql 儲存格魔法會向 Neptune 端點發出 SPARQL 查詢。它會接受以下選用的行輸入：

- **-h** 或 **--help** – 傳回有關這些參數的說明文字。
- **--path** – 在 SPARQL 端點的路徑前面加上字首。例如，如果您指定 --path "abc/def"，則呼叫的端點將是 *host*:*port*/abc/def。

- **`--expand-all`** – 這是一個查詢視覺化提示，其會告訴視覺化工具在圖形圖表中包含所有 `?s ?p ?o` 結果，而不論繫結類型為何。

  根據預設，SPARQL 視覺化僅包含三重模式，其中 `o?` 為 `uri` 或 bnode (空白節點)。所有其他 `?o` 繫結類型 (例如常值字串或整數) 都會被視為 `?s` 節點的屬性，您可以使用圖形索引標籤中的詳細資訊窗格檢視這些屬性。

  當您想要改在視覺化中包含這類常值作為頂點時，請使用 `--expand-all` 查詢提示。

  請不要將此視覺化提示與 explain 參數結合，因為 explain 查詢不會進行視覺化。
- **`--explain-type`** – 用來指定要使用的 explain 模式 (其中一個:`dynamic`、`static` 或 `details`)。
- **`--explain-format`** – 用來指定 explain 查詢的回應格式 (`text/csv` 或 `text/html` 之一)。
- --store-to – 用來指定要儲存查詢結果的變數。

`explain` 查詢的範例：

```
%%sparql explain

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

具有 `--expand-all` 視覺化提示參數的視覺化查詢範例 (請參閱 SPARQL 視覺化)：

```
%%sparql --expand-all

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

`%%gremlin` 儲存格魔法

`%%gremlin`細胞魔術發出一個 Grimlin 查詢到 Neptune 端點使用. WebSocket 它接受要切換至 Gremlin `explain` /> 模式或 Gremlin `profile` API 的選用行輸入，以及要修改視覺化輸出行為的個別選用視覺化提示 (請參閱 Gremlin 視覺化)。

`explain` 查詢的範例：

```
%%gremlin explain

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

profile 查詢的範例：

```
%%gremlin profile

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

具有視覺化提示參數的視覺化查詢範例：

```
%%gremlin -p v,outv

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

%%gremlin profile 查詢的選用參數

- **--chop** – 指定設定檔結果字串的長度上限。如果您未提供此引數，則預設值為 250。
- **--serializer** – 指定要結果的序列化程式。允許的值是任何有效的 MIME 類型或 TinkerPop 驅動程序「序列化器」枚舉值。如果您未提供此引數，則預設值為 application.json。
- **--no-results** – 僅顯示結果計數。如果未使用，則根據預設，所有查詢結果都會顯示在設定檔報告中。
- **--indexOps** – 顯示所有索引操作的詳細報告。

%%opencypher 儲存格魔法 (也是 %%oc)

%%opencypher 儲存格魔法 (縮寫形式為 %%oc) 會向 Neptune 端點發出 OpenCypher 查詢。它會接受以下選用的行輸入引數：

- mode – 查詢模式：query 或 bolt。如果您未提供此引數，則預設值為 query。
- **--group-by** 或 **-g** – 指定用來將節點分組的屬性。例如 code, ~id。如果您未提供此引數，則預設值為 ~labels。
- **--ignore-groups** – 如果存在，則忽略所有分組選項。
- **--display-propery** 或 **-d** – 指定應為每個頂點顯示其值的屬性。如果您未提供此引數，則預設值為 ~labels。
- **--edge-display-propery** 或 **-de** – 指定應為每個邊緣顯示其值的屬性。如果您未提供此引數，則預設值為 ~labels。
- **--label-max-length** 或 **-l** – 指定要顯示之頂點標籤的字元數上限。如果您未提供此引數，則預設值為 10。

- **--store-to** 或 **-s** – 指定要在其中儲存查詢結果的變數名稱。
- **--plan-cache** 或 **-pc** – 指定要使用的計畫快取模式。預設值為auto。 (* 計劃快取僅適用於 Neptune 分析)
- **--query-timeout** 或 **-qt** – 指定查詢逾時上限 (以毫秒為單位)。預設值為 1800000。
- **--query-parameters** 或 **qp** – 要套用至查詢的參數定義。此選項可以接受單一變數名稱，或可接受映射的字串表示法。

  --query-parameters 使用範例

  1. 在一個筆記本儲存格中定義 OpenCypher 參數的映射。

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

  2. 透過 %%oc 將參數傳遞到另一個儲存格中的 --query-parameters。

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

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

- --explain in-type — 用於指定要使用的解釋模式（以下其中一種：動態，靜態或詳細信息）。

%%graph_notebook_vis_options 儲存格魔法

%%graph_notebook_vis_options 儲存格魔法可讓您設定筆記本的視覺化選項。您可以將 %graph-notebook-vis-options 行魔法傳回的設定複製到新的儲存格、對其進行更改，然後使用 %%graph_notebook_vis_options 儲存格魔法設定新值。

這些選項會在 vis.js 文件中加以說明。

若要將視覺化設定還原為其預設值，您可以搭配 reset 參數執行 %graph_notebook_vis_options 行魔法。這會重設所有視覺化設定：

```
%graph_notebook_vis_options reset
```

%neptune_ml 行魔法

您可以使用 %neptune_ml 行魔法來啟動並管理各種 Neptune ML 操作。

> **Note**
>
> 您可以使用 %%neptune_ml 儲存格魔法來啟動並管理一些 Neptune ML 操作。

- **%neptune_ml export start** – 啟動新的匯出工作。

  參數

  - **--export-url** *exporter-endpoint* – (選用) 可在其中呼叫匯出程式的 Amazon API Gateway 端點。
  - **--export-iam** – (選用) 指示對匯出 URL 的請求必須使用 SIGv4 進行簽署的旗標。
  - **--export-no-ssl** – (選用) 指示連線至匯出程式時不應使用 SSL 的旗標。
  - **--wait** – (選用) 指示操作應等到匯出完成的旗標。
  - **--wait-interval***interval-to-wait*— (選擇性) 設定匯出狀態檢查之間的時間 (以秒為單位) (預設值：60)。
  - **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待匯出工作完成的時間 (以秒為單位) (預設值：3,600)。
  - **--store-to***location-to-store-result*— (選用) 用於儲存匯出結果的變數。如果指定 `--wait`，最終狀態將儲存在該處。
- **%neptune_ml export status** – 擷取匯出工作的狀態。

  參數

  - **--job-id**#### *ID* – 要擷取其狀態之匯出工作的 ID。
  - **--export-url** *exporter-endpoint* – (選用) 可在其中呼叫匯出程式的 Amazon API Gateway 端點。
  - **--export-iam** – (選用) 指示對匯出 URL 的請求必須使用 SIGv4 進行簽署的旗標。
  - **--export-no-ssl** – (選用) 指示連線至匯出程式時不應使用 SSL 的旗標。
  - **--wait** – (選用) 指示操作應等到匯出完成的旗標。
  - **--wait-interval***interval-to-wait*— (選擇性) 設定匯出狀態檢查之間的時間 (以秒為單位) (預設值：60)。

- **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待匯出工作完成的時間 (以秒為單位) (預設值：3,600)。
- **--store-to*location-to-store-result*** — (選用) 用於儲存匯出結果的變數。如果指定 `--wait`，最後狀態將儲存在該處。

- **%neptune_ml dataprocessing start** – 啟動 Neptune ML 資料處理步驟。

  參數

  - **--job-id** *#### ID* – (選用) 指派給此工作的 ID。
  - **--s3-input-uri** *S3 URI* – (選用) 要在其中尋找此資料處理工作之輸入的 S3 URI。
  - **--config-file-name** *####* – (選用) 此資料處理工作的組態檔名稱。
  - **--store-to*location-to-store-result*** — (選擇性) 用於儲存資料處理結果的變數。
  - **--instance-type** *(######)* – (選用) 用於此資料處理工作的執行個體大小。
  - **--wait** – (選用) 指示操作應等到資料處理完成的旗標。
  - **--wait-interval*interval-to-wait*** — (選擇性) 設定資料處理狀態檢查之間的時間 (以秒為單位) (預設值：60)。
  - **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待資料處理工作完成的時間 (以秒為單位) (預設值：3,600)。
- **%neptune_ml dataprocessing status** – 擷取資料處理工作的狀態。

  參數

  - **--job-id** *## ID* – 要擷取其狀態之工作的 ID。
  - **--store-to** *######* – (選用) 儲存模型訓練結果的變數。
  - **--wait** – (選用) 指示操作應等到模型訓練完成的旗標。
  - **--wait-interval*interval-to-wait*** — (選擇性) 設定模型訓練狀態檢查之間的時間 (以秒為單位) (預設值：60)。
  - **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待資料處理工作完成的時間 (以秒為單位) (預設值：3,600)。
- **%neptune_ml training start** – 啟動 Neptune ML 模型訓練程序。

  參數

  - **--job-id** *#### ID* – (選用) 指派給此工作的 ID。
  - **--data-processing-id** *###### ID* – (選用) 建立要用於訓練之成品的資料處理工作 ID。

- **--s3-output-uri** *S3 URI* – (選用) 儲存此模型訓練工作之輸出的 S3 URI。
- **--instance-type** *(######)* – (選用) 用於此模型訓練工作的執行個體大小。
- **--store-to*location-to-store-result*** — (選擇性) 用於儲存模型訓練結果的變數。
- **--wait** – (選用) 指示操作應等到模型訓練完成的旗標。
- **--wait-interval*interval-to-wait*** — (選擇性) 設定模型訓練狀態檢查之間的時間 (以秒為單位) (預設值：60)。
- **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待模型訓練工作完成的時間 (以秒為單位) (預設值：3,600)。

- **%neptune_ml training status** – 擷取 Neptune ML 模型訓練工作的狀態。

  參數

  - **--job-id** *## ID* – 要擷取其狀態之工作的 ID。
  - **--store-to** *######* – (選用) 儲存狀態結果的變數。
  - **--wait** – (選用) 指示操作應等到模型訓練完成的旗標。
  - **--wait-interval*interval-to-wait*** — (選擇性) 設定模型訓練狀態檢查之間的時間 (以秒為單位) (預設值：60)。
  - **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待資料處理工作完成的時間 (以秒為單位) (預設值：3,600)。

- **%neptune_ml endpoint create** – 建立 Neptune ML 模型的查詢端點。

  參數

  - **--job-id** *#### ID* – (選用) 指派給此工作的 ID。
  - **--model-job-id** *###### ID* – (選用) 要為其建立查詢端點之模型訓練工作的 ID。
  - **--instance-type** *(######)* – (optional) 用於查詢端點的執行個體大小。
  - **--store-to*location-to-store-result*** — (選用) 用於儲存端點建立結果的變數。
  - **--wait** – (選用) 指示操作應等到端點建立完成的旗標。
  - **--wait-interval*interval-to-wait*** — (選擇性) 設定狀態檢查之間的時間 (以秒為單位) (預設值：60)。
  - **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待端點建立工作完成的時間 (以秒為單位) (預設值：3,600)。

- **%neptune_ml endpoint status** – 擷取 Neptune ML 查詢端點的狀態。

參數

- **--job-id** *#### ID* – (選用) 要報告其狀態之端點建立工作的 ID。
- **--store-to***location-to-store-result*— (選擇性) 用來儲存狀態結果的變數。
- **--wait** – (選用) 指示操作應等到端點建立完成的旗標。
- **--wait-interval***interval-to-wait*— (選擇性) 設定狀態檢查之間的時間 (以秒為單位) (預設值：60)。
- **--wait-timeout** *interval-to-wait* – (選用) 設定在傳回最新狀態之前等待端點建立工作完成的時間 (以秒為單位) (預設值：3,600)。

%%neptune_ml 儲存格魔法

%%neptune_ml 儲存格魔法會忽略行輸入，例如 --job-id 或 --export-url。相反，它可以讓您在儲存格本文內提供這些輸入和其他輸入。

您也可以將此類輸入儲存在指派給 Jupyter 變數的另一個儲存格中，然後使用該變數將它們注入至儲存格本文。如此一來，您可以一再使用此類輸入，而不必每次都重新輸入它們。

這只有在注入變數是儲存格的唯一內容時才有效。您不能在一個儲存格中使用多個變數，也不能使用文字和變數的組合。

例如，%%neptune_ml export start 儲存格魔法可以使用儲存格本文中的 JSON 文件，其中包含用來控制 Neptune 匯出程序的參數 中所述的所有參數。

在 Neptune-ML-01-Introduction-to-Node-Classification-Gremlin 筆記本中，於匯出資料和模型組態區段的設定功能下，您可以看到下列儲存格如何在指派給 Jupyter 變數的文件中保留匯出參數 (名為 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"}
```

當您執行此儲存格時，Jupyter 會在該名稱下儲存參數文件。然後，您可以使用 ${export_params} 將 JSON 文件注入至 %%neptune_ml export start cell 的本文，如下所示：

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

${export_params}
```

%%neptune_ml 儲存格魔法的可用形式

%%neptune_ml 儲存格魔法可以用於以下形式：

- **%%neptune_ml export start** – 啟動 Neptune ML 匯出程序。
- **%%neptune_ml dataprocessing start** – 啟動 Neptune ML 資料處理工作。
- **%%neptune_ml training start** – 啟動 Neptune ML 模型訓練工作。
- **%%neptune_ml endpoint create** – 建立模型的 Neptune ML 查詢端點。

Neptune 工作台中的圖形視覺化

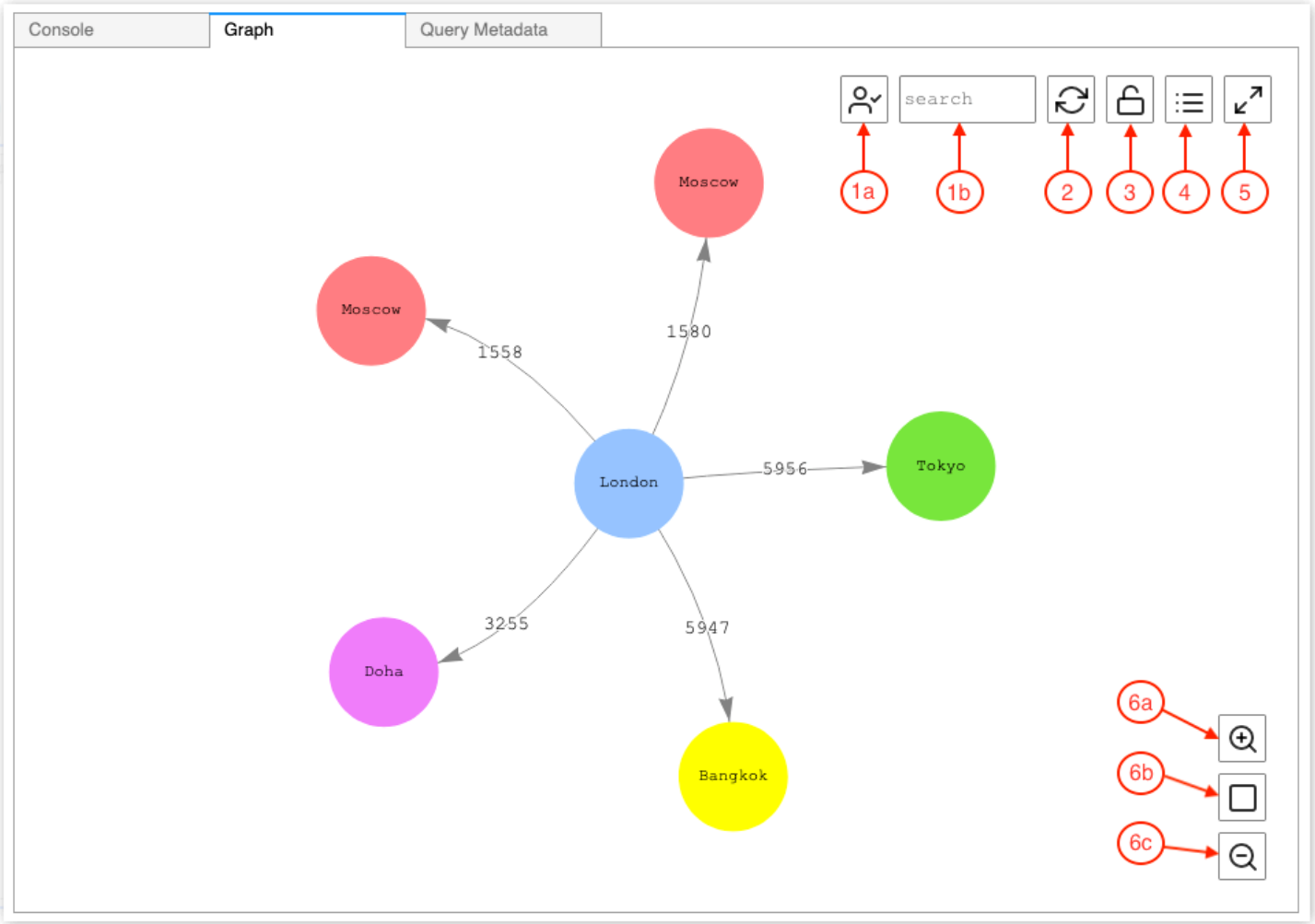
在許多情況下，Neptune 工作台可以建立查詢結果的視覺化圖表，以及以表格形式傳回它們。每當可以進行視覺化，便可在查詢結果中的圖形索引標籤中使用圖形視覺化。

除了這裡描述的內建視覺化功能之外，您還可以搭配 Neptune 圖形筆記本使用更進階的視覺化工具。

> **Note**
>
> 若要在您已使用的筆記本中存取最近新增的功能和修正，請先停止再重新啟動筆記本執行個體。

圖形索引標籤介面概觀

此圖表會識別 [圖表] 索引標籤中存在的使用者介面元素：



1. 圖形搜尋
   a. UUID 切換：切換圖形搜尋中包含 ID 屬性值。預設會啟用包含 ID。如果停用，則符合 ID 屬性 (包括參考節點 ID 的邊緣屬性) 的項目不會導致元素醒目提示。
   b. 搜尋文字欄位：醒目提示包含您在這裡指定之文字字串的所有頂點和邊緣屬性值。
2. 圖形重設 – 重新執行圖形物理模擬，並將縮放設定為符合視窗中的圖形。
3. 切換圖形物理 – 切換圖形物理模擬的執行。物理預設為啟用，讓圖形可以動態變更。如果停用，則移動其他頂點時，頂點會保持鎖定位置。
4. 詳細資訊檢視 – 選取節點或邊緣時，這會顯示元素的屬性索引鍵和值清單 (如果可在查詢結果中取得的話)。
5. 全螢幕檢視 – 展開圖形索引標籤視窗以符合螢幕大小。再按一下將圖形索引標籤最小化。
6. 縮放選項
   a. 放大
   b. 縮放重置：設定縮放以使圖形索引標籤視窗可以容納所有頂點。
   c. 縮小

將 Gremlin 查詢結果視覺化

Neptune 工作台會為傳回 `path` 的任何 Gemlin 查詢建立查詢結果的視覺化。若要查看視覺化，請在執行查詢之後選取查詢下主控台索引標籤右側的圖形索引標籤。

您可以使用查詢視覺化提示，來控制視覺化工具圖表查詢輸出的方式。這些提示會遵循 `%%gremlin` 儲存格魔法，且其前面會加上 `--path-pattern` (或其簡短格式 `-p`) 參數名稱：

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

您也可以使用 `--group-by` (或 `-g`) 旗標來指定頂點的屬性，以依此屬性將這些頂點分組。這允許為不同的頂點群組指定顏色或圖示。

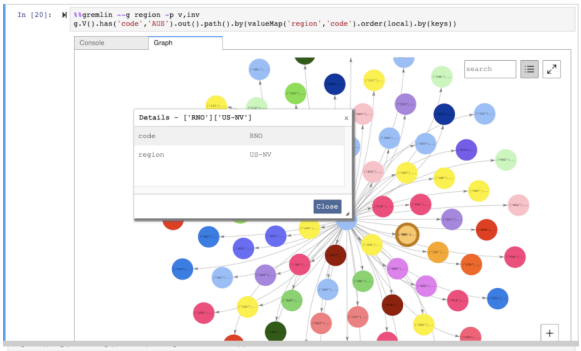
提示的名稱反映了在頂點之間周遊時常用的 Grimlin 步驟，並且它們會相應地有所表現。多個提示可以結合使用，以逗號分隔，它們之間沒有任何空格。使用的提示應該符合要視覺化之查詢中的對應 Gemlin 步驟。請見此處範例：

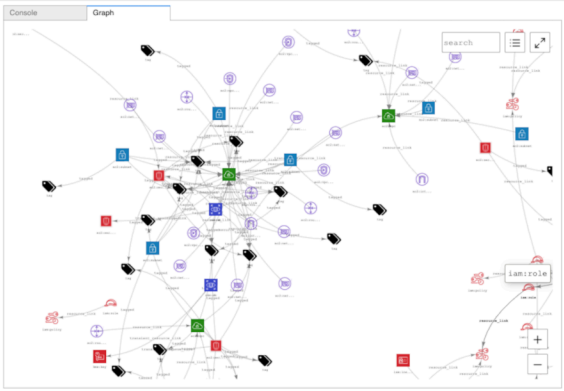
```
%%gremlin -p v,oute,inv
g.V().hasLabel('airport').outE().inV().path().by('code').by('dist').limit(5)
```

可用的視覺化提示如下：

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

以下是使用群組進行圖形視覺化的一些範例：





將 SPARQL 查詢結果視覺化

Neptune 工作台會為採用下列其中一種形式的任何 SPARQL 查詢建立查詢結果的視覺化：

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

若要查看視覺化，請在執行查詢之後選取查詢下資料表索引標籤右側的圖形索引標籤。

根據預設，SPARQL 視覺化僅包含三重模式，其中 o? 為 `uri` 或 bnode (空白節點)。所有其他 `?o` 繫結類型 (例如常值字串或整數) 都會被視為 `?s` 節點的屬性，您可以使用圖形索引標籤中的詳細資訊窗格檢視這些屬性。

不過，在許多情況下，您可能想要在視覺化中包含這類常值作為頂點。若要這樣做，請在 `%%sparql` 儲存格魔法之後使用 `--expand-all` 查詢提示：

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

這會告訴視覺化工具在圖形圖表中包含所有 `?s ?p ?o` 結果，而不論繫結類型為何。

您可以看到整個 `Air-Routes-SPARQL.ipynb` 筆記本中使用這個提示，也可以透過執行具有或沒有提示的查詢進行實驗，以查看它在視覺化中產生的差異。

在 Neptune 工作台中存取視覺化教學課程

Neptune 工作台隨附的兩個視覺化教學課程筆記本，其會在 Gremlin 和 SPARQL 中提供如何有效地查詢圖形資料並將結果視覺化的大量範例。

導覽至視覺化筆記本

1. 在左側的導覽窗格中，選擇右側的開啟記事本按鈕。
2. 一旦工作台開啟，執行 Jupyter，您就會在最上層看到 Neptune 資料夾。選擇此資料夾予以開啟。
3. 下一層是名為 02-Visualization 的資料夾。開啟此資料夾。裡面有幾個筆記本，其會透過不同的方式逐步引導你在 Gremlin 和 SPARQL 中查詢圖形資料，以及如何將查詢結果視覺化：

   - Air-Routes-Gremlin
   - Air-Routes-SPARQL
   - 工作台視覺化部落格
   - EPL-Gremlin
   - EPL-SPARQL

   選取要嘗試其包含之查詢的筆記本。

設定 Neptune

歡迎使用 Amazon Neptune。本節可協助您建立新的 Neptune 資料庫叢集，並找到您在 Neptune 文件中尋找的項目。

> **Note**
>
> 如需 AWS 圖形資料庫參考架構和參考部署架構，請參閱 Amazon Neptune 資源。這些資源可協助您了解所選的圖形資料模型及查詢語言，加速您的部署過程。

主題

選擇正確的 Neptune 資料庫執行個體

Amazon Neptune 提供許多不同的執行個體大小和系列，提供適合於不同圖形工作負載的不同功能。本節旨在協助您選擇符合需求的最佳執行個體類型。

如需這些系列中每個執行個體類型的定價，請參閱 Neptune 定價頁面。

執行個體資源配置的概觀

Neptune 中使用的每個 Amazon EC2 執行個體類型和大小都會提供定義的運算 (vCPU) 和系統記憶體數量。Neptune 的主要儲存體位於叢集中的資料庫執行個體外部，可讓運算和儲存容量彼此獨立擴展。

本節專注於運算資源的擴展方式，以及各種不同執行個體系列之間的差異。

在所有執行個體系列中，都會配置 vCPU 資源以支援每個 vCPU 兩 (2) 個查詢執行執行緒。此支援是由執行個體大小決定。確定給定 Neptune 資料庫執行個體的適當大小時，您需要考慮應用程式可能的並行性，以及查詢的平均延遲。您可以按照以下方式估計所需的 vCPU 數目，其中延遲是以平均查詢延遲來測量 (以秒為單位)，而並行性是以每秒的目標查詢數目來測量：

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

> **Note**
>
> 使用 DFE 查詢引擎的 SPARQL 查詢、OpenCypher 查詢和 Gramlin 讀取查詢，可以在某些情況下每個查詢使用多個執行緒。在最初調整資料庫叢集的大小時，請先假設每個查詢每次執行都會取用單一執行緒，並且如果您觀察到有背壓進入查詢佇列，則會縱向擴展。這可以透過使用`/gremlin/status`/`oc/status`、或 `/sparql/status` API 來觀察，也可以使用`MainRequestsPendingRequestsQueue` CloudWatch 量度來觀察。

每個執行個體上的系統記憶體分為兩個主要配置：緩衝集區快取和查詢執行緒記憶體。

執行個體中約有三分之二的可用記憶體會配置給緩衝集區快取。緩衝集區快取用來快取圖形的最近使用元件，以便重複存取這些元件的查詢可以進行更快的存取。具有較大系統記憶體數量的執行個體擁有較大的緩衝集區快取，可在本機儲存更多圖形。使用者可以監控中提供的緩衝區快取命中和遺漏測量結果，以調整適當的緩衝區集區快取數量。 CloudWatch

如果快取命中率始終在一段時間內掉至 99.9% 以下，您可能會想要增加執行個體的大小。這表明緩衝集區不夠大，而且引擎必須頻繁地從基礎儲存磁碟區擷取資料。

剩餘的三分之一系統記憶體會平均分佈在查詢執行緒之間，還有一些記憶體留給作業系統和小型動態集區，供執行緒視需要使用。每個執行緒可用的記憶體會從一個執行個體大小略為增加到下一個執行個體大小，直到 8xl 執行個體類型，此大小會使每個執行緒配置的記憶體達到上限。

新增更多執行緒記憶體的時機，就是您遇到 `OutOfMemoryException` (OOM)。當一個執行緒需要超過配置給它的記憶體上限時，就會發生 OOM 例外狀況 (這與整個執行個體耗盡記憶體不同)。

t3 和 t4g 執行個體類型

執行個體的 t3 和 t4g 系列提供低成本選項，用於開始使用圖形資料庫，也可用於初始開發和測試。這些執行個體符合 Neptune 免費方案的資格，可讓新客戶在獨立 AWS 帳戶中使用的前 750 個執行個體小時免費使用 Neptune，或在具有合併帳單 (付款人帳戶) 的 AWS 組織下彙整。

t3 和 t4g 執行個體僅以中型組態 (t3.medium 和 t4g.medium) 提供。

它們不適用於生產環境。

因為這些執行個體具有的資源非常有限，所以不建議用於測試查詢執行時間或整體資料庫效能。若要評估查詢效能，請升級至其他執行個體系列之一。

執行個體類型的 r4 系列

已棄用 – r4 系列是在 Neptune 於 2018 年推出時提供的，但現在有較新的執行個體類型可提供更好的價格/效能。從引擎 1.1.0.0 版開始，Neptune 不再支援 r4 執行個體類型。

執行個體類型的 r5 系列

r5 系列包含記憶體優化執行個體類型，適用於大多數圖形使用案例。r5 系列包含從 r5.large 最高可至 r5.24xlarge 的執行個體。隨著大小的增加，它們會以線性方式擴展運算效能。例如，r5.xlarge (4 個 vCPU 和 32GiB 記憶體) 具有的 vCPU 和記憶體是 r5.large (2 個 vCPU 和 16GiB 記憶體) 的兩倍，而 r5.2xlarge (8 個 vCPU 和 64GiB 記憶體) 具有的 vCPU 和記憶體是 r5.xlarge 的兩倍 您可以預期查詢效能會隨著運算容量直接最高擴展至 r5.12xlarge 執行個體類型。

r5 執行個體系列具有雙插槽 Intel CPU 架構。r5.12xlarge 和較小的類型會使用單插槽和該單插槽處理器所擁有的系統記憶體。r5.16xlarge 和 r5.24xlarge 類型會使用這兩個插槽和可用記憶體。因為雙插槽架構中的兩個實體處理器之間需要一些記憶體管理負荷，所以從 r5.12xlarge 擴展到 r5.16xlarge 或 r5.24xlarge 執行個體類型的效能提升並不像您以較小的大小縱向擴展那樣呈線性提升。

執行個體類型的 r5d 系列

Neptune 具有查詢快取功能，可以用來改善需要擷取和傳回大量屬性值和常值之查詢的效能。此功能主要是由需要使用查詢傳回許多屬性的客戶使用。查詢快取會本機擷取這些屬性值，而不是在 Neptune 索引儲存區中反覆查詢每個屬性值，藉此提升這些查詢的效能。

查詢快取是使用 r5d 執行個體類型上 NVMe 連接的 EBS 磁碟區來實作。它是使用叢集的參數群組來啟用。從 Neptune 索引儲存區擷取資料時，會在此 NVMe 磁碟區內快取屬性值和 RDF 常值。

如果您不需要查詢快取功能，請使用標準 r5 執行個體類型，而非 r5d，以避免更高成本的 r5d。

r5d 系列具有與 r5 系列大小相同的執行個體類型 (從 r5d.large 到 r5d.24xlarge)。

執行個體類型的 r6g 系列

AWS 已經開發了自己的基於 ARM 的處理器，稱為引力，提供比英特爾和 AMD 等價物更好的價格/性能。r6g 系列使用 Graviton2 處理器。在我們的測試中，Graviton2 處理器可為 OLTP 樣式 (受限) 圖形查詢提供 10-20% 的效能提升。不過，由於記憶體分頁效能的表現稍差，因此使用 Graviton2 處理器時，較大的 OLAP-ish 查詢效能可能會略低於 Intel 處理器的查詢效能。

另外務必要注意的是，r6g 系列具有單插槽架構，這意味著效能會隨著運算容量以線性方式從 r6g.large 擴展到 r6g.16xlarge (系列中最大的類型)。

執行個體類型的 r6i 系列

Amazon R6i 執行個體是由第三代 Intel Xeon 可擴展處理器 (名為 Ice Lake 的程式碼) 提供，非常適合於記憶體密集型工作負載。一般而言，與同類 R5 執行個體類型相比，每個 vCPU 可提升高達 15% 的運算價格效能，以及高達 20% 的記憶體頻寬。

執行個體類型的 x2g 系列

當執行個體具有較大的緩衝集區快取時，某些圖形使用案例可以看到更好的效能。推出 x2g 系列是為了更好地支援這些使用案例。此x2g系列的 memory-to-v CPU 比例大於r5或r6g系列。x2g 執行個體也會使用 Graviton2 處理器，並具有許多與 r6g 執行個體類型相同的效能特性，以及較大的緩衝集區快取。

如果您是 CPU 使用率低且緩衝集區快取遺漏率高的 r5 或 r6g 執行個體類型，請嘗試改用 x2g 系列。如此一來，您將取得所需的額外記憶體，而無需為更多 CPU 容量支付費用。

serverless 執行個體類型

Neptune Serverless 功能可以根據工作負載的資源需求動態擴展執行個體大小。Neptune Serverless 可讓您為資料庫叢集中的執行個體設定運算容量的下限和上限 (以 Neptune 容量單位測量)，而不是計算您的應用程式需要多少 vCPU。具有不同使用率的工作負載可以使用無伺服器而不是佈建的執行個體進行成本最佳化

您可以在相同的資料庫叢集中同時設定佈建執行個體和無伺服器執行個體，以實現最佳的成本效能組態。

為 Neptune 資料庫叢集選擇正確的儲存類型

Neptune 提供兩種不同定價模式的儲存類型：

- 標準儲存 — 標準儲存體可為中至低 I/O 使用量的應用程式提供具經濟效益的資料庫儲存。
- I/O 最佳化儲存 — 使用 I/O 最佳化儲存 (從引擎版本 1.3.0.0 提供)，您只需為使用的儲存和執行個體付費。儲存成本高於標準儲存的成本，而且您無需為使用的 I/O 付費。如果您的 I/O 使用量很高，佈建的 IOP 儲存可以大幅降低成本。

  I/O 優化儲存的設計符合 I/O 密集型圖形工作負載的需求，並具有低延遲要求和一致的 I/O 輸送量的低延遲。您每 30 天只能在 I/O 最佳化和標準儲存類型之間切換一次。

  如需 I/O 優化儲存的定價資訊，請參閱 Neptune 定價頁面。下節說明如何為 Neptune 資料庫叢集設定 I/O 優化儲存。

選擇適用於 Neptune 資料庫叢集的 I/O 優化儲存

依預設，Neptune 資料庫叢集使用標準儲存體。您可以在建立資料庫叢集時在該叢集上啟用 I/O 優化儲存，如下所示：

以下範例說明如何在使用 AWS CLI建立叢集時啟用 I/O 優化儲存：

```
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
```

然後，您建立的任何執行個體都會自動啟用 I/O 優化儲存：

```
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
```

您也可以修改現有的資料庫叢集，以啟用 I/O 優化儲存體，如下所示：

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

您可以在啟用 I/O 優化儲存體的情況下將備份快照還原到資料庫叢集：

```
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
```

您可以使用任何 `describe-` 呼叫來判斷叢集是否正在使用 I/O 優化儲存體。如果已啟用 I/O 優化儲存，呼叫會傳回設定為 `iop1` 的儲存類型欄位。

建立新的 Neptune 資料庫叢集

建立新的 Amazon Neptune 資料庫叢集最簡單的方法是使用可為您建立所有必要資源的 AWS CloudFormation 範本，而無需手動執行所有操作。 AWS CloudFormation 範本會為您執行大部分的設定，包括建立 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體：

使用 AWS CloudFormation 範本啟動新的 Neptune 資料庫叢集

1. 建立新的 IAM 使用者，其具有您使用 Neptune 資料庫叢集所需的許可，如 IAM 使用者許可 中所述。
2. 設定使用 AWS CloudFormation 範本所需的其他必要條件，如中所述使用 AWS CloudFormation 設定 Neptune 的先決條件。
3. 呼叫 AWS CloudFormation 堆疊，如中所述使用 AWS CloudFormation 堆疊建立 Neptune 資料庫叢集。

您也可以建立橫跨多個 Neptune 全域資料庫 AWS 區域，以啟用低延遲的全域讀取，並在中斷影響整 AWS 區域體的極少數情況下提供快速復原。

如需使用手動建立 Amazon Neptune 叢集的相關資訊 AWS Management Console，請參閱使用 AWS Management Console 啟動 Neptune 資料庫叢集。

您也可以使用 AWS CloudFormation 範本建立 Lambda 函數以搭配 Neptune 使用 (請參閱使用 AWS CloudFormation 建立要在 Neptune 中使用的 Lambda 函數)。

如需有關在 Neptune 中管理叢集和執行個體的一般資訊，請參閱 管理 Amazon Neptune 資料庫。

使用 AWS CloudFormation 設定 Neptune 的先決條件

使用 AWS CloudFormation 範本建立 Amazon Neptune 叢集之前，您必須具備下列項目：

- 一個 Amazon EC2 金鑰對。
- 使用所需的權限 AWS CloudFormation。

建立 Amazon EC2 金鑰配對，以便使用下列方式啟動 Neptune 叢集 AWS CloudFormation

若要使用 AWS CloudFormation 範本啟動 Neptune 資料庫叢集，您必須在建立堆疊的區域中提供 Amazon EC2Key 配對 (及其關聯的 PEM 檔案)。 AWS CloudFormation

如果您需要建立 key pair，請參閱 Amazon EC2 使用者指南中的使用 Amazon EC2 建立金鑰配對，或參閱 Amazon EC2 使用者指南中的使用 Amazon EC2 建立金鑰配對以取得指示。

新增 IAM 政策以授予使用 AWS CloudFormation 範本所需的許可

首先，您需要設定 IAM 使用者，讓其具有使用 Neptune 所需的許可，如 建立具有 Neptune 許可的 IAM 使用者 中所述。

然後，`AWSCloudFormationReadOnlyAccess`您需要將 AWS 受管理的策略新增至該使用者。

最後，您需要建立下列客戶受管政策，並將其新增至該使用者：

```
{
  "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**
>
> 刪除堆疊僅需下列許可：iam:DeleteRole、iam:RemoveRoleFromInstanceProfile、iam:DeleteRolePolicy、i
> 及 ec2:DeleteVpcEndpoints。
> 亦請注意 ec2:*Vpc授予 ec2:DeleteVpc 許可。

使用 AWS CloudFormation 堆疊建立 Neptune 資料庫叢集

您可以使用 AWS CloudFormation 範本來設定 Neptune 資料庫叢集。

1. 若要在 AWS CloudFormation 主控台上啟動 AWS CloudFormation 堆疊，請選擇下表中的其中一個「啟動堆疊」按鈕。

區域	檢視	在設計工具中檢視	啟動
美國東部 (維吉尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國東部 (俄亥俄)	檢視	在設計工具中檢視	Launch Stack
美國西部 (加利佛尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國西部 (奧勒岡)	檢視	在設計工具中檢視	Launch Stack
加拿大 (中部)	檢視	在設計工具中檢視	Launch Stack
南美洲 (聖保羅)	檢視	在設計工具中檢視	Launch Stack
歐洲 (斯德哥爾摩)	檢視	在設計工具中檢視	Launch Stack
歐洲 (愛爾蘭)	檢視	在設計工具中檢視	Launch Stack
歐洲 (倫敦)	檢視	在設計工具中檢視	Launch Stack
Europe (Paris)	檢視	在設計工具中檢視	Launch Stack
歐洲 (法蘭克福)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
Middle East (Bahrain)	檢視	在設計工具中檢視	Launch Stack
中東 (阿拉伯聯合大公國)	檢視	在設計工具中檢視	Launch Stack
以色列 (特拉維夫)	檢視	在設計工具中檢視	Launch Stack
非洲 (開普敦)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (香港)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (東京)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (首爾)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (新加坡)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (悉尼)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (孟買)	檢視	在設計工具中檢視	Launch Stack
中國 (北京)	檢視	在設計工具中檢視	Launch Stack
中國 (寧夏)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國西部)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國東部)	檢視	在設計工具中檢視	Launch Stack

2. 在 Select Template (選取範本) 頁面上，請選擇 Next (下一步)。

3. 在 [指定詳細資訊] 頁面上，選擇 EC2SSH KeyPair 名稱的 key pair。

   存取 EC2 執行個體需要用到這個金鑰對。確認您具有所選金鑰對的 PEM 檔案。

4. 選擇下一步。
5. 在選項頁面上，選擇下一步。
6. 在檢閱頁面上，選取第一個核取方塊以確認 AWS CloudFormation 將建立 IAM 資源。選取第二個核取方塊，確認新堆疊 `CAPABILITY_AUTO_EXPAND`。

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` 明確確認在建立堆疊時，無需事先檢閱，即可擴充巨集。使用者通常會在處理過的範本中建立變更集，以便在實際建立堆疊之前，檢閱巨集所做的變更。如需詳細資訊，請參閱 AWS CloudFormation CreateStack API。

   然後選擇 Create (建立)。

> **Note**
>
> 您也可以使用 AWS CloudFormation 範本升級資料庫叢集的引擎版本。

設定 Amazon Neptune 資料庫叢集所在的 Amazon VPC

只能在 Amazon Virtual Private Cloud (Amazon VPC) 中建立的 Amazon Neptune 資料庫叢集。在該 VPC 內可存取其端點。

有許多不同的方法可以設定 VPC，取決於您想要存取資料庫叢集的方式。

設定 Neptune 資料庫叢集所在的 VPC 時，要謹記以下幾點：

- 您的 VPC 必須至少要有兩個子網路。這些子網路必須位於兩個不同的可用區域 (AZ)。透過將叢集執行個體分散到至少兩個 AZ，Neptune 可協助確保即使在不大可能發生可用區域失敗的情況下，您的資料庫叢集內始終都有執行個體可用。Neptune 資料庫叢集的叢集磁碟區一律橫跨三個可用區域，以提供資料遺失可能性極低的耐久性儲存體。
- 每個子網路中的 CIDR 區塊必須大到足以提供在維護活動、容錯移轉和擴展期間 Neptune 可能需要的 IP 地址。

- VPC 必須具有一個資料庫子網路群組，其中包含您已建立的子網路。Neptune 會選擇子網路群組中的其中一個子網路，以及該子網路內的 IP 地址，以與資料庫叢集內的資料庫執行個體建立關聯。然後，資料庫執行個體位於與子網路相同的可用區域中。
- VPC 應該已啟用 DNS (DNS 主機名稱和 DNS 解析)。
- 您的 VPC 必須具有 VPC 安全群組，允許存取您的資料庫叢集。
- Neptune VPC 中的租用應設定為預設值。

將子網路新增至 Neptune 資料庫叢集所在的 VPC

子網是您的 VPC 中的 IP 地址範圍。您可以將 Neptune 資料庫叢集或 EC2 執行個體等資源啟動至特定子網路。建立子網時，您需為該子網指定 IPv4 CIDR 區塊，其即是 VPC CIDR 區塊的子網。每個子網路必須完全位於某一可用區域 (AZ) 內，而且不得跨越多個區域。藉由在單獨的可用區域中啟動執行個體，您可以保護應用程式免於在單一區域發生失敗。如需詳細資訊，請參閱 VPC 子網路文件。

一個 Neptune 資料庫叢集需要至少兩個 VPC 子網路。

將子網路新增至 VPC

1. 登入 AWS Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
2. 在導覽窗格中，選擇 Subnets (子網)。
3. 在 VPC 儀表板中選擇子網路，然後選擇建立子網路。
4. 在建立子網路頁面上，選擇要在其中建立子網路的 VPC。
5. 在子網路設定下，進行下列選擇：
    a. 在子網路名稱下輸入新子網路的名稱。
    b. 選擇子網路的可用區域 (AZ)，或保留無偏好設定中的選擇。
    c. 在 IPv4 CIDR 區塊下輸入子網路的 IP 地址區塊。
    d. 如有需要，請將標籤新增至子網路。
    e. 選擇
6. 如果您想要同時建立另一個子網路，請選擇新增子網路。
7. 選擇建立子網路以建立新的子網路。

在 VPC 中建立子網路群組

建立子網路群組。

建立 Neptune 子網路群組

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 選擇 Subnet groups (子網路群組)，然後選擇 Create DB Subnet Group (建立資料庫子網路群組)。
3. 輸入新子網路群組的名稱和描述 (描述是必要的)。
4. 在 VPC 下，選擇您要在其中放置此子網路群組的 VPC。
5. 在可用區域下，選擇您要在其中放置此子網路群組的 AZ。
6. 在子網路下，將此可用區域中的一或多個子網路新增至此子網路群組。
7. 選擇建立來建立新的子網路群組。

使用 VPC 主控台建立安全群組

安全群組提供 VPC 中 Neptune 資料庫叢集的存取權。其作用就像相關聯資料庫叢集的防火牆，從執行個體層級控制傳入和傳出流量。根據預設，建立的資料庫執行個體具有防火牆和預設安全群組，可防止對其進行任何存取。若要啟用存取，您必須擁有具備其他規則的 VPC 安全群組。

下列程序說明如何新增自訂的 TCP 規則，指定 Amazon EC2 執行個體用來存取 Neptune 資料庫叢集的連接埠範圍和 IP 地址。您可以使用指派給 EC2 執行個體的 VPC 安全群組，而不是其 IP 地址。

在主控台上建立 Neptune 的 VPC 安全群組

1. 登入 AWS Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
2. 在主控台的右上角，選擇您要為 Neptune 建立 VPC 安全性群組的 AWS 區域。在該區域的 Amazon VPC 資源清單中，應該會顯示至少一個 VPC 和多個子網路。如果沒有，表示您在該區域中沒有預設 VPC。
3. 在導覽窗格中，於安全下，選擇安全群組。
4. 選擇建立安全群組。在建立安全群組視窗中，輸入安全群組名稱、描述，以及 Neptune 資料庫叢集將位於其中之 VPC 的識別符。
5. 為要連線至 Neptune 資料庫叢集之 Amazon EC2 執行個體的安全群組新增傳入規則：

a. 在傳入規則區域中，選擇新增規則。

b. 在類型清單中，保持選取自訂 TCP。

c. 在連接埠範圍方塊中，輸入 8182，這是 Neptune 的預設連接埠值。

d. 在來源下，輸入您要從中存取 Neptune 的 IP 地址範圍 (CIDR 值)，或選擇現有的安全群組名稱。

e. 如果需要新增更多 IP 地址或不同的連接埠範圍，請再次選擇新增規則。

6. 在 [傳出規則] 區域中，您也可以在需要時新增一或多個傳出規則。

7. 完成後，請選擇 Create security group (建立安全群組)。

當建立新的 Neptune 資料庫叢集時，您可以使用此新的 VPC 安全群組。

如果您使用預設 VPC，系統已經為您建立橫跨所有 VPC 子網路的預設子網路群組。當您在 Neptune 主控台中選擇建立資料庫時，除非您指定不同的 VPC，否則會使用預設 VPC。

確定您在 VPC 中具有 DNS 支援

網域名稱系統 (DNS) 是一種在網際網路上用於解析為對應 IP 地址的標準名稱。DNS 主機名稱為獨特的電腦名稱並由主機名稱和網域名稱組成。DNS 伺服器會將 DNS 主機名稱解析為對應的 IP 地址。

請檢查以確定 VPC 中已同時啟用 DNS 主機名稱和 DNS 解析。VPC 網路屬性 `enableDnsHostnames` 和 `enableDnsSupport` 必須設定為 `true`. 要查看和修改這些屬性，請移至位於 https://console.aws.amazon.com/vpc/ 的 VPC 主控台。

如需詳細資訊，請參閱以 VPC 使用 DNS。

> **Note**
>
> 如果您使用的是 Route 53，請確認您的組態不會覆寫 VPC 中的 DNS 網路屬性。

連線至您的 Amazon Neptune 圖形

一旦建立了 Neptune 資料庫叢集，下一步就是設定您要連線至其中的方式。

設定 curl 或 awscurl 以與 Neptune 端點通訊

如本文件中的許多範例所示，具有將查詢提交至 Neptune 資料庫叢集的命令列工具非常方便。curl 命令列工具是在未啟用 IAM 身分驗證時與 Neptune 端點通訊的絕佳選項。從 7.75.0 開始的版本支援在啟用 IAM 身分驗證時簽署請求的 `--aws-sigv4` 選項。

對於已啟用 IAM 身分驗證的端點，您還可以使用 awscurl，其會使用與 `curl` 幾乎完全相同的語法，但支援簽署 IAM 身分驗證所需的請求。由於 IAM 身分驗證提供了新增的安全性，因此啟用它通常是個好主意。

如需如何使用 `curl` (`awscurl`) 的相關資訊，請參閱 curl man 頁面和 Everything curl 一書。

若要使用 Neptune 需要的 HTTPS 進行連線，`curl` 需要存取適當的憑證。只要 `curl` 可以找到適當的憑證，它處理 HTTPS 連線的方式就跟 HTTP 連線一樣，無需額外參數。`awscurl` 也是一樣。此文件的範例是以該案例為基礎。

若要了解如何取得這類憑證，以及如何將它們正確格式化為 `curl` 可以使用的憑證授權機構 (CA) 憑證存放區，請參閱 `curl` 文件中的 SSL 憑證驗證。

然後，您可以使用 `CURL_CA_BUNDLE` 環境變數來指定此 CA 憑證存放區的位置。在 Windows 上，`curl` 會自動在名為 `curl-ca-bundle.crt` 的檔案中尋找它。它會先在和 `curl.exe` 相同的目錄中尋找，然後再尋找路徑的其他位置。如需詳細資訊，請參閱 SSL Certificate Verification。

連線至 Neptune 資料庫叢集的不同方式

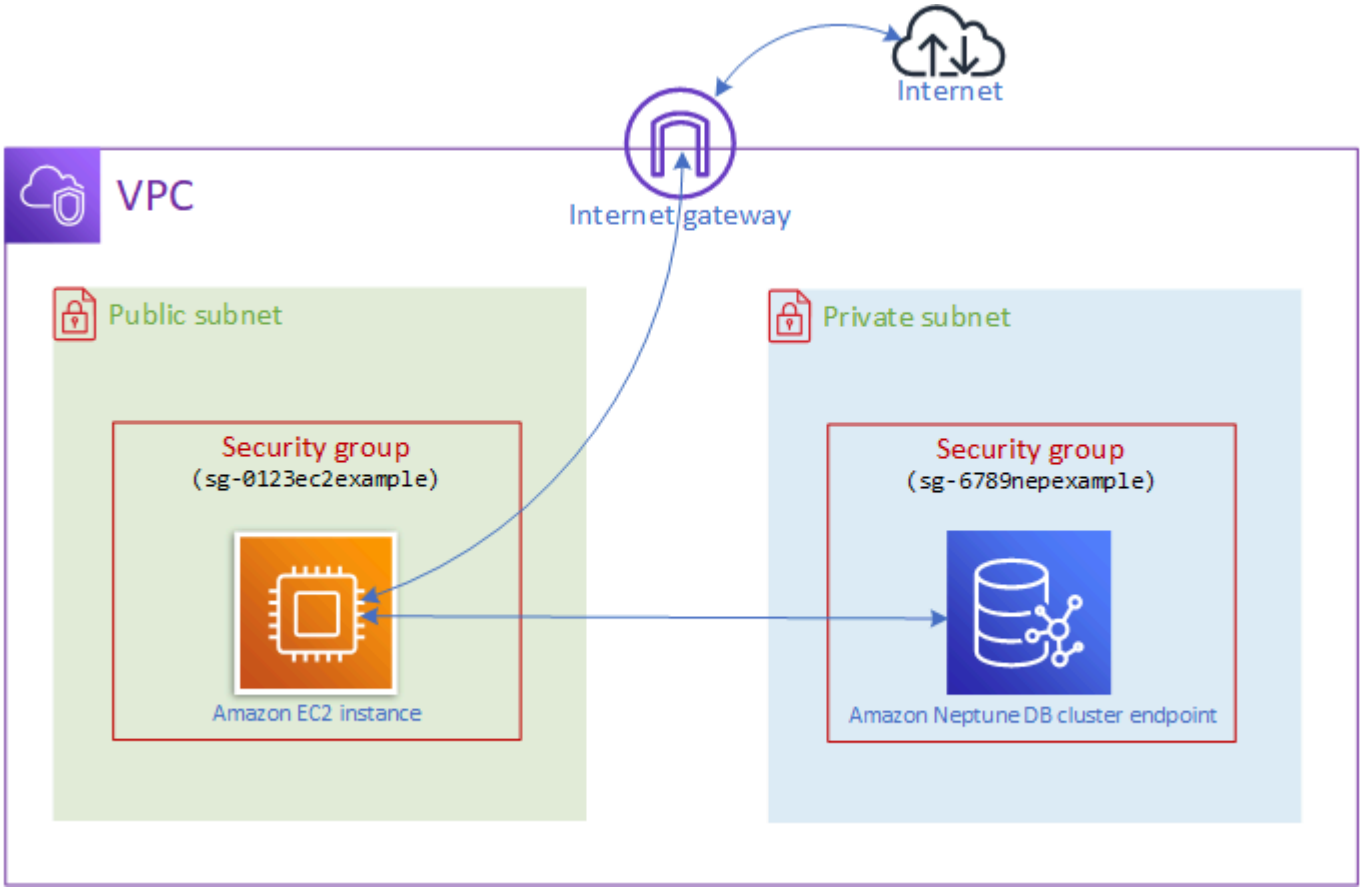
只能在 Amazon Virtual Private Cloud (Amazon VPC) 中建立的 Amazon Neptune 資料庫叢集。除非您為資料庫叢集啟用並設定 Neptune 公用端點，否則只能在該 VPC 內存取其端點。

有數種不同的方法，可在其 VPC 中設定對 Neptune 資料庫叢集的存取。

- 從同一 VPC 中的 Amazon EC2 執行個體連線
- 從另一個 VPC 中的 Amazon EC2 執行個體連線
- 從私有網路連線

從同一 VPC 中的 Amazon EC2 執行個體連線至 Neptune 資料庫叢集

連線至 Neptune 資料庫的最常見方法之一，就是從與 Neptune 資料庫叢集相同的 VPC 中的 Amazon EC2 執行個體連線。例如，EC2 執行個體可能正在執行與網際網路連線的 Web 伺服器。在此情況下，只有 EC2 執行個體才能存取 Neptune 資料庫叢集，而網際網路只能存取 EC2 執行個體：



若要啟用此組態，您必須設定正確的 VPC 安全群組和子網路群組。Web 伺服器是在公有子網路中託管，以便它可以連線公有網際網路，而您的 Neptune 叢集執行個體是在私有子網路中託管，以確保其安全。請參閱設定 Amazon Neptune 資料庫叢集所在的 Amazon VPC。

為了讓 Amazon EC2 執行個體連線至您的 Neptune 端點 (例如連接埠 8182)，您需要設定安全群組來執行此動作。例如，如果您的 Amazon EC2 執行個體使用名為 db-sg1 的安全群組，您需要建立另一個 Amazon EC2 安全群組 (假設是 ec2-sg1)，其中具有連接埠 8182 的傳入規則，以及具有 ec2-sg1 作為其來源。然後，將 db-sg1 新增至您的 Neptune 叢集以允許連線。

在建立 Amazon EC2 執行個體之後，您可以使用 SSH 登入該執行個體，然後連線至您的 Neptune 資料庫叢集。如需使用安全殼層連線至 EC2 執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到 Linux 執行個體。

如果您使用 Linux 或 macOS 命令列連線至 EC2 執行個體，您可以從堆疊的「輸出」區段中，從 SSHAccess 項目貼上 SSH 命令。 AWS CloudFormation 您必須有目前目錄的 PEM 檔案，而且此 PEM 檔案許可必須設定為 400 (`chmod 400 keypair.pem`)。

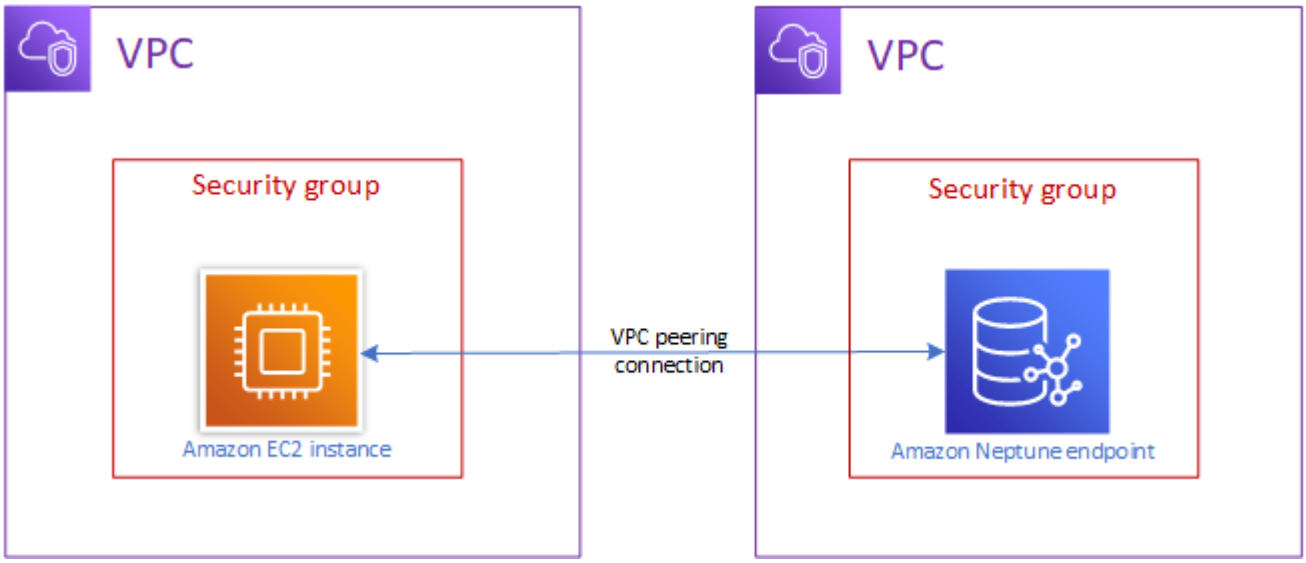
建立含私有和公有子網路的 VPC

1. 登入 AWS Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
2. 在的右上角 AWS Management Console，選擇要在其中建立 VPC 的地區。
3. 在 VPC 儀表板中，選擇啟動 VPC 精靈。
4. 完成建立 VPC 頁面的 VPC 設定 區域：
   a. 在 Resources to create (建立資源) 下，選擇 VPC, subnets, etc. (VPC、子網路等)。
   b. 保持預設名稱標籤不變，或輸入您選擇的名稱，或取消勾選自動產生核取方塊以停用產生名稱標籤。
   c. 將 IPv4 CIDR 區塊值保留為 `10.0.0.0/16`。
   d. 將 IPv6 CIDR 區塊值保留為沒有 IPv6 CIDR 區塊。
   e. 將租用保留為預設值。
   f. 將可用區域 (AZ) 的數目保留為 2。
   g. 除非您需要一或多個 NAT 閘道，否則將 NAT 閘道 ($) 保留為無。
   h. 除非您要使用 Amazon S3，否則將 VPC 端點設定為無。
   i. 應該同時勾選啟用 DNS 主機名稱和啟用 DNS 解析。
5. 選擇建立 VPC。

從另一個 VPC 中的 Amazon EC2 執行個體存取您的資料庫叢集

Amazon Neptune 資料庫叢集只能在 Amazon Virtual Private Cloud (Amazon VPC) 中建立，而且其端點只能在該 VPC 內存取，通常從在該 VPC 中執行的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體進行。

當您的資料庫叢集與您用來存取它的 EC2 執行個體不在相同 VPC 中時，您可以使用 對等互連來進行連線。



VPC 對等互連連線是兩個 VPC 之間的聯網連線，其會以私有方式在它們之間路由流量，以便任一 VPC 中的執行個體可以如同它們在同一個網路內進行通訊。您可以在帳戶中的 VPC 之間、帳戶中的 VPC 與其 AWS 他帳戶中的 VPC 之間建立 VPC 對等連線，或使用不同區域中的 VPC 建立 VPC 對等連線。 AWS AWS

AWS 使用 VPC 的現有基礎結構來建立 VPC 對等連線。它既不是閘道器，也不是 AWS Site-to-Site VPN 連線，而且不依賴於單獨的實體硬體。它不會有通訊的單一故障點或頻寬瓶頸問題。

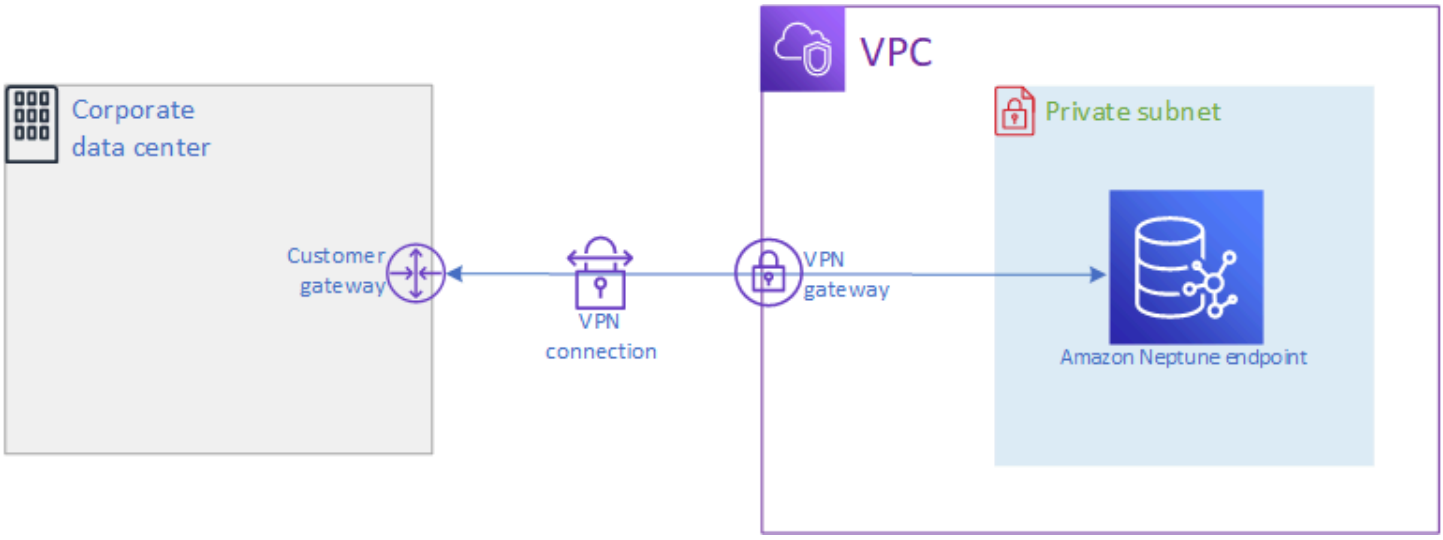
如需如何使用 VPC 對等互連的詳細資訊，請參閱 Amazon VPC 對等互連指南。

從私有網路存取您的資料庫叢集

您可以使用兩種不同的方式從私有網路存取 Neptune 資料庫叢集：

- 使用 AWS 站台對站台 VPN 連線。
- 使用 AWS Direct Connect 連線。

上面連結具有這些連線方法以及如何設定它們的相關資訊。站 AWS 台對站台連線的設定可能如下所示：



在 Amazon Neptune 中保護您的資料

有多種方式可供您保護 Amazon Neptune 叢集。

使用 IAM 政策來限制對 Neptune 資料庫叢集的存取

若要控制誰可以在 Neptune 資料庫叢集和資料庫執行個體上執行 Neptune 管理動作，請使用 AWS Identity and Access Management (IAM)。

當您使用 IAM 帳戶存取 Neptune 主控台時，必須先 AWS Management Console 使用您的 IAM 帳戶登入，然後才能在 https://console.aws.amazon.com/neptune/home 開啟 Neptune 主控台。

當您連線到 AWS 使用 IAM 登入資料時，您的 IAM 帳戶必須具有 IAM 政策，以授與執行 Neptune 管理作業所需的許可。如需詳細資訊，請參閱 使用不同種類的 IAM 政策來控制對 Neptune 的存取。

使用 VPC 安全群組限制對 Neptune 資料庫叢集的存取

必須在 Amazon Virtual Private Cloud (Amazon VPC) 中建立 Neptune 資料庫叢集。若要控制哪些裝置和 EC2 執行個體可以開放對 VPC 中 Neptune 資料庫叢集之資料庫執行個體端點和連接埠的連線，您可以使用 VPC 安全群組。如需 VPCs 的詳細資訊，請參閱「使用 VPC 主控台建立安全群組」。

使用 IAM 身分驗證來限制對 Neptune 資料庫叢集的存取

如果您在 Neptune 資料庫叢集中啟用 AWS Identity and Access Management (IAM) 身份驗證，則存取資料庫叢集的任何人都必須先通過驗證。如需設定 IAM 身分驗證的相關資訊，請參閱 Amazon Neptune AWS Identity and Access Management （ IAM ）概述。

如需使用臨時登入資料進行驗證的相關資訊，包括 AWS CLI AWS Lambda、和 Amazon EC2 的範例，請參閱the section called “暫時登入資料”。

下列連結提供有關使用 IAM 身分驗證搭配個別查詢語言，來連線至 Neptune 的其他資訊：

使用 Gremlin 搭配 IAM 身分驗證

- the section called “Gremlin 主控台”
- the section called “Gremlin Java”
- the section called “Python 範例”

> **Note**
> 此範例適用於 Gremlin 和 SPARQL。

使用 openCypher 搭配 IAM 身分驗證

- the section called “Gremlin 主控台”
- the section called “Gremlin Java”
- the section called “Python 範例”

> **Note**
> 此範例適用於 Gremlin 和 SPARQL。

使用 SPARQL 搭配 IAM 身分驗證

- the section called “SPARQL Java (RDF4J 和 Jena)”
- the section called “Python 範例”

> **Note**
> 此範例適用於 Gremlin 和 SPARQL。

開始存取您的 Neptune 圖形

一旦您建立了 Neptune 資料庫叢集並設定與它的連線，下一步就是與它通訊，以便載入資料、進行查詢等。若要這樣做，大多數人都會使用 `curl` 或 `awscurl` 命令列工具。

設定 `curl` 以與 Neptune 端點通訊

如本文件中的許多範例所述，curl 命令列工具是可與 Neptune 端點通訊的便利選項。如需工具資訊，請參閱 curl man 頁面和 Everything curl 一書。

若要依我們的建議與 Neptune 在大部分區域的要求使用 HTTPS 連線，`curl` 需要存取適當的憑證。若要了解如何取得這些憑證，以及如何將它們正確格式化為 `curl` 可以使用的憑證授權機構 (CA) 憑證存放區，請參閱 `curl` 文件中的 SSL 憑證驗證。

然後，您可以使用 `CURL_CA_BUNDLE` 環境變數來指定此 CA 憑證存放區的位置。在 Windows 上，`curl` 會自動在名為 `curl-ca-bundle.crt` 的檔案中尋找它。它會先在和 `curl.exe` 相同的目錄中尋找，然後再尋找路徑的其他位置。如需詳細資訊，請參閱 SSL Certificate Verification。

只要 `curl` 可以找到適當的憑證，它處理 HTTPS 連線的方式就跟 HTTP 連線一樣，無需額外參數。此文件的範例是以該案例為基礎。

使用查詢語言存取 Neptune 資料庫叢集中的圖形資料

一旦連線，您可以使用 Gremlin 和 OpenCypher 查詢語言，來建立和查詢屬性圖，或者使用 SPARQL 查詢語言，來建立和查詢包含 RDF 資料的圖形。

Neptune 支援的圖形查詢語言

- Gremlin 是一種適用於屬性圖的圖形周遊語言。Grimlin 中的查詢是由離散步驟組成的周遊，每個步驟都沿著一個邊緣到一個節點。如需詳細資訊，請參閱 Apache TinkerPop 3 的小鬼文件。

  Gremlin 的 Neptune 實作與其他實作有一些不同，尤其是使用 Gremlin-Groovy 時 (以序列化的文字傳送的 Gremlin 查詢)。如需詳細資訊，請參閱 Amazon Neptune 中的 Gremlin 標準合規。
- OpenCypher 是屬性圖的宣告式查詢語言，最初由 Neo4j 開發，然後在 2015 年成為開放原始碼，並在 Apache 2 開放原始碼授權下投入 OpenCypher 專案。其語法記載於 Cypher 查詢語言參考第 9 版。
- SPARQL 是 RDF 資料的宣告式查詢語言，以全球資訊網協會 (W3C) 制定的標準圖形模式配對為基礎，於 SPARQL 1.1 概觀和 SPARQL 1.1 查詢語言規格中描述。

> **Note**
>
> 您可以使用 Gremlin 和 OpenCypher，但不使用 SPARQL，來存取 Neptune 屬性圖資料。同樣地，您只能使用 SPARQL，而不是 Gremlin 或 openCypher 存取 RDF 資料。

使用 Gremlin 存取 Amazon Neptune 中的圖形

您可以使用 Grinlin 控制台在 REPL（read-eval-print 循環）環境中嘗試 TinkerPop 圖形和查詢。

以下教學會逐步解說如何使用 Gremlin 主控台在 Neptune 圖形中新增頂點、邊緣、屬性及更多，反白 Neptune 特定 Gremlin 實作中的一些差異。

> **Note**
>
> 此範例假設您已完成下列各項：
>
> - 您已使用 SSH 連線到 Amazon EC2 執行個體。
> - 您已建立 Neptune 叢集，如 建立資料庫叢集 中所詳述。
> - 您已如安裝 Gremlin 主控台所述安裝 Gremlin 主控台。

使用 Gremlin 主控台

1. 將目錄變更為 Gremlin 主控台檔案解壓縮的資料夾。

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

2. 輸入下列命令以執行 Gremlin 主控台。

   ```
   bin/gremlin.sh
   ```

   您應該會看到下列輸出：

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

```
gremlin>
```

您現在進入 gremlin> 提示。您會在這個提示下進入其餘的步驟。

3. 在 gremlin> 提示下，輸入以下命令以連線到 Neptune 資料庫執行個體。

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

4. 在 gremlin> 提示下，輸入以下內容以切換為遠端模式。這會傳送所有 Gremlin 查詢到遠端連線。

```
:remote console
```

5. 新增頂點以及標籤和屬性。

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

頂點會獲指派一個 string ID，包含 GUID。所有頂點 ID 都是 Neptune 中的字串。

6. 新增頂點與自訂 ID。

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

id 屬性不使用括號括住。這是頂點 ID 的關鍵字。此處的頂點 ID 是一個字串，其中有數字 1。

正常屬性名稱必須包含在引號中。

7. 如果屬性不存在，請變更屬性或新增屬性。

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

您在這裡變更前一個步驟頂點的 name 屬性。這會移除 name 屬性所有現有的值。

如果您未指定 single，則會改為附加值到 name 屬性 (如果尚未這樣做)。

8. 新增屬性，但如果屬性已有值則附加屬性。

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

Neptune 使用成組基數做為預設動作。

此命令會新增 age 屬性以及值 29，但不會取代任何現有的值。

如果 age 屬性已有值，這個命令會附加 29 到屬性。例如，如果 age 屬性原本是 27，新的值會是 [27, 29]。

9. 新增多個頂點。

```
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)
```

您可以同時將多個陳述式傳送至 Neptune。

陳述式可以使用換行符號 ('\n')、空格 (' ')、分號 ('; ') 來隔開，也可以不使用任何分隔符號 (例如：g.addV('person').iterate()g.V() 為有效)。

> **Note**
>
> Gremlin 主控台會在每個新行 ('\n') 處傳送個別命令，所以在這種情況下每個都是單獨的交易。此範例將所有命令放在不同的行以方便閱讀。移除新行 ('\n') 字元可透過 Gremlin 主控台將其做為單一命令傳送。

最後一個陳述式除外的所有陳述式都必須結束在終止步驟，例如 .next() 或 .iterate()，否則不會執行。Gremlin 主控台不需要這些終止步驟。每當您不需要序列化結果時就使用 .iterate。

一起傳送的所有陳述式會包含在單一交易中，一起成功或一起失敗。

10. 新增邊緣。

```
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)
```

以下是新增邊緣的兩個不同方式。

11. 新增其他現代圖形。

```
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. 刪除頂點。

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

移除具有 name 屬性的頂點等於 justin。

> **Important**
>
> 停在這裡，你有完整的 Apache TinkerPop 現代圖。 TinkerPop 文件「遍歷」區段中的範例使用「現代」圖表。

13. 執行周遊。

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

傳回所有 person 頂點。

14. 使用值 (valueMap ()) 執行周遊。

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

傳回 marko「知道」的所有頂點的索引鍵值組。

15. 指定多個標籤。

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

Neptune 支援頂點的多個標籤。當您建立標籤時，您可以指定多重標籤並用 :: 分隔。

此範例新增有三種不同標籤的頂點。

hasLabel 步驟將比對此頂點和這三個標籤：hasLabel("Label1")、hasLabel("Label2") 和 hasLabel("Label3")。

:: 分隔符號僅針對本用途保留。

您不能在 hasLabel 步驟中指定多重標籤。例如，hasLabel("Label1::Label2") 不符合任何內容。

16. 指定時間/日期。

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

Neptune 不支援 Java 日期。請改用 datetime() 函數。datetime() 接受 ISO8061 相容 datetime 字串。

支援格式如下：YYYY-MM-DD, YYYY-MM-DDTHH:mm、YYYY-MM-DDTHH:mm:SS 和 YYYY-MM-DDTHH:mm:SSZ

17. 刪除頂點、屬性或邊緣。

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

以下是數個 drop 範例。

> **Note**
>
> .next() 步驟不適用於 .drop()。請改用 .iterate()。

18. 完成後，輸入以下內容以退出 Gremlin 主控台。

```
:exit
```

> **Note**
>
> 使用分號 (;) 或換行符號字元 (\n) 來分隔每個陳述式。
> 最終周遊之前的每個周遊節尾必須為 iterate()，才能執行。但只有最後的周遊資料會傳回。

使用 openCypher 存取 Amazon Neptune 中的圖形

若要開始使用 OpenCypher，請參閱openCypher或使用 Neptune 圖形筆記本存放庫中的 OpenCypher 筆記本。 GitHub

使用 RDF 和 SPARQL 存取 Amazon Neptune 中的圖形

SPARQL 是一種資源描述架構 (RDF) 的查詢語言，其是專為網路設計的圖形資料格式。Amazon Neptune 與 SPARQL 1.1 相容。這表示您可以連線到 Neptune 資料庫執行個體，並使用 SPARQL 1.1 查詢語言規格所述的查詢語言來查詢圖形。

SPARQL 的查詢包含 SELECT 子句，用於指定要傳回的變數，和 WHERE 子句，用於指定要比對圖形中的哪些資料。如果您不熟悉 SPARQL 查詢，請參閱 SPARQL 1.1 查詢語言中的編寫簡易查詢。

對 Neptune 資料庫執行個體進行 SPARQL 查詢時所用的 HTTP 端點為 `https://your-neptune-endpoint:port/sparql`。

若要連接到 SPARQL

1. 您可以從 AWS CloudFormation 堆疊的 [輸出] 區段中的SparqlEndpoint項目取得 Neptune 叢集的 SPARQL 端點。
2. 輸入下列命令，以使用 HTTP POST 和 curl 命令提交 SPARQL **UPDATE**。

   ```
   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
   ```

   上述範例插入以下三元組到 SPARQL 預設圖形：`<https://test.com/s> <https://test.com/p> <https://test.com/o>`
3. 輸入下列命令，以使用 HTTP POST 和 curl 命令提交 SPARQL **QUERY**。

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

   先前範例使用 `?s ?p ?o` 查詢和限制 10，以傳回圖形中的最多 10 個三元組 (subject-predicate-object)。若要查詢其他項目，請將查詢換成其他 SPARQL 查詢。

> **Note**
>
> SELECT 和 ASK 查詢回應的預設 MIME 類型為 application/sparql-results+json。
> CONSTRUCT 和 DESCRIBE 查詢回應的預設 MIME 類型為 application/n-quads。
> 如需所有可用 MIME 類型的清單，請參閱 SPARQL HTTP API。

將資料載入至 Neptune

Amazon Neptune 提供直接從外部檔案將資料載入 Neptune 資料庫執行個體的程序。您可以使用此程序，而非執行大量 INSERT 陳述式、addV 和 addE 步驟，或其他 API 呼叫。

以下是額外載入資訊的連結。

- 載入資料的方法 – 載入資料
- 大量載入器支援的資料格式 – the section called “資料格式”
- 載入範例 – the section called “載入範例”

監控 Amazon Neptune

Amazon Neptune 支援下列監控方法。

- 亞馬遜 CloudWatch — Amazon Neptune 會自動將指標傳送到警報， CloudWatch 並支援 CloudWatch 警報。如需詳細資訊，請參閱 the section called “使用 CloudWatch”。
- AWS CloudTrail— Amazon Neptune 支持使用 CloudTrail. 如需詳細資訊，請參閱 the section called “使用記錄 Neptune API 呼叫 AWS CloudTrail”。
- 標記 – 使用標籤將中繼資料新增到 Neptune 資源，並根據標籤追蹤使用情形。如需詳細資訊，請參閱 the section called “標記 Neptune 資源”。
- 稽核日誌檔 – 使用 Neptune 主控台檢視、下載或監看資料庫日誌檔。如需詳細資訊，請參閱 the section called “稽核日誌搭配 Neptune”。
- 執行個體狀態 – 檢查 Neptune 執行個體圖形資料庫引擎的運作狀態，了解已安裝的引擎版本，並使用執行個體狀態 API 取得其他引擎狀態資訊。

Neptune 中的疑難排解和最佳實務

以下連結可能有助於解決 Amazon Neptune 的問題。

- 最佳實務 – 如需常見問題的解決辦法和效能建議，請參閱 最佳實務。
- 服務錯誤 – 如需管理 API 和圖形資料庫連線的問題清單，請參閱 Neptune 错误。
- 服務限制 – 如需 Neptune 限制的相關資訊，請參閱 Neptune 限制。
- 引擎版本 – 如需圖形引擎版本的相關資訊，包括版本備註，請參閱 引擎版本。
- 支援論壇 – 若要加入 Neptune 的相關討論，請參閱 Amazon Neptune 論壇。
- 定價 – 如需使用 Amazon Neptune 的成本相關資訊，請參閱 Amazon Neptune 定價。
- AWS Sup@@ port — 如需專家的說明和指導，請參閱AWS Support。

建立 Neptune 全球資料庫

Amazon Neptune 全球資料庫橫跨多個 AWS 區域，可在中斷影響整個 AWS 區域 的罕見情況下，啟用低延遲全域讀取並提供快速復原。

Neptune 全球資料庫由一個區域中的主要資料庫叢集，以及不同區域中最多五個次要資料庫叢集所組成。

寫入只能在主要區域中發生。次要區域僅支援讀取。每個次要區域最多可有 16 個讀取器執行個體。

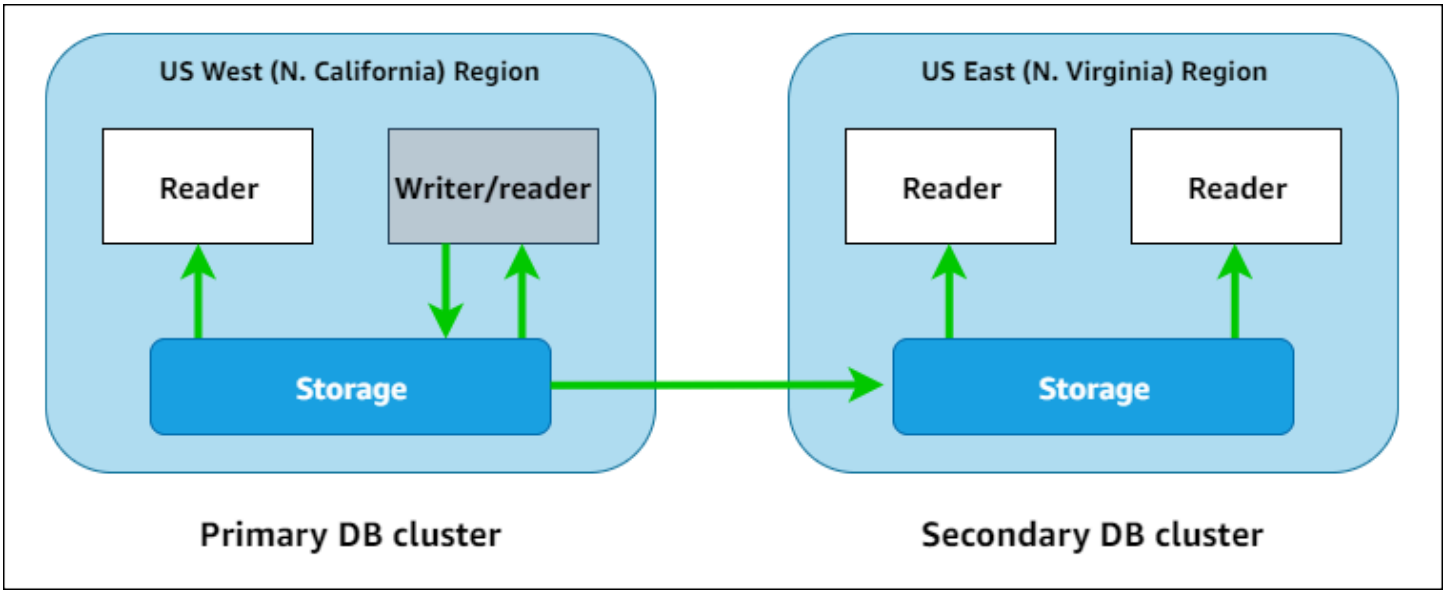
主題

Amazon Neptune 中全球資料庫的概觀

使用 Neptune 全球資料庫，您可以在跨越多個 AWS 區域 的單一資料庫上執行全域分散式應用程式。

Neptune 全球資料庫由資料寫入所在之主要 AWS 區域中的一個資料庫叢集，以及次要 AWS 區域中最多五個唯讀資料庫叢集所組成。當您在主要資料庫叢集上執行寫入操作時，Neptune 會使用專用基礎架構將寫入的資料庫叢集複寫到所有次要資料庫叢集，其延遲通常不到一秒鐘。

下圖顯示跨越兩個 AWS 區域的範例全球資料庫。



您可以新增一或多個僅供讀取複本執行個體，來獨立擴展每個次要叢集，以處理唯讀工作負載。

若要執行寫入操作，您必須連線到主要資料庫叢集的資料庫叢集端點。只有主要叢集才能執行寫入操作。然後，如上圖所示，複寫是由叢集儲存磁碟區執行，而不是由資料庫引擎執行。

Neptune 全球資料庫專為覆蓋全世界的應用程式而設計。唯讀次要資料庫叢集支援更接近應用程式使用者的讀取操作。

Aurora 全球資料庫支援兩種不同的容錯移轉方法。

- 若要從主要區域的中斷復原，請使用手動非計劃分離並提升程序，您可以在其中分離其中一個次要叢集、將其轉換為獨立叢集，然後將其提升為新的主要叢集。
- 針對規劃的操作程序 (例如維護)，請使用受管的規劃容錯移轉，在這裡您會將主要叢集重新放置到其中一個次要區域，而不會遺失資料。

在 Amazon Neptune 使用全球資料庫的優點

使用全球資料庫，具有下列優點：

- 以本機延遲提供全域讀取 — 如果您在世界各地設有辦公室，則全球資料庫可讓次要區域中的辦公室以本機延遲方式存取其自己區域中的資料。
- 可擴展的次要 Neptune 資料庫叢集 — 您可以新增僅供讀取複本資料庫執行個體來擴展次要叢集。因為次要叢集是唯讀的，所以這些叢集每一個最多可支援 16 個僅供讀取複本，而不是一般限制的 15 個。

- 快速複寫到次要資料庫叢集 — 從主要資料庫叢集複寫到次要資料庫叢集很快，其延遲通常不到一秒鐘，對主要資料庫叢集的效能影響不大。因為複寫是在儲存層級執行，所以資料庫執行個體資源完全可供應用程式讀取和寫入工作負載使用。
- 從全區域中斷中復原 — 次要資料庫叢集可讓您更快地將主要叢集移至新區域，比起傳統複寫解決方案，其 RTO 更低，且更不會遺失資料 (RPO 更低)。

Amazon Neptune 中全球資料庫的限制

全球資料庫目前有下列限制：

- Neptune 全球資料庫僅適用於以下 AWS 區域：
  - 美國東部 (維吉尼亞北部)：`us-east-1`
  - 美國東部 (俄亥俄)：`us-east-2`
  - 美國西部 (加利佛尼亞北部)：`us-west-1`
  - 美國西部 (奧勒岡)：`us-west-2`
  - 歐洲 (愛爾蘭)：`eu-west-1`
  - 歐洲 (倫敦)：`eu-west-2`
  - 亞太區域 (東京)：`ap-northeast-1`
- Neptune 全球資料庫不支援次要資料庫叢集的自動擴展。
- 在執行該全球資料庫的主要版本升級時，無法將自訂參數群組套用至全球資料庫叢集。相反，請在全域叢集的每個區域中建立自訂參數群組，然後在升級後手動將其套用至區域叢集。
- 您無法個別停止或啟動全球資料庫中的資料庫叢集。
- 在特定情況下，次要資料庫叢集中的僅供讀取複本執行個體可以重新啟動。如果主要叢集的寫入器資料庫執行個體重新啟動或容錯移轉，次要區域中的所有執行個體也會重新啟動。然後，次要叢集將無法使用，直到其所有執行個體都與主要資料庫叢集的寫入器執行個體重新同步為止。

在 Amazon Neptune 中設定全球資料庫

您可以使用下列其中一種方式建立 Neptune 全球資料庫：

- 使用所有新的資料庫叢集和執行個體建立全球資料庫。
- 使用現有的 Neptune 資料庫叢集作為主要叢集來建立全球資料庫。

主題

Amazon Neptune 中全球資料庫的組態需求

Neptune 全球資料庫至少跨越兩個 AWS 區域。主要 AWS 區域 包含具有一個寫入器執行個體的 Neptune 資料庫叢集。一至五個 AWS 區域，每一個都包含完全由僅供讀取複本執行個體組成的唯讀 Neptune 資料庫叢集。需要至少一個 AWS 區域。

構成全球資料庫的 Neptune 資料庫叢集具有下列特定需求：

- 資料庫執行個體類別需求 — 全球資料庫需要針對記憶體密集型工作負載最佳化的 `r5` 或 `r6g` 資料庫執行個體類別，例如 `db.r5.large` 執行個體類型。
- AWS 區域需求 — 全球資料庫需要一個 AWS 區域中有一個主要 Neptune 資料庫叢集，而在不同區域中至少有一個次要 Neptune 資料庫叢集。您最多可以建立五個次要唯讀 Neptune 資料庫叢集，而且每個叢集必須位於不同的區域。換句話說，Neptune 全球資料庫中沒有兩個 Neptune 資料庫叢集可以在同一個 AWS 區域中。
- 引擎版本需求 — 全球資料庫中所有資料庫叢集使用的 Neptune 引擎版本應該相同，且必須大於或等於 `1.2.0.0`。如果您在建立新的全球資料庫或叢集或執行個體時未指定引擎版本，則會使用最新的引擎版本。

> **Important**
>
> 雖然可以針對全球資料庫中的每個資料庫叢集個別設定資料庫叢集參數群組，但是最好讓叢集之間的設定保持一致，以避免次要叢集必須升級為主要叢集時發生意外的行為變更。例如，在所有資料庫叢集中，對物件索引、串流等使用相同的設定。

使用 AWS CLI 在 Amazon Neptune 中建立全球資料庫

> **Note**
>
> 本節中的範例遵循使用反斜線 (\) 作為行擴充器字元的 UNIX 慣例。若為 Windows，請將反斜線取代為插入符號 (^)。

使用 AWS CLI 建立全球資料庫

1. 首先使用 `create-global-cluster` AWS CLI 命令 (其中包裝 CreateGlobalCluster API) 建立空的全球資料庫。指定您要其成為主要的 AWS 區域 名稱、將 Neptune 設定為資料庫引擎，以及選擇性地指定您想要使用的引擎版本 (這必須是 1.2.0.0 版或更高版本)：

```
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. 全球資料庫可能需要幾分鐘的時間才能使用，因此在進行下一個步驟之前，請使用 `describe-global-clusters` CLI 命令 (其中包裝 DescribeGlobalClusters API) 來檢查全球資料庫是否可用：

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

3. 在 Neptune 全球資料庫變成可用之後，您可以建立新的 Neptune 資料庫叢集成為其主要叢集：

```
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. 使用 `describe-db-clusters` AWS CLI 命令，確認新的資料庫叢集已準備好讓您新增其主要資料庫執行個體：

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

當回應顯示 "Status": "available" 時，請繼續下一個步驟。

5. 使用 create-db-instance AWS CLI 命令建立主要叢集的主要資料庫執行個體。您必須使用其中一種記憶體最佳化的 r5 或 r6g 執行個體類型，例如 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**
>
> 如果您打算使用 Neptune 大量載入器，將資料新增至新的主要資料庫叢集，請在新增次要區域之前執行此動作。這比在全球資料庫完全設定之後執行大量載入更快速且更具成本效益。

現在，將一或多個次要區域新增至新的全球資料庫，如 使用 AWS CLI 新增次要區域 中所述。

將現有的資料庫叢集轉換為全球資料庫

若要將現有的資料庫叢集轉換為全球資料庫，請使用 create-global-cluster AWS CLI 命令，在與現有資料庫叢集所在位置相同的 AWS 區域 中建立新的全球資料庫，並將其 --source-db-cluster-identifier 參數設定為位於該處的現有叢集的 Amazon Resource Name (ARN)：

```
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)
```

現在，將一或多個次要區域新增至新的全球資料庫，如 使用 AWS CLI 新增次要區域 中所述。

使用從快照還原的資料庫叢集作為主要叢集

您可以將從快照還原的資料庫叢集轉換為 Neptune 全球資料庫。還原完成後，將其建立的資料庫叢集轉換成新全球資料庫的主要叢集，如上所述。

將次要全球資料庫區域新增至 Amazon Neptune 的主要區域

Neptune 全球資料庫至少需要一個次要 Neptune 資料庫叢集位在與主要資料庫叢集不同的 AWS 區域中。您最多可以將五個次要資料庫叢集附加至主要資料庫叢集。

您新增的每個次要資料庫叢集會將您可在主要叢集上具有的僅供讀取複本執行個體數目上限減一。例如，若有 4 個次要叢集，則您可在主要叢集具有的僅供讀取複本執行個體數目上限為 `15 - 4 = 11`。這表示，如果您在主要資料庫叢集及一個次要叢集中已有 14 個讀取器執行個體，則無法新增另一個次要叢集。

使用 AWS CLI 將次要區域新增至 Neptune 中的全球資料庫

使用 AWS CLI 將次要 AWS 區域新增至 Neptune 全球資料庫

1. 使用 `create-db-cluster` AWS CLI 命令，在與主要叢集不同的區域中建立新的資料庫叢集，並設定其 `--global-cluster-identifier` 參數以指定全球資料庫的 ID：

   ```
   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. 使用 `describe-db-clusters` AWS CLI 命令，確認新的資料庫叢集已準備好讓您新增其主要資料庫執行個體：

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

   當回應顯示 `"Status": "available"` 時，請繼續下一個步驟。

3. 使用 `create-db-instance` AWS CLI 命令搭配 `r5` 或 `r6g` 執行個體類別中的執行個體類型，來建立主要叢集的主要資料庫執行個體：

```
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**
>
> 如果您不想要在次要區域中提供大量讀取請求，而且主要關心的是能在該處可靠地備份您的資料，則可以建立不含資料庫執行個體的次要資料庫叢集。這樣可以節省費用，因為您只需支付次要叢集的儲存體費用，Neptune 將與主要資料庫叢集中的儲存體保持同步。

連線到全球資料庫

連線到 Neptune 全球資料庫的方式取決於您需要寫入其中還是從中讀取：

- 對於唯讀請求或查詢，請連線至 AWS 區域中 Neptune 叢集的讀取器端點。
- 若要執行變動查詢，請連線至主要資料庫叢集的叢集端點，該端點可能位於與您應用程式不同的 AWS 區域 中。

管理 Amazon Neptune 全球資料庫

除了受管計劃容錯移轉程序外，您還可在構成 Neptune 全球資料庫的個別叢集上執行大部分的管理操作。受管計劃容錯移轉程序僅適用於 Neptune 全球資料庫，不適用於個別的 Neptune 資料庫叢集。如需進一步了解，請參閱執行 Neptune 全球資料庫的受管計劃容錯移轉。

若要從其主要區域的非計劃中斷復原 Neptune 全球資料庫，請參閱 在發生非計劃中斷時分離並提升 Neptune 全球資料庫。

雖然您可以針對全球資料庫中的每個 Neptune 叢集個別設定資料庫叢集參數群組，但是最好讓叢集之間的設定保持一致，以避免次要叢集升級為主要叢集時發生意外的行為變更。例如，在所有資料庫叢集中，對物件索引、串流等使用相同的設定。

從 Neptune 全球資料庫中移除資料庫叢集

有幾個您可能想要從全球資料庫移除資料庫叢集的原因。例如：

- 如果主要叢集降級或遭隔離，您可以將其從全球資料庫中移除，而且其會變成可用來建立新全球資料庫的獨立佈建叢集 (請參閱 在發生非計劃中斷時分離並提升 Neptune 全球資料庫)。
- 如果想要刪除全球資料庫，首先您必須從中刪除 (分離) 所有相關聯的叢集，最後保留主要叢集 (請參閱 刪除 Neptune 全球資料庫)。

您可以使用 remove-from-global-cluster CLI 命令 (其中包裝 RemoveFromGlobalCluster API)，將 Neptune 資料庫叢集從全球資料庫中分離：

```
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)
```

然後，分離的資料庫叢集會變成獨立資料庫叢集。

刪除 Neptune 全球資料庫

您無法在單一步驟中刪除全球資料庫及其相關聯的叢集。相反，您必須逐一刪除其元件：

1. 從全球資料庫中分離所有次要資料庫叢集，如 移除叢集 所述。如果想要的話，您現在可以個別刪除它們。
2. 從全球資料庫中分離主要資料庫叢集。
3. 從主要叢集中刪除所有僅供讀取複本資料庫執行個體。
4. 從主要叢集中刪除主要 (寫入器) 資料庫執行個體。如果您在主控台上執行此操作，它也會刪除資料庫叢集。
5. 刪除全球資料庫本身。若要使用 AWS CLI 執行此操作，請使用 delete-global-cluster CLI 命令 (其中包裝 DeleteGlobalCluster API)，如下所示：

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

修改 Neptune 全球資料庫

雖然可以針對全球資料庫中的每個 Neptune 資料庫叢集個別設定資料庫叢集參數群組，但是最好讓叢集之間的設定保持一致，以避免次要叢集必須升級為主要叢集時發生意外的行為變更。

您可以使用 modify-global-cluster CLI 命令 (其中包裝 ModifyGlobalCluster API) 來修改全球資料庫本身的設定。例如，您可以變更全球資料庫識別符，同時關閉刪除保護，如下所示：

```
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
```

在 Neptune 全球資料庫中使用容錯移轉

Neptune 全球資料庫提供的容錯移轉功能比獨立 Neptune 資料庫叢集提供的更加全面。使用全球資料庫，您可以相當快速地規劃並從災難復原。評估災難復原的方式通常會使用復原時間點目標 (RTO) 和復原點目標 (RPO) 的評估：

- 復原時間點目標 (RTO) — 這是系統在災難發生後回到工作狀態的速度。換言之，RTO 會測量停機時間。對於 Neptune 全球資料庫，RTO 可能需要幾分鐘。
- 復原點目標 (RPO) — 遺失資料期間的時間量。對於 Neptune 全球資料庫，RPO 通常以秒為單位進行測量 (請參閱 執行 Neptune 全球資料庫的受管計劃容錯移轉)。

對於 Neptune 全球資料庫，有兩種不同的容錯移轉方法：

- 分離並提升 (手動非計劃復原) — 若要從非計畫中斷復原或執行災難復原測試 (DR 測試)，請對全球資料庫中的其中一個次要資料庫叢集執行跨區域分離並提升。此手動程序的 RTO 取決於您在 分離並提升 中執行所列任務的速度。RPO 通常為幾秒，但這取決於發生故障時網路上的儲存體複寫延遲。
- 受管計劃容錯移轉 — 此方法適用於操作維護和其他計劃操作程序，例如將全球資料庫的主要資料庫叢集重新放置到其中一個次要區域。由於此程序會將次要資料庫叢集與主要資料庫叢集同步，再做出任何其他變更，因此 RPO 實際為 0 (亦即，不會遺失任何資料)。請參閱 執行 Neptune 全球資料庫的受管計劃容錯移轉。

在發生非計劃中斷時分離並提升 Neptune 全球資料庫

在極少數情況下，Neptune 全球資料庫會在其主要 AWS 區域 遭遇非預期的中斷，您的主要 Neptune 資料庫叢集及其寫入器節點會變得無法使用，而主要叢集與次要叢集之間的複寫則會停止。若要將產生的停機時間 (RTO) 和資料遺失 (RPO) 減至最低，請快速執行跨區域分離並提升，以重建全球資料庫。

> **Tip**
>
> 最好在使用此程序之前先對其有所了解，並制定一個適當的計劃，以在全區域問題第一次出現時便立即快速進行該計劃。
>
> - 定期使用 Amazon CloudWatch 追蹤次要叢集的延遲時間，以便您可以在需要容錯移轉時，識別延遲時間最短的次要區域。
> - 請務必測試您的計畫，以檢查您的程序是否完整且準確。
> - 使用模擬環境來確保您的員工受過訓練，並準備好在必要時快速執行 DR 容錯移轉。

在主要區域發生非計劃中斷之後容錯移轉至次要叢集

1. 停止在主資料庫叢集上發出變動查詢和其他寫入操作。
2. 識別次要 AWS 區域 中哪個資料庫叢集要用作全球資料庫的新主要資料庫叢集。如果全球資料庫有兩個以上的次要 AWS 區域，請選擇延遲時間最低的次要叢集。
3. 從 Neptune 全球資料庫中分離您選擇的次要資料庫叢集。

   從 Neptune 全球資料庫中移除次要資料庫叢集，會立即停止將資料從主要資料庫叢集複寫到該次要資料庫叢集，並將其提升為具有完整讀取/寫入功能的獨立資料庫叢集。全球資料庫中的任何其他次要叢集仍可供使用，而且可以接受來自您應用程式的讀取呼叫。

   在重新建立 Neptune 全球資料庫之前，您還必須分離其他次要叢集，以避免叢集之間的資料不一致 (請參閱 移除叢集)。
4. 重新設定您的應用程式，以使用其新端點將所有寫入操作傳送至您選擇成為新主要叢集的獨立 Neptune 資料庫叢集。如果您在建立 Neptune 全球資料庫時接受預設名稱，則可以透過從應用程式中叢集的端點字串移除 `-ro` 來變更端點。

   例如，當該叢集從全球資料庫分離時，次要叢集的端點 `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` 會變成 `my-global.cluster-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com`。

當您開始在下一個步驟中將區域新增至這個 Neptune 資料庫叢集時，其會成為新 Neptune 全球資料庫的主要叢集。

5. 將 AWS 區域 新增至資料庫叢集。當您執行這項操作時，從主要到次要的複寫程序即會開始。請參閱 將次要全球資料庫區域新增至 Amazon Neptune 的主要區域 。
6. 視需新增更多 AWS 區域，以重新建立支援您的應用程式所需的拓撲。

請確定應用程式寫入會在做出這些變更之前、期間和之後，傳送至正確的 Neptune 資料庫叢集。這樣可以避免 Neptune 全球資料庫中的資料庫叢集之間出現資料不一致的情況 (這些稱為核心分裂問題)。

執行 Neptune 全球資料庫的受管計劃容錯移轉

受管計劃容錯移轉會讓您將 Neptune 全球資料庫的主要叢集重新放置到不同的 AWS 區域，無論您做何選擇都一樣。有些組織會想要定期輪換其主要叢集位置。

> **Note**
>
> 這裡描述的受管計劃容錯移轉旨在用於運作狀態良好的 Neptune 全球資料庫。若要從非計劃中斷復原或執行災難復原 (DR) 測試，請改為遵循分離並提升程序。

在受管計劃容錯移轉期間，您的主要叢集會容錯移轉至您選擇的次要區域，同時保留全球資料庫現有的複寫拓撲。在受管計劃容錯移轉程序開始之前，全球資料庫會將所有次要叢集與其主要叢集同步。確保所有叢集都已同步之後，受管計劃容錯移轉即會開始。主要區域中的資料庫叢集會變成唯讀，而選擇的次要叢集會將其中一個唯讀執行個體提升為完整寫入器狀態，從而允許叢集擔任主要叢集的角色。由於所有次要叢集都在程序開始時與主要叢集同步，因此新的主要叢集會繼續執行全球資料庫的操作，而不會遺失任何資料。此資料庫在短時間內無法使用，因為同時間主要叢集和選取的次要叢集會擔任其新角色。

若要最佳化應用程式可用性，請在寫入主要資料庫叢集最少的非尖峰時段執行容錯移轉。此外，在開始容錯移轉之前採取下列步驟：

- 盡可能讓應用程式離線，以減少對主要叢集的寫入。
- 檢查全球資料庫中所有次要 Neptune 資料庫叢集的延遲時間，並選擇整體延遲時間最低的次要叢集成為主要叢集。使用 Amazon CloudWatch 來檢視所有次要資料庫叢集的 `NeptuneGlobalDBProgressLag` 指標。此指標會告訴您，次要資料庫叢集與主要資料庫叢集的差距 (以毫秒為單位)。其值直接與 Neptune 完成容錯移轉所需的時間成正比。換言之，延遲值越

大，容錯移轉中斷時間就越長，因此請選擇延遲最低的次要叢集。如需詳細資訊，請參閱Neptune CloudWatch 度量。

在受管計劃容錯移轉期間，選擇的次要資料庫叢集會提升為作為主要資料庫叢集的新角色，但其不會繼承主要資料庫叢集的完整組態。組態不相符可能會導致效能問題、工作負載不相容，以及其他異常行為。若要避免這類問題，請在容錯移轉之前先解決全球資料庫叢集之間下列種類的組態差異：

- 在新的主要叢集中設定參數以符合目前的主要叢集。
- 設定監控工具、選項和警示 — 設定將成為新主要叢集的資料庫叢集，其具有的記錄功能、警示等等與目前主要叢集具有的相同。
- 設定與其他 AWS 服務的整合 — 如果您的 Neptune 全球資料庫與 AWS 服務 (例如 AWS Identity and Access Management (IAM)、Amazon S3 或 AWS Lambda) 整合，請確定這些服務會視需設定為與新的主要資料庫叢集整合。

當容錯移轉程序完成且提升的資料庫叢集已準備好處理全球資料庫的寫入操作時，請務必變更您的應用程式，以將新的端點用於新的主要叢集。

使用 AWS CLI 啟動受管計劃容錯移轉

使用 failover-global-cluster CLI 命令 (其中包裝 FailoverGlobalCluster API) 來容錯移轉 Neptune 全球資料庫：

```
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**
>
> 預覽版中無法使用 `failover-global-cluster` API。它將是 GA 版本的一部分。

使用 CloudWatch 指標監控 Neptune 全球資料庫

Neptune 支援下列您可以用來監控 Neptune 全球資料庫的 CloudWatch 指標：

- **`GlobalDbDataTransferBytes`** – 在 Neptune 全球資料庫中重做日誌資料從主要 AWS 區域傳輸到次要 AWS 區域 的位元組數目。

- **GlobalDbReplicatedWriteIO** – 從全球資料庫中主要 AWS 區域複寫到次要 AWS 區域中叢集磁碟區的寫入 I/O 操作次數。

  Neptune 全球資料庫中每個資料庫叢集的計費計算會使用 `VolumeWriteIOPS`，來說明該叢集內執行的寫入。對於主要資料庫叢集，計費計算會使用 `NeptuneGlobalDbReplicatedWriteIO` 來說明對次要資料庫叢集的跨區域複寫。
- **GlobalDbProgressLag** – 對於使用者交易和系統交易，次要叢集在主要叢集後面的毫秒數。

指標	維度	發佈於	個單位
`GlobalDbDataTransferBytes`	SourceRegion、DBClusterIdentifier	Secondary	位元組
`GlobalDbReplicatedWriteIO`	SourceRegion、DBClusterIdentifier	Secondary	計數
`GlobalDbProgressLag`	DBClusterIdentifier、SecondaryRegion：在主要中 DBClusterIdentifier、SourceRegion：在次要中	主要、次要	毫秒

Amazon Neptune 功能概觀

本節提供特定 Neptune 功能的概觀，包括：

- Neptune 符合查詢語言標準。
- Neptune 的圖形資料模型。
- Neptune 交易語義的說明。
- Neptune 叢集和執行個體簡介。
- Neptune 的儲存體、可靠性和可用性。
- Neptune 端點的說明。
- Neptune 的自訂查詢 ID 如何讓您檢查查詢狀態。
- 使用 Neptune 的實驗室模式啟用實驗功能。
- Neptune DFE 引擎的描述。
- Neptune 的 JDBC 連線能力。
- Neptune 引擎版本清單以及如何更新引擎。

> **Note**
>
> 本節未涵蓋使用您可以用來存取 Neptune 圖形中資料的查詢語言。
> 如需如何使用 Gremlin 連線到執行中 Neptune 資料庫叢集的相關資訊，請參閱 使用 Gremlin 存取 Neptune 圖形。
> 如需如何使用 openCypher 連線到執行中 Neptune 資料庫叢集的相關資訊，請參閱 使用 openCypher 存取 Neptune 圖形。
> 如需如何使用 SPARQL 連線到執行中 Neptune 資料庫叢集的相關資訊，請參閱 使用 SPARQL 存取 Neptune 圖形。

主題

- Amazon Neptune 標準合規的注意事項
- Neptune 圖形資料模型。
- Neptune 查詢快取可加速讀取查詢
- Neptune 中的交易語意
- Amazon Neptune 資料庫叢集和執行個體

- Amazon Neptune 儲存體、可靠性和可用性
- 連線至 Amazon Neptune 端點
- 將自訂 ID 注入至 Neptune Gremlin 或 SPARQL 查詢
- Neptune 實驗室模式
- Amazon Neptune 替代查詢引擎（DFE）
- 管理要供 Neptune DFE 使用的統計資料
- 取得有關圖形的快速摘要報告
- Amazon Neptune JDBC 連線能力
- Amazon Neptune 引擎更新

Amazon Neptune 標準合規的注意事項

在大多數情況下，Amazon Neptune 符合實作 Gremlin 和 SPARQL 圖形查詢語言的適用標準。

以下各節說明這些標準，以及 Neptune 擴展或從中分出來的領域。

主題

Amazon Neptune 中的 Gremlin 標準合規

以下各節提供了 Grinlin 的 Neptune 實作概觀，以及它與 Apache TinkerPop 實作的不同之處。

Neptune 在其引擎中以原生方式實現了一些小鬼步驟，並使用 Apache TinkerPop Grimlin 實現來處理其他步驟（請參閱）。Amazon Neptune 的原生 Gremlin 步驟支援

> **Note**
>
> 如需 Gremlin 主控台和 Amazon Neptune 中所顯示實作差異的一些具體範例，請參閱快速入門的 the section called “使用 Gremlin” 一節。

主題

- 頂點屬性 ID
- 頂點屬性的基數
- 更新頂點屬性
- 標籤
- 逸出字元
- Groovy 限制
- 序列化
- Lambda 步驟
- 不支援的 Gremlin 方法
- 不支援的 Gremlin 步驟
- Neptune 中的 Gremlin 圖形功能

Gremlin 的適用標準

- 怪靈語言是由 Apache TinkerPop 文檔和 Apache TinkerPop 實現的小靈，而不是由正式的規範定義。
- 對於數值格式，Gremlin 遵循 IEEE 754 標準 (IEEE 754-2019 - 浮點數運算的 IEEE 標準。如需詳細資訊，另請參閱 Wikipedia IEEE 754 頁面)。

指令碼中的變數和參數

如果與預先繫結的變數有關，則周遊物件 g 在 Neptune 中是預先繫結的，而且不支援 `graph` 物件。

雖然 Neptune 不支援指令碼中的 Gremlin 變數或參數化，但是您可能經常會在網際網路上遇到 Gemlin 伺服器的範例指令碼，其中包含變數宣告，例如：

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

在提交查詢時，還有許多使用參數化 (或繫結) 的範例，例如：

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

參數範例通常與警告相關聯，這些警告關於盡可能不參數化所產生的效能損失。有很多這樣的例 TinkerPop 子，你可能會遇到，他們都聽起來很有說服力的需要參數化。

不過，變數宣告功能和參數化功能 (以及警告) 都只適用於 TinkerPop的 Gemlin 伺服器使用. `GremlinGroovyScriptEngine` 當 Gremlin 伺服器使用 Gramlin 的 `gremlin-language` ANTLR 文法來解析查詢時，它們不適用。ANTLR 語法不支援變數宣告或參數化，因此在使用 ANTLR 時，您不必擔心無法參數化。由於 ANTLR 語法是較新的元件 TinkerPop，因此您可能會在網際網路上遇到的較舊內容通常不會反映這種區別。

Neptune 會在其查詢處理引擎中使用 ANTLR 文法，而不是 `GremlinGroovyScriptEngine`，因此它不支援變數或參數化或 `bindings` 屬性。因此，與無法參數化有關的問題不適用於 Neptune。使用 Neptune，只需在通常參數化的位置按原狀提交查詢就非常安全。因此，前面的範例可以簡化，而不會造成任何效能損失，如下所示：

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

TinkerPop 枚舉

Neptune 不支援列舉值的完全合格類別名稱。例如，您必須在 Groovy 請求中使用 `single`，而不是 `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single`。

列舉類型由參數類型決定。

下表顯示允許的列舉值和相關的 TinkerPop 完整名稱。

允許值	類別
`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`	或者. 修補程式. VertexProperty. 基數
`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`	修補程式. 格林. 過程遍歷. SackFunctions. 障礙物
`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`	或者. 修補程序. 格林. 進程. TraversalOption父系。挑選

Java 程式碼

Neptune 不支援任意 Java 所定義以外的呼叫方法或除了支援的 Gremlin API 以外的 Java 程式庫呼叫。例如，不允許 `java.lang.*`、`Date()` 和 `g.V().tryNext().orElseGet()`。

元素上的屬性

Neptune 不支持 TinkerPop 3.7.0 中引入的`materializeProperties`標誌以返回元素的屬性。其結果是，Neptune 仍然只返回頂點或邊作為引用只是他們的`id`和`label`。

指令碼執行

所有查詢開頭必須為周遊物件 `g`。

在字串查詢提交中，您可以發出多重周遊，並以分號 (`;`) 或換行符號字元 (\n) 分隔。若要執行，除了最後一個以外的每個陳述式結尾必須是 `.iterate()` 步驟。只有最後的周遊資料會傳回。請注意，這不適用於 GLV ByteCode 查詢提交。

工作階段

Neptune 中的工作階段僅限持續 10 分鐘。如需詳細Gremlin 指令碼型工作階段資訊，請參閱〈TinkerPop工作階段參考〉。

交易

Neptune 會在每個 Gremlin 周遊開始時開啟新交易，並在周遊成功完成時關閉交易。發生錯誤時交易將還原。

以分號 (;) 或換行符號字元 (\n) 分隔的多重陳述式包含在單一交易內。除了最後一個以外的每個陳述式結尾必須為要執行的 `next()` 步驟。只有最後的周遊資料會傳回。

使用 `tx.commit()` 和 `tx.rollback()` 的手動交易邏輯不受支援。

> **Important**
>
> 這「只」適用於以「文字字串」傳送 Gremlin 查詢的方法 (請參閱 Gremlin 交易)。

頂點和邊緣 ID

Neptune Gremlin 頂點和邊緣 ID 必須為 `String` 類型。這些 ID 字串支援 Unicode 字元，且大小不能超過 55 MB。

使用者提供的 ID 受到支援，但它們在正常使用狀況下為選用。如果您在新增頂點或邊緣時未提供 ID，Neptune 會產生 UUID 並將其轉換為字串，格式如下：`"48af8178-50ce-971a-fc41-8c9a954cea62"`。這些 UUID 不符合 RFC 標準，因此，如果您需要標準 UUID，則應在外部產生它們，並在您新增頂點或邊緣時提供它們。

> **Note**
>
> 不過，Neptune Load 命令要求您使用 Neptune CSV 格式的 ~id 欄位提供 ID。

使用者提供的 ID

使用者提供的 ID 允許使用在 Neptune Gremlin，條文如下。

- 提供的 ID 是選用的。
- 僅支援頂點和邊緣。

- 僅支援 String 類型。

若要建立使用自訂 ID 的新頂點，請使用 property 步驟搭配 id 關鍵字：g.addV().property(id, 'customid')。

> **Note**
>
> 不要將引號放在 id 關鍵字旁邊。它指的是 T.id。

所有的頂點 ID 必須各不相同，且所有的邊緣 ID 必須各不相同。不過，Neptune 確實允許頂點和邊緣具有相同的 ID。

如果您嘗試使用 g.addV() 建立新頂點，而使用該 ID 的頂點已存在，則操作會失敗。例外狀況為，如果您為頂點指定新的標籤，操作即會成功，但會將新標籤及指定的任何其他屬性新增至現有的頂點。不會覆寫任何項目。此舉不會建立新的頂點。頂點 ID 將維持不變，仍保有其唯一性。

例如，以下的 Gremlin 主控台命令會成功：

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

頂點屬性 ID

Vertex 屬性 ID 將自動產生，且查詢時可顯示為正數或負數。

頂點屬性的基數

Neptune 支援成組基數和單一基數。如未指定，則選取設定基數。這表示，如果您設定屬性值，它將新增新的值至屬性，但前提是其不能出現在值組內。此為 Set 的 Gremlin 列舉值。

不支援 List。如需有關屬性基數的詳細資訊，請參閱 Girmlin 中的頂點主題。 JavaDoc

更新頂點屬性

若要更新屬性值，而不新增額外的值給值組，請在 property 步驟中指定 single 基數。

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

這會移除屬性所有現有的值。

標籤

Neptune 支援頂點的多個標籤。當您建立標籤時，您可以指定多重標籤並用 `::` 分隔。例如，`g.addV("Label1::Label2::Label3")` 將新增有三種不同標籤的頂點。`hasLabel` 步驟將比對此頂點和這三個標籤：`hasLabel("Label1")`、`hasLabel("Label2")` 和 `hasLabel("Label3")`。

> **Important**
>
> `::` 分隔符號僅針對本用途保留。您不能在 `hasLabel` 步驟中指定多重標籤。例如，`hasLabel("Label1::Label2")` 不符合任何內容。

逸出字元

Neptune 會解析所有逸出字元，如 Apache Groovy 語言文件的逸出特殊字元一節所述。

Groovy 限制

Neptune 不支援開頭不是 g 的 Groovy 命令。這包括數學 (例如 1+1)、系統呼叫 (例如 `System.nanoTime()`) 和變數定義 (例如 1+1)。

> **Important**
>
> Neptune 不支援完整的類別名稱。例如，您必須在 Groovy 請求中使用 `single`，而不是 `org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single`。

序列化

Neptune 根據請求的 MIME 類型支援下列序列化。

MIME 類型	序列化	組態
`application/vnd.gremlin-v1.0+gryo`	`GryoMessageSerializerV1`	`ioRegistries: [org.apache.tinkerpop.gremlin.tinker`

		graph.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 （僅適用於 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.gr      GraphBinaryMessage
aphbinary-v1.0          SerializerV1
```

雖然 Neptune 支援這些不同的序列化器類型，但其使用指引非常簡單。如果您透過 HTTP 連線到 Neptune，請在替代版本 GraphSON 3 中將其作`application/vnd.gremlin-v3.0+json;types=false`為嵌入式類型的使用優先順序，因此使用起來很複雜。如果您使用的是 Apache TinkerPop 驅動程式，您可能不需要像使用預設值那樣做任何選擇`application/vnd.graphbinary-v1.0`。由於舊有原因，剩餘的格式仍然存在。

> **Note**
>
> 這裡顯示的序列化表是指命名為 TinkerPop 3.7.0。如果您想進一步了解此變更，請參閱TinkerPop 升級文件。Gryo 序列化支援已在 3.4.3 中棄用，並在 3.6.0 中正式移除。如果您明確使用 Gryo 或預設使用 Gryo 的驅動程式版本，則應切換至 GraphBinary 或升級驅動程式。

Lambda 步驟

Neptune 不支援 Lambda 步驟。

不支援的 Gremlin 方法

Neptune 不支援以下 Gremlin 方法：

- `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`

例如，不允許以下周遊：`g.V().addE('something').from(__.V().next()).to(__.V().next())`。

> **Important**
>
> 這「只」適用於以「文字字串」傳送 Gremlin 查詢的方法。

不支援的 Gremlin 步驟

Neptune 不支援以下 Gremlin 步驟：

- Neptune 中僅部分支援 Gremlin io( ) 步驟。它可以在讀取內容中使用，如 `g.io(`*`url`*`).read()` 中所示，但不能寫入。

Neptune 中的 Gremlin 圖形功能

Gremlin 的 Neptune 實作不會公開 `graph` 物件。下表列出 Grimlin 功能，並指出 Neptune 是否支持它們。

Neptune 對 **`graph`** 功能的支援

Neptune 圖形功能 (如果支援) 與 `graph.features()` 命令將傳回的功能相同。

圖形功能	已啟用？
`Transactions`	true
`ThreadedTransactions`	false
`Computer`	false
`Persistence`	true
`ConcurrentAccess`	true

Neptune 對變數功能的支援

變數功能	已啟用？
`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

Neptune 對頂點功能的支援

頂點功能	已啟用？
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

Neptune 對頂點屬性功能的支援

頂點屬性功能	已啟用？
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

Neptune 對邊緣功能的支援

邊緣功能	已啟用？
AddEdges	true
RemoveEdges	true

UserSuppliedIds	true
AddProperty	true
RemoveProperty	true
NumericIds	false
StringIds	true
UuidIds	false
CustomIds	false
AnyIds	false

Neptune 對邊緣屬性功能的支援

邊緣屬性功能	已啟用？
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

Amazon Neptune 中的 SPARQL 標準合規

在列出適用的 SPARQL 標準之後，以下各節提供有關 Neptune 的 SPARQL 實作如何擴展或從這些標準分出來的特定詳細資料。

主題

- SPARQL 的適用標準
- Neptune SPARQL 中的預設命名空間字首
- SPARQL 預設圖形和具名圖形
- Neptune 支援的 SPARQL XPath 建構子函數
- 查詢和更新的預設基礎 IRI
- Neptune 中的 xsd:dateTime 值
- 特殊浮點值的 Neptune 處理
- Neptune 任意長度值的限制
- Neptune 擴充 SPARQL 中的等於比較
- 處理 Neptune SPARQL 中超出範圍的常值

Amazon Neptune 在實作 SPARQL 圖形查詢語言時符合下列標準。

SPARQL 的適用標準

- SPARQL 是依 2013 年 3 月 21 日的 W3C SPARQL 1.1 查詢語言建議定義。
- SPARQL 更新協定和查詢語言是依 W3C SPARQL 1.1 更新規格定義。
- 對於數值格式，SPARQL 遵循 W3C XML 結構描述定義語言 (XSD) 1.1 第 2 部分：Datatypes 規格，其與 IEEE 754 規格一致 (IEEE 754-2019 - 浮點數運算的 IEEE 標準；如需詳細資訊，另請參閱 Wikipedia IEEE 754 頁面)。不過，此規格不包含於 IEEE 754-1985 版本後引入的功能。

Neptune SPARQL 中的預設命名空間字首

Neptune 定義下列 SPARQL 查詢預設使用的字首。如需詳細資訊，請參閱 SPARQL 規格中的 Prefixed Names。

- `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#`

SPARQL 預設圖形和具名圖形

Amazon Neptune 會將每個三元組與一個具名圖形建立關聯。預設圖形是定義成所有具名圖形的聯集。

查詢的預設圖形

若您提交 SPARQL 查詢而未透過 `GRAPH` 關鍵字或 `FROM NAMED` 之類的建構式明確指定圖形，Neptune 一律會考慮資料庫執行個體中的所有三元組。例如，以下查詢將從 Neptune SPARQL 端點傳回所有三元組：

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

顯現於多個圖形中的三元組將僅傳回一次。

如需預設圖形規格的相關資訊，請參閱 SPARQL 1.1 查詢語言規格的 RDF 資料集一節。

指定具名圖形以進行載入、插入或更新

如果您在載入、插入或更新三元組時未指定具名圖形，Neptune 會使用由 URI `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` 定義的備用具名圖形。

當您使用三元組類型的格式發出 Neptune Load 請求時，可使用 `parserConfiguration: namedGraphUri` 參數指定具名圖形以用於所有三元組。如需 Load 命令語法的相關資訊，請參閱 the section called “載入器命令”。

> **Important**
>
> 若您既沒有使用此參數也未指定具名圖形，則將使用備用 URI：`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`。

如果透過 SPARQL UPDATE 載入三元組而未明確提供具名圖形目標，則亦將使用上述的備用具名圖形。

您可以使用四元組格式的 N-Quads，為資料庫中的每個三元組指定具名圖形。

> **Note**
>
> 使用 N-Quads 允許您將具名圖形留空不填。在此情況下即會使用 `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`。
> 您可以使用 `namedGraphUri` 剖析器組態選項，覆寫 N-Quads 的預設具名圖形。

Neptune 支援的 SPARQL XPath 建構子函數

SPARQL 標準允許 SPARQL 引擎支援一組可擴展的 XPath 建構子函數。Neptune 目前支援以下建構子函數，其中 `xsd` 字首定義為 `http://www.w3.org/2001/XMLSchema#`：

- `xsd:boolean`
- `xsd:integer`
- `xsd:double`
- `xsd:float`
- `xsd:decimal`
- `xsd:long`
- `xsd:unsignedLong`

查詢和更新的預設基礎 IRI

因為 Neptune 叢集具有數個不同的端點，所以在解析相對 IRI 時，使用查詢或更新的請求 URL 做為基礎 IRI 可能會產生非預期的結果。

從引擎 1.2.1.0 版開始，如果明確的基礎 IRI 不是請求的一部分，Neptune 會使用 `http://aws.amazon.com/neptune/default/` 做為基礎 IRI。

在以下請求中，基礎 IRI 是請求的一部分：

```
BASE <http://example.org/default/>
INSERT DATA { <node1> <id> "n1" }

BASE <http://example.org/default/>
SELECT * { <node1> ?p ?o }
```

結果將是：

```
?p                                                  ?o
http://example.org/default/id                       n1
```

不過，在此請求中，不包括基礎 IRI：

```
INSERT DATA { <node1> <id> "n1" }

SELECT * { <node1> ?p ?o }
```

在此情況下，結果將是：

```
?p                                                  ?o
http://aws.amazon.com/neptune/default/id            n1
```

Neptune 中的 xsd:dateTime 值

基於效能考量，Neptune 一律會將日期/時間值儲存為國際標準時間 (UTC)。這能讓直接比較很有效率。

這也表示如果您輸入指定特定時區的 `dateTime` 值，Neptune 就會將值轉換為 UTC 並捨棄該時區資訊。接下來，當您收到 `dateTime` 後，該執會以 UTC 呈現而非原始時區的時間，而且您也無法再判斷原始時區。

特殊浮點值的 Neptune 處理

Neptune 處理 SPARQL 特殊浮點值的方式如下。

Neptune 中的 SPARQL NaN 處理

在 Neptune 中，SPARQL 可以接受查詢中的 NaN 值。signalling 和 quiet NaN 值之間沒有任何區別。Neptune 會將所有 NaN 值視為 quiet。

在語意上，不可能進行 NaN 的比較，因為沒有任何值大於、小於或等於 NaN。這表示理論上，比較一端的 NaN 值和另一端的「任何內容」永遠不相符。

不過，XSD 規格確實將兩個 `xsd:double` 或 `xsd:float` NaN 值視為相等。對於 IN 篩選條件、篩選條件表達式的等於運算子，以及完全相符語義 (在三重模式的物件位置中具有 NaN)，Neptune 皆遵循此原則。

Neptune 中的 SPARQL 無限值處理

在 Neptune 中，SPARQL 可接受查詢中的 `INF` 或 `-INF` 值。`INF` 會相比為大於任何其他數值，而 `-INF` 會相比為小於任何其他數值。

比較具有相符符號的兩個 INF 值時，無論其類型如何皆彼此相等 (例如，float `-INF` 等於 double `-INF`)。

當然，因為沒有任何值大於、小於或等於 NaN，所以不可能和 NaN 比較。

Neptune 中的 SPARQL 負零處理

Neptune 會將負零值標準化為不帶正負號的零。負零值可用於查詢中，但不會原樣記錄在資料庫中，而且不等於不帶正負號的零。

Neptune 任意長度值的限制

Neptune 將 SPARQL 中的 XSD 整數、浮點數和小數值的儲存大小限制在 64 位元。使用較大的值會導致 `InvalidNumericDataException` 錯誤。

Neptune 擴充 SPARQL 中的等於比較

SPARQL 標準為值運算式定義了三元邏輯，其中，值運算式可評估為 `true`、`false` 或 `error`。SPARQL 1.1 規格中定義的術語相等性的預設語義適用於 FILTER 條件中的 = 與 != 比較，這會在比較規格中運算子資料表中未明確比較的資料類型時產生一個 `error`。

這種行為可能會導致不直觀的結果，如下列範例所示。

資料：

```
<http://example.com/Server/1> <http://example.com/ip> "127.0.0.1"^^<http://example.com/
datatype/IPAddress>
```

查詢 1：

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o = "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

查詢 2：

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o != "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

透過 Neptune 在 1.0.2.1 版本以前所使用的預設 SPARQL 語義，這兩個查詢都會傳回空結果。原因是 `?o = "127.0.0.2"^^<http://example.com/IPAddress>` 在評估 `?o := "127.0.0.1"^^<http://example.com/IPAddress>` 時會產生 `error`，而不是 `false`，因為沒有為自訂資料類型 `<http://example.com/IPAddress>` 指定明確的比較規則。因此，第二個查詢中的否定版本也會產生 `error`。在這兩個查詢中，`error` 會導致篩選出候選解決方案。

從 1.0.2.1 版開始，Neptune 已根據規格擴展 SPARQL 不等式運算子。請參閱 SPARQL 第 1.1 節，運算子可擴充性，讓引擎定義如何跨使用者定義和不可比較內建資料類型進行比較的其他相關規則。

使用此選項，如果常值和資料類型在語法上相等，則 Neptune 會立即將在運算子對應表中未明確定義的任何兩個資料類型的比較視同評估為 `true`，否則為 false。任何情況下都不會發生 `error`。

使用這些新的語意，第二個查詢就會傳回 `"127.0.0.1"^^<http://example.com/IPAddress>`，而不是空的結果。

處理 Neptune SPARQL 中超出範圍的常值

XSD 語義會定義每個數值類型的值空間，但 `integer` 和 `decimal` 除外。這些定義會將每種類型限制在某個範圍的值內。例如，範圍 `xsd:byte` 的範圍為 -128 到 +127 (頭尾皆含)。任何超出此範圍的值都視為無效。

如果您嘗試在類型的值空間之外指派常值 (例如，如果您嘗試將文字值設定`xsd:byte`為 999)，Neptune 會以原樣接受該 out-of-range 值，而不會四捨五入或截斷它。但因為指定的類型不能代表此值，所以不會保存為數值。

也就是說，即使 `"999"^^xsd:byte` 是已定義之 `xsd:byte` 值範圍外的值，Neptune 也會接受它。但是，在資料庫中保存該值之後，此值只能用於三重模式物件位置的完全相符語義。沒有範圍篩選可以在其上執行，因為 out-of-range 常值不會被視為數值。

SPARQL 1.1 規格會以格式 `numeric-operator-numeric`、`string-operator-string`、`literal-operator-literal` 等等定義範圍運算子。Neptune 無法執行如 `invalid-literal-operator-numeric-value` 之類的範圍比較運算子。

Amazon Neptune 中的開放密碼規範合規

OpenCypher 的 Amazon Neptune 版本通常支援目前 OpenCypher 規格 (即 Cypher 查詢語言參考第 9 版) 中定義的子句、運算子、運算式、函數和語法。下面列出了 Neptune 對 OpenCypher 支援的限制和差異。

> **Note**
>
> Cypher 的目前 Neo4j 實作包含了上述 OpenCypher 規格中未包含的功能。如果您要將目前 Cypher 程式碼遷移至 Neptune，請參閱 Neptune 與 Neo4j 的相容性 和 重寫 Cypher 查詢以在 Neptune 上的 OpenCpher 中執行 以取得詳細資訊。

Neptune 中對 openCypher 子句的支援

Neptune 支援下列子句，除非另有說明：

- `MATCH` – 支援，但目前不支援 *`shortestPath()`* 和 *`allShortestPaths()`*。
- `OPTIONAL MATCH`
- *`MANDATORY MATCH`* – 目前在 Neptune 中不支援。不過，Neptune 確實支援 `MATCH` 查詢中的自訂 ID 值。
- `RETURN` – 支援，但與 `SKIP` 或 `LIMIT` 的非靜態值搭配使用時除外。例如，目前下列項目無法運作：

```
MATCH (n)
RETURN n LIMIT toInteger(rand())    // Does NOT work!
```

- `WITH` – 支援，但與 `SKIP` 或 `LIMIT` 的非靜態值搭配使用時除外。例如，目前下列項目無法運作：

```
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 可讓您在 `CREATE` 查詢中建立自訂 ID 值。
- `DELETE`
- `SET`
- `REMOVE`
- `MERGE` – Neptune 支援 `MERGE` 查詢中的自訂 ID 值。
- *`CALL[YIELD...]`* – 目前在 Neptune 中不支援。
- `UNION, UNION ALL` – 支援唯讀查詢，但目前不支援變動查詢。

Neptune 中對 openCypher 運算子的支援

Neptune 支援下列運算子，除非另有說明：

一般運算子

- `DISTINCT`
- 用於存取巢狀常值映射屬性的 **.** 運算子。

數學運算子

- **+** 加法運算子。
- **-** 減法運算子。
- ***** 乘法運算子。
- **/** 除法運算子。

- **%** 模數除法運算子。
- **^** 指數運算子####。

比較運算子

- **=** 加法運算子。
- **<>** 不等式運算子。
- 支援 **<** 小於運算子，但其中一個引數是 Path、List 或 Map 除外。
- 支援 **>** 大於運算子，但其中一個引數是 Path、List 或 Map 除外。
- 除非其中一個引數是「路徑」、「清單」或「對映」，否則支援 **<=** less-than-or-equal-to 運算子。
- 除非其中一個引數是「路徑」、「清單」或「對映」，否則支援 **>=** greater-than-or-equal-to 運算子。
- `IS NULL`
- `IS NOT NULL`
- 如果要搜尋的資料是字串，則支援 `STARTS WITH`。
- 如果要搜尋的資料是字串，則支援 `ENDS WITH`。
- 如果要搜尋的資料是字串，則支援 `CONTAINS`。

布林值運算子

- `AND`
- `OR`
- `XOR`
- `NOT`

字串運算子

- **+** 串連運算子。

清單運算子

- **+** 串連運算子。
- `IN` (檢查清單中是否存在項目)

Neptune 中對 openCypher 運算式的支援

Neptune 支援下列運算式，除非另有說明：

- CASE
- Neptune 中目前不支援 **[]** 運算式，用於存取節點、關係或映射內動態計算的屬性索引鍵。例如，下列項目無法運作：

```
MATCH (n)
WITH [5, n, {key: 'value'}] AS list
RETURN list[1].name
```

Neptune 中對 openCypher 函數的支援

Neptune 支援下列函數，除非另有說明：

述詞函數

- exists()

純量函數

- coalesce()
- endNode()
- epochmillis()
- head()
- id()
- last()
- length()
- randomUUID()
- properties()
- removeKeyFromMap
- size() – 此過載方法目前僅適用於模式運算式、清單和字串
- startNode()
- timestamp()

- toBoolean()
- toFloat()
- toInteger()
- type()

彙總函數

- avg()
- collect()
- count()
- max()
- min()
- percentileDisc()
- stDev()
- percentileCont()
- stDevP()
- sum()

列出函數

- join() (將清單中的字串串連為單一字串)
- keys()
- labels()
- nodes()
- range()
- relationships()
- reverse()
- tail()

數學函數 – 數字

- abs()

- ceil()
- floor()
- rand()
- round()
- sign()

數學函數 – 對數

- e()
- exp()
- log()
- log10()
- sqrt()

數學函數 – 三角函數

- acos()
- asin()
- atan()
- atan2()
- cos()
- cot()
- degrees()
- pi()
- radians()
- sin()
- tan()

字串函數

- join() (將清單中的字串串連為單一字串)

- left()
- lTrim()
- replace()
- reverse()
- right()
- rTrim()
- split()
- substring()
- toLower()
- toString()
- toUpper()
- trim()

使用者定義的函數

Neptune 中目前不支援########。

Neptune 特定的 openCypher 實作詳細資訊

以下各節描述了其中 OpenCypher 的 Neptune 實作可能不同於或超出 OpenCypher 規格的方式。

Neptune 中的可變長度路徑（VLP）評估

可變長度路徑 (VLP) 評估會探索圖形中節點之間的路徑。路徑長度可在查詢中不受限制。為了防止循環，OpenCypher 規格指定每個解決方案必須周遊每個邊緣最多一次。

對於 VLP，Neptune 實作會偏離 OpenCypher 規格，因為它只支援屬性等式篩選條件的常數值。採取下列查詢：

```
MATCH (x)-[:route*1..2 {dist:33, code:x.name}]->(y) return x,y
```

因為 x.name 屬性等式篩選條件值不是常數，所以此查詢會產生 UnsupportedOperationException 並顯示訊息：Property predicate over variable-length relationships with non-constant expression is not supported in this release.

Neptune 開放密碼實現中的時間支持（Neptune 數據庫 1.3.1.0 及以下版本）

Neptune 目前為 OpenCypher 中的時間函數提供了有限的支援。它支援時間類型的 `DateTime` 資料類型。

`datetime()` 函數可以用來取得目前 UTC 日期和時間，如下所示：

```
RETURN  datetime() as res
```

日期和時間值可以從儲存在 Neptune 中的資料轉換而來，如下所示：

```
MATCH (n) RETURN datetime(n.createdDate)
```

日期和時間值可從 "dateTtime" 格式的字串中剖析而來，其中 date 和 time 皆以下面的其中一種支援形式表示：

支援的日期格式

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

支援的時間格式

- `HH:mm:ssZ`
- `HHmmssZ`
- `HH:mm:ssZ`
- `HH:mmZ`
- `HHmmZ`
- `HHZ`
- `HHmmss`
- `HH:mm:ss`

- HH:mm
- HHmm
- HH

例如：

```
RETURN datetime('2022-01-01T00:01')      // or another example:
RETURN datetime('2022T0001')
```

請注意，Neptune OpenCypher 中的所有日期/時間值都以 UTC 值的形式儲存和擷取。

Neptune OpenCypher 使用 `statement` 時鐘，這表示在整個查詢期間使用相同的時刻。同一交易中的不同查詢可能會使用不同的時刻。

Neptune 不支援在呼叫 `datetime()` 時使用函數 。例如，下列項目無法運作：

```
CREATE (:n {date:datetime(tostring(2021))})  // ---> NOT ALLOWED!
```

Neptune 確實支援將 `datetime` 轉換為 `epochmillis` 的 `epochmillis()` 函數。例如：

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

Neptune 目前不支援 `DateTime` 物件上的其他函數和操作，例如加法和減法。

Neptune OpenCypher 實現中的時間支持（Neptune 分析和 Neptune 數據庫 1.3.2.0 及更高版本）

下列適用於的日期時間功能 OpenCypher 適用於 Neptune 分析。或者，您也可以使用 labmode 參數`DatetimeMillisecond=enabled`，在 Neptune 引擎版本 1.3.2.0 及更新版本上啟用下列日期時間功能。如需在 labmode 中使用此功能的詳細資訊，請參閱擴展的日期時間支持。

- Support 毫秒。日期時間文字將始終以毫秒返回，即使毫秒為 0。（以前的行為是截斷毫秒。）

```
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 通過存儲的屬性或中間結果調用 datetime ( ) 函數。例如，在此功能之前，無法執行下列查詢。

  日期時間 ( ) 超過屬性：

  ```
  // 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
  ```

  日期時間 ( ) 超過中間結果：

  ```
  // Parse datetime from parameter
  UNWIND $list as myDate
  RETURN datetime(myDate) as d
  ```

- 現在也可以保存在上述情況下創建的日期時間屬性。

  將日期時間從一個屬性的字符串屬性保存到另一個屬性：

  ```
  // 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)
```

從具有日期時間屬性的參數 Batch 創建節點：

```
// 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 較大的 ISO8601 日期時間格式子集。請參閱以下 。

支援的格式

日期時間值的格式是 [日期] T [時間] [時區]，其中 T 是分隔符。如果未提供明確的時區，則假設 UTC (Z) 為預設值。

時區

支援的時區格式如下：

- +/-HH: 毫米
- +/#-嗯
- +/-H

日期時間字符串中的時區是可選的。如果時區偏移量為 0，則可以使用 Z 代替上面的時區後綴來表示 UTC 時間。支持的時區範圍是從-下午 14 點到下午 2 點之間。

日期

如果沒有時區，或時區為 UTC (Z)，則支援的日期格式如下：

> **Note**
>
> DDD 指的是序數日期，代表從 001 到 365（閏年 366）的一年中的某一天。例如，2024-002 代表二零二四年一月二日。

- yyyy-MM-dd
- yyyyMMdd
- yyyy-MM
- yyyyMM
- yyyy-DDD
- yyyyDDD
- yyyy

如果選擇 Z 以外的時區，則支援的日期格式僅限於以下內容：

- yyyy-MM-dd
- yyyy-DDD
- yyyyDDD

支援的日期範圍為 1400-1 月 1 日至 99 年 12 月 31 日。

時間

如果沒有時間段存在，或時區為 UTC (Z)，則支援的時間格式為：

- HH:mm:ss.SSS
- HH:mm:ss
- HHmmss.SSS
- HHmmss
- HH:mm
- HHmm
- HH

如果選擇 Z 以外的時區，則支援的時間格式僅限於以下內容：

- `HH:mm:ss`
- `HH:mm:ss.SSS`

Neptune openCypher 語言語義中的差異

Neptune 會將節點和關係 ID 表示為字串，而非整數。ID 等於透過資料載入器提供的 ID。如果資料欄有命名空間，則命名空間加上 ID。因此，`id` 函數會傳回字串而非整數。

`INTEGER` 資料類型限制為 64 位元。使用 `TOINTEGER` 函數將較大的浮點數值或字串值轉換為整數時，負值會截斷為 `LLONG_MIN`，而正值會截斷為 `LLONG_MAX`。

例如：

```
RETURN TOINTEGER(2^100)
>  9223372036854775807

RETURN TOINTEGER(-1 * 2^100)
>  -9223372036854775808
```

Neptune 特定 **`join()`** 函數

Neptune 實作一個不存在於 openCypher 規格中的 `join()` 函數。它從字串常值清單和字串分隔符號中建立一個字串常值。它需要兩個引數：

- 第一個引數是字串常值的清單。
- 第二個引數是分隔符號，它可以由零個、一個或多個字元組成。

範例：

```
join(["abc", "def", "ghi"], ", ")    // Returns "abc, def, ghi"
```

Neptune 特定 **`removeKeyFromMap()`** 函數

Neptune 實作一個不存在於 openCypher 規格中的 `removeKeyFromMap()` 函數。它從映射中刪除指定的索引鍵，並傳回產生的新映射。

函數需要兩個引數：

- 第一個引數是要從中移除索引鍵的映射。
- 第二個引數是要從映射中移除的索引鍵。

在您想要透過展開映射清單來設定節點或關係值的情況下，removeKeyFromMap() 函數特別有用。例如：

```
UNWIND [{`~id`: 'id1', name: 'john'}, {`~id`: 'id2', name: 'jim'}] as val
CREATE (n {`~id`: val.`~id`})
SET n = removeKeyFromMap(val, '~id')
```

節點和關係屬性的自訂 ID 值

從引擎 1.2.0.2 版開始，Neptune 已擴展 OpenCypher 規格，因此您現在可以在 CREATE、MERGE 和 MATCH 子句中指定節點和關係的 id 值。這可讓您指派易於使用的字串，而不是系統產生的 UUID 來識別節點和關係。

> **Warning**
>
> OpenCypher 規格的這個延伸模組不與舊版相容，因為 ~id 現在被視為保留的屬性名稱。如果您已在資料和查詢中使用 ~id 做為屬性，則必須將現有屬性遷移至新的屬性索引鍵，並移除舊的屬性索引鍵。請參閱如果您目前使用 ~id 做為屬性，該怎麼辦。

以下範例展示如何建立具有自訂 IDS 的節點和關聯性：

```
CREATE (n {`~id`: 'fromNode', name: 'john'})
  -[:knows {`~id`: 'john-knows->jim', since: 2020}]
  ->(m {`~id`: 'toNode', name: 'jim'})
```

如果您嘗試建立已在使用中的自訂 ID，Neptune 會擲回 DuplicateDataException 錯誤。

以下是在 MATCH 子句中使用自訂 ID 的範例：

```
MATCH (n {`~id`: 'id1'})
RETURN n
```

以下是在 MERGE 子句中使用自訂 ID 的範例：

```
MATCH (n {name: 'john'}), (m {name: 'jim'})
```

```
MERGE (n)-[r {`~id`: 'john->jim'}]->(m)
RETURN r
```

如果您目前使用 **~id** 做為屬性，該怎麼辦

使用引擎 1.2.0.2 版，OpenCypher 子句中的 ~id 金鑰現在被視為 id，而不是做為一個屬性。這表示，如果您有一個名為 ~id 的屬性，將無法存取它。

如果您使用的是 ~id 屬性，則在升級至引擎版本 1.2.0.2 或更高版本之前必須做的事情，首先是將現有 ~id 屬性遷移至新的屬性索引鍵，然後刪除 ~id 屬性。例如，下面查詢：

- 為所有節點建立一個名為 'newId' 的新屬性、
- 將 '~id' 屬性的值複製到 'NewID' 屬性，
- 然後從資料中刪除 '~id' 屬性

```
MATCH (n)
WHERE exists(n.`~id`)
SET n.newId = n.`~id`
REMOVE n.`~id`
```

對於資料中具有 ~id 屬性的任何關係，也需要執行相同的操作。

您還必須變更您正在使用且參考 ~id 屬性的任何查詢。例如，這個查詢：

```
MATCH (n)
WHERE n.`~id` = 'some-value'
RETURN n
```

...將會變更這樣：

```
MATCH (n)
WHERE n.newId = 'some-value'
RETURN n
```

Neptune openCypher 與 Cypher 之間的其他差異

- Neptune 僅支援 Bolt 通訊協定的 TCP 連線。 WebSockets不支援螺栓的連接對。
- Neptune openCypher 會移除 trim()、ltrim() 和 rtrim() 函數中由 Unicode 定義的空格。

- 在 Neptune OpenCypher 中，`tostring(double)` 不會自動切換到 E 表示法，表示 double 的大值。
- 雖然 OpenCypher CREATE 不會建立多值屬性，但它們可以存在於使用 Grimlin 建立的資料中。如果 Neptune OpenCypher 遇到一個多值屬性，其中一個值是任意選擇的，這會建立一個非確定性的結果。

Neptune 圖形資料模型。

Amazon Neptune 圖形資料的基本單位是四個位置 (quad) 元素，類似資源描述架構 (RDF) quad。以下是 Neptune quad 的四個位置：

- `subject  (S)`
- `predicate  (P)`
- `object  (O)`
- `graph  (G)`

每個 quad 是一個陳述式，可做出關於一個或多個資源的宣告。陳述式可以宣告兩個資源之間的關係，或者將屬性 (鍵/值組) 連接到資源。您一般可以將 quad 述詞值視為陳述式的動詞。它會描述正在定義的關係或屬性的類型。物件是關係的目標，或是屬性的值。範例如下：

- 可透過將來源頂點識別符存放在 S 位置中、將目標頂點識別符存放在 O 位置中，以及將邊緣標籤存放在 P 位置，來表示兩個頂點之間的關係。
- 可透過將元素識別符存放在 S 位置中、將屬性索引鍵存放在 P 位置中，以及將屬性值存放在 O 位置中，來表示屬性。

圖形位置 G 在不同堆疊中的使用方式不同。對於 Neptune 中的 RDF 資料，G 位置包含具名圖形識別符。對於 Gremlin 中的屬性圖，它用於在邊緣的情況下存放邊緣 ID 值。在所有其他情況下，它會預設為固定的值。

具有共用資源識別符的一組 quad 陳述式可建立圖形。

使用者面向值的字典

Neptune 不會將大多數使用者面向值直接儲存在其維護的各種索引中。相反地，它會將它們個別儲存在字典中，並使用 8 位元組識別符在索引中取代它們。

- 將進入 S、P 或 G 索引的所有使用者面向值都以這種方式儲存在字典中。
- 在 O 索引中，數值會直接儲存在索引中 (內嵌)。這包括 `date` 和 `datetime` 值 (以 Epoch 中的毫秒表示)。
- 將進入 O 索引的所有其他使用者面向值都會儲存在字典中，並在索引中以 ID 表示。

字典包含使用者面向值與 `value_to_id` 索引中 8 位元組 ID 的正向映射。

它會將 8 位元組 ID 與值的反向映射儲存在兩個索引之一，取決於值的大小：

- `id_to_value` 索引會將 ID 映射至內部編碼後小於 767 個位元組的使用者面向值。
- `id_to_blob` 索引會將 ID 映射至較大的使用者面向值。

陳述式在 Neptune 中如何編製索引

當您查詢 quad 圖表，您可以對每個 quad 指定值限制，也可以不指定。查詢會傳回符合您指定之值限制的所有 quad。

Neptune 會使用索引來解決查詢。Andreas Harth 和 Stefan Decker 在其 2005 年的文章 Optimized Index Structures for Querying RDF from the Web 中指出，四 quad 位置有 16 個 (2^4) 可能的存取模式。您可以有效率地查詢所有 16 個模式，而無需使用六個 quad 陳述式索引進行掃描和篩選。每個 quad 陳述式索引使用由以不同順序串連的四個位置值組成的索引鍵。

```
     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
```

根據預設，Neptune 只會建立和維護這六個索引中的三個：

- SPOG – 使用 Subject + Predicate + Object + Graph 組成的索引鍵。
- POGS – 使用 Predicate + Object + Graph + Subject 組成的索引鍵。
- GPSO – 使用 Graph + Predicate + Subject + Object 組成的索引鍵。

這三個索引會處理許多最常見的存取模式。只維護三個完整陳述式索引 (而非六個)，可大幅減少支援不經掃描和篩選而快速存取所需的資源。例如，每當頂點或頂點和屬性識別符等位置字首繫結時，SPOG 索引會允許有效查詢。只有存放於 P 位置的邊緣或屬性標籤繫結時，POGS 索引才會允許有效存取。

尋找陳述式的低層級 API 採用的陳述式模式，其中有些位置已知，而其餘則保留供索引搜尋探索。透過根據其中一個陳述式索引的索引鍵順序，在索引鍵字首索引鍵中構成已知位置，Neptune 執行範圍掃描來擷取符合已知位置的所有陳述式。

不過，根據預設，Neptune 未建立的其中一個陳述式索引是反向周遊 OSGP 索引，它可以跨物件和主題收集述詞。Neptune 預設會改為在個別索引中追蹤它用來進行 {all P x POGS} 聯合掃描的不同述詞。使用 Gremlin 時，述詞對應到屬性或邊緣標籤。

如果圖形中的不同述詞量很大時，預設的 Neptune 存取策略會變得沒有效率。例如，在 Gremlin 中，未指定邊緣標籤的 in() 步驟，或內部使用 in() 的任何步驟 (例如 both() 或 drop()) 可能會變得沒有效率。

使用實驗室模式啟用 OSGP 索引建立

如果您的資料模型建立大量不同的述詞，可能會遇到效能降低和營運成本增加的情況，這些可藉由使用實驗室模式來啟用 OSGP 索引 (除了 Neptune 根據預設維護的三個索引以外) 而大幅改善。

> **Note**
>
> 此功能從 Neptune 引擎版本 1.0.1.0.200463.0開始可用。

啟用 OSGP 索引可能會有幾個缺點：

- 插入速率最多可能減慢 23%。
- 儲存量增加高達 20%。
- 均等觸及所有索引的讀取查詢 (這是很少見的情況) 可能會增加延遲。

不過，一般而言，啟用具有大量不同述詞之資料庫叢集的 OSGP 索引是值得的。物件型搜尋變成很有效率 (例如，尋找頂點的所有傳入邊緣，或連線到給定物件的所有主體)，因此，降低頂點也變得更有效率。

> **Important**
>
> 在載入任何資料之前，您只能在空白的資料庫叢集中啟用 OSGP 索引。

Neptune 資料模型中的 Gremlin 陳述式

Gremlin 屬性圖資料在 SPOG 模型中使用三類陳述式來表示，即：

- 頂點標籤陳述式
- 邊緣陳述式
- 屬性陳述式

如需如何在 Gemlin 查詢中使用這些陳述式的說明，請參閱 了解 Gremlin 查詢如何在 Neptune 中運作。

Neptune 查詢快取可加速讀取查詢

Amazon Neptune 會實作查詢快取，其可使用 R5d 執行個體的 NVMe 型 SSD，針對經常重複查詢屬性值或 RDF 常值的查詢改善讀取效能。查詢快取會暫時將這些值儲存在 NVMe SSD 磁碟區中，在這裡可以快速存取它們。

從 Amazon Neptune 引擎版本 1.0.4.2.R2 (2021 年 6 月 1 日) 開始就可以使用這項功能。

如果需要從叢集儲存磁碟區而非記憶體擷取屬性值或常值，則傳回大量頂點和邊緣或許多 RDF 三元組之屬性的讀取查詢可能會有較高的延遲。範例包括長時間執行的讀取查詢，這些查詢會從身分圖傳回大量完整名稱，或從詐騙偵測圖傳回 IP 地址。隨著查詢傳回的屬性值或 RDF 常值數量增加，可用記憶體會減少，而且您的查詢執行效能可能會大幅降低。

Neptune 查詢快取的使用案例

查詢快取僅在您的讀取查詢傳回非常大量頂點和邊緣或 RDF 三元組的屬性時才有幫助。

為了最佳化查詢效能，Amazon Neptune 會使用 R5d 執行個體類型，為這類屬性值或常值建立大型快取。然後，從快取擷取它們會比從叢集儲存磁碟區擷取它們快得多。

根據經驗法則，只有在滿足下列三個條件的情況下，才能啟用查詢快取：

- 您一直觀察到讀取查詢中的延遲增加。
- 執行讀取查詢時，您也會觀察`BufferCacheHitRatio`CloudWatch 量度下降 (請參閱使用 Amazon 監控 Neptune CloudWatch)。
- 在呈現結果之前，您的讀取查詢會花費大量時間實體化傳回值 (請參閱下面的 Gremlin-profile 範例，取得確定要針對一個查詢具體化多少屬性值的方法)。

> **Note**
>
> 此功能「僅」在上述特定案例中有幫助。例如，查詢快取根本無法幫助彙總查詢。除非您正在執行將從查詢快取得益的查詢，否則沒有理由使用 R5d 執行個體類型，而非同等且費用較低的 R5 執行個體類型。

如果您使用的是 Gremlin，您可以使用 Gremlin `profile` API 評估查詢的具體化成本。在「索引操作」下，它會顯示執行期間具體化的術語數量：

```
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
```

具體化的非數字術語數量直接與 Neptune 必須執行的術語查詢數量成正比。

使用查詢快取

查詢快取僅適用於 R5d 執行個體類型，預設會自動將其啟用。Neptune R5d 執行個體具有與 R5 執行個體相同的規格，加上高達 1.8 TB 的本機 NVMe 型 SSD 儲存體。查詢快取是執行個體特定的，且受益的工作負載可以專門導向至 Neptune 叢集中的 R5d 執行個體，而其他工作負載則可以導向至 R5 或其他執行個體類型。

若要在 Neptune 執行個體上使用查詢快取，只要將該執行個體升級至 R5d 執行個體類型即可。當您這樣做時，Neptune 會自動將 neptune_lookup_cache 資料庫叢集參數設定為 `'enabled'`，並在該特定執行個體上建立查詢快取。然後，您可以使用 執行個體狀態 API 來確認快取已啟用。

同樣地，若要在給定執行個體上停用查詢快取，請將執行個體從 R5d 執行個體類型縮減至相等的 R5 執行個體類型。

啟動 R5d 執行個體時，會啟用查詢快取且其處於冷啟動模式中，表示它是空白的。處理查詢時，Neptune 首先會在查詢快取中檢查是否有屬性值或 RDF 常值，並新增它們 (如果尚未存在的話)。這會逐漸預熱快取。

當您將需要屬性值或 RDF 常值查詢的讀取查詢導向至 R5d「讀取器」執行個體時，讀取效能會在其快取預熱時稍微降低。不過，當快取預熱時，讀取效能會大幅加快，而且您也可能會看到與命中快取而非叢集儲存體的查詢相關的 I/O 成本下降。記憶體使用率也會改善。

如果您的「寫入器」執行個體是 R5d，則它會在每次寫入操作時自動預熱其查詢快取。此方法確實會稍微增加寫入查詢的延遲，但會更有效率地預熱查詢快取。然後，如果將需要屬性值或 RDF 常值查詢的讀取查詢導向至寫入器執行個體，則您會立即開始獲得改善的讀取效能，因為已在該處快取這些值。

此外，如果您在 R5d 寫入器執行個體上執行大量載入器，則可能會注意到其效能由於快取而略為降低。

因為查詢快取是每個節點特定的，所以主機更換會將快取重設為冷啟動。

您可以將 neptune_lookup_cache 資料庫叢集參數設定為 `'disabled'`，在資料庫叢集中的所有執行個體上暫時停用查詢快取。不過，一般而言，在特定執行個體上停用快取會更有意義，方法是將它們從 R5d 縮減至 R5 執行個體類型。

Neptune 中的交易語意

Amazon Neptune 旨在透過資料圖形支援高度並行線上交易處理 (OLTP) 工作負載。適用於 RDF 規格的 W3C SPARQL 查詢語言和 Apache TinkerPop 格林圖遍歷語言文件不會定義並行查詢處理的交易語意。由於 ACID 支援和定義良好的交易保證非常重要，因此我們會強制執行嚴格的語義，以協助避免資料異常。

本節會定義這些語義，並說明它們如何套用到 Neptune 中的一些常用案例。

主題

- 隔離層級的定義
- Neptune 中的交易隔離層級
- Neptune 交易語義範例
- 例外狀況處理和重試

隔離層級的定義

ACID 中的 "I" 代表隔離。交易的隔離程度決定了其他並行交易可能影響其操作的資料有多少。

SQL:1992 標準 建立了描述隔離層級的詞彙。它定義了兩個並行交易 (Tx1 和 Tx2) 之間可能發生的三種互動 (它稱為「現象」)：

- `Dirty read` – 當 Tx1 修改項目，然後 Tx2 在 Tx1 遞交了變更之前讀取該項目，就會發生此情況。然後，如果 Tx1 從未成功遞交變更或將其回復，則 Tx2 已讀取從未進入資料庫的值。
- `Non-repeatable read` – 當 Tx1 讀取項目，接著 Tx2 修改或刪除該項目並遞交變更，然後 Tx1 嘗試重新讀取該項目時，便會發生此情況。Tx1 現在會讀取與之前不同的值，或發現該項目不再存在。
- `Phantom read` – 當 Tx1 讀取一組滿足搜尋條件的項目，接著 Tx2 新增符合搜尋條件的新項目，然後 Tx1 重複搜尋時，便會發生此情況。Tx1 現在取得與之前不同的一組項目。

這三種類型的互動，每一種都可能導致資料庫中產生的資料不一致。

SQL:1992 標準定義了四種隔離層級，在三種互動類型以及它們可能產生的不一致性方面具有不同的保證。在所有四個層級，可以保證交易完全執行或完全不執行：

- `READ UNCOMMITTED` – 允許所有三種互動 (亦即，已變更讀取、不可重複讀取及幽靈讀取)。
- `READ COMMITTED` – 已變更讀取是不可行，但不可重複讀取和幽靈讀取卻可行。

- REPEATABLE READ – 已變更讀取或不可重複讀取全都不可行，但幽靈讀取仍然可行。
- SERIALIZABLE – 三種類型的互動現象都不能發生。

多版本並行控制 (MVCC) 允許另一種隔離，即 SNAPSHOT 隔離。這保證在交易開始時，交易可在存在之資料的快照上操作，而且其他交易都不能變更該快照。

Neptune 中的交易隔離層級

Amazon Neptune 針對唯讀查詢和變動查詢實作不同的交易隔離層級。SPARQL 和 Gremlin 查詢會根據下列條件分類為唯讀或變動：

- 在 SPARQL 中，讀取查詢 (SELECT、ASK、CONSTRUCT 和 DESCRIBE，如 SPARQL 1.1 查詢語言規格中所定義) 與變動查詢 (INSERT 和 DELETE，如 SPARQL 1.1 更新規格中所定義) 之間有明確的區別。

  請注意，Neptune 將多個一起提交的變動查詢 (例如，在 POST 訊息中，以分號分隔) 視為單一交易。其為保證作用的原子單位，不是成功就是失敗，若是失敗則會轉返部分變更。
- 不過，在 Gremlin 中，Neptune 會根據是否包含任何查詢路徑步驟 (例如操縱資料的 addE()、addV()、property() 或 drop()) 將查詢分類為唯讀查詢或變動查詢。如果查詢包含任何這類路徑步驟，則會將其分類並做為變動查詢執行。

也可以在 Gremlin 中使用常設工作階段。如需詳細資訊，請參閱 Gremlin 指令碼型工作階段。在這些工作階段中，所有查詢 (包括唯讀查詢) 都會在與寫入器端點上變動查詢相同的隔離下執行。

在 openCypher 中使用 Bolt 讀寫工作階段，所有查詢 (包括唯讀查詢) 都會在與寫入器端點上變動查詢相同的隔離下執行。

主題

- Neptune 中的唯讀查詢隔離
- Neptune 中的變動查詢隔離
- 使用鎖定等待逾時的衝突解決機制
- 範圍鎖定和錯誤衝突

Neptune 中的唯讀查詢隔離

Neptune 會依據快照隔離語義來評估唯讀查詢。這表示唯讀查詢會在邏輯上對查詢評估開始時所取得之資料庫的一致快照進行操作。然後，Neptune 可以保證不會發生以下任何現象：

- `Dirty reads` – Neptune 中的唯讀查詢絕不會看到來自並行交易的未遞交資料。
- `Non-repeatable reads` – 多次讀取相同資料的唯讀交易一律會取回相同的值。
- `Phantom reads` – 唯讀交易絕不會讀取交易開始後新增的資料。

由於快照隔離是使用多版本並行控制 (MVCC) 來實現，因此唯讀查詢不需要鎖定資料，因此不會封鎖變動查詢。

僅供讀取複本僅接受唯讀查詢，因此針對僅供讀取複本的所有查詢都會依據 SNAPSHOT 隔離語意執行。

查詢僅供讀取複本時，唯一額外的考量是寫入器和僅供讀取複本之間可能會有較短的複寫延遲。這表示對寫入器所做的更新可能需要很短的時間傳播到您正在從中讀取的僅供讀取複本。實際複寫時間取決於針對主要執行個體的寫入負載。Neptune 架構支援低延遲複寫，並在 Amazon CloudWatch 指標中測試複寫延遲。

儘管如此，由於 SNAPSHOT 隔離層級，讀取查詢永遠都會看到資料庫的一致狀態，即使不是最新的查詢也一樣。

如果您需要強力保證查詢會觀察先前更新的結果，請將查詢傳送到寫入器端點本身，而不是傳送到僅供讀取複本。

Neptune 中的變動查詢隔離

做為變動查詢一部分進行的讀取會在 `READ COMMITTED` 交易隔離下執行，以排除已變更讀取的可能性。超越對 `READ COMMITTED` 交易隔離提供的一般保證，Neptune 強力保證，既不會發生 `NON-REPEATABLE` 讀取，也不會發生 `PHANTOM` 讀取。

這些強力保證是透過讀取資料時鎖定記錄和記錄範圍來實現。這可防止並行交易在讀取後對索引範圍進行插入或刪除，因而保證可重複讀取。

> **Note**
>
> 不過，並行變動交易 `Tx2` 可能會在變動交易 `Tx1` 啟動之後開始，並可能在 `Tx1` 鎖定資料以讀取變更之前遞交變更。在此情況下，`Tx1` 會看到 `Tx2` 的變更，宛如在 `Tx1` 啟動之前，`Tx2` 就已完成。因為這只適用於遞交的變更，所以 `dirty read` 永遠不會發生。

若要了解 Neptune 用於變動查詢的鎖定機制，首先了解 Neptune 圖形資料模型 和 索引策略 的詳細資訊很有幫助。Neptune 會使用三個索引 (即 SPOG、POGS 和 GPSO) 管理資料。

為了實現 READ COMMITTED 交易層級的可重複讀取，Neptune 會在使用中的索引中採取範圍鎖定。例如，如果變動查詢讀取名為 person1 之頂點的所有屬性和傳出邊緣，則節點會在讀取資料之前，鎖定 SPOG 索引中字首 S=person1 所定義的整個範圍。

使用其他索引時，也適用相同的機制。例如，當變動交易使用 POGS 索引，查詢所指定邊緣標籤的所有來源目標頂點時，P 位置中的邊緣標籤範圍將被鎖定。任何並行交易，無論它是唯讀或變動查詢，仍可在鎖定的範圍內執行讀取。不過，任何涉及在鎖定字首範圍內插入或刪除新記錄的變動都需要獨佔鎖定，而且將予以防止。

換言之，當變動交易讀取某個索引範圍時，強力保證在讀取交易結束之前，任何並行交易都不會修改此範圍。這可保證 non-repeatable reads 不會發生。

使用鎖定等待逾時的衝突解決機制

如果第二個交易嘗試修改第一個交易已鎖定之範圍中的記錄，Neptune 會立即偵測到衝突並封鎖第二個交易。

如果未偵測到相依性死結，Neptune 會自動套用鎖定等待逾時機制。其中封鎖的交易會等待最多 60 秒，讓保留鎖定的交易完成並釋出鎖定。

- 如果鎖定等待逾時在釋出鎖定之前過期，則會復原封鎖的交易。
- 如果鎖定在鎖定等待逾時內釋出，則第二個交易會解除封鎖並可以成功完成，而不需要重試。

不過，如果 Neptune 在兩個交易之間偵測到相依性死結，則無法自動調解衝突。在此情況下，Neptune 會立即取消並復原兩個交易之一，而不會啟動鎖定等待逾時。Neptune 會盡最大努力復原已插入或刪除最少記錄的交易。

範圍鎖定和錯誤衝突

Neptune 會使用間隙鎖定採取範圍鎖定。間隙鎖定是鎖定索引記錄之間的間隙，或鎖定第一個索引記錄之前或最後一個索引記錄之後的間隙。

Neptune 使用所謂的字典資料表，將數值 ID 值與特定字串常值建立關聯。以下是這類 Neptune 字典的範例狀態：資料表：

字串	ID
type	1

字串	ID
default_graph	2
person_3	3
person_1	5
knows	6
person_2	7
age	8
edge_1	9
lives_in	10
New York	11
個人	12
Place	13
edge_2	14

上面的字串屬於一個屬性圖模型，但這些概念也同樣適用於所有 RDF 圖形模型。

SPOG (Subject-Predicate-Object_Graph) 索引的對應狀態會顯示在下面的左側。右側會顯示對應字串，以協助了解索引資料的含義。

S (ID)	P (ID)	O (ID)	G (ID)		S (字串)	P (字串)	O (字串)	G (字串)
3	1	12	2		person_3	type	個人	default_graph
5	1	12	2		person_1	type	個人	default_graph
5	6	3	9		person_1	knows	person_3	edge_1

S (ID)	P (ID)	O (ID)	G (ID)		S (字串)	P (字串)	O (字串)	G (字串)
5	8	40	2		person_1	age	40	default_graph
5	10	11	14		person_1	lives_in	New York	edge_2
7	1	12	2		person_2	type	個人	default_graph
11	1	13	2		New York	type	Place	default_graph

現在，如果變動查詢讀取名為 `person_1` 之頂點的所有屬性和傳出邊緣，則節點會在讀取資料之前，鎖定 SPOG 索引中字首 `S=person_1` 所定義的整個範圍。範圍鎖定會在所有相符的記錄和第一筆不相符的記錄上放置間隙鎖定。相符記錄將遭到鎖定，而不相符的記錄將不會遭到鎖定。Neptune 將放置間隙鎖定，如下所示：

- `5 1 12 2` (間隙 1)
- `5 6 3 9` (間隙 2)
- `5 8 40 2` (間隙 3)
- `5 10 11 14` (間隙 4)
- `7 1 12 2` (間隙 5)

這會鎖定下列記錄：

- `5 1 12 2`
- `5 6 3 9`
- `5 8 40 2`
- `5 10 11 14`

在此狀態下，下列操作會遭合法封鎖：

- 插入 S=person_1 的新屬性或邊緣。不同於 type 的新屬性或新邊緣必須進入間隙 2、間隙 3、間隙 4 或間隙 5，它們全都遭到鎖定。
- 刪除任何現有記錄。

同時，一些並行作將遭錯誤地封鎖 (產生錯誤衝突）：

- S=person_3 的任何屬性或邊緣插入都會遭到封鎖，因為它們必須進入間隙 1。
- 在 3 與 5 之間獲指派 ID 的任何新頂點插入將遭封鎖，因為它必須進入間隙 1。
- 在 5 與 7 之間獲指派 ID 的任何新頂點插入將遭封鎖，因為它必須進入間隙 5。

間隙鎖定不夠精確，無法鎖定一個特定述詞的間隙 (例如，鎖定述詞 S=5 的間隙 5)。

範圍鎖定只會放置在發生讀取的索引中。在上述情況下，記錄僅鎖定在 SPOG 索引中，而不是在 POGS 或 GPSO 中。查詢的讀取可能跨所有索引上執行，取決於存取模式，這些模式可以使用 explain API 來列出 (適用於 Sparql 和 Gremlin)。

> **Note**
>
> 也可以採取間隙鎖定，對基礎索引進行安全的並行更新，這也可能導致錯誤的衝突。放置這些間隙鎖定與交易所執行的隔離層級或讀取操作無關。

不僅在「並行」交易由於間隙鎖定而發生衝突時，可能發生錯誤衝突，還可能在某些情況下，於任何類型的失敗之後重試交易時也會發生。如果失敗觸發的復原仍在進行中，且先前針對交易採取的鎖定尚未完全釋放，則重試將會遇到錯誤的衝突並失敗。

在高負載下，您通常會發現 3-4% 的寫入查詢會由於錯誤的衝突而失敗。對於外部用戶端，這類錯誤的衝突很難預測，應該使用重試來處理。

Neptune 交易語義範例

以下範例說明 Amazon Neptune 中交易語義的不同使用案例。

主題

- 範例 1 - 僅在屬性不存在時插入屬性
- 範例 2 - 宣告屬性值在全域上是唯一的

範例 1 - 僅在屬性不存在時插入屬性

假設您想要確保屬性只設定一次。例如，假設多個查詢正在嘗試同時將信用分數指派給人員。您只想要插入一個屬性執行個體，而其他查詢會失敗，因為已設定屬性。

```
# 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 .} }
```

Gremlin `property()` 步驟會插入具有所指定索引鍵和值的屬性。`coalesce()` 步驟會在第一個步驟中執行第一個引數，如果失敗，則會執行第二個步驟：

在插入所指定 `person1` 頂點的 `creditScore` 屬性值之前，交易必須嘗試為 `person1` 讀取可能不存在的 `creditScore` 值。此嘗試讀取會鎖定 SPOG 索引中 `S=person1` 和 `P=creditScore` 的 SP 範圍，其中 `creditScore` 值存在或將被寫入。

採取此範圍鎖定可防止任何並行交易同時插入 `creditScore` 值。有多個平行交易時，一次最多一個平行交易可以更新該值。這會排除建立多個 `creditScore` 屬性的異常。

範例 2 - 宣告屬性值在全域上是唯一的

假設您想要插入社會安全號碼做為主索引鍵的人員。您想要變動查詢保證，在全域層級中，資料庫中沒有任何其他人具有相同的社會安全號碼：

```
# 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 } }
```

此範例與上述範例類似。主要差別在於範圍鎖定是針對 POGS 索引而非 SPOG 索引採取的。

執行查詢的交易必須讀取模式 (?person :ssn 123456789)，其中繫結 P 和 O 位置。範圍鎖定是針對 P=ssn 和 O=123456789 的 POGS 索引採取的。

- 如果模式的確存在，則不會採取任何動作。
- 如果不存在，鎖定會防止任何並行交易也插入該社會安全號碼

範例 3 - 如果另一個屬性具有指定值，則變更屬性

假設遊戲中的各種事件將人物從第一層移到第二層，並將設為零的新 level2Score 屬性指派給他們。您需要確保這類交易的多個並行執行個體無法建立層級二分數屬性的多個執行個體。Gremlin 和 SPARQL 中的查詢可能看起來如下所示。

```
# 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 .}
```

在 Gremlin 中，指定 Cardinality.single 時，property() 步驟會新增屬性，或將現有的屬性值取代為指定的新值。

屬性值的任何更新 (例如將 level 從 1 增加到 2) 會實作為刪除目前記錄，並使用新的屬性值插入新記錄。在此情況下，會刪除層級編號 1 的記錄，並重新插入層級編號 2 的記錄。

若要讓交易能夠新增 level，並將 level2Score 從 1 更新到 2，必須先驗證 level 值目前是否等於 1。執行此動作時，它會對 SPOG 索引中 S=person1、P=level 和 O=1 的 SPO 字首採取範圍鎖定。此鎖定可防止並行交易刪除版本 1 三元組，因此不可能發生起衝突的並行更新。

範例 4 - 取代現有屬性

特定事件可能會將人員的信用分數更新為新值 (這裡指的是 BBB)。但您想要確保該類型的並行事件無法為人員建立多個信用分數屬性。

```
# 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 .}
```

此案例與範例 3 類似，差別在於不是鎖定 SPO 字首，而是 Neptune 只會鎖定具有 S=person1 和 P=creditScore 的 SP 字首。這可防止並行交易插入或刪除任何具有 person1 主旨之 creditScore 屬性的三元組。

範例 5 - 避免懸置屬性或邊緣

實體上的更新不應留下懸置元素，也就是與未輸入之實體相關聯的屬性或邊緣。這只是 SPARQL 中的問題，因為 Gremlin 具有內建限制條件來防止懸置元素。

```
# SPARQL:
tx1: INSERT { :person1 :age 23 } WHERE { :person1 rdf:type :Person }
tx2: DELETE { :person1 ?p ?o }
```

INSERT 查詢必須讀取和鎖定 SPOG 索引 O=Person 中具有 S=person1 和 P=rdf:type,的 SPO 字首。鎖定可防止 DELETE 查詢平行成功。

在這兩個查詢 (嘗試刪除 :person1 rdf:type :Person 記錄的 DELETE 查詢，以及讀取記錄並在 SPOG 索引中的 SPO 上建立範圍的 INSERT 查詢) 之間的競爭中，可能會產生以下結果：

- 如果 INSERT 查詢在 DELETE 查詢讀取並刪除 :person1 的所有記錄之前遞交，則 :person1 會完全從資料庫中移除，包括新插入的記錄。
- 如果 DELETE 查詢在 INSERT 查詢嘗試讀取 :person1 rdf:type :Person 記錄之前遞交，則讀取會觀察遞交的變更。也就是說，它找不到任何 :person1 rdf:type :Person 記錄，因此變成無操作。

- 如果 INSERT 查詢在 DELETE 查詢讀取之前讀取，則 `:person1 rdf:type :Person` 三元組會遭到鎖定，而且 DELETE 查詢會遭到封鎖，直到 INSERT 查詢遞交為止，如先前的第一個案例一樣。
- 如果 DELETE 在 INSERT 查詢之前讀取，而且 INSERT 查詢嘗試讀取，並對記錄的 SPO 字首採取鎖定，則會偵測到衝突。這是因為三元組已標記為移除，然後 INSERT 失敗。

在所有這些不同的可能事件序列中，不會建立懸置邊緣。

例外狀況處理和重試

交易由於無法解決的衝突或鎖定等待逾時而取消時，Amazon Neptune 會以 `ConcurrentModificationException` 回應。如需詳細資訊，請參閱 引擎錯誤代碼。根據最佳實務，用戶端應一律截獲和處理這些例外狀況。

在許多情況下，當 `ConcurrentModificationException` 執行個體數量很低時，以指數退避為基礎的重試機制也很適合做為處理它們的方式。在這類重試方法中，重試次數和等待時間上限通常取決於交易的大小上限和持續時間。

不過， 如果您的應用程式具有高度並行更新工作負載，而且您觀察到大量 `ConcurrentModificationException` 事件，則您或許能夠修改應用程式，以減少發生衝突之並行修改的數量。

例如，考量有一個應用程式經常更新一組頂點，並使用多個並行執行緒進行這些更新，以最佳化寫入輸送量。如果每個執行緒持續執行更新一或多個節點屬性的查詢，則相同節點的並行更新可能會產生 `ConcurrentModificationException`。這樣一來，可能會降低寫入效能。

如果您可以序列化可能彼此衝突的更新，則可以大幅降低此類衝突的可能性。例如，如果您可以確保指定節點的所有更新查詢都是在同一個執行緒上進行 (可能使用雜湊型指派)，則可以確保它們逐一執行，而不是同時執行。雖然對相鄰節點採取的範圍鎖定仍有可能導致 `ConcurrentModificationException`，但您可以消除對相同節點的並行更新。

Amazon Neptune 資料庫叢集和執行個體

Amazon Neptune「資料庫叢集」會透過查詢管理對資料的存取。一個叢集包含：

- 一個「主要資料庫執行個體」。
- 最多 15 個「僅供讀取複本資料庫執行個體」。

叢集中的所有執行個體共用相同的基礎受管儲存磁碟區，這是專為可靠性和高可用性而設計的。

您可以透過 Neptune 端點連線至資料庫叢集中的資料庫執行個體。

Neptune 資料庫叢集中的主要資料庫執行個體

主要資料庫執行個體會協調對資料庫叢集的基礎儲存磁碟區的所有寫入操作。它也支援讀取操作。

Neptune 資料庫叢集中只能有一個主要資料庫執行個體。如果主要執行個體變得無法使用，Neptune 會自動容錯移轉至具有您所指定優先順序的其中一個僅供讀取複本執行個體。

Neptune 資料庫叢集中的僅供讀取複本資料庫執行個體

在建立資料庫叢集的主要執行個體之後，您可以在資料庫叢集中最多建立 15 個僅供讀取複本執行個體，來支援唯讀查詢。

Aurora 僅供讀取複本很適用於擴展讀取容量，因為完全專用於叢集磁碟區上的讀取操作。所有寫入操作是由主要執行個體管理。每個僅供讀取複本資料庫執行個體都有自己的端點。

由於叢集儲存磁碟區是在叢集中的所有執行個體之間共用，因此所有僅供讀取複本執行個體都會針對查詢結果傳回相同的資料，複寫延遲非常短。在主要執行個體寫入更新之後，此延遲通常遠低於 100 毫秒，但在寫入操作的數量非常大時，此延遲通常會稍長一些。

在不同的可用區域中具有一或多個可用的僅供讀取複本執行個體可以提高可用性，因為僅供讀取複本會做為主要執行個體的容錯移轉目標。亦即，如果主要執行個體失敗，則 Neptune 會提升僅供讀取複本以成為主要執行個體。當發生此情況時，若重新啟動提升的執行個體，會發生短暫的中斷，在此期間對主要執行個體提出的讀取和寫入請求會由於例外狀況而失敗。

相反地，如果您的資料庫叢集不包含任何僅供讀取複本執行個體，則當主要執行個體失敗時，資料庫叢集仍無法使用，直到重新建立為止。重新建立主要執行個體所花費的時間比提升僅供讀取複本要長得多。

為了確保高可用性，建議您建立一或多個僅供讀取複本執行個體，這些執行個體具有與主要執行個體相同的資料庫執行個體類別，且位於與主要執行個體不同的可用區域。請參閱Neptune 資料庫叢集的容錯能力。

透過主控台，您只要在建立資料庫叢集時指定 Multi-AZ (異地同步備份)，即可輕鬆建立異地同步備份部署。如果資料庫叢集位於單一可用區域中，您可以在不同的可用區域新增 Neptune 複本，使其變成多重可用區資料庫叢集。

> **Note**
>
> 您無法為未加密的 Neptune 資料庫叢集建立加密的僅供讀取複本執行個體，也無法為加密的 Neptune 資料庫叢集建立未加密的僅供讀取複本執行個體。

如需如何建立 Neptune 僅供讀取複本資料庫執行個體的詳細資訊，請參閱 使用主控台建立 Neptune 讀取器執行個體。

調整 Neptune 資料庫叢集中資料庫執行個體的大小

根據您的 CPU 和記憶體需求調整 Neptune 資料庫叢集中執行個體的大小。執行個體上的 vCPU 數目會決定處理傳入查詢的查詢執行緒數目。執行個體上的記憶體數量會決定緩衝區快取的大小，用於儲存從基礎儲存磁碟區擷取的資料頁面副本。

每個 Neptune 資料庫執行個體都具有相當於該執行個體上 vCPU 數目 2 倍的查詢執行緒數目。例如，具有 16 個 vCPU 的 `r5.4xlarge` 具有 32 個查詢執行緒，因此可以同時處理 32 個查詢。

在所有查詢執行緒都被佔用時到達的其他查詢會放入伺服器端佇列中，並在查詢執行緒變成可用時以 FIFO 的方式進行處理。此伺服器端佇列可保留約 8000 個待定請求。一旦裝滿，Neptune 會以 `ThrottlingException` 回應其他請求。您可以使用`MainRequestQueuePendingRequests` CloudWatch 指標來監視擱置的要求數目，或使用具有參數的 Gemlin 查詢狀態端點。`includeWaiting`

從用戶端的角度來看，查詢執行時間除了實際執行查詢所花費的時間之外，還包括花費在佇列中的任何時間。

利用主要資料庫執行個體上所有查詢執行緒的持續並行寫入負載，理想情況下會顯示 90% 以上的 CPU 使用率，這指示伺服器上的所有查詢執行緒都在積極參與執行有用的工作。不過，即使在持續的並行寫入負載下，實際 CPU 使用率通常也會稍低一些。這通常是因為查詢執行緒正在等待基礎儲存磁碟區的 I/O 操作完成。Neptune 會使用仲裁寫入，跨三個可用區域製作六個資料副本，而這六個儲存節點中的

四個必須確認寫入，才能將其視為持久性。當查詢執行緒從儲存磁碟區等待此仲裁時，它會停滯，從而降低 CPU 使用率。

如果您有一個序列寫入負載，您正在其中執行一個接一個的寫入，並等待第一個寫入完成後再開始下一個，您可以預期 CPU 使用率仍會降低。確切的數量將是 vCPU 和查詢執行緒數目的函數 (查詢執行緒越多，每個查詢的整體 CPU 就越少)，而等待 I/O 會導致減少一些。

如需有關如何最好地調整資料庫執行個體大小的詳細資訊，請參閱 選擇正確的 Neptune 資料庫執行個體。如需每個執行個體類型的定價，請參閱 Neptune 定價頁面。

在 Neptune 中監控資料庫執行個體

您可以使用 Neptune 中的 CloudWatch 指標來監視資料庫執行個體的效能，並追蹤用戶端觀察到的查詢延遲。請參閱用 CloudWatch 來監視 Neptune 中的資料庫執行個體效。

Amazon Neptune 儲存體、可靠性和可用性

Amazon Neptune 會使用分散式和共用儲存架構，該架構會隨著資料庫儲存需求的增長而自動擴展。

Neptune 資料會儲存在叢集磁碟區中，該磁碟區是單一虛擬磁碟區，使用非揮發性記憶體快速 (NVMe) SSD 型磁碟機。叢集磁碟區包含邏輯區塊 (稱為區段) 的集合。這些區段的每一個都會獲配置 10 GB 的儲存體。每個區段中的資料會複寫至六個副本，然後在資料庫叢集所在 AWS 區域中的三個可用區域 (AZ) 之間進行配置。

建立 Neptune 資料庫叢集時，會為其配置 10 GB 的單一區段。當資料量增加並超過目前配置的儲存體時，Neptune 會透過新增區段來自動擴充叢集磁碟區。除了中國以外的所有支援區域，Neptune 叢集磁碟區的大小上限為 128 TB (TiB) GovCloud，且限制為 64 TiB。不過，對於 版本：1.0.2.2 (2020 年 3 月 9 日) 之前的版本引擎，所有區域中叢集磁碟區的大小限制為 64 TiB。

資料庫叢集磁碟區包含您的所有使用者資料、索引和字典 (如 Neptune 圖形資料模型。 一節所述)，以及內部中繼資料，例如內部交易日誌。所有此圖形資料 (包括索引和內部日誌) 不得超過叢集磁碟區的大小上限。

I/O 優化儲存選項

Neptune 提供兩種儲存定價模式：

- 標準儲存 — 標準儲存體可為中至低 I/O 使用量的應用程式提供具經濟效益的資料庫儲存。
- I/O 優化儲存 — 若使用 I/O 優化儲存，您只需以高於標準儲存的成本為所使用的儲存體付費，而且無需為使用的 I/O 付費。

  I/O 優化儲存的設計符合 I/O 密集型圖形工作負載的需求，並具有低延遲要求和一致的 I/O 輸送量的低延遲。

  如需詳細資訊，請參閱 I/O 優化儲存。

Neptune 儲存體配置

即使 Neptune 叢集磁碟區可以增長至 128 TiB (或在少數區域中，64 TiB)，您也只需為實際配置的空間支付費用。配置的總空間是由儲存高水位決定，這是在叢集磁碟區存在期間隨時配置給叢集磁碟區的最大容量。

這表示，即使從叢集磁碟區移除使用者資料，例如透過捨棄查詢 (例如 `g.V().drop()`)，總配置的空間仍保持不變。Neptune 會自動最佳化未使用的配置空間，以供日後重複使用。

除了使用者資料之外，另外兩種類型的內容會耗用內部儲存空間，即字典資料和內部交易日誌。雖然字典資料與圖形資料一起儲存，但即使刪除了它支援的圖形資料後，它還是會無限期地存在，這表示如果重新引入資料，可以重複使用這些項目。內部日誌資料儲存在個別的內部存儲空間中，該儲存空間具有自己的高水位。當內部日誌到期時，它所佔用的儲存體可以重複用於其他日誌，但不能用於圖形資料。已配置給記錄檔的內部空間量會包含在VolumeBytesUsedCloudWatch 指標報告的總空間中。

檢查 儲存體最佳實務 取得將配置的儲存體保持在下限並重複使用空間的方法。

Neptune 儲存計費

儲存成本是根據儲存高水位計費，如上所述。雖然您的資料會複寫為六個副本，但您只需為一份資料副本支付費用。

您可以透過監視VolumeBytesUsed CloudWatch 指標來判斷資料庫叢集目前的儲存高水位標記為何 (請參閱使用 Amazon 監控 Neptune CloudWatch)。

其他可能會影響 Neptune 儲存成本的因素包括資料庫快照和備份，這些快照和備份會以備份儲存體分別計費，並以 Neptune 儲存成本為基礎 (請參閱 有助於管理 Neptune 備份儲存體的 CloudWatch 指標)。

不過，如果您建立資料庫的複製，複製會指向資料庫叢集本身使用的同一叢集磁碟區，因此原始資料不會產生額外的儲存費用。複製後續變更會使用copy-on-write 通訊協定，並且會產生額外的儲存成本。

如需更多的 Neptune 定價資訊，請參閱 Amazon Neptune 定價。

Neptune 儲存體最佳實務

由於某些類型的資料會耗用 Neptune 中的永久儲存體，因此請使用下列最佳實務來避免儲存體增長時出現大量激增的情況：

- 在設計圖形資料模型時，請盡可能避免使用本質為暫時性的屬性索引鍵和使用者面向值。
- 如果您打算對資料模型進行變更，請不要使用新模型將資料載入至現有的資料庫叢集，直到使用快速重設 API 清除該資料庫叢集中的資料為止。最好的做法通常是將使用新模型的資料載入至新的資料庫叢集。
- 對大量資料操作的交易會產生相應的大型內部日誌，這可能會永久增加內部日誌空間的高水位。例如，刪除資料庫叢集中所有資料的單一交易可能會產生巨大的內部日誌，該日誌需要配置大量內部儲存空間，進而永久減少可用於圖形資料的空間。

  為了避免發生這種情況，請將大型交易分割為較小的交易，並在它們之間留出時間，以便相關聯的內部日誌有機會過期並釋放其內部存儲空間，以供後續日誌重複使用。

- 若要監視 Neptune 叢集磁碟區的成長，您可以在`VolumeBytesUsed` CloudWatch 指標上設定 CloudWatch 警示。如果您的資料達到叢集磁碟區的大小上限，這可能會特別有用。如需詳細資訊，請參閱使用 Amazon CloudWatch 警示。

當您有大量未使用的配置空間時，縮減資料庫叢集使用的儲存空間的唯一方法是匯出圖形中的所有資料，然後將其重新載入至新的資料庫叢集。如需從資料庫叢集匯出資料的簡單方法，請參閱 Neptune 的資料匯出服務和公用程式，以及如需將資料匯回 Neptune 的簡單方法，請參閱 Neptune 的大量載入器。

> **Note**
>
> 建立和還原快照並不會減少為資料庫叢集配置的儲存空間量，因為快照會保留叢集基礎儲存體的原始映像。如果有大量配置的儲存裝置未使用，則縮減所配置之儲存空間量的唯一方法是匯出您的圖形資料，然後將其重新載入至新的資料庫叢集。

Amazon Neptune 儲存體可靠性和高可用性

Amazon Neptune 的設計目的是要既可靠、耐用且容錯。

在三個可用區域 (AZ) 之間維護六個 Neptune 資料副本的事實，可確保資料的儲存體具有高耐用性，因而資料遺失的可能性極低。無論可用區域中是否有資料庫執行個體，資料都會跨這些可用區域自動複寫，而且複寫數量與您叢集中的資料庫執行個體數目無關。

這表示您可以快速新增僅供讀取複本，因為 Neptune 不會製作圖形資料的新副本。相反地，僅供讀取複本會連線至已包含您資料的叢集磁碟區。同樣地，移除僅供讀取複本並不會移除任何基礎資料。

您只能在刪除叢集磁碟區的所有資料庫執行個體之後刪除該叢集磁碟區及其資料。

Neptune 也會自動偵測在組成叢集磁碟區的區段中所發生的失敗。當區段中的資料副本損毀時，Neptune 會立即修復該區段，進而使用相同區段內的其他資料副本，以確保修復的資料是最新的。因此，Neptune 可避免資料遺失，並減少執行 point-in-time 還原以從磁碟故障復原的需求。

連線至 Amazon Neptune 端點

Amazon Neptune 會使用資料庫執行個體的叢集，而非單一執行個體。每個 Neptune 連線都由特定的資料庫執行個體處理。當您連線至 Neptune 叢集時，您指定的主機名稱和連接埠會指向稱為「端點」的中繼處理常式。端點是包含主機地址和連接埠的 URL。Neptune 端點會使用加密的 Transport Layer Security/Secure Sockets Layer (TLS/SSL) 連線。

Neptune 會使用端點機制來抽象化這些連線，以便您不必對主機名稱進行硬式編碼，或撰寫您自己的邏輯，在某些資料庫執行個體無法使用時重新路由連線。

使用端點，您可以將每個連線映射至適當的執行個體或執行個體群組，取決於您的使用案例。自訂端點可讓您連線至資料庫執行個體的子集。下列端點可在 Neptune 資料庫叢集中使用。

Neptune 叢集端點

叢集端點為 Neptune 資料庫叢集的端點，其會連線至該資料庫叢集目前的主要資料庫執行個體。每個 Neptune 資料庫叢集具有一個叢集端點和一個主要資料庫執行個體。

叢集端點可為資料庫叢集的讀寫連線提供容錯移轉支援。對資料庫叢集上的所有寫入操作，包括插入、更新、刪除和資料定義語言 (DDL) 變更使用叢集端點。您也可以對讀取操作 (例如查詢) 使用叢集端點。

如果資料庫叢集目前的主要資料庫執行個體失敗，Neptune 會自動容錯移轉至新的主要資料庫執行個體。容錯移轉期間，資料庫叢集會繼續從新的主要資料庫執行個體對叢集端點提供連線請求，將對服務的中斷降到最低。

以下範例說明 Neptune 資料庫叢集的叢集端點。

```
mydbcluster.cluster-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Neptune 讀取器端點

讀取器端點為 Neptune 資料庫叢集的端點，其會連線至該資料庫叢集其中一個可用的 Neptune 複本。每個 Neptune 資料庫叢集都有一個讀取器端點。如果有多個 Neptune 複本，讀取器端點會將每個連線請求導向其中一個 Neptune 複本。

讀取器端點可為資料庫叢集的唯讀連線提供循環配置資源路由。對讀取操作 (例如 查詢) 使用讀取器端點。

除非您有單一執行個體叢集 (沒有僅供讀取複本的叢集)，否則無法使用讀取器端點進行寫入作業。在這種情況下 (以及只有在這種情況下)，讀取器可以用於寫入操作以及讀取操作。

讀取器端點循環配置資源路由的運作方式是改變 DNS 項目指向的主機。每當您解析 DNS，就會取得不同的 IP，且對這些 IP 開放連線。建立連線後，所有對該連線的請求都會傳送到同一主機。用戶端必須再次建立新的連線和解析 DNS 記錄，以取得可能不同的僅供讀取複本的連線。

> **Note**
>
> WebSockets 連接通常長時間保持活動狀態。若要獲得不同的僅供讀取複本，請執行下列動作：
>
> - 確保您的用戶端於每次連線時解析 DNS 項目。
> - 關閉連線並重新連線。

不同的用戶端軟體可能會以不同的方式解析 DNS。例如，如果您的用戶端解析 DNS，然後將 IP 用於每一個連線，則軟體會將所有請求導向單一主機。

用戶端或代理伺服器的 DNS 快取，會從快取將 DNS 名稱解析至同一個端點。這對循環配置資源路由和容錯移轉情況都構成問題。

> **Note**
>
> 停用所有 DNS 快取設定，以強制每次執行 DNS 解析。

資料庫叢集會在可用的 Neptune 複本間對讀取器端點分配連線請求。如果資料庫叢集僅包含一個主要資料庫執行個體，讀取器端點會從主要資料庫執行個體提供連接請求。如果已為該資料庫叢集建立 Neptune 複本，讀取器端點會繼續從新的 Neptune 複本對讀取器端點提供連線請求，將對服務的中斷降到最低。

下列範例說明 Neptune 資料庫叢集的讀取器端點。

```
mydbcluster.cluster-ro-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Neptune 執行個體端點

執行個體端點為 Neptune 資料庫叢集中資料庫執行個體的端點，其會連線至該特定資料庫執行個體。資料庫叢集中的每個資料庫執行個體，不論執行個體類型為何，都有自己的唯一執行個體端點。因此，資料庫叢集目前的主要資料庫執行個體有一個執行個體端點。資料庫叢集的每一個 Neptune 複本也有一個執行個體端點。

執行個體端點對資料庫叢集的連線提供直接控制，使用叢集端點或讀取器端點的案例可能不適用。例如，根據工作負載類型而定，您的用戶端應用程式可能需要精細負載平衡。在此情況下，您可以設定多個用戶端連線至資料庫叢集中的不同 Neptune 複本，以分配讀取工作負載。

下列範例說明 Neptune 資料庫叢集中資料庫執行個體的執行個體端點。

```
mydbinstance.123456789012.us-east-1.neptune.amazonaws.com:8182
```

Neptune 自訂端點

Neptune 叢集的自訂端點代表您選擇的一組資料庫執行個體。連線至端點時，Neptune 會選擇群組的其中一個執行個體來處理連線。您可以定義此端點要參考的執行個體，並且決定端點的運作目的。

Neptune 資料庫叢集沒有自訂端點，直到您建立一個自訂端點，而且您可以為每個佈建的 Neptune 叢集建立最多五個自訂端點。

自訂端點可根據資料庫執行個體的唯讀或讀寫功能以外的準則來提供負載平衡的資料庫連線。由於連線可能前往與端點相關聯的任何資料庫執行個體，因此確定該群組內的所有執行個體分享相同的效能和記憶體容量特性。使用自訂端點時，您一般不會對該叢集使用讀取器端點。

此功能預期供具有專精類型工作負載的進階使用者使用，在其工作負載中，讓叢集中的所有 Neptune 複本保持相同並不切實際。利用自訂端點，您可以調整與每個連線搭配使用的資料庫執行個體容量。

例如，如果您定義數個自訂端點，而這些端點連線到具有不同執行個體類別的執行個體群組，則可以將效能需求不同的使用者導向至最適合其使用案例的端點。

下列範例說明 Neptune 資料庫叢集中資料庫執行個體的自訂端點：

```
myendpoint.cluster-custom-123456789012.us-east-1.neptune.amazonaws.com:8182
```

如需詳細資訊，請參閱使用自訂端點。

Neptune 端點考量

使用 Neptune 端點時，請考慮下列情況：

- 使用執行個體端點來連接至資料庫叢集中的特定資料庫執行個體之前，請考慮改為針對資料庫叢集使用叢集端點或讀取器端點。

  叢集端點和讀取器端點提供高可用性案例的支援。如果資料庫叢集的主要資料庫執行個體失敗，Neptune 會自動容錯移轉至新的主要資料庫執行個體。它會透過將現有的 Neptune 複本提升為新的主要資料庫執行個體，或建立新的主要資料庫執行個體來完成。如果發生容錯移轉，您可以使用

叢集端點重新連線至新提升或建立的主要資料庫執行個體，或使用讀取器端點重新連線至資料庫叢集中其中一個其他的 Neptune 複本。

如果您不採取此方法，您仍可以針對預期的操作確定已連接至資料庫叢集中的正確資料庫執行個體。若要這麼做，您可以手動或以程式設計方式探索資料庫叢集中可用資料庫執行個體的結果集，並在容錯移轉之後，於使用特定資料庫執行個體的執行個體端點之前確認其執行個體類型。

如需容錯移轉的詳細資訊，請參閱 Neptune 資料庫叢集的容錯能力。

- 讀取器端點只會將連線導向 Neptune 資料庫叢集中可用的 Neptune 複本。它不會將特定查詢導向。

  > **Important**
  >
  > Neptune 不會進行負載平衡。

  如果您想要平衡查詢負載以分配資料庫叢集的讀取工作負載，您必須管理您應用程式的負載。您必須使用執行個體端點直接連線到 Neptune 複本，以平衡負載。

- 讀取器端點循環配置資源路由的運作方式是改變 DNS 項目指向的主機。用戶端必須建立新的連線並再次解析 DNS 紀錄，以取得可能是新的僅供讀取複本的連線。

- 在容錯移轉期間，當 Neptune 複本提升為新的主要資料庫執行個體時，讀取器端點可能會短暫直接連線至資料庫叢集的新主要資料庫執行個體。

使用 Neptune 中的自訂端點

新增資料庫執行個體至自訂端點或將它從自訂端點移除時，對該資料庫執行個體的任何現有連線會保持作用中。

您可以定義要包含在自訂端點中的資料庫執行個體清單 (「靜態」清單)，或定義要從自訂端點中排除的資料庫執行個體清單 (「排除」清單)。您可以使用包含/排除機制，將資料庫執行個體細分為群組，以及確定自訂端點涵蓋叢集中的所有資料庫執行個體。每個自訂端點只能包含這些清單類型的其中一個。

在中 AWS Management Console，選擇以 [附加 future 新增至此叢集的執行個體] 核取方塊表示。將該核取方塊保持未選取時，自訂端點會使用僅包含對話方塊中所指定資料庫執行個體的靜態清單。當您勾

選該核取方塊時，自訂端點會使用排除清單。在此情況下，自訂端點會呈現叢集中的所有資料庫執行個體 (包括您未來所新增的任何項目)，在對話方塊中保持未選取的那些除外。

當主要執行個體與 Neptune 複本之間因為容錯移轉或提升，而造成資料庫執行個體變更角色時，Neptune 不會變更靜態或排除清單中指定的資料庫執行個體。

您可以將一個資料庫執行個體與多個自訂端點建立關聯。例如，假設您將新的資料庫執行個體新增至叢集。在這些情況下，資料庫執行個體會新增至符合其資格的所有自訂端點。針對其定義的靜態或排除清單會確定哪個資料庫執行個體可以新增至其中。

如果端點包含資料庫執行個體的靜態清單，新增加的 Neptune 複本不會新增至其中。相反地，如果端點具有排除清單，剛新增的 Neptune 複本會新增至其中，前提是它們未在排除清單中具名。

如果 Neptune 複本變得無法使用，則會保持與其自訂端點相關聯。無論其是運作狀態不佳、已停止、重新啟動或由於其他原因而無法使用，都是如此。不過，只要它仍然無法使用，您就無法透過任何端點連線至其中。

由於新建立的 Neptune 叢集沒有自訂端點，因此您必須自行建立和管理這些自訂端點。對於從快照還原的 Neptune 叢集也是如此，因為自訂端點未包含在快照中。如果還原的叢集位於與原始叢集相同的區域，則您已在還原之後建立它們，並選擇端點名稱。

建立自訂端點

使用 Neptune 主控台管理自訂端點。導覽至 Neptune 叢集的詳細資訊頁面，並使用自訂端點區段中的控制項來執行此操作。

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 導覽至叢集詳細資訊頁面。
3. 在端點區段中選擇 `Create custom endpoint` 動作。
4. 選擇自訂端點的名稱，該名稱是您的使用者 ID 和區域的唯一。名稱長度必須在 63 個字元以內，並採取下列格式：

   `endpointName``.cluster-custom-`*`customerDnsIdentifier`*`.`*`dnsSuffix`*

   由於自訂端點名稱未包含您的叢集名稱，如果將叢集重新命名，便不需變更這些名稱。不過，您不可以對同一區域中的多個叢集重複使用相同的自訂端點名稱。提供每個自訂端點一個名稱，該名稱在您的使用者 ID 於特定區域內所擁有的叢集間都是唯一的。

5. 若要選取即使在叢集展開時仍保持相同的資料庫執行個體清單，請將核取方塊 Attach future instances added to this cluster (連線新增至此叢集的未來執行個體) 保持未選取。勾選該核取方塊時，自訂端點會在任何新的執行個體新增至叢集時動態新增這些執行個體。

檢視自訂端點

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 導覽至資料庫叢集的叢集詳細資訊頁面。
3. 端點區段僅包含自訂端點的相關資訊 (有關內建端點的詳細資訊會列示在主要詳細資訊區段中)。若要查看特定自訂端點的詳細資訊，請選取其名稱，以帶出該端點的詳細資訊頁面。

編輯自訂端點

您可以編輯自訂端點的屬性，以變更與其相關聯的資料庫執行個體。您也可以在靜態清單與排除清單之間進行切換。

您無法在編輯動作的變更進行中時連線至或使用自訂端點。在端點狀態回到可用，且您可以重新連線之前，可能需要幾分鐘的時間。

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 導覽至叢集詳細資訊頁面。
3. 在端點區段中，選擇您要編輯之自訂端點的名稱。
4. 在該端點的詳細資訊頁面中，選擇編輯動作。

刪除自訂端點

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 導覽至叢集詳細資訊頁面。
3. 在端點區段中，選擇您要刪除之自訂端點的名稱。
4. 在該端點的詳細資訊頁面中，選擇刪除動作。

將自訂 ID 注入至 Neptune Gremlin 或 SPARQL 查詢

根據預設，Neptune 會將唯一 `queryId` 值指派給每個查詢。您可以使用此 ID 來取得執行中查詢的相關資訊 (請參閱Gremlin 查詢狀態 API 或 SPARQL 查詢狀態 API)，或取消它 (請參閱Gremlin 查詢取消或SPARQL 查詢取消)。

Neptune 也可讓您在 HTTP 標頭中，或使用 `queryId` 查詢提示為 SPARQL 查詢指定自己的 `queryId` 值，以用於 Gremlin 或 SPARQL 查詢。指派您自己的 `queryID` 可讓您輕鬆追蹤查詢，以取得狀態或取消該查詢。

> **Note**
>
> 從 版本 1.0.1.0.200463.0 (2019 年 10 月 15 日) 開始就可以使用這項功能。

使用 HTTP 標頭來注入自訂 **`queryId`** 值

對於 Gremlin 和 SPARQL 二者，HTTP 標頭可用來將您自己的 `queryId` 值注入至查詢。

Gremlin 範例

```
curl -XPOST https://your-neptune-endpoint:port \
    -d "{\"gremlin\": \
        \"g.V().limit(1).count()\" , \
        \"queryId\":\"4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47\"  }"
```

SPARQL 範例

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=SELECT * WHERE { ?s ?p ?o } " \
       --data-urlencode \
       "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

使用 SPARQL 查詢提示來注入自訂 **`queryId`** 值

以下範例說明如何使用 SPARQL `queryId` 查詢提示，將自訂 `queryId` 值注入至 SPARQL 查詢：

```
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}}"
```

使用 queryId 值檢查查詢狀態

Gremlin 範例

```
curl https://your-neptune-endpoint:port/gremlin/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

SPARQL 範例

```
curl https://your-neptune-endpoint:port/sparql/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Neptune 實驗室模式

您可以使用 Amazon Neptune「實驗室模式」，以啟用目前 Neptune 引擎版本中的新功能，但這些功能尚未準備好用於生產用途，因此預設為未啟用。這可讓您在開發和測試環境中試用這些功能。

> **Note**
>
> 從 版本 1.0.1.0.200463.0 (2019 年 10 月 15 日) 開始就可以使用這項功能。

使用 Neptune 實驗室模式

使用 neptune_lab_mode 資料庫叢集參數來啟用或停用功能。做法是在資料庫叢集參數群組的 neptune_lab_mode 參數值中包含 *(feature name)*=enabled 或 *(feature name)*=disabled。

例如，在此引擎版本中，您可將 neptune_lab_mode 參數設為 Streams=disabled,ReadWriteConflictDetection=enabled。

如需如何針對您的資料庫編輯資料庫叢集參數群組的相關資訊，請參閱編輯參數群組。請注意，您無法編輯預設資料庫叢集參數群組；如果您是使用預設群組，則必須建立新的資料庫叢集參數群組，才能設定 neptune_lab_mode 參數。

> **Note**
>
> 當您對靜態資料庫叢集參數 (例如 neptune_lab_mode) 進行變更時，必須重新啟動叢集的主要 (寫入器) 執行個體，變更才會生效。在 版本：1.2.0.0 (2022 年 7 月 21 日) 之前，當主要執行個體重新啟動時，資料庫叢集中的所有僅供讀取複本都會自動重新啟動
> 從 版本：1.2.0.0 (2022 年 7 月 21 日) 開始，重新啟動主要執行個體並不會導致任何複本重新啟動。這表示您必須個別重新啟動每個執行個體，才能取得資料庫叢集參數變更 (請參閱 參數群組)。

> **Important**
>
> 目前，如果您提供了錯誤的實驗室模式參數，或者您的請求由於其他原因而失敗，則可能不會通知您發生失敗。您應該一律透過呼叫狀態 API 來驗證實驗室模式變更請求是否成功，如下所示：

```
curl -G https://your-neptune-endpoint:port/status
```

狀態結果包括實驗室模式資訊，這些資訊會顯示您請求的變更是否已進行：

```
{
  "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"}
}
```

目前可使用實驗室模式存取下列功能：

OSGP 索引

Neptune 現在可以維護第四個索引，即 OSGP 索引，這對於具有大量述詞的資料集很有用 (請參閱 啟用 OSGP 索引)。

> **Note**
>
> 此功能從 Neptune 引擎 1.0.2.1 版開始提供。

您可以在 neptune_lab_mode 資料庫叢集參數中設定 ObjectIndex=enabled，在新的空白 Neptune 資料庫叢集中啟用 OSGP 索引。OSGP 索引只能在新的空白資料庫叢集中啟用。

根據預設，停用 OSGP 索引。

> **Note**
>
> 在設定 neptune_lab_mode 資料庫叢集參數，以便啟用 OSGP 索引之後，您必須重新啟動叢集的寫入器執行個體，變更才會生效。

> **Warning**
>
> 如果您透過設定 ObjectIndex=disabled 來停用已啟用的 OSGP 索引，稍後又在新增更多資料之後重新啟用它，則將無法正確建立索引。不支援隨需重建索引，因此您應該只在資料庫為空時才啟用 OSGP 索引。

正規化的交易語意

Neptune 已更新並行交易的正式語義 (請參閱 Neptune 中的交易語意)。

在啟用或停用格式化交易語義的 neptune_lab_mode 參數中，使用 ReadWriteConflictDetection 做為名稱。

根據預設，已啟用正規化的交易語意。如果您想要還原為先前行為，請將 ReadWriteConflictDetection=disabled 併入針對資料庫叢集 neptune_lab_mode 參數設定的值中。

擴展的日期時間支持

Neptune 已擴展對日期時間功能的支持。若要啟用具有延伸格式的日期時間，請將資料庫叢集neptune_lab_mode參數的值包含DatetimeMillisecond=enabled在設定的值中。

Amazon Neptune 替代查詢引擎（DFE）

Amazon Neptune 具有稱為 DFE 的替代查詢引擎，它比原始 Neptune 引擎更能有效率地使用資料庫執行個體資源 (例如 CPU 核心、記憶體和 I/O)。

> **Note**
>
> 使用大型資料集，DFE 引擎可能無法在 t3 執行個體上很好地執行。

DFE 引擎會執行 SPARQL、Gremlin 和 OpenCypher 查詢，並支援廣泛的各種計畫類型，包括 left-deep、bushy 和 hybrid 計畫類型。計畫運算子可以調用在一組保留運算核心上執行的運算操作，也可以調用 I/O 操作，每個操作都會在 I/O 執行緒集區的自己執行緒上執行。

DFE 會使用有關 Neptune 圖形資料的預先產生統計資料，做出有關如何建構查詢的明智決策。如需如何產生這些統計資料的相關資訊，請參閱 DFE 統計資料。

系統會根據預先產生的統計資料和 Neptune 前端節點中可用的資源，自動選擇計畫類型和已使用的運算執行緒數目。結果的順序不是針對具有內部運算平行處理的計畫預先確定的。

控制 Neptune DFE 引擎的使用位置

根據預設，執行個體的 neptune_dfe_query_engine 執行個體參數會設定為 `viaQueryHint`，這會導致 DFE 引擎僅用於 OpenCypher 查詢，以及明確包含 `useDFE` 查詢提示 (設定為 `true`) 的 Gemlin 和 SPARQL 查詢。

您可以透過將 `neptune_dfe_query_engine` 執行個體參數設定為 `enabled` 來完全啟用 DFE 引擎，以便盡可能使用該引擎。

您也可以透過包含特定 Gremlin 查詢或 SPARQL 查詢的 `useDFE` 查詢提示來停用 DFE。此查詢提示可讓您防止 DFE 執行該特定查詢。

您可以使用如下的 執行個體狀態 呼叫，判斷是否在執行個體中已啟用 DFE：

```
curl -G https://your-neptune-endpoint:port/status
```

然后，狀態回應會指定是否已啟用 DFE：

```
{
```

```
  "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"}
}
```

Grimlin `explain` 和 `profile` 結果會告訴您 DFE 是否正在執行查詢。請參閱 Gremlin explain 報告中包含的資訊 (若為 `explain`) 和 DFE profile 報告 (若為 `profile`)。

同樣地，SPARQL `explain` 會告訴您 DFE 是否正在執行 SPARQL 查詢。如需詳細資訊，請參閱 DFE 啟用時的 SPARQL explain 輸出範例 和 DFENode運算子。

Neptune DFE 支援的查詢建構模組

目前，Neptune DFE 支援 SPARQL 和 Gremlin 查詢建構模組的子集。

對於 SPARQL，這是結合的基本圖形模式子集。

對於 Gemlin，它通常是查詢的子集，而這些查詢包含的周遊鏈未包含一些更複雜的步驟。

您可以找出 DFE 是全部還是部分執行您的其中一個查詢，如下所示：

- 在 Gremlin 中，`explain` 和 `profile` 結果會告訴您 DFE 正在執行查詢的哪些部分 (如果有的話)。請參閱 Gremlin explain 報告中包含的資訊 (若為 `explain`) 和 DFE profile 報告 (若為 `profile`)。另請參閱 使用 explain 和 profile 調校 Gremlin 查詢。

  有關 Neptune 引擎對個別 Gemlin 步驟之支援的詳細資訊，記載於 Gremlin 步驟支援 中。

- 同樣地，SPARQL `explain` 會告訴您 DFE 是否正在執行 SPARQL 查詢。如需詳細資訊，請參閱 DFE 啟用時的 SPARQL `explain` 輸出範例 和 DFENode運算子。

管理要供 Neptune DFE 使用的統計資料

> **Note**
>
> 對 openCypher 的支援取決於 Neptune 中的 DFE 查詢引擎。
> DFE 引擎首先在 Neptune 引擎 1.0.3.0 版的實驗室模式中使用，而且從 Neptune 引擎 1.0.5.0 版開始，預設為啟用，但僅能與查詢提示搭配使用，以及僅適用於 OpenCypher 支援。
> 從 Neptune 引擎 1.1.1.0 版開始，DFE 引擎不再處於實驗室模式，而且現在可以使用執行個體資料庫參數群組中的 neptune_dfe_query_engine 執行個體參數來控制此引擎。

DFE 引擎會在規劃查詢執行時，使用 Neptune 圖形中的資料相關資訊，進行有效的權衡。這項資訊採取統計資料的形式，其中包括所謂的特性集和述詞統計資料，可以引導查詢規劃。

從引擎版本 1.2.1.0 開始，您可以使用摘要 API 或端點，從這些統計資料擷取有關圖表的GetGraph摘要資訊。`summary`

目前，每當圖形中超過 10% 的資料已變更或最新統計資料已過了 10 天時，就會重新產生這些 DFE 統計資料。不過，這些觸發條件日後可能會變更。

> **Note**
>
> 統計資料產生會在 T3 和 T4g 執行個體上停用，因為它可能超過這些執行個體類型的記憶體容量。

您可以透過下列其中一個端點管理 DFE 統計資料的產生：

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics` (適用於 SPARQL)。
- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics` (適用於 Gremlin 和 openCypher)，以及它的替代版本：`https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics`。

> **Note**
>
> 從引擎 1.1.1.0 版開始，Gramlin 統計資料端點 (`https://`*`your-neptune-host`*`:`*`port`*`/gremlin/statistics`) 已被棄用以支持 `propertygraph` 或 `pg` 端點。基於回溯相容性仍支援它，但可能會在未來的版本中將其移除。

從引擎 1.2.1.0 版開始，SPARQL 統計資料端點 (`https://your-neptune-host:port/sparql/statistics`) 已被棄用以支持 `rdf` 端點。基於回溯相容性仍支援它，但可能會在未來的版本中將其移除。

在下面範例中，`$STATISTICS_ENDPOINT` 代表這些端點 URL 中的任何一個。

> **Note**
>
> 如果 DFE 統計資料端點位於讀取器執行個體上，則它可以處理的唯一請求是狀態請求。其他請求將由於 `ReadOnlyViolationException` 而失敗。

對 DFE 統計資料產生的大小限制

目前，如果達到下列任一大小限制，DFE 統計資料產生便會中止：

- 所產生的特性集數目不得超過 50,000 個。
- 所產生的述詞統計資料數目不得超過一百萬個。

這些限制可能會變更。

DEF 統計資料的目前狀態。

您可使用下列 `curl` 請求檢查 DFE 統計資料的目前狀態：

```
curl -G "$STATISTICS_ENDPOINT"
```

對狀態請求的回應包含下列欄位：

- `status` – 請求的 HTTP 傳回碼。如果請求成功，則傳回碼為 `200`。如需常見錯誤的清單，請參閱常見錯誤。
- `payload`:
  - `autoCompute` – (布林值) 指示是否啟用自動產生統計資料。
  - `active` – (布林值) 指示是否完全啟用 DFE 統計資料產生。
  - `statisticsId` – 報告目前統計資料產生執行的 ID。值 `-1` 指示尚未產生任何統計資料。
  - `date` – 最近產生 DFE 統計資料的 UTC 時間，以 ISO 8601 格式表示。

> **Note**
>
> 在引擎 1.2.1.0 版之前，這是以分鐘精確度表示，但從引擎 1.2.1.0 版開始，它是以毫秒精確度表示 (例如，2023-01-24T00:47:43.319Z)。

- note – 關於統計資料無效情況下問題的說明。
- signatureInfo – 包含統計資料中所產生之特性集的相關資訊 (在引擎 1.2.1.0 版之前，此欄位已命名為 summary)。這些通常不能直接採取行動：
  - signatureCount – 所有特性集的簽章總數。
  - instanceCount – 特性集執行個體的總數。
  - predicateCount – 唯一述詞的總數。

未產生統計資料時，對狀態請求的回應如下所示：

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : -1
   }
}
```

如果 DFE 統計資料可用，回應如下所示：

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : true,
    "statisticsId" : 1588893232718,
    "date" : "2020-05-07T23:13Z",
    "summary" : {
      "signatureCount" : 5,
      "instanceCount" : 1000,
      "predicateCount" : 20
    }
  }
```

```
}
```

如果 DFE 統計資料產生失敗，例如因為其超出統計資料大小限制，回應如下所示：

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : 1588713528304,
    "date" : "2020-05-05T21:18Z",
    "note" : "Limit reached: Statistics are not available"
  }
}
```

停用自動產生 DFE 統計資料

根據預設，啟用 DFE 時，會啟用自動產生 DFE 統計資料。

您可以停用自動產生，如下所示：

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "disableAutoCompute" }'
```

如果請求成功，HTTP 回應代碼為 200，且回應為：

```
{
  "status" : "200 OK"
}
```

您可以發出狀態請求並檢查回應中的 autoCompute 欄位是否設定為 false，以確認已停用自動產生。

停用自動產生統計資料並不會終止進行中的統計資料計算。

如果您提出請求，停用讀取器執行個體的自動產生，而不是資料庫叢集的寫入器執行個體的自動產生，則請求會失敗，HTTP 傳回碼為 400，且輸出如下所示：

```
{
  "detailedMessage" : "Writes are not permitted on a read replica instance",
  "code" : "ReadOnlyViolationException",
  "requestId":"8eb8d3e5-0996-4a1b-616a-74e0ec32d5f7"
```

```
}
```

如需其他常見錯誤的清單，請參閱 常見錯誤。

重新啟用自動產生 DFE 統計資料

根據預設，啟用 DFE 時，已啟用自動產生 DFE 統計資料。如果您停用自動產生，您可以稍後重新啟用它，如下所示：

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "enableAutoCompute" }'
```

如果請求成功，HTTP 回應代碼為 200，且回應為：

```
{
  "status" : "200 OK"
}
```

您可以發出狀態請求並檢查回應中的 autoCompute 欄位是否設定為 true，以確認已啟用自動產生。

手動觸發 DFE 統計資料的產生

您可以手動啟動 DFE 統計資料產生，如下所示：

```
curl -X POST "$STATISTICS_ENDPOINT" -d '{ "mode" : "refresh" }'
```

如果請求成功，則輸出如下，HTTP 傳回碼為 200：

```
{
  "status" : "200 OK",
  "payload" : {
    "statisticsId" : 1588893232718
  }
}
```

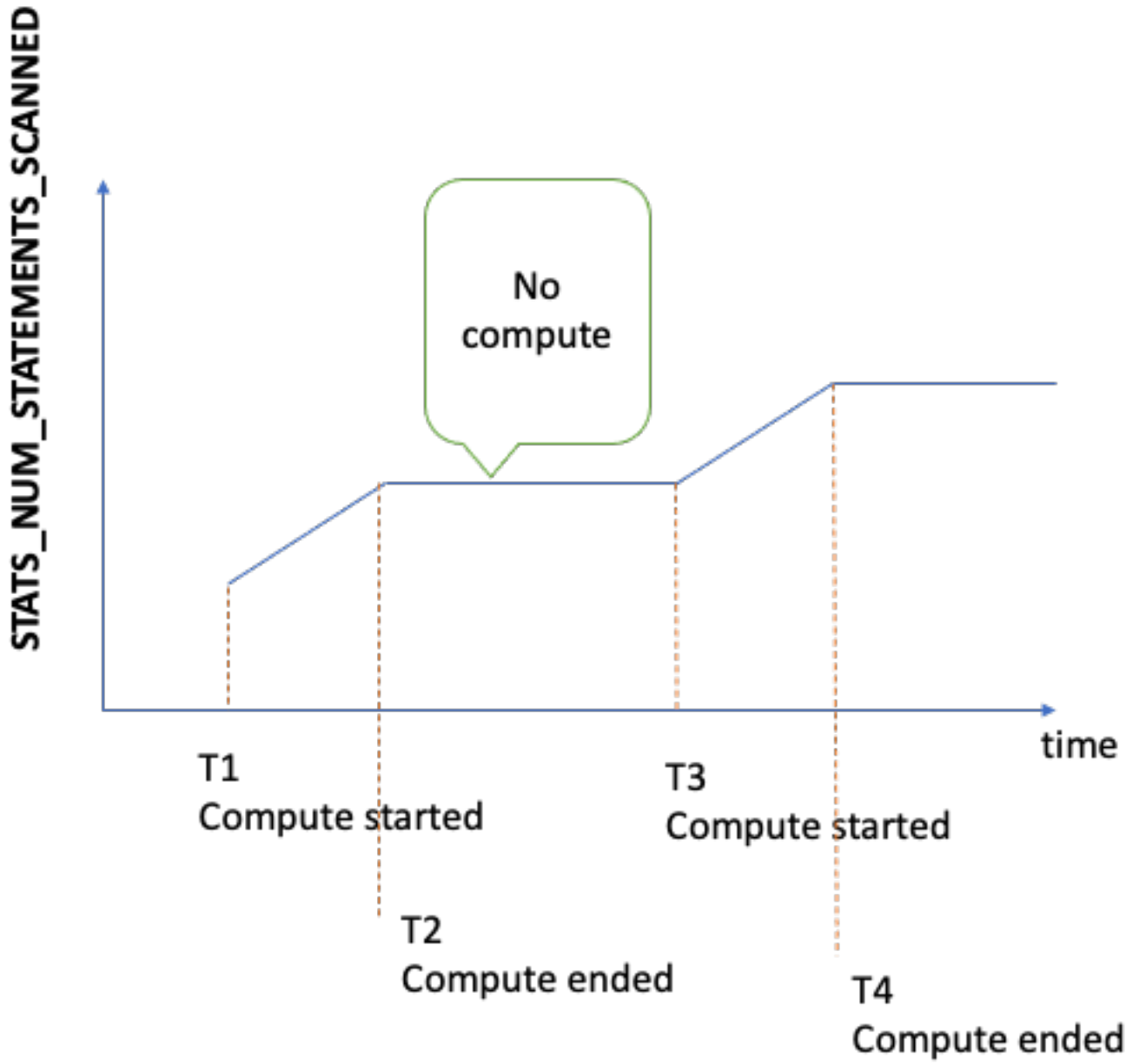
輸出中的 statisticsId 是目前正在發生之統計資料產生執行的識別符。如果執行已在請求時已處理中，則請求會傳回該執行的 ID，而不是啟動新的 ID。一次僅能發生一個統計資料產生執行。

如果在產生 DFE 統計資料時發生容錯移轉，新的寫入器節點會取得上次處理的檢查點，並從該處繼續統計資料執行。

使用**StatsNumStatementsScanned** CloudWatch 測量結果監督統計資料運算

此StatsNumStatementsScanned CloudWatch 測量結果會傳回伺服器啟動後針對統計資料運算而掃描的敘述句總數。它是在每個統計資料計算部分處更新。

每次觸發統計資料計算時，這個數字都會增加，並且當沒有計算發生時，它會保持不變。因此，查看隨時間變化的 StatsNumStatementsScanned 值圖，可以很清楚地了解統計資料計算何時發生以及有多快：



進行計算時，圖形的斜坡會向您顯示有多快 (斜坡越陡，統計資料的計算速度越快)。

如果圖形只是一條位於 0 的平直線，表示已啟用統計資料功能，但根本沒有計算任何統計資料。如果已停用統計資料功能，或者如果您使用的引擎版本不支援統計資料計算，則 `StatsNumStatementsScanned` 不存在。

如先前所述，您可以使用統計資料 API 停用統計資料計算，但是將其保留關閉狀態可能會導致統計資料不是最新的，進而導致針對 DFE 引擎產生不良的查詢計畫。

如需有使用 Amazon 監控 Neptune CloudWatch關如何使用的資訊，請參閱 CloudWatch。

將 AWS Identity and Access Management (IAM) 身份驗證與 DFE 統計資料端點搭配使用

您可以使用 awscurl 或任何使用 HTTPS 和 IAM 的任何工具，搭配 IAM 身分驗證安全地存取 DFE 統計資料端點。請參閱 使用 awscurl 搭配臨時憑證，安全地連線至啟用 IAM 身分驗證的資料庫叢集 以了解如何設定適當的憑證。一旦完成了該操作，您就可以提出如下的狀態請求：

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

或者，您可以建立名為 `request.json` 的 JSON 檔案，其中包含：

```
{ "mode" : "refresh" }
```

您可以接著手動啟動統計資料產生，如下所示：

```
awscurl "$STATISTICS_ENDPOINT" \
    --region (your region) \
    --service neptune-db \
    -X POST -d @request.json
```

刪除 DEF 統計資料

您可以對統計資料端點發出 HTTP DELETE 請求，來刪除資料庫中的所有統計資料：

```
curl -X "DELETE" "$STATISTICS_ENDPOINT"
```

有效的 HTTP 傳回碼為：

- 200 – 刪除成功。

  在此情況下，典型的回應如下所示：

```
{
  "status" : "200 OK",
  "payload" : {
      "active" : false,
      "statisticsId" : -1
  }
}
```

- 204 – 沒有任何要刪除的統計資料。

  在此情況下，回應是空白的 (沒有回應)。

如果將刪除請求傳送至讀取器節點上的統計資料端點，則會擲回 ReadOnlyViolationException。

DFE 統計資料請求的常見錯誤代碼

以下是當您對統計資料端點提出請求時可能發生的常見錯誤清單：

- AccessDeniedException – 傳回碼：400。訊息：Missing Authentication Token。
- BadRequestException (適用於 Gremlin 和 openCypher) – 傳回碼：400。訊息：Bad route: /pg/statistics。
- BadRequestException (適用於 RDF 資料) – 傳回碼：400。訊息：Bad route: /rdf/statistics。
- InvalidParameterException – 傳回碼：400。訊息：Statistics command parameter 'mode' has unsupported value '*the invalid value*'。
- MissingParameterException – 傳回碼：400。訊息：Content-type header not specified.。
- ReadOnlyViolationException – 傳回碼：400。訊息：Writes are not permitted on a read replica instance。

例如，如果您在未啟用 DFE 和統計資料時提出請求，您會得到如下的回應：

```
{
  "code" : "BadRequestException",
```

```
  "requestId" : "b2b8f8ee-18f1-e164-49ea-836381a3e174",
  "detailedMessage" : "Bad route: /sparql/statistics"
}
```

取得有關圖形的快速摘要報告

Neptune 圖形摘要 API 會擷取圖形的下列相關資訊：

- 對於屬形 (PG) 圖，圖形摘要 API 會傳回節點和邊緣標籤以及屬性索引鍵的唯讀清單，也會傳回節點、邊緣和屬性的計數。
- 對於資源描述架構 (RDF) 圖形，圖形摘要 API 會傳回類別和述詞索引鍵的唯讀清單，也會傳回四元組、主旨和述詞的計數。

> **Note**
>
> 圖形摘要 API 是在 Neptune 引擎 1.2.1.0 版中引進的。

使用圖形摘要 API，您可以快速了解圖形資料大小和內容。您也可以使用 `%summary` Neptune 工作台魔法，在 Neptune 筆記本內以互動方式使用 API。在圖形應用程式中，API 可以用來改善搜尋結果，方法是提供探索到的節點或邊緣標籤做為搜尋的一部分。

圖形摘要資料取自 Neptune DFE 引擎在執行期所計算的 DFE 統計資料，而且每當 DFE 統計資料可用時，就可以使用此圖形摘要資料。當您建立新的 Neptune 資料庫叢集時，預設會啟用統計資料。

> **Note**
>
> `t3` 和 `t4` 執行個體類型 (也就是，`db.t3.medium` 和 `db.t4g.medium` 執行個體類型) 上會停用產生統計資料，以節省記憶體。因此，這些執行個體類型上都無法使用圖形摘要資料。

您可以使用統計資料狀態 API 檢查 DFE 統計資料的狀態。只要未停用自動產生統計資料，就會定期自動更新統計資料。

如果您想要在請求圖形摘要時確定統計資料盡可能是最新的，則可以在擷取摘要之前，手動觸發統計資料更新。如果圖形在統計資料計算時變更，則它們一定會稍微落後，但不會太多。

使用圖形摘要 API 擷取圖形摘要資訊

對於您使用 Gremlin 或 OpenCypher 查詢的屬性圖形，您可以從屬性圖摘要端點擷取圖形摘要。此端點既有長 URI 也有短 URI：

- https://*your-neptune-host*:*port*/propertygraph/statistics/summary
- https://*your-neptune-host*:*port*/pg/statistics/summary

對於您使用 SPARQL 查詢的 RDF 圖形，您可以從 RDF 摘要端點擷取圖形摘要：

- https://*your-neptune-host*:*port*/rdf/statistics/summary

這些端點是唯讀端點，且僅支援 HTTP GET 操作。如果 $GRAPH_SUMMARY_ENDPOINT 設定為您要查詢之任何端點的地址，則您可以使用 curl 和 HTTP GET 擷取摘要資料，如下所示：

```
curl -G "$GRAPH_SUMMARY_ENDPOINT"
```

如果在嘗試擷取圖形摘要時沒有可用的統計資料，則回應如下所示：

```
{
  "detailedMessage": "Statistics are not available. Summary can only be generated after
 statistics are available.",
  "requestId": "48c1f788-f80b-b69c-d728-3f6df579a5f6",
  "code": "StatisticsNotAvailableException"
}
```

圖形摘要 API 的 **mode** URL 查詢參數

圖形摘要 API 接受名為 mode 的 URL 查詢參數，該參數可以採取兩個值之一，即 basic (預設值) 和 detailed。對於 RDF 圖，detailed 模式圖形摘要回應包含一個額外的 subjectStructures 欄位。對於屬性圖，詳細的圖形摘要回應包含兩個額外的欄位，即 nodeStructures 和 edgeStructures。

若要請求 detailed 圖形摘要回應，請包含 mode 參數，如下所示：

```
curl -G "$GRAPH_SUMMARY_ENDPOINT?mode=detailed"
```

如果 mode 參數不存在，則預設會使用 basic 模式，因此儘管可以明確指定 ?mode=basic，但這不是必需的。

屬性圖 (PG) 的圖形摘要回應

對於空的屬性圖，詳細的圖形摘要回應如下所示：

```
{
  "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" : [ ]
    }
  }
}
```

屬性圖 (PG) 摘要回應具有下列欄位：

- **`status`** – 請求的 HTTP 傳回碼。如果請求成功，則傳回碼為 200。

  如需常見錯誤的清單，請參閱 常見的圖形摘要錯誤。
- **`payload`**
  - **`version`** – 此圖形摘要回應的版本。
  - **`lastStatisticsComputationTime`** – Neptune 上次計算統計資料之時間的時間戳記 (採用 ISO 8601 格式)。
  - **`graphSummary`**
    - **`numNodes`** – 圖形中節點的數目。
    - **`numEdges`** – 圖形中邊緣的數目。
    - **`numNodeLabels`** – 圖形中不同節點標籤的數目。
    - **`numEdgeLabels`** – 圖形中不同邊緣標籤的數目。
    - **`nodeLabels`** – 圖形中不同節點標籤的清單。

- **edgeLabels** – 圖形中不同邊緣標籤的清單。
- **numNodeProperties** – 圖形中不同節點屬性的數目。
- **numEdgeProperties** – 圖形中不同邊緣屬性的數目。
- **nodeProperties** – 圖形中不同節點屬性的清單，以及其中使用每個屬性的節點計數。
- **edgeProperties** – 圖形中不同邊緣屬性的清單，以及其中使用每個屬性的邊緣計數。
- **totalNodePropertyValues** – 所有節點屬性的使用總數。
- **totalEdgePropertyValues** – 所有邊緣屬性的使用總數。
- **nodeStructures** – 只有在請求中指定 *mode=detailed* 時，才會顯示此欄位。它包含節點結構的清單，每個結構都包含下列欄位：
  - **count** – 具有此特定結構的節點數目。
  - **nodeProperties** – 此特定結構中存在之節點屬性的清單。
  - **distinctOutgoingEdgeLabels** – 此特定結構中存在之不同傳出邊緣標籤的清單。
- **edgeStructures** – 只有在請求中指定 *mode=detailed* 時，才會顯示此欄位。它包含邊緣結構的清單，每個結構都包含下列欄位：
  - **count** – 具有此特定結構的邊緣數目。
  - **edgeProperties** – 此特定結構中存在之邊緣屬性的清單。

RDF 圖形的圖形摘要回應

對於空的 RDF 圖形，詳細的圖形摘要回應如下所示：

```
{
  "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" : [ ]
    }
  }
```

```
}
```

RDF 圖形摘要回應具有下列欄位：

- **`status`** – 請求的 HTTP 傳回碼。如果請求成功，則傳回碼為 200。

  如需常見錯誤的清單，請參閱 常見的圖形摘要錯誤。
- **`payload`**
  - **`version`** – 此圖形摘要回應的版本。
  - **`lastStatisticsComputationTime`** – Neptune 上次計算統計資料之時間的時間戳記 (採用 ISO 8601 格式)。
  - **`graphSummary`**
    - **`numDistinctSubjects`** – 圖形中不同主旨的數目。
    - **`numDistinctPredicates`** – 圖形中不同述詞的數目。
    - **`numQuads`** – 圖形中四元組的數目。
    - **`numClasses`** – 圖形中類別的數目。
    - **`classes`** – 圖形中類別的清單。
    - **`predicates`** – 圖形中述詞的清單，以及述詞計數。
    - **`subjectStructures`** – 只有在請求中指定 *`mode=detailed`* 時，才會顯示此欄位。它包含主旨結構的清單，每個結構都包含下列欄位：
      - **`count`** – 此特定結構的出現次數。
      - **`predicates`** – 此特定結構中存在之述詞的清單。

範例屬性圖 (PG) 摘要回應

以下是屬性圖的詳細摘要回應，其中包含範例屬性圖航線資料集：

```
{
  "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"
          ]
        }
      ]
    }
  }
}
```

範例 RDF 圖形摘要回應

以下是 RDF 圖形的詳細摘要回應，其中包含範例 RDF 航線資料集：

```
{
  "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"
          ]
        }
      ]
    }
```

```
  }
}
```

搭配圖形摘要端點使用 AWS Identity and Access Management (IAM) 身份驗證

您可以使用 awscurl 或任何使用 HTTPS 和 IAM 的任何工具，搭配 IAM 身分驗證安全地存取圖形摘要端點。請參閱 使用 awscurl 搭配臨時憑證，安全地連線至啟用 IAM 身分驗證的資料庫叢集 以了解如何設定適當的憑證。一旦完成了該操作，您就可以提出如下的請求：

```
awscurl "$GRAPH_SUMMARY_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

> **Important**
>
> 建立臨時登入資料的 IAM 身分或角色，必須附加允許「GetGraph摘要 IAM」動作的 IAM 政策。

如需您可能遇到的常見 IAM 錯誤清單，請參閱 IAM 身分驗證錯誤。

圖形摘要請求可能傳回的常見錯誤代碼

Neptune 服務錯誤代碼	HTTP 狀態	訊息	錯誤案例	緩解
AccessDeniedException	403	缺少身分驗證字符。	未簽署或未正確簽署的請求已傳送至啟用 IAM 的 Neptune 資料庫。	在傳送之前使用 Sigv4 簽署請求 (請參閱 IAM 和圖形摘要)。
	403	用戶：*### ARN#*未被授權執行：海王星數據庫：資源：*#GetGraph*	IAM 政策不允許在啟用 IAM 的情況下將圖形摘要請求傳送至 Neptune 資料庫時執	確定附加至提出請求之使用者或角色的 IAM 政策允許 GetGraphSummary 動作。

Neptune 服務錯誤代碼	HTTP 狀態	訊息	錯誤案例	緩解
		Summary 資源 ARN)。	GetGraph行動作摘要。	
BadRequestException	400	統計資料已停用，因此也會停用圖形摘要。	嘗試在已停用統計資料的高載執行個體類型 (t3 或 t4g) 上擷取摘要。	使用已啟用統計資料產生的執行個體類型 (除了 t3 和 t4g 以外的所有受支援執行個體)。
	400	錯誤的路線：*/rdf/statistics/summarypathapi*	請求已傳送至無效路徑。	使用圖形摘要端點的正確路由。
InvalidParameterException	400	請求包含未知的參數：'*(########)*' 。	在請求中指定了無效參數時。	僅在請求中使用有效參數 (例如 mode)。
InvalidParameterException	400	URI 查詢參數 'mode' 具有不支援的值 '*(###)*'。	當請求中的 URL 參數 'mode' 後面跟著一個無效值時。	指定 URL 參數 'mode' 時，請使用有效值 (例如 basic 或 detailed)。
MethodNotAllowedException	405	不允許方法。	使用 GET (例如 POST 或 DELETE) 以外的任何 HTTP 方法呼叫摘要端點。	呼叫摘要端點時，請使用 HTTP GET 方法。
StatisticsNotAvailableException	400	統計資料尚未計算，在統計資料計算完成後將可使用圖形摘要。	當請求傳送至摘要端點時，沒有可用的統計資料。	等到統計資料產生完成。您可以使用統計資料狀態 API 檢查統計資料產生的狀態。

Neptune 服務錯誤代碼	HTTP 狀態	訊息	錯誤案例	緩解
	400	已達統計資料限制，因此無法使用圖形摘要。	統計資料產生已停止，因為它達到了統計資料大小限制。	此圖形上沒有可用的圖形摘要。

例如，如果您在已啟用 IAM 身分驗證的 Neptune 資料庫中對圖形化摘要端點提出請求，且請求者的 IAM 政策中沒有必要的許可，則您會得到如下的回應：

```
{
  "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"
}
```

Amazon Neptune JDBC 連線能力

Amazon Neptune 已發行開放原始碼 JDBC 驅動程式，支援 openCypher、Gremlin、SQL-Gremlin 和 SPARQL 查詢。JDBC 連線能力可讓您透過商業智慧 (BI) 工具 (例如 Tableau) 輕鬆連線至 Neptune。使用 JDBC 驅動程式搭配 Neptune 無需額外成本 — 您仍然只需為耗用的 Neptune 資源支付費用即可。

此驅動程式與 JDBC 4.2 相容，並且至少需要 Java 8。如需如何使用 JDBC 的相關資訊，請參閱 JDBC API 文件。

該 GitHub 項目，您可以在其中提交問題和打開功能請求，包含驅動程序的詳細文檔：

適用於 Amazon Neptune 的 JDBC 驅動程式

- 使用 SQL 搭配 JDBC 驅動程式
- 使用 Gremlin 搭配 JDBC 驅動程式
- 使用 openCypher 搭配 JDBC 驅動程式
- 使用 SPARQL 搭配 JDBC 驅動程式

Neptune JDBC 驅動程式入門

若要使用 Neptune JDBC 驅動程式連線至 Neptune 執行個體，JDBC 驅動程式必須部署在與 Neptune 資料庫叢集相同的 VPC 中，或者執行個體必須透過 SSH 通道或負載平衡器使用。SSH 通道可以在驅動程式內部設定，也可以在外部設定。

您可以在這裡下載驅動程式。驅動程式會包裝成單一 JAR 檔案，其名稱類似 `neptune-jdbc-1.0.0-all.jar`。若要使用它，請將 JAR 檔案放在應用程式的 `classpath` 中。或者，如果您的應用程式使用 Maven 或 Gradle，則您可以使用適當的 Maven 或 Gradle 命令，從 JAR 安裝驅動程式。

驅動程式需要一個 JDBC 連線 URL 與 Neptune 連線，形式如下：

```
jdbc:neptune:(connection
 type)://(host);property=value;property=value;...;property=value
```

GitHub 專案中每個查詢語言的區段會說明您可以在該查詢語言的 JDBC 連線 URL 中設定的屬性。

如果 JAR 檔案位於應用程式的 `classpath` 中，則不需要任何其他組態。您可以使用 JDBC `DriverManager` 介面和 Neptune 連線字串來連線驅動程式。例如，如果可以透過連接埠 8182 上的

端點 neptune-example.com 存取您的 Neptune 資料庫叢集，您將能夠像這樣與 OpenCypher 連線：

```
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();
}
```

GitHub 專案中每種查詢語言的文件章節說明如何在使用該查詢語言時建構連接字串。

使用 Tableau 搭配 Neptune JDBC 驅動程式

若要使用 Tableau 搭配 Neptune JDBC 驅動程式，請先下載並安裝最新版本的 Tableau Desktop。下載 Neptune JDBC 驅動程式的 JAR 檔案，以及 Neptune Tableau 連接器檔案 (.taco 檔案)。

在 Mac 上連線到適用於 Neptune 的 Tableau

1. 將 Neptune JDBC 驅動程式 JAR 檔案放在 /Users/*(your user name)*/Library/Tableau/Drivers 資料夾中。
2. 將 Neptune Tableau 連接器 .taco 檔案放在 /Users/*(your user name)*/Documents/My Tableau Repository/Connectors 資料夾中。
3. 如果您已啟用 IAM 身分驗證，請為其設定環境。請注意，在 .zprofile/、.zshenv/、.bash_profile 等中設定的環境變數將無法運作。您必須設定環境變數，GUI 應用程式才能載入它們。

   設定憑證的方法是將您的存取金鑰和私密金鑰放在 /Users/*(your user name)*/.aws/credentials 檔案中。

   設定服務區域的簡單方法就是開啟終端機，並使用應用程式的區域輸入下列命令 (例如 us-east-1)：

   ```
   launchctl setenv SERVICE_REGION region name
   ```

還有其他方法可以設定在重新啟動之後持續存在的環境變數，但是無論您使用哪種技術，都必須設定 GUI 應用程式可以存取的變數。

4. 若要在 Mac 上將環境變數載入至 GUI，請在終端機上輸入以下命令：

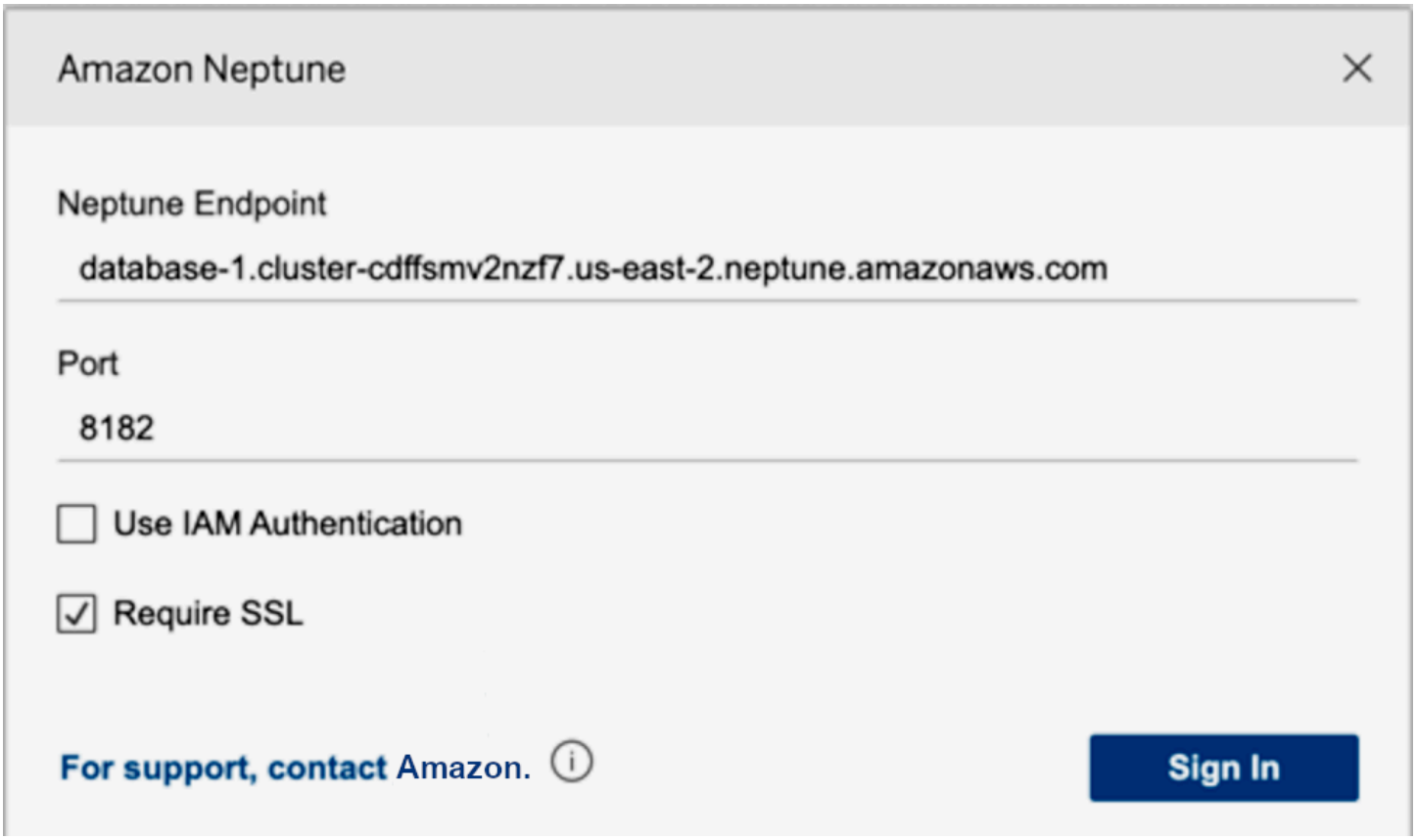
```
/Applications/Tableau/Desktop/2021.1.app/Contents/MacOS/Tableau
```

在 Windows 電腦上連線到適用於 Neptune 的 Tableau

1. 將 Neptune JDBC 驅動程式 JAR 檔案放在 `C:\Program Files\Tableau\Drivers` 資料夾中。
2. 將 Neptune Tableau 連接器 `.taco` 檔案放在 `C:\Users\`*`(your user name)`*`\Documents\My Tableau Repository\Connectors` 資料夾中。
3. 如果您已啟用 IAM 身分驗證，請為其設定環境。

   這可以像設定使用者 `ACCESS_KEY`、`SECRET_KEY` 和 `SERVICE_REGION` 環境變數一樣簡單。

在 Tableau 開啟的情況下，選取視窗左側的更多。如果 Tableau 連接器檔案位置正確，您可以在出現的清單中選取依 AWS的 Amazon Neptune：



您應該不必編輯連接埠，或新增任何連線選項。輸入您的 Neptune 端點並設定 IAM 和 SSL 組態 (如果您使用的是 IAM，則必須啟用 SSL)。

當您選取登入時，如果圖形很大，連線可能需要 30 秒以上的時間。Tableau 正在收集頂點和邊緣，並連接邊緣上的頂點，以及建立視覺效果。

對 JDBC 驅動程式連線進行疑難排解

如果驅動程式無法連線到伺服器，請使用 JDBC `Connection` 物件的 `isValid` 函數來檢查連線是否有效。如果此函數傳回 `false` (表示連線無效)，請檢查連線的端點是否正確，以及您是否位於 Neptune DB 叢集的 VPC 中，或者您是否具有叢集的有效 SSH 通道。

如果您收到來自 `DriverManager.getConnection` 呼叫的 `No suitable driver found for` *`(connection string)`* 回應，則連線字串開頭處可能發生問題。確保您的連線字串開頭如下所示：

```
jdbc:neptune:opencypher://...
```

若要收集有關連線的詳細資訊，您可以將 `LogLevel` 新增至連線字串，如下所示：

```
jdbc:neptune:opencypher://(JDBC URL):(port);logLevel=trace
```

或者，您也可以在輸入屬性中新增 properties.put("logLevel", "trace") 來記錄追蹤資訊。

Amazon Neptune 引擎更新

Amazon Neptune 會定期發行引擎更新。您可以使用執行個體狀態 API 來判斷您目前安裝的引擎發行版本。

引擎版本列在 Amazon Neptune 的引擎版本，而修補程式列在最新更新。

您可以在叢集維護處找到有關如何發行更新，以及如何在您的資料庫中升級 Neptune 引擎的詳細資訊。例如，在引擎版本編號中會說明版本編號方式。

Amazon Neptune 中的安全

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同的責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。在 AWS 合規計畫中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 Amazon Neptune 的合規計劃，請參閱合規計劃範圍內的AWS 服務。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Neptune 時套用共同責任模型。下列主題說明如何將 Neptune 設定為實現您的安全及合規目標。您也會學到如何使用其他可 AWS 協助您監控和保護 Neptune 資源的服務。

主題

- Amazon Neptune 中的資料保護
- Amazon Neptune AWS Identity and Access Management （ IAM ）概述
- 在 Neptune 中啟用 IAM 資料庫身分驗證
- 使用簽 AWS 名版本 4 連接和簽名
- 使用 IAM 政策管理存取
- 使用 Neptune 的服務連結角色
- 使用暫時登入資料的 IAM 身分驗證
- 記錄和監控 Amazon Neptune 資源
- Amazon Neptune 的合規驗證
- Amazon Neptune 中的恢復能力

Amazon Neptune 中的資料保護

AWS 共同責任模型適用於 Amazon Neptune 中的資料保護。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全

組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您 AWS 服務 使用主控台、API 或 AWS SDK 與 Neptune 或其他 AWS CLI人合作時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

> **Important**
>
> TLS 1.3 僅支援 Neptune 引擎版本 1.3.2.0 及更新版本。

您可以使用 AWS 已發佈的 API 呼叫透過網路管理 Neptune。用戶端必須使用強式加密套件支援 Transport Layer Security (TLS) 1.2 或更新版本，如 傳輸中加密 中所述。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

以下幾節描述如何保護 Neptune 資料。

主題

- 每個 Amazon Neptune 資料庫叢集都位於 Amazon VPC 中
- 傳輸中加密：使用 SSL/HTTPS 連線至 Neptune
- 靜態加密 Neptune 資源

每個 Amazon Neptune 資料庫叢集都位於 Amazon VPC 中

Amazon Neptune 資料庫叢集只能在 Amazon Virtual Private Cloud (Amazon VPC) 中建立，而且其端點只能在該 VPC 內存取，通常從在該 VPC 中執行的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體進行。

您可以限制對 Neptune 資料庫叢集所在 VPC 的存取，以保護 Neptune 資料，如中 連線至您的 Amazon Neptune 圖形 所述。

傳輸中加密：使用 SSL/HTTPS 連線至 Neptune

從引擎 1.0.4.0 版開始，Amazon Neptune 只允許透過 HTTPS 對任何執行個體或叢集端點進行 Secure Sockets Layer (SSL) 連線。

Neptune 至少需要使用 TLS 1.2 版，使用以下強式密碼套件：

- `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`

從 Neptune 引擎版本 1.3.2.0 開始，Neptune 使用下列加密套件支援 TLS 1.3 版：

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384

即使在舊版引擎中允許 HTTP 連線的情況下，任何使用新資料庫叢集參數群組的資料庫叢集根據預設都必須使用 SSL。為了保護您的資料，引擎版本 *1.0.4.0* 及更新版本的 Neptune 端點僅支援 HTTPS 請求。如需詳細資訊，請參閱使用 HTTP REST 端點連線到 Neptune 資料庫執行個體。

Neptune 會自動為您的 Neptune 資料庫執行個體提供 SSL 憑證。您不需要請求任何憑證。憑證會在您建立新執行個體時提供。

Neptune 會為每個 AWS 區域的帳戶中的執行個體指派單一萬用字元 SSL 憑證。此憑證為叢集端點、叢集唯讀端點、執行個體端點提供項目。

憑證詳細資訊

提供的憑證包含以下項目：

- 叢集端點 - *.cluster-*a1b2c3d4wxyz*.*region*.neptune.amazonaws.com
- 唯讀端點 - *.cluster-ro-*a1b2c3d4wxyz*.*region*.neptune.amazonaws.com
- 執行個體端點 - *.*a1b2c3d4wxyz*.*region*.neptune.amazonaws.com

只支援此處所列的項目。

代理連線

憑證只支援上一節所列的主機名稱。

如果您使用負載平衡器或代理伺服器 (例如 HAProxy)，您必須在代理伺服器上使用 SSL 終止，並有您自己的 SSL 憑證。

SSL 傳遞沒有作用，因為提供的 SSL 憑證與代理伺服器主機名稱不符。

根 CA 憑證

Neptune 執行個體的憑證通常使用作業系統或 SDK (例如 Java SDK) 的本機信任存放區進行驗證。

如果您需要手動提供根憑證，可以從 Amazon Trust Services 政策儲存庫下載 PEM 格式的 Amazon 根 CA 憑證。

詳細資訊

如需使用 SSL 連線至 Neptune 端點的詳細資訊，請參閱 the section called “安裝 Gremlin 主控台” 和 the section called “HTTP REST”。

靜態加密 Neptune 資源

Neptune 加密的執行個體可以協助保護您的資料免於發生未經授權的基礎儲存體存取，為資料提供另一層保護。您可以使用 Neptune 加密來提高部署在雲端之應用程式的資料保護。您也可以使用它來滿足 data-at-rest 加密的合規性要求。

若要管理用於加密和解密 Neptune 資源的金鑰，請使用 AWS Key Management Service ()AWS KMS。 AWS KMS 結合安全、高可用性的硬體和軟體，提供專為雲端擴充的金鑰管理系統。您可以使用建立加密金鑰 AWS KMS，並定義控制如何使用這些金鑰的原則。 AWS KMS 支持 AWS CloudTrail，因此您可以審核密鑰使用情況以驗證密鑰是否正確使用。您可以將 AWS KMS 金鑰與 Neptune 和支援的 AWS 服務結合使用，例如 Amazon 簡單儲存服務 (Amazon S3)、亞馬遜 Elastic

Block Store (Amazon EBS) 和 Amazon Redshift。如需支援的服務清單 AWS KMS，請參閱AWS Key Management Service 開發人員指南 AWS KMS中的 AWS 服務使用方式。

對於 Neptune 加密的執行個體，所有日誌、備份和快照都會加密。

啟用 Neptune 資料庫執行個體的加密

若要針對新的 Neptune 資料庫執行個體啟用加密，請在 Neptune 主控台的啟用加密區段中選擇是。如需建立 Neptune 資料庫執行個體的相關資訊，請參閱 建立新的 Neptune 資料庫叢集。

建立加密的 Neptune 資料庫執行個體時，也可以提供加密 AWS KMS 金鑰的金鑰識別碼。如果您未指定 AWS KMS 金鑰識別碼，Neptune 會針對新的 Neptune 資料庫執行個體使用預設的 Amazon RDS 加密金鑰 (`aws/rds`)。 AWS KMS 為您的 AWS 帳戶建立 Neptune 的預設加密金鑰。您的 AWS 帳戶每個 AWS 區域都有不同的預設加密金鑰。

建立加密的 Neptune 資料庫執行個體之後，您將無法變更該執行個體的加密金鑰。因此，務必在加密 Neptune 資料庫執行個體之前先確定您的加密金鑰需求。

您可以使用另一個帳戶的金鑰的 Amazon Resource Name (ARN) 來加密 Neptune 資料庫執行個體。如果您使用擁有用於加密該新 Neptune 資料庫執行個體的 AWS KMS 加密金鑰的相同 AWS 帳戶建立 Neptune 資料庫執行個體，則您傳遞的 AWS KMS 金鑰識別碼可以是 AWS KMS 金鑰別名，而不是金鑰的 ARN。

> **Important**
>
> 如果 Neptune 失去對 Neptune 資料庫執行個體加密金鑰的存取 (例如，當 Neptune 的金鑰存取被撤銷)，加密的資料庫執行個體將進入結束狀態，且只能從備份恢復。強烈建議您一律針對加密的 Neptune 資料庫執行個體啟用備份，以免遺失資料庫中的加密資料。

啟用加密時所需的金鑰許可

建立加密 Neptune 資料庫執行個體的 IAM 使用者或角色必須至少具有 KMS 金鑰的下列許可：

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:GenerateDataKey"`
- `"kms:ReEncryptTo"`
- `"kms:GenerateDataKeyWithoutPlaintext"`

- "kms:CreateGrant"
- "kms:ReEncryptFrom"
- "kms:DescribeKey"

以下是包含必要許可的金鑰政策範例：

```
{
  "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"
        }
      }
    }
  ]
}
```

- 此政策中的第一個聲明是選用的。它可讓您存取使用者的根主體。
- 第二個陳述式可讓您存取此角色的所有必要 AWS KMS API，範圍為 RDS 服務主體。
- 第三個陳述式強制執行此金鑰無法由此角色用於任何其他 AWS 服務，藉此加強安全性。

您還可以進一步限制 `createGrant` 許可範圍，方法為新增：

```
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
```

Neptune 加密的限制

加密 Neptune 叢集存在以下限制：

- 您無法將未加密的資料庫叢集轉換為已加密的叢集。

  但是，您可以將未加密的 資料庫叢集快照還原至加密的 資料庫叢集。若要執行此作業，請在從未加密資料庫叢集快照還原時指定 KMS 加密金鑰。

- 您無法將未加密的資料庫執行個體轉換為已加密的執行個體。您只能在建立資料庫執行個體時，針對此資料庫執行個體啟用加密。
- 此外，無法修改已加密的資料庫執行個體叢集來停用加密。
- 未加密資料庫執行個體不可以有加密僅供讀取複本，加密資料庫執行個體也不可以有未加密僅供讀取複本。
- 加密的讀取本必須使用與來源資料庫執行個體相同的索引鍵進行加密。

Amazon Neptune AWS Identity and Access Management （ IAM ）概述

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員可以控制誰能完成身分驗證 (登入) 和獲得授權 (具有許可)，而得以使用 Neptune 資源。您可以使用 IAM AWS 服務 ，無需額外付費。

您可以使用 AWS Identity and Access Management (IAM) 向 Neptune 資料庫執行個體或資料庫叢集進行驗證。啟用 IAM 資料庫身份驗證後，每個請求都必須使用 AWS 簽名版本 4 簽署。

AWS 簽名版本 4 將驗證信息添加到 AWS 請求中。基於安全，在啟用 IAM 身分驗證的情況下，所有對 Neptune 資料庫叢集提出的請求都必須使用存取金鑰進行簽署。此金鑰由存取金鑰 ID 和私密存取金鑰組成。身分驗證會使用 IAM 政策從外部進行管理。

Neptune 會在連線上進行驗證，對於 WebSockets 連線，它會定期驗證權限，以確保使用者仍然具有存取權。

> **Note**
>
> - 不建議撤銷、刪除或輪換與 IAM 使用者相關聯的憑證，因為它不會終止任何已開啟的連線。
> - 每個資料庫執行個體的並行 WebSocket 連線數目有限制，以及連線可以保持開啟的時間長度。如需詳細資訊，請參閱 WebSockets 限制。

IAM 用法取決於您的角色

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在 Neptune 中所做的工作。

服務使用者 - 如果您使用 Neptune 服務執行工作，則管理員會為您提供使用 Neptune 資料平面所需的憑證和許可。當您需要更多存取權來執行工作時，了解資料存取的管理方式可協助您向管理員請求正確的許可。

服務管理員 - 如果您負責公司的 Neptune 資源，則可能有權存取 Neptune 管理動作，這些動作對應於 Neptune 管理 API。確定使用者需要哪個 Neptune 資料存取動作和資源服務才能完成其工作，這也可能是您的工作。IAM 管理員接著可以套用 IAM 政策來變更服務使用者的許可。

IAM 管理員 - 如果您是 IAM 管理員，您將需要撰寫 IAM 政策，來同時管理對 Neptune 的管理和資料存取。若要檢視您可以使用的範例 Neptune 身分型政策，請參閱 使用不同種類的 IAM 政策來控制對 Neptune 的存取。

使用身分來驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以便使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。

- 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
- 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。
- 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者)的時機。

在 Neptune 中啟用 IAM 資料庫身分驗證

根據預設，當您建立 Amazon Neptune 資料庫叢集時，系統會停用 IAM 資料庫身分驗證。您可以使用 AWS Management Console啟用 IAM 資料庫身分驗證 (或再次停用它)。

若要使用主控台建立具備 IAM 身分驗證的新 Neptune 資料庫叢集，請遵循 使用 AWS Management Console 啟動 Neptune 資料庫叢集 中建立 Neptune 資料庫叢集的指示。

在建立程序的第二頁，針對 Enable IAM DB Authentication (啟用 IAM 資料庫身分驗證)，選擇 Yes (是)。

為現有資料庫執行個體或叢集啟用或停用 IAM 身分驗證

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在導覽窗格中，選擇叢集。

3. 選擇您想要修改的 Neptune 資料庫叢集，然後選擇叢集動作。然後選擇 Modify cluster (修改叢集)。
4. 在 Database options (資料庫選項) 區段的 IAM DB Authentication (IAM 資料庫身分驗證)，選擇 Enable IAM DB authorization (啟用 IAM 資料庫授權) 或 No (否) (以進行停用)。然後選擇 Continue (繼續)。
5. 若要立即套用變更，請選擇 Apply immediately (立即套用)。
6. 選擇修改叢集。

使用簽 AWS 名版本 4 連接和簽名

啟用 IAM 資料庫身份驗證的 Amazon Neptune 資源需要使用簽 AWS 名版本 4 簽署所有 HTTP 請求。如需使用簽 AWS 名版本 4 簽署要求的一般資訊，請參閱簽署 AWS API 要求。

AWS 簽名版本 4 是將驗證信息添加到 AWS 請求的過程。為了安全起見，大多數的請求都 AWS 必須使用訪問密鑰進行簽名，該訪問密鑰包括訪問密鑰 ID 和秘密訪問密鑰。

> **Note**
>
> 如果您使用臨時登入資料，登入資料會在指定的時間後過期，包括工作階段字符。
> 當您請求新的登入資料時，必須更新工作階段字符。如需詳細資訊，請參閱使用臨時安全登入資料來要求 AWS 資源的存取權。

> **Important**
>
> 透過 IAM 型身分驗證存取 Neptune 會要求您建立 HTTP 請求並自行簽署這些請求。

Signature 第 4 版的運作方式

1. 建立正式的請求。
2. 您可以使用規範要求和其他一些資訊來建立. string-to-sign
3. 您可以使用 AWS 密鑰訪問密鑰來導出簽名密鑰，然後使用該簽名密鑰和創建簽名。 string-to-sign
4. 您可以對標頭中的 HTTP 請求新增所產生的簽章，或做為查詢字串參數。

當 Neptune 收到請求時，便會執行您計算簽章所做的相同步驟。接著 Neptune 就會比較計算簽章與跟請求一併傳送的簽章。如果簽章相符，就會受理請求。如果簽章不相符，則會拒絕請求。

如需有關使用簽 AWS 名版本 4 簽署要求的一般資訊，請參閱 AWS 一般參考.

下列章節會包含範例，示範如何在啟用 IAM 身分驗證的情況下，將已簽署的請求傳送至 Neptune 資料庫執行個體的 Gremlin 和 SPARQL 端點。

主題

Amazon Linux EC2 的事前準備

以下說明如何在 Amazon EC2 執行個體上安裝 Apache Maven 和 Java 8。這些是 Amazon Neptune Signature 第 4 版身分驗證範例的必要項目。

若要在 EC2 執行個體上安裝 Apache Maven 和 Java 8。

1. 使用 SSH 用戶端連線至 Amazon EC2 執行個體。
2. 在 EC2 執行個體上安裝 Apache Maven。首先，輸入以下內容，新增包含 Maven 套件的儲存庫。

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   輸入以下內容以設定套件的版本號碼。

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   接著使用 yum 安裝 Maven。

   ```
   sudo yum install -y apache-maven
   ```

3. Gremlin 二進位碼需要 Java 8。輸入以下內容將 Java 8 安裝在您的 EC2 執行個體上。

```
sudo yum install java-1.8.0-devel
```

4. 輸入以下內容將 Java 8 設定為 EC2 執行個體上預設的執行時間。

```
sudo /usr/sbin/alternatives --config java
```

出現提示時，輸入 Java 8 的數字。

5. 輸入以下內容，將 Java 8 設為 EC2 執行個體上的預設編譯器。

```
sudo /usr/sbin/alternatives --config javac
```

出現提示時，輸入 Java 8 的數字。

使用命令列工具將查詢提交至 Neptune 資料庫叢集

如本文件中的許多範例所示，具有將查詢提交至 Neptune 資料庫叢集的命令列工具非常方便。如果未啟用 IAM 身分驗證，curl 工具是與 Neptune 端點通訊的絕佳選擇。

不過，若要保護您的資料，最好啟用 IAM 身分驗證。

當 IAM 身分驗證啟用時，每個請求都必須使用 Signature 第 4 版 (Sig4) 進行簽署。第三方 awscurl 命令列工具會使用與 `curl` 相同的語法，並且可以使用 Sig4 簽署來簽署查詢。以下 使用 `awscurl` 一節說明如何安全地使用 `awscurl` 搭配臨時憑證。

設定命令列工具來使用 HTTPS

Neptune 要求所有連線都使用 HTTPS。任何命令行工具 (例如 `curl` 或 `awscurl`) 需要存取適當的憑證，才能使用 HTTPS。只要 `curl` 或 `awscurl` 可以找到適當的憑證，它們處理 HTTPS 連線的方式就跟 HTTP 連線一樣，無需額外參數。此文件的範例是以該案例為基礎。

若要了解如何取得這類憑證，以及如何將它們正確格式化為 `curl` 可以使用的憑證授權機構 (CA) 憑證存放區，請參閱 `curl` 文件中的 SSL 憑證驗證。

然後，您可以使用 `CURL_CA_BUNDLE` 環境變數來指定此 CA 憑證存放區的位置。在 Windows 上，`curl` 會自動在名為 `curl-ca-bundle.crt` 的檔案中尋找它。它會先在和 `curl.exe` 相同的目錄中尋找，然後再尋找路徑的其他位置。如需詳細資訊，請參閱 SSL Certificate Verification。

使用 awscurl 搭配臨時憑證，安全地連線至啟用 IAM 身分驗證的資料庫叢集

awscurl 工具會使用與 curl 相同的語法，但也需要額外的資訊：

- **--access_key** – 有效的存取金鑰。如果未使用此參數提供，則必須在 AWS_ACCESS_KEY_ID 環境變數或在組態檔案中提供它。
- **--secret_key** – 對應至存取金鑰的有效私密存鑰。如果未使用此參數提供，則必須在 AWS_SECRET_ACCESS_KEY 環境變數或在組態檔案中提供它。
- **--security_token** – 有效的工作階段權杖。如果未使用此參數提供，則必須在 AWS_SECURITY_TOKEN 環境變數或在組態檔案中提供它。

在過去，常見的做法是使用永久憑證搭配 awscurl (例如 IAM 使用者憑證，甚至是根憑證)，但不建議這麼做。相反地，使用其中一個 AWS Security Token Service (STS) API 或其中一個 AWS CLI 包裝函式來產生臨時憑證。

最好將 STS 呼叫所傳回的 AccessKeyId、SecretAccessKey 和 SessionToken 值放入 Shell 工作階段中的適當環境變數中，而不是放入組態檔案中。然後，當 Shell 終止時，憑證會自動捨棄，而組態檔案不是這種情況。同樣地，請不要為臨時憑證請求長於您可能需要的持續時間。

下列範例展示您可能在 Linux Shell 中採取的步驟，以使用 sts assume-role 取得有效半小時的臨時憑證，然後將它們放在 awscurl 可以找到它們的環境變數中：

```
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
```

然後，您可以使用 awscurl 對資料庫叢集提出簽署的請求，如下所示：

```
awscurl (your cluster endpoint):8182/status \
    --region us-east-1 \
    --service neptune-db
```

搭配 Signature 第 4 版簽署，使用 Gremlin 主控台連線到 Neptune

使用具有簽名版本 4 身份驗證的 Gemlin 控制台連接到 Amazon Neptune 的方式取決於您使用的是版本還是更高 TinkerPop 版本`3.4.11`，還是較早的版本。無論是哪種情況，都需要下列先決條件：

- 您必須具有簽署請求所需的 IAM 憑證。請參閱 AWS SDK for Java 開發人員指南中的使用預設憑證提供者鏈結。
- 您必須已安裝與資料庫叢集所使用的 Neptune 引擎版本相容的 Gemlin 主控台版本。

如果您使用的是臨時憑證，它們會在指定的間隔後過期，而工作階段權杖也一樣，因此在您請求新憑證時必須更新您的工作階段權杖。請參閱 IAM 使用者指南中的使用臨時安全登入 AWS 資料要求存取資源。

如需使用 SSL/TLS 進行連線的說明，請參閱 SSL/TLS 組態。

使用 TinkerPop 3.4.11 或更高版本通過 Sig4 簽名連接到 Neptune

使用 TinkerPop 3.4.11 或更高版本時，您將使用`handshakeInterceptor()`，它提供了一種將 Sigv4 簽署者插入命令建立的連接的方法。`:remote`與用於 Java 的方法一樣，它需要您手動設定`Cluster`物件，然後將其傳遞至`:remote`命令。

請注意，這與`:remote`命令需要組態檔案形成連線的典型情況完全不同。組態檔案方法將無法運作，因為`handshakeInterceptor()`必須以程式設計方式設定，並且無法從檔案中載入其組態。

使用 Sig4 簽名 Connect 小鬼控制台（TinkerPop 3.4.11 及更高版本）

1. 啟動 Gremlin 主控台：

   ```
   $ bin/gremlin.sh
   ```

2. 在`gremlin>`提示中，安裝`amazon-neptune-sigv4-signer`程式庫 (只需要為主控台完成一次此操作)：

   ```
   :install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
   ```

   如果您在此步驟中遇到問題，可能有助於查閱有關葡萄配置的TinkerPop 文檔。

> **Note**
>
> 如果您使用的是 HTTP Proxy，在此步驟中可能會遇到 `:install` 命令未完成的錯誤。若要解決此問題，請執行下列命令，以告知主控台有關 Proxy 的相關資訊：
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```

3. 將處理簽署所需的類別匯入至 `handshakeInterceptor()`：

```
:import com.amazonaws.auth.DefaultAWSCredentialsProviderChain
:import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer
```

4. 如果您使用的是臨時憑證，則還需要提供工作階段權杖，如下所示：

```
System.setProperty("aws.sessionToken","(your session token)")
```

5. 如果您尚未以其他方式建立帳戶憑證，則可以指派它們，如下所示：

```
System.setProperty("aws.accessKeyId","(your access key)")
System.setProperty("aws.secretKey","(your secret key)")
```

6. 手動建構要連線至 Neptune 的 `Cluster` 物件：

```
cluster = Cluster.build("(host name)")  \
                 .enableSsl(true) \
                 .handshakeInterceptor { r ->  \
                   def sigV4Signer = new NeptuneNettyHttpSigV4Signer("(Amazon
 region)", \
                                        new DefaultAWSCredentialsProviderChain()); \
                   sigV4Signer.signRequest(r); \
                   return r; } \
                 .create()
```

如需尋找 Neptune 資料庫執行個體主機名稱的說明，請參閱 連線至 Amazon Neptune 端點。

7. 使用上一個步驟中 `Cluster` 物件的變數名稱來建立 `:remote` 連線：

```
:remote connect tinkerpop.server cluster
```

8. 輸入以下命令以切換至遠端模式。這會將所有 Gremlin 查詢傳送至遠端連線：

```
:remote console
```

使用 3.4.11 之 TinkerPop前的版本通過 Sig4 簽名連接到 Neptune

使用 TinkerPop 3.4.10 或更低版本時，請使用 Neptune 提供的程式`amazon-neptune-gremlin-java-sigv4`庫，透過 Sigv4 簽章將主控台連線到 Neptune，如下所述：

使用 Sig4 簽名 Connect 小鬼控制台（ 3.4.11 之前的TinkerPop 版本 ）

1. 啟動 Gremlin 主控台：

```
$ bin/gremlin.sh
```

2. 在 `gremlin>` 提示中，安裝 `amazon-neptune-sigv4-signer` 程式庫 (只需要為主控台完成一次此操作)：

```
:install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
```

> **Note**
>
> 如果您使用的是 HTTP Proxy，在此步驟中可能會遇到 `:install` 命令未完成的錯誤。若要解決此問題，請執行下列命令，以告知主控台有關 Proxy 的相關資訊：
>
> ```
> System.setProperty("https.proxyHost", "(the proxy IP address)")
> System.setProperty("https.proxyPort", "(the proxy port)")
> ```
>
> 它也可能有助於查閱有關葡萄配置的TinkerPop 文檔。

3. 在解壓縮目錄的 `conf` 子資料夾中，建立名為 `neptune-remote.yaml` 的檔案。

   如果您使用 AWS CloudFormation 範本建立 Neptune 資料庫叢集，則`neptune-remote.yaml`檔案將已存在。在此情況下，您要做的就是編輯現有檔案，以包含通道分離器設定，如下所示。

   否則，請將下列文字複製到檔案，將 *(####)* 取代為 Neptune 資料庫執行個體的主機名稱或 IP 地址。請注意，包含主機名稱的方括號 ([]) 是必要的。

```
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**
>
> 必須提供 IAM 登入資料才能簽署請求。輸入以下命令來將您的登入資料設為環境變數，並使用您的登入資料取代相關項目。
>
> ```
> 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
> ```
>
> Neptune 第 4 版簽署者使用預設憑證提供者鏈結。如需提供憑證的其他方法，請參閱《AWS SDK for Java 開發人員指南》中的使用預設憑證提供者鏈結。
> 即使使用登入資料檔案，SERVICE_REGION 變數也仍是必要項目。

5. 使用 .yaml 檔案建立 :remote 連線：

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

6. 輸入以下命令以切換至遠端模式，此模式會將所有 Gemlin 查詢傳送至遠端連線：

```
:remote console
```

搭配 Signature 第 4 版簽署，使用 Java 和 Gremlin 連線到 Neptune

使用 TinkerPop 3.4.11 或更高版本通過 Sig4 簽名連接到 Neptune

以下是一個示例，說明如何在使用 TinkerPop 3.4.11 或更高版本時使用帶有 Sig4 簽名的小鬼 Java API 連接到 Neptune（它假定有關使用 Maven 的一般知識）。首先，將相依性定義為 pom.xml 檔案的一部分：

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-sigv4-signer</artifactId>
  <version>2.4.0</version>
</dependency>
```

然後，使用如下程式碼：

```
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**
>
> 如果您要從 3.4.11 升級，請移除對 amazon-neptune-gremlin-java-sigv4 程式庫的參考。如上面範例所示，使用 handshakeInterceptor() 時它不再是必要的。不要嘗試搭配通道分離器 (SigV4WebSocketChannelizer.class) 使用 handshakeInterceptor()，因為它會產生錯誤。

使用 3.4.11 之 TinkerPop前的版本通過 Sig4 簽名連接到 Neptune

TinkerPop 之前的版本3.4.11不支援上一節所示的handshakeInterceptor()組態，因此必須依賴amazon-neptune-gremlin-java-sigv4套件。這是一個包含該SigV4WebSocketChannelizer類的 Neptune 庫，它將標準 TinkerPop 通道器替換為可以自動注入 Sigv4 簽名的通道器。在可能的情況下，升級到 TinkerPop 3.4.11 或更高版本，因為該amazon-neptune-gremlin-java-sigv4庫已被棄用。

以下是使用 3.4.11 之前的 TinkerPop 版本（假設有關如何使用 Maven 的一般知識）時，如何使用具有 Sig4 簽名的 Gremlin Java API 連接到 Neptune 的示例。

首先，將相依性定義為 pom.xml 檔案的一部分：

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-gremlin-java-sigv4</artifactId>
  <version>2.4.0</version>
</dependency>
```

上述相依性將包括 Gremlin 驅動程式版本3.4.10。雖然可以使用較新的 Gemlin 驅動程式版本 (直到 3.4.13)，但是升級到 3.4.10 以後的驅動程式應該包含使用上述 handshakeInterceptor() 模型的變更。

然後，gremlin-driver 叢集物件應該在 Java 程式碼中設定，如下所示：

```
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();
```

搭配 Signature 第 4 版簽署，使用 Java 和 SPARQL 連線到 Neptune (RDF4J 和 Jena)

本節說明如何搭配 Signature 第 4 版身分驗證，使用 RDF4J 或 Apache Jena 連線到 Neptune。

必要條件

- Java 8 或更高版本。
- Apache Maven 3.3 或更高版本。

  如需在執行 Amazon Linux 的 EC2 執行個體上安裝這些先決條件的詳細資訊，請參閱Amazon Linux EC2 的事前準備。
- 用於簽署請求的 IAM 登入資料。如需詳細資訊，請參閱《AWS SDK for Java 開發人員指南》中的使用預設憑證提供者鏈結。

  > **Note**
  >
  > 如果您使用臨時登入資料，登入資料會在指定的時間後過期，包括工作階段字符。
  > 當您請求新的登入資料時，必須更新工作階段字符。如需詳細資訊，請參閱 IAM 使用者指南中的使用臨時安全登入資料申請 AWS 資源存取權。
- 將 SERVICE_REGION 變數設為下列其中一項，指出 Neptune 資料庫執行個體的區域：
  - 美國東部 (維吉尼亞北部)：us-east-1
  - 美國東部 (俄亥俄)：us-east-2
  - 美國西部 (加利佛尼亞北部)：us-west-1
  - 美國西部 (奧勒岡)：us-west-2
  - 加拿大 (中部)：ca-central-1

- 南美洲 (聖保羅) : sa-east-1
- 歐洲 (斯德哥爾摩) : eu-north-1
- 歐洲 (愛爾蘭) : eu-west-1
- 歐洲 (倫敦) : eu-west-2
- 歐洲 (巴黎) : eu-west-3
- 歐洲 (法蘭克福) : eu-central-1
- 中東 (巴林) : me-south-1
- 中東 (阿拉伯聯合大公國) : me-central-1
- 以色列 (特拉維夫) : il-central-1
- 非洲 (開普敦) : af-south-1
- 亞太區域 (香港) : ap-east-1
- 亞太區域 (東京) : ap-northeast-1
- 亞太區域 (首爾) : ap-northeast-2
- 亞太區域 (大阪): ap-northeast-3
- 亞太區域 (新加坡) : ap-southeast-1
- 亞太區域 (雪梨) : ap-southeast-2
- 亞太區域 (孟買) : ap-south-1
- 中國 (北京) : cn-north-1
- 中國 (寧夏) : cn-northwest-1
- AWS GovCloud (美國西部): us-gov-west-1
- AWS GovCloud (美國東部): us-gov-east-1

搭配 Signature 第 4 版簽署，使用 RDF4J 或 Apache Jena 連線到 Neptune

1. 從複製範例存放庫 GitHub。

```
git clone https://github.com/aws/amazon-neptune-sparql-java-sigv4.git
```

2. 變更到複製的目錄。

```
cd amazon-neptune-sparql-java-sigv4
```

3. 透過簽出具有 latest (最新) 標籤的分支，取得專案的最新版本。

```
git checkout $(git describe --tags `git rev-list --tags --max-count=1`)
```

4. 輸入以下其中一個命令來編譯及執行範例程式碼。

   將 *your-neptune-endpoint* 取代為 Neptune 資料庫執行個體的主機名稱或 IP 地址。預設連接埠為 8182。

   > **Note**
   >
   > 如需尋找 Neptune 資料庫執行個體主機名稱的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

   Eclipse RDF4J

   輸入以下內容來執行 RDF4J 範例。

   ```
   mvn compile exec:java \
       -Dexec.mainClass="com.amazonaws.neptune.client.rdf4j.NeptuneRdf4JSigV4Example"
    \
       -Dexec.args="https://your-neptune-endpoint:port"
   ```

   Apache Jena

   輸入以下內容來執行 Apache Jena 範例。

   ```
   mvn compile exec:java \
       -Dexec.mainClass="com.amazonaws.neptune.client.jena.NeptuneJenaSigV4Example" \
       -Dexec.args="https://your-neptune-endpoint:port"
   ```

5. 若要查看範例的原始程式碼，請參閱 `src/main/java/com/amazonaws/neptune/client/` 目錄中的範例。

若要在您自己 Java 應用程式中使用 SigV4 簽署驅動程式，請將 `amazon-neptune-sigv4-signer` Maven 套件新增到您 `pom.xml` 的 `<dependencies>` 區段。我們建議您使用範例做為起點。

搭配 Signature 第 4 版簽署，使用 SPARQL 和 Node.js 連線到 Neptune

使用簽名 V4 簽名和 AWS SDK 進行查詢

以下是如何使用具有簽名版本 4 身份驗證的 Node.js 連接到 Neptune SPARQL 的示例： AWS

```
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);
    }
  );
}
```

使用簽名 V4 簽名和 AWS SDK 進行查詢

以下是如何使用具有簽名版本 4 身份驗證的 Node.js 連接到 Neptune SPARQL 的示例： AWS

```
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);
    });
```

```
}
```

範例：搭配 Signature 第 4 版簽署，使用 Python 連線到 Neptune

本節顯示以 Python 撰寫的程式範例，說明如何針對 Amazon Neptune 使用 Signature 第 4 版。此範例是根據 https://docs.aws.amazon.com/general/latest/gr/sigv4-signed-request-examples.html 中的 Amazon Web Services 一般參考 Signature 第 4 版簽署程序部分。

若要使用此範例程式，您需要以下項目：

- 將 Python 3.x 安裝在電腦上，這個程式可從 Python 網站取得。這些程式已使用 Python 3.6 來測試。
- Python requests library，在範例指令碼中用來進行 web 請求。Python 套件安裝方便，只要使用 `pip` 即可從 Python 套件索引網站取得套件。然後，您可以在命令列執行 `pip install requests`，即可安裝 `requests`。
- 環境變數中的存取金鑰 (存取金鑰 ID 和私密存取金鑰) 名為 `AWS_ACCESS_KEY_ID` `AWS_SECRET_ACCESS_KEY`。根據最佳實務，建議您請勿 在程式碼中內嵌登入資料。如需詳細資訊，請參閱《AWS Account Management 參考指南》中的 AWS 帳戶的最佳實務。

  名為 `SERVICE_REGION` 的環境變數中 Neptune 資料庫叢集的區域。

  如果您使用臨時的登入資料，除了指定 `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、`SERVICE_REGION`，還必須指定 `AWS_SESSION_TOKEN`。

> **Note**
>
> 如果您使用臨時登入資料，登入資料會在指定的時間後過期，包括工作階段字符。當您請求新的登入資料時，必須更新工作階段字符。如需詳細資訊，請參閱使用臨時安全憑證來請求對 AWS 資源的存取。

以下範例說明如何使用 Python 對 Neptune 提出已簽署的請求。此請求提出 GET 或 POST 請求。身分驗證資訊會使用 `Authorization` 請求標頭來傳遞。

這個例子也可以作為一個 AWS Lambda 函數。如需詳細資訊，請參閱 the section called “設定 Lambda”。

對 Gremlin 和 SPARQL Neptune 端點提出已簽署的請求

1. 建立名為 neptunesigv4.py 的新檔案，並在文字編輯器中開啟。
2. 複製以下程式碼，並在 neptunesigv4.py 檔案中貼上。

```
# 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. 在終端機瀏覽到 neptunesigv4.py 檔案的位置。
4. 輸入以下命令，將存取金鑰、私有金鑰和區域取代成正確值。

```
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
```

如果您使用臨時的登入資料，除了指定 AWS_ACCESS_KEY_ID、AWS_SECRET_ACCESS_KEY、SERVICE_REGION，還必須指定 AWS_SESSION_TOKEN。

```
export AWS_SESSION_TOKEN=MY_AWS_SESSION_TOKEN
```

> **Note**
>
> 如果您使用臨時登入資料，登入資料會在指定的時間後過期，包括工作階段字符。

當您請求新的登入資料時，必須更新工作階段字符。如需詳細資訊，請參閱使用臨時安全憑證來請求對 AWS 資源的存取。

5. 輸入以下其中一個命令，將已簽署的請求傳送到 Neptune 資料庫執行個體。這些範例使用 Python 3.6 版。

端點狀態

```
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()"
```

Gremlin 狀態

```
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> }"
```

SPARQL 狀態

```
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;"
```

openCypher 狀態

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p 8182 -a GET -q openCypher/
status
```

載入器

```
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. 執行 Python 指令碼的語法如下：

```
python3.6 neptunesigv4.py -ho your-neptune-endpoint -p port -a GET|POST -q gremlin|
sparql|sparqlupdate|loader|status -d "string0data"
```

SPARQL UPDATE 要求 POST。

使用 IAM 政策管理存取

IAM 政策是 JSON 物件，可定義使用動作和資源的許可。

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS

以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

搭 AWS 配組織使用服務控制原則 (SCP)

服務控制策略（SCP）是 JSON 策略，用於指定中AWS Organizations組織或組織單位（OU）的最大權限。 AWS Organizations 是一種用於分組和集中管理您企業擁有的多個 AWS 帳戶的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者。如需有關 Organizations 和 SCP 的詳細資訊，請參閱《 AWS Organizations 使用者指南》中的 SCP 如何運作。

在 AWS 組織內的 AWS 帳戶中部署 Amazon Neptune 的客戶可以利用 SCP 控制哪些帳戶可以使用 Neptune。若要確定可在成員帳戶內存取 Neptune，請務必分別 `neptune:*` 和 `neptune-db:*`，允許存取控制平面和資料平面 IAM 動作。

使用 Amazon Neptune 主控台所需的許可

對於使用 Amazon Neptune 主控台的使用者，該使用者必須擁有一組符合最低限制的許可。這些許可允許使用者描述其 AWS 帳戶的 Neptune 資源，並提供其他相關資訊，包括 Amazon EC2 安全性和網路資訊。

如果您建立比最基本必要許可更嚴格的 IAM 政策，則對於採取該 IAM 政策的使用者而言， 主控台就無法如預期運作。為了確保這些使用者仍可使用 Neptune 主控台，也請將 `NeptuneReadOnlyAccess` 受管政策連線至使用者，如 AWS Amazon Neptune 的受管 (預先定義) 政策 中所述。

您不需要為僅對 AWS CLI 或 Amazon Neptune API 進行呼叫的使用者允許最低主控台許可。

將 IAM 政策附加至 IAM 使用者

若要套用受管或自訂政策，請將它附加至 IAM 使用者。如需此主題的教學，請參閱 IAM 使用者指南中的建立和連接您的第一個客戶受管政策。

在您進行教學課程時，可使用本節所示的其中一個政策範例做為起點，並依您的需求進行自訂。教學課程結束時，您將有一個連接政策的 IAM 使用者可使用 `neptune-db:*` 動作。

> **Important**
>
> - 最多需要 10 分鐘才能將 IAM 政策的變更套用到指定的 Neptune 資源。
> - 已套用到 Neptune 資料庫叢集的 IAM 政策也會套用到該叢集中的所有執行個體。

使用不同種類的 IAM 政策來控制對 Neptune 的存取

若要提供 Neptune 管理動作或 Neptune 資料庫叢集中資料的存取權，您可以將政策附加至 IAM 使用者或角色。如需如何將 IAM 政策附加至使用者的相關資訊，請參閱 將 IAM 政策附加至 IAM 使用者。如需將政策附加到角色的相關資訊，請參閱《IAM 使用者指南》中的新增與移除 IAM 政策。

如需對 Neptune 的一般存取，您可以使用 Neptune 的其中一個受管政策。如需更受限制的存取，您可以使用 Neptune 支援的管理動作和資源來建立自己的自訂政策。

在自訂 IAM 政策中，您可以使用兩個不同種類的政策陳述式，控制 Neptune 資料庫叢集的不同存取模式：

- 管理政策陳述式 – 管理政策陳述式可讓您存取 Neptune 管理 API，您可以使用這些 API 來建立、設定和管理資料庫叢集及其執行個體。

  因為 Neptune 與 Amazon RDS 共用功能，所以 Neptune 政策中的管理動作、資源和條件金鑰會依設計使用 `rds:` 字首。
- 資料存取政策聲明 – 資料存取政策陳述式會使用資料存取動作、資源和條件金鑰，控制如何存取資料庫叢集包含的資料。

Neptune 資料存取動作、資源和條件金鑰會使用 neptune-db: 字首。

在 Amazon Neptune 中使用 IAM 條件內容金鑰

您可以在控制 Neptune 存取權的 IAM 政策陳述式中指定條件。然後，政策陳述式只有在條件成立時才會生效。

例如，您可能想要政策陳述式只在特定日期之後生效，或者只有在請求中存在特定值時才允許存取。

若要表示條件，請在政策陳述式的 Condition 中使用預先定義的條件金鑰，搭配 IAM 條件政策運算子，例如等於或小於。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 政策元素：變數和標籤。

條件金鑰的資料類型會判斷您可以使用哪些條件運算子，來將請求中的值與政策陳述式中的值比較。如果您使用與該資料類型不相容的條件運算子，則比對一律失敗且政策陳述式永不套用。

Neptune 支援管理政策陳述式與資料存取政策陳述式各有不同的條件金鑰集：

- 管理政策陳述式的條件金鑰
- 資料存取政策的條件金鑰

支援 Amazon Neptune 中的 IAM 政策和存取控制功能

下表顯示 Neptune 針對管理政策陳述式和資料存取政策陳述式支援的 IAM 功能：

您可以搭配 Neptune 使用的 IAM 功能

IAM 功能	管理	資料存取
身分型政策	是	是

IAM 功能	管理	資料存取
資源型政策	否	否
政策動作	是	是
政策資源	是	是
全域條件鍵	是	(子集)
標籤型條件金鑰	是	否
存取控制清單 (ACL)	否	否
服務控制政策 (SCP)	是	是
服務連結角色	是	否

IAM 政策限制

最多需要 10 分鐘才能將 IAM 政策的變更套用到指定的 Neptune 資源。

已套用到 Neptune 資料庫叢集的 IAM 政策也會套用到該叢集中的所有執行個體。

Neptune 目前不支援跨帳戶存取控制。

AWS Amazon Neptune 的受管 (預先定義) 政策

AWS 透過提供由建立和管理的獨立 IAM 政策來解決許多常見使用案例 AWS。受管政策授與常見使用案例中必要的許可，讓您免於查詢需要哪些許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

您可以將下列 AWS 受管政策附加到帳戶中的使用者，適用於使用 Amazon Neptune 管理 API：

- NeptuneReadOnlyAccess— 授予對所有 Neptune 資源的唯讀存取權，以用於根 AWS 帳戶中的管理和資料存取目的。
- NeptuneFull存取權 — 授予對所有 Neptune 資源的完整存取權，以用於根 AWS 帳戶中的管理和資料存取目的。如果您需要從 AWS CLI 或 SDK 進行完整的 Neptune 存取權，但不能存 AWS Management Console 取，則建議使用此選項。

- NeptuneConsoleFullAccess— 授與根 AWS 帳戶中的所有 Neptune 管理動作和資源的完整存取權，但不授與任何資料存取動作或資源。它也會包括額外的許可，簡化來自主控台的 Neptune 存取，包括有限的 IAM 和 Amazon EC2 (VPC) 許可。
- NeptuneGraphReadOnlyAccess — 提供對所有 Amazon Neptune 分析資源的唯讀存取，以及相依服務的唯讀許可
- AWSServiceRoleForNeptuneGraphPolicy-讓 Neptune 分析圖形發布 CloudWatch 操作和使用指標和日誌。

Neptune IAM 角色和政策會授與 Amazon RDS 資源的部分存取權，因為 Neptune 和 Amazon RD 共用操作技術以提供某些管理功能。這包括管理 API 許可，這就是為什麼 Neptune 管理動作具有 `rds:` 字首的原因。

Neptune 受 AWS 管原則的更新

下表追蹤 Neptune 受管政策的更新，從 Neptune 開始追蹤這些變更的時間開始：

政策	描述	日期
AWS Amazon Neptune 的受管政策-更新現有政策	`NeptuneReadOnlyAccess` 和受`NeptuneFullAccess` 管理的政策現在會在原則陳述式中包含 `Sid` (陳述式 ID) 做為識別碼。	2024-01-22
NeptuneGraphReadOnly訪問權限（已發布）	發行以提供 Neptune Analytics 圖形和資源的唯讀存取權。	2023-11-29
AWSServiceRoleForNeptuneGraphPolicy(已發行)	發佈以允許 Neptune Analytics 圖形存取 CloudWatch 以發佈操作和使用量指標和記錄。請參閱 Neptune Analytics 中的使用服務連結角色 (SLR)。	2023-11-29
NeptuneConsoleFullAccess（添加權限）	新增的許可權提供與 Neptune Analytics 圖形互動所需的所有存取權。	2023-11/29

政策	描述	日期
NeptuneFull存取權限 (新增權限)	已新增資料存取許可和新全球資料庫 API 的許可。	2022-07-28
NeptuneConsoleFullAccess（添加權限）	已新增新全球資料庫 API 的許可。	2022-07-21
Neptune 已開始追蹤變更	Neptune 開始追蹤其受 AWS 管理原則的變更。	2022-07-21

NeptuneReadOnlyAccessAWS 受管政策

以下NeptuneReadOnlyAccess受管理政策授予對所有 Neptune 動作和資源的唯讀存取權，以用於管理和資料存取目的。

> **Note**
>
> 此政策已於 2022 年 7 月 21 日更新，以包含唯讀資料存取許可以及唯讀管理許可，也包含全球資料庫動作的許可。

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

NeptuneFullAccessAWS 受管政策

以下NeptuneFull存取受管理政策授予對所有 Neptune 動作和資源的完整存取權，以供系統管理和資料存取用途使用。如果您需要從 SDK AWS CLI 或從 SDK 進行完整存取，但不需要從 AWS Management Console.

> **Note**
>
> 此政策已於 2022 年 7 月 21 日更新，以包含完整資料存取許可以及完整管理許可，也包含全球資料庫動作的許可。

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

NeptuneConsoleFullAccessAWS 受管政策

出於NeptuneConsoleFullAccess管理目的，以下受管理策略授予對所有 Neptune 操作和資源的完整訪問權限，但不能用於數據訪問目的。它也會包括額外的許可，簡化來自主控台的 Neptune 存取，包括有限的 IAM 和 Amazon EC2 (VPC) 許可。

> **Note**
>
> 此政策已於 2023-11-29 更新，包含與 Neptune Analytics 圖形互動所需的許可權。
> 此政策已於 2022-07-21 更新，包含全球資料庫動作的許可。

```
{
  "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"
        }
      }
    }
  ]
}
```

NeptuneGraphReadOnlyAccessAWS 受管政策

以下NeptuneGraphReadOnly存取受管政策提供對所有 Amazon Neptune Analytics 資源的唯讀存取權限，以及相依服務的唯讀許可。

此政策包含執行以下動作的許可：

- 對於 Amazon EC2 — 擷取 VPC、子網路、安全群組和可用區域的相關資訊。
- 用於 AWS KMS — 擷取 KMS 金鑰和別名的相關資訊。
- 用於 CloudWatch — 擷取有關 CloudWatch 測量結果的資訊。
- 對於 CloudWatch 記錄檔 — 擷取有關記 CloudWatch 錄資料流和事件的資訊。

> **Note**
>
> 這項政策已於 2023-11-29 發布。

```
{
  "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:*"
      ]
    }
  ]
}
```

AWSServiceRoleForNeptuneGraphPolicyAWS 受管政策

下列AWSServiceRoleForNeptuneGraphPolicy受管理政策可讓圖形存 CloudWatch 取發佈作業和使用量度和記錄檔。請參閱 nan-service-linked-roles。

> **Note**
>
> 這項已政策於 2023-11-29 發布。

```
{
  "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}"
        }
      }
    }
  ]
}
```

Amazon Neptune 支援的 IAM 條件內容金鑰

您可以在 IAM 政策中指定條件，控制對 Neptune 管理動作和資源的存取。然後，政策陳述式只有在條件成立時才會生效。

例如，您可能想要政策陳述式只在特定日期之後生效，或者只有在 API 請求中存在特定值時才允許存取。

若要表示條件，請在政策陳述式的 `Condition` 中使用預先定義的條件金鑰，搭配 IAM 條件政策運算子，例如等於或小於。

若您在陳述式中指定多個 `Condition` 元素，或是在單一 `Condition` 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 政策元素：變數和標籤。

條件金鑰的資料類型會判斷您可以使用哪些條件運算子，來將請求中的值與政策陳述式中的值比較。如果您使用與該資料類型不相容的條件運算子，則比對一律失敗且政策陳述式永不套用。

Neptune 管理政策陳述式的 IAM 條件金鑰

- 全域條件金鑰 — 您可以在 Neptune 管理原則陳述式中使用大部分的 AWS 全域條件金鑰。
- 服務特定條件金鑰 — 這些是針對特定 AWS 服務定義的金鑰。Neptune IAM 管理政策陳述式中可用的條件金鑰 中列出了 Neptune 針對管理政策陳述式支援的條件金鑰。

Neptune 資料存取政策陳述式的 IAM 條件金鑰

- 全域條件金鑰 – AWS Neptune 在資料存取原則陳述式中支援的全域條件內容金鑰 中列出了 Neptune 在資料存取政策陳述式中所支援的這些金鑰的子集。
- 條件金鑰 中列出了 Neptune 針對資料存取政策陳述式定義的服務特定條件金鑰。

Amazon Neptune 的自訂 IAM 管理政策陳述式

管理政策陳述式可讓您控制 IAM 使用者可以做什麼來管理 Neptune 資料庫。

Neptune 管理政策陳述式會授與 Neptune 支援的一或多個管理動作和管理資源的存取權。您也可以使用 條件金鑰 讓管理許可更具體。

> **Note**
>
> 因為 Neptune 與 Amazon RDS 共用功能，所以管理政策陳述式中的管理動作、資源和服務特定條件金鑰會依設計使用 `rds:` 字首。

主題

Neptune IAM 管理政策陳述式中可用的動作

您可以在 IAM 政策陳述式的 `Action` 元素中使用下面列出的管理動作，來控制對 Neptune 管理 API 的存取。在政策中使用動作時，通常會允許或拒絕存取相同名稱的 API 操作或 CLI 命令。不過，在某些情況下，單一動作可控制對多個操作的存取。或者，某些操作需要多種不同的動作。

下列清單中的 `Resource type` 欄位指示每個動作是否支援資源層級許可。如果此欄位沒有值，您必須在政策陳述式的 `Resource` 元素中指定所有資源 ("*")。如果資料欄包含資源類型，則您可以在具有該動作的陳述式中指定該類型的資源 ARN。此頁面上列出了 Neptune 管理資源類型。

下列清單中的必要資源會以星號 (*) 表示。如果您在使用此動作的陳述式中指定資源層級許可 ARN，則它必須屬於此類型。某些動作支援多種資源類型。如果資源類型是選用的 (換句話說，沒有以星號標記)，則您不必包含它。

如需此處所列欄位的詳細資訊，請參閱《IAM 使用者指南》中的動作表。

rds：目AddRole標資料庫叢集

`AddRoleToDBCluster` 會將 IAM 角色與 Neptune 資料庫叢集建立關聯。

存取層級：`Write`。

相依動作：`iam:PassRole`。

資源類型：cluster (必要)。

rds: AddSource IdentifierTo 訂閱

AddSourceIdentifierToSubscription 會將來源識別符新增至現有的 Neptune 事件通知訂閱。

存取層級：Write。

資源類型：es (必要)

RDS：AddTagsToResource

AddTagsToResource 會將 IAM 角色與 Neptune 資料庫叢集建立關聯。

存取層級：Write。

資源類型：

- db
- es
- pg
- cluster-snapshot
- subgrp

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

RDS：ApplyPendingMaintenanceAction

ApplyPendingMaintenanceAction 會將待定維護動作套用到資源。

存取層級：Write。

資源類型：db (必要)。

RDS: 複ClusterParameter製資料庫群組

CopyDBClusterParameterGroup 會複製指定的資料庫叢集參數群組。

存取層級：Write。

資源類型：cluster-pg (必要)。

RDS: 複製資料庫 ClusterSnapshot

CopyDBClusterSnapshot 會複製資料庫叢集的快照。

存取層級：Write。

資源類型：cluster-snapshot (必要)。

RDS: 複製資料庫 ParameterGroup

CopyDBParameterGroup 會複製指定的資料庫參數群組。

存取層級：Write。

資源類型：pg (必要)。

rds:CreateDBCluster

CreateDBCluster 會建立新的 Neptune 資料庫叢集。

存取層級：Tagging。

相依動作：iam:PassRole。

資源類型：

- cluster (必要)。
- cluster-pg (必要)。
- subgrp (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys
- 海王星-RDS_ DatabaseEngine

RDS: ClusterParameter 建立資料庫群組

CreateDBClusterParameterGroup 會建立新的資料庫叢集參數群組。

存取層級：Tagging。

資源類型：cluster-pg (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

RDS: 建立資料庫 ClusterSnapshot

CreateDBClusterSnapshot 會建立資料庫叢集的快照。

存取層級：Tagging。

資源類型：

- cluster (必要)。
- cluster-snapshot (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

rds:CreateDBInstance

CreateDBInstance 會建立新的資料庫執行個體。

存取層級：Tagging。

相依動作：iam:PassRole。

資源類型：

- db (必要)。
- pg (必要)。
- subgrp (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

RDS: 建立資料庫 ParameterGroup

CreateDBParameterGroup 會建立新的資料庫參數群組。

存取層級：Tagging。

資源類型：pg (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

RDS: 建立資料庫 SubnetGroup

CreateDBSubnetGroup 會建立新的資料庫子網路群組。

存取層級：Tagging。

資源類型：subgrp (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

rds: CreateEvent 訂閱

CreateEventSubscription 會建立 Neptune 事件通知訂閱。

存取層級：Tagging。

資源類型：es (必要)

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

rds:DeleteDBCluster

DeleteDBCluster 會刪除現有的 Neptune 資料庫叢集。

存取層級：Write。

資源類型：

- cluster (必要)。
- cluster-snapshot (必要)。

RDS: 刪除資ClusterParameter料庫群組

DeleteDBClusterParameterGroup 會刪除指定的資料庫叢集參數群組。

存取層級：Write。

資源類型：cluster-pg (必要)。

RDS: 刪除資料庫 ClusterSnapshot

DeleteDBClusterSnapshot 會刪除資料庫叢集快照。

存取層級：Write。

資源類型：cluster-snapshot (必要)。

rds:DeleteDBInstance

DeleteDBInstance 會刪除指定的資料庫執行個體。

存取層級：Write。

資源類型：db (必要)。

RDS: 刪除資料庫 ParameterGroup

DeleteDBParameterGroup刪除指定的數據庫ParameterGroup。

存取層級：Write。

資源類型：pg (必要)。

RDS: 刪除資料庫 SubnetGroup

DeleteDBSubnetGroup 會刪除資料庫子網路群組。

存取層級：Write。

資源類型：subgrp (必要)。

rds: DeleteEvent 訂閱

DeleteEventSubscription 會刪除事件通知訂閱。

存取層級：Write。

資源類型：es (必要)

RDS: 描述 B 群ClusterParameter組

DescribeDBClusterParameterGroups返回數據庫ClusterParameterGroup 描述的列表。

存取層級：List。

資源類型：cluster-pg (必要)。

RDS: 描述 B ClusterParameters

DescribeDBClusterParameters 會傳回特定資料庫叢集參數群組的詳細參數清單。

存取層級：List。

資源類型：cluster-pg (必要)。

RDS: 描述屬性 ClusterSnapshot

DescribeDBClusterSnapshotAttributes 會傳回手動資料庫叢集快照之資料庫叢集快照屬性名稱和值的清單。

存取層級：List。

資源類型：cluster-snapshot (必要)。

RDS: 描述 B ClusterSnapshots

DescribeDBClusterSnapshots 會傳回資料庫叢集快照的相關資訊。

存取層級：Read。

rds:DescribeDBClusters

DescribeDBClusters 會傳回佈建 Neptune 資料庫叢集的相關資訊。

存取層級：List。

資源類型：cluster (必要)。

RDS: 描述 B EngineVersions

DescribeDBEngineVersions 會傳回可用的資料庫引擎清單。

存取層級：List。

資源類型：pg (必要)。

rds:DescribeDBInstances

DescribeDBInstances 會傳回資料庫執行個體的相關資訊。

存取層級：List。

資源類型：es (必要)

RDS: 描述 B ParameterGroups

DescribeDBParameterGroups返回數據庫ParameterGroup 描述的列表。

存取層級：List。

資源類型：pg (必要)。

rds:DescribeDBParameters

DescribeDBParameters 會傳回特定資料庫參數群組的詳細參數清單。

存取層級：List。

資源類型：pg (必要)。

RDS: 描述 B SubnetGroups

DescribeDBSubnetGroups返回數據庫SubnetGroup 描述的列表。

存取層級：List。

資源類型：subgrp (必要)。

rds: DescribeEvent 類別

DescribeEventCategories 會傳回所有事件來源類型或特定來源類型 (如果指定) 的類別清單。

存取層級：List。

rds：DescribeEvent訂閱

DescribeEventSubscriptions 會列出客戶帳戶的所有訂閱描述。

存取層級：List。

資源類型：es (必要)

RDS：DescribeEvents

DescribeEvents 會傳回過去 14 天與資料庫執行個體、資料庫安全群組和資料庫參數群組相關的事件。

存取層級：List。

資源類型：es (必要)

RDS: DescribeOrderable 資料庫 InstanceOptions

DescribeOrderableDBInstanceOptions 會傳回指定引擎的可排序資料庫執行個體選項清單。

存取層級：List。

RDS：DescribePendingMaintenanceActions

DescribePendingMaintenanceActions 會傳回至少有一個待處理維護動作的資源清單 (例如，資料庫執行個體)。

存取層級：List。

資源類型：db (必要)。

RDS: DescribeValid 資料庫 InstanceModifications

DescribeValidDBInstanceModifications 會列出您對資料庫執行個體可做的修改。

存取層級：List。

資源類型：db (必要)。

rds:FailoverDBCluster

FailoverDBCluster 會強制資料庫叢集進行容錯移轉。

存取層級：Write。

資源類型：cluster (必要)。

RDS：ListTagsForResource

ListTagsForResource 會列出 Neptune 資源的所有標籤。

存取層級：Read。

資源類型：

- cluster-snapshot
- db
- es
- pg
- subgrp

rds:ModifyDBCluster

ModifyDBCluster

修改 Neptune 資料庫叢集的設定。

存取層級：Write。

相依動作：iam:PassRole。

資源類型：

- cluster (必要)。
- cluster-pg (必要)。

RDS: 修改資ClusterParameter料庫群組

ModifyDBClusterParameterGroup 會修改資料庫叢集參數群組的參數。

存取層級：Write。

資源類型：cluster-pg (必要)。

RDS: 修改資ClusterSnapshot料庫屬性

ModifyDBClusterSnapshotAttribute 會在手動資料庫叢集快照中新增或移除屬性和值。

存取層級：Write。

資源類型：cluster-snapshot (必要)。

rds:ModifyDBInstance

ModifyDBInstance 會修改資料庫執行個體的設定。

存取層級：Write。

相依動作：iam:PassRole。

資源類型：

- db (必要)。
- pg (必要)。

RDS: 修改資料庫 ParameterGroup

ModifyDBParameterGroup 會修改資料庫參數群組的參數。

存取層級：Write。

資源類型：pg (必要)。

RDS: 修改資料庫 SubnetGroup

ModifyDBSubnetGroup 會修改現有的資料庫子網路群組。

存取層級：Write。

資源類型：subgrp (必要)。

rds: ModifyEvent 訂閱

ModifyEventSubscription 會修改現有的 Neptune 事件通知訂閱。

存取層級：Write。

資源類型：es (必要)

rds:RebootDBInstance

RebootDBInstance 會重新啟動執行個體的資料庫引擎服務。

存取層級：Write。

資源類型：db (必要)。

rds: RemoveRole 從叢集

RemoveRoleFromDBCluster取消 AWS Identity and Access Management (IAM) 角色與 Amazon Neptune 資料庫叢集的關聯性。

存取層級：Write。

相依動作：iam:PassRole。

資源類型：cluster (必要)。

rds: RemoveSource IdentifierFrom 訂閱

RemoveSourceIdentifierFromSubscription 會從現有的 Neptune 事件通知訂閱移除來源識別符。

存取層級：Write。

資源類型：es (必要)

RDS：RemoveTagsFromResource

RemoveTagsFromResource 會從 Neptune 資源中移除中繼資料標籤。

存取層級：Tagging。

資源類型：

- cluster-snapshot

- db
- es
- pg
- subgrp

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

RDS: ClusterParameter 重設資料庫群組

ResetDBClusterParameterGroup 會將資料庫叢集參數群組的參數修改為預設值。

存取層級：Write。

資源類型：cluster-pg (必要)。

RDS: 重設資料庫 ParameterGroup

ResetDBParameterGroup 會將資料庫參數群組的參數修改為引擎/系統的預設值。

存取層級：Write。

資源類型：pg (必要)。

RDS: ClusterFrom 還原快照

RestoreDBClusterFromSnapshot 會從資料庫叢集快照建立新的資料庫叢集。

存取層級：Write。

相依動作：iam:PassRole。

資源類型：

- cluster (必要)。
- cluster-snapshot (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

RDS ClusterToPointIn: 恢復時間

RestoreDBClusterToPointInTime 會將資料庫叢集還原到任意時間點。

存取層級：Write。

相依動作：iam:PassRole。

資源類型：

- cluster (必要)。
- subgrp (必要)。

條件金鑰：

- AWS：RequestTag/###
- AWS：TagKeys

rds:StartDBCluster

StartDBCluster 會啟動指定的資料庫叢集。

存取層級：Write。

資源類型：cluster (必要)。

rds:StopDBCluster

StopDBCluster 會停止指定的資料庫叢集。

存取層級：Write。

資源類型：cluster (必要)。

IAM Neptune 管理政策陳述式中可用的資源類型

Neptune 支援下表中的資源類型，用於 IAM 管理政策陳述式的 Resource 元素。如需 Resource 元素的詳細資訊，請參閱 IAM JSON 政策元素：Resource。

Neptune 管理動作清單會識別可隨每個動作指定的資源類型。資源類型也會確定您可在政策中包括哪些條件金鑰，如下表的最後一欄所指定。

下表中的 ARN 欄會指定參考此類型資源必須使用的 Amazon Resource Name (ARN) 格式。 $ 後面的部分必須取代為您案例的實際值。例如，如果您在 ARN 中看到 $user-name，您必須將該字串取代為實際 IAM 使用者的名稱，或取代為包含 IAM 使用者名稱的政策變數。如需 ARN 的詳細資訊，請參閱 IAM ARN 和 在 Amazon Neptune 中使用管理 ARN。

Condition Keys 欄會指定只有在此陳述式中同時包含此資源和相容的支援動作時，您才能在 IAM 政策陳述式中包含的條件內容金鑰。

資源類型	ARN	條件金鑰
cluster (資料庫叢集)	arn:*partition* :rds:*region*:*account-id* :cluster: *instance-name*	AWS：ResourceTag/### rds:cluster-tag/*tag-key*
cluster-pg (資料庫叢集參數群組)	arn:*partition* :rds:*region*:*account-id* :cluster-pg: *neptune-DBClusterParameterGroupName*	AWS：ResourceTag/###
cluster-snapshot (資料庫叢集快照)	arn:*partition* :rds:*region*:*account-id* :cluster-snapshot: *neptune-DBClusterSnapshotName*	AWS：ResourceTag/### rds:cluster-snapshot-tag/*tag-key*
db (資料庫執行個體)	arn:*partition* :rds:*region*:*account-id* :db:*neptune-DbInstanceName*	AWS：ResourceTag/### RDS：DatabaseClass RDS：DatabaseEngine rds:db-tag/*tag-key*

資源類型	ARN	條件金鑰
es (事件訂閱)	arn:*partition* :rds:*region*:*account-id* :es:*neptune-CustSubscriptionId*	AWS：ResourceTag/### rds:es-tag/*tag-key*
pg (資料庫參數群組)	arn:*partition* :rds:*region*:*account-id* :pg:*neptune-ParameterGroupName*	AWS：ResourceTag/### rds:pg-tag/*tag-key*
subgrp (資料庫子網路群組)	arn:*partition* :rds:*region*:*account-id* :subgrp:*neptune-DBSubnetGroupName* }	AWS：ResourceTag/### rds:subgrp-tag/*tag-key*

Neptune IAM 管理政策陳述式中可用的條件金鑰

使用條件金鑰，您可以在 IAM 政策陳述式中指定條件，以便陳述式只在條件成立時才生效。您可以在 Neptune 管理政策陳述式中使用的條件金鑰分為下列類別：

- 全域條件金鑰 — 這些金鑰由定義，以 AWS 供 AWS 服務一般使用。大部分可以用於 Neptune 管理政策陳述式。
- 管理資源屬性條件金鑰 – 下面列出的這些金鑰是以管理資源的屬性為基礎。
- 標籤型存取條件金鑰 – 下面列出的這些金鑰是以附加至管理資源的 AWS 標籤為基礎。

Neptune 管理資源屬性條件金鑰

條件索引鍵	描述	Type
rds:DatabaseClass	依資料庫執行個體類別的類型篩選存取	字串
rds:DatabaseEngine	依資料庫引擎來篩選存取權限。如需可能的值，請參閱建立資料庫執行個體 API 中的引擎參數	字串

條件索引鍵	描述	Type
rds:DatabaseName	依資料庫執行個體上的資料庫使用者定義名稱來篩選存取權限。	字串
rds:EndpointType	依端點類型篩選存取權限。READER、WRITER、CUSTOM 的其中一個	字串
rds:Vpc	依此值指定資料庫執行個體是否在 Amazon Virtual Private Cloud (Amazon VPC) 中執行來篩選存取權限。若要指示資料庫執行個體在 Amazon VPC 中執行，請指定 true。	Boolean

管理標籤型條件金鑰

Amazon Neptune 支援在 IAM 政策中使用自訂標籤來指定條件，以透過 管理 API 參考 控制對 Neptune 的存取。

例如，如果您將名為 environment 的標籤新增至資料庫執行個體，而此標籤具有 beta、staging 和 production 等值，則您可以建立一個政策，根據該標籤的值限制對執行個體的存取。

> **Important**
>
> 如果您使用標記來管理對 Neptune 資源的存取，請務必保護對標籤的存取。您可以建立 AddTagsToResource 和 RemoveTagsFromResource 動作的政策，來限制對標籤的存取。例如，您可以使用下列政策，拒絕使用者可對所有資源新增或移除標籤的能力。然後，您可以建立政策來允許特定使用者新增或移除標籤。

```
{ "Version": "2012-10-17",
  "Statement":[
    { "Sid": "DenyTagUpdates",
      "Effect": "Deny",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource":"*"
    }
  ]
}
```

下列標籤型條件金鑰僅會在管理政策陳述式中使用管理資源。

標籤型管理條件金鑰

條件索引鍵	描述	Type
aws:RequestTag/${TagKey}	根據請求中存在的標籤金鑰值對來篩選存取。	字串
aws:ResourceTag/${TagKey}	根據附加到資源的標籤金鑰值對來篩選存取。	字串
aws:TagKeys	根據請求中存在的標籤金鑰來篩選存取。	字串
rds:cluster-pg-tag/${TagKey}	依附加到資料庫叢集參數群組的標籤來篩選存取。	字串
rds:cluster-snapshot-tag/${TagKey}	依附加到資料庫叢集快照的標籤來篩選存取。	字串
rds:cluster-tag/${TagKey}	依附加到資料庫叢集的標籤來篩選存取。	字串
rds:db-tag/${TagKey}	依附加到資料庫執行個體的標籤來篩選存取。	字串
rds:es-tag/${TagKey}	依附加到事件訂閱的標籤來篩選存取。	字串
rds:pg-tag/${TagKey}	依附加到資料庫參數群組的標籤來篩選存取。	字串

條件索引鍵	描述	Type
rds:req-tag/${TagKey}	依限制可用來標記資源的一組標籤金鑰和值來篩選存取。	字串
rds:secgrp-tag/${TagKey}	依附加到資料庫安全群組的標籤來篩選存取。	字串
rds:snapshot-tag/${TagKey}	依附加到資料庫快照的標籤來篩選存取。	字串
rds:subgrp-tag/${TagKey}	依附加至資料庫子網路群組的標籤來篩選存取權限	字串

Neptune 的 IAM 管理政策陳述式範例

一般管理政策範例

以下範例說明如何建立 Neptune 管理政策，授與對資料庫叢集執行各種管理動作的許可。

防止 IAM 使用者刪除所指定資料庫執行個體的政策

以下是防止 IAM 使用者刪除所指定 Neptune 資料庫執行個體的範例政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteOneInstance",
      "Effect": "Deny",
      "Action": "rds:DeleteDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-instance-name"
    }
  ]
}
```

授與許可來建立新資料庫執行個體的政策

以下是允許 IAM 使用者在指定的 Neptune 資料庫叢集中建立資料庫執行個體的範例政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstance",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster"
    }
  ]
}
```

授與許可來建立使用特定資料庫參數群組之新資料庫執行個體的政策

以下範例政策允許 IAM 使用者僅使用指定的資料庫參數群組，在指定的 Neptune 資料庫叢集 (此處的 us-west-2) 中建立資料庫執行個體。

```
{
  "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"
      ]
    }
  ]
}
```

授與許可來描述任何資源的政策

以下是允許 IAM 使用者描述任何 Neptune 資源的範例政策。

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

```
  "Statement": [
    {
      "Sid": "AllowDescribe",
      "Effect": "Allow",
      "Action": "rds:Describe*",
      "Resource": *
    }
  ]
}
```

標籤型管理政策範例

以下範例說明如何建立 Neptune 管理政策，使用標籤來篩選資料庫叢集上各種管理動作的許可。

範例 1：使用可以採取多個值的自訂標籤，對資源上的動作授與許可

以下政策允許在 env 標籤設定為 dev 或 test 的任何資料庫執行個體上使用 ModifyDBInstance、CreateDBInstance 或 DeleteDBInstance API：

```
{ "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"
        }
      }
    }
  ]
}
```

範例 2：限制可用來標記資源的一組標籤金鑰和值

此政策會使用 Condition 金鑰，允許將具有金鑰 env 和值 test、qa 或 dev 的標籤新增至資源：

```
{ "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"
        }
      }
    }
  ]
}
```

範例 3：允許根據 `aws:ResourceTag` 完整存取 Neptune 資源

以下政策與上述第一個範例類似，但會改用 aws:ResourceTag：

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullAccessToDev",
      "Effect": "Allow",
      "Action": [
        "rds:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/env": "dev",
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Amazon Neptune 的自訂 IAM 資料存取政策陳述式

Neptune 資料存取政策陳述式會使用資料存取動作、資源和條件金鑰，它們前面全都會加上字首 `neptune-db:`。

主題

- 在 Neptune 資料存取政策陳述式中使用查詢動作
- Neptune IAM 資料存取政策陳述式中可用的動作
- 在 Neptune IAM 資料存取政策陳述式中指定資源
- Neptune IAM 資料存取政策陳述式中可用的條件金鑰
- Neptune IAM 資料存取政策的範例

在 Neptune 資料存取政策陳述式中使用查詢動作

有三個 Neptune 查詢動作可以用於資料存取政策陳述式，即 `ReadDataViaQuery`、`WriteDataViaQuery` 和 `DeleteDataViaQuery`。特定查詢可能需要許可才能執行其中多個動作，而且必須允許這些動作的哪個組合才能執行查詢，可能並不總是顯而易見。

在執行查詢之前，Neptune 會確定執行查詢每個步驟所需的許可，並將這些許可合併成查詢所需的完整許可集。請注意，這個完整許可集包括查詢可能會執行的所有動作，這不一定是查詢在資料上執行時將實際執行的動作集。

這表示若要允許給定的查詢執行，您必須為查詢可能執行的每個動作提供許可，不論查詢是否實際執行這些動作。

以下是一些範例 Gremlin 查詢，其中更詳細地解釋了此情況：

- ```
  g.V().count()
  ```

  `g.V()` 和 `count()` 只需要讀取存取權，因此整體查詢只需要 `ReadDataViaQuery` 存取權。
  ```

- ```
  g.addV()
  ```

  addV() 需要在插入新頂點之前檢查具有給定 ID 的頂點是否存在。這表示它同時需要 ReadDataViaQuery 和 WriteDataViaQuery 存取權。

- ```
  g.V('1').as('a').out('created').addE('createdBy').to('a')
  ```

  g.V('1').as('a') 和 out('created') 只需要讀取存取權，但 addE().from('a') 同時需要讀取和寫入存取權，因為 addE() 需要讀取 from 和 to 頂點，並檢查具有相同 ID 的邊緣是否已經存在，然後再新增邊緣。因此，整體查詢同時需要 ReadDataViaQuery 和 WriteDataViaQuery 存取權。

- ```
  g.V().drop()
  ```

  g.V() 僅需要讀取存取權。drop() 同時需要讀取和刪除存取權，因為其需要在刪除頂點或邊緣之前讀取它，所以整體查詢同時需要 ReadDataViaQuery 和 DeleteDataViaQuery 存取權。

- ```
  g.V('1').property(single, 'key1', 'value1')
  ```

  g.V('1') 只需要讀取存取權，但 property(single, 'key1', 'value1') 需要讀取、寫入和刪除存取權。在這裡，property() 步驟會插入金鑰和值，如果它們不存在於頂點的 話，但如果它們確實已經存在，它會刪除現有的屬性值，並在其位置插入一個新值。因此，整個查詢需要 ReadDataViaQuery、WriteDataViaQuery 和 DeleteDataViaQuery 存取權。

  任何包含 property() 步驟的查詢都需要 ReadDataViaQuery、WriteDataViaQuery 和 DeleteDataViaQuery 許可。

以下是一些 openCypher 範例：

- ```
  MATCH (n)
  RETURN n
  ```

  此查詢讀取資料庫中的所有節點並傳回它們，這只需要 ReadDataViaQuery 存取權。

- ```
  MATCH (n:Person)
  SET n.dept = 'AWS'
  ```

此查詢需要 ReadDataViaQuery、WriteDataViaQuery 和 DeleteDataViaQuery 存取權。它會讀取標籤為 'Person' 的所有節點，並將具有金鑰 dept 和值 AWS 的新屬性新增至其中，或者如果 dept 屬性已經存在，則其會刪除舊值並改為插入 AWS。此外，如果要設定的值為 null，則 SET 會完全刪除屬性。

因為 SET 子句在某些情況下可能需要刪除現有值，所以它始終需要 DeleteDataViaQuery 許可以及 ReadDataViaQuery 和 WriteDataViaQuery 許可。

- ```
  MATCH (n:Person)
  DETACH DELETE n
  ```

  此查詢需要 ReadDataViaQuery 和 DeleteDataViaQuery 許可。它會找出所有具有標籤 Person 的節點，然後刪除它們以及連線至這些節點的邊緣和任何相關聯的標籤和屬性。

- ```
  MERGE (n:Person {name: 'John'})-[:knows]->(:Person {name: 'Peter'})
  RETURN n
  ```

  此查詢需要 ReadDataViaQuery 和 WriteDataViaQuery 許可。MERGE 子句會比對指定的模式或建立它。因為，如果模式不符，則可能會發生寫入，所以需要寫入許可以及讀取許可。

Neptune IAM 資料存取政策陳述式中可用的動作

請注意，Neptune 資料存取動作具有字首 neptune-db:，而 Neptune 中的管理動作則具有字首 rds:。

IAM 中資料資源的 Amazon Resource Name (ARN) 與在建立時指派給叢集的 ARN 不同。您必須建構 ARN，如指定資料資源中所示。這類資料資源 ARN 可以使用萬用字元來包含多個資源。

資料存取原則陳述式也可以包含 neptune-db: QueryLanguage 條件索引鍵，以便依據查詢語言限制存取。

從 版本：1.2.0.0 (2022 年 7 月 21 日) 開始，Neptune 支援將許可限制為一或多個特定 Neptune 動作。這提供了可能比以前更精細的存取控制。

> **Important**
>
> - 最多需要 10 分鐘才能將 IAM 政策的變更套用到指定的 Neptune 資源。
> - 已套用到 Neptune 資料庫叢集的 IAM 政策也會套用到該叢集中的所有執行個體。

查詢型資料存取動作

> **Note**
>
> 需要哪些許可才能執行給定查詢並不總是顯而易見，因為查詢可能會採取多個動作，取決於它們處理的資料。如需詳細資訊，請參閱使用查詢動作。

neptune-db:ReadDataViaQuery

ReadDataViaQuery 允許使用者透過提交查詢從 Neptune 資料庫讀取資料。

動作群組：唯讀、讀寫。

動作內容金鑰：neptune-db:QueryLanguage。

必要資源：資料庫。

neptune-db:WriteDataViaQuery

WriteDataViaQuery 允許使用者透過提交查詢將資料寫入 Neptune 資料庫。

動作群組：讀寫。

動作內容金鑰：neptune-db:QueryLanguage。

必要資源：資料庫。

neptune-db:DeleteDataViaQuery

DeleteDataViaQuery 允許使用者透過提交查詢從 Neptune 資料庫刪除資料。

動作群組：讀寫。

動作內容金鑰：neptune-db:QueryLanguage。

必要資源：資料庫。

neptune-db:GetQueryStatus

GetQueryStatus 允許使用者檢查所有作用中查詢的狀態。

動作群組：唯讀、讀寫。

動作內容金鑰：neptune-db:QueryLanguage。

必要資源：資料庫。

neptune-db:GetStreamRecords

GetStreamRecords 允許使用者從 Neptune 擷取串流記錄。

動作群組：讀寫。

動作內容金鑰：neptune-db:QueryLanguage。

必要資源：資料庫。

neptune-db:CancelQuery

CancelQuery 允許使用者取消查詢。

動作群組：讀寫。

必要資源：資料庫。

一般資料存取動作

neptune-db:GetEngineStatus

GetEngineStatus 允許使用者檢查 Neptune 引擎的狀態。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:GetStatisticsStatus

GetStatisticsStatus 允許使用者檢查正為資料庫收集之統計資料的狀態。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:GetGraphSummary

GetGraphSummary 圖形摘要 API 可讓您擷取圖形的唯讀摘要。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:ManageStatistics

ManageStatistics 允許使用者管理資料庫的統計資料集合。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:DeleteStatistics

DeleteStatistics 允許使用者刪除資料庫中的所有統計資料。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:ResetDatabase

ResetDatabase 允許使用者取得重設所需的權杖，以及重設 Neptune 資料庫。

動作群組：讀寫。

必要資源：資料庫。

大量載入器資料存取動作

neptune-db:StartLoaderJob

StartLoaderJob 允許使用者啟動大量載入器工作。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:GetLoaderJobStatus

GetLoaderJobStatus 允許使用者檢查大量載入器工作的狀態。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:ListLoaderJobs

ListLoaderJobs 允許使用者列出所有大量載入器工作。

動作群組：僅列出、唯讀、讀寫。

必要資源：資料庫。

neptune-db:CancelLoaderJob

CancelLoaderJob 允許使用者取消載入器工作。

動作群組：讀寫。

必要資源：資料庫。

機器學習資料存取動作

neptune-db:StartMLDataProcessingJob

StartMLDataProcessingJob 允許使用者啟動 Neptune ML 資料處理工作。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:StartMLModelTrainingJob

StartMLModelTrainingJob 允許使用者啟動 ML 模型訓練工作。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:StartMLModelTransformJob

StartMLModelTransformJob 允許使用者啟動 ML 模型轉換工作。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:CreateMLEndpoint

CreateMLEndpoint 允許使用者建立 Neptune ML 端點。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:GetMLDataProcessingJobStatus

GetMLDataProcessingJobStatus 允許使用者檢查 Neptune ML 資料處理工作的狀態。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:GetMLModelTrainingJobStatus

GetMLModelTrainingJobStatus 允許使用者檢查 Neptune ML 模型訓練工作的狀態。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:GetMLModelTransformJobStatus

GetMLModelTransformJobStatus 允許使用者檢查 Neptune ML 模型轉換工作的狀態。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:GetMLEndpointStatus

GetMLEndpointStatus 允許使用者檢查 Neptune ML 端點的狀態。

動作群組：唯讀、讀寫。

必要資源：資料庫。

neptune-db:ListMLDataProcessingJobs

ListMLDataProcessingJobs 允許使用者列出所有 Neptune ML 資料處理工作。

動作群組：僅列出、唯讀、讀寫。

必要資源：資料庫。

neptune-db:ListMLModelTrainingJobs

ListMLModelTrainingJobs 允許使用者列出所有 Neptune ML 模型訓練工作。

動作群組：僅列出、唯讀、讀寫。

必要資源：資料庫。

neptune-db:ListMLModelTransformJobs

ListMLModelTransformJobs 允許使用者列出所有 ML 模型轉換工作。

動作群組：僅列出、唯讀、讀寫。

必要資源：資料庫。

neptune-db:ListMLEndpoints

ListMLEndpoints 允許使用者列出所有 Neptune ML 端點。

動作群組：僅列出、唯讀、讀寫。

必要資源：資料庫。

neptune-db:CancelMLDataProcessingJob

CancelMLDataProcessingJob 允許使用者取消 Neptune ML 資料處理工作。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:CancelMLModelTrainingJob

CancelMLModelTrainingJob 允許使用者取消 Neptune ML 模型訓練工作。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:CancelMLModelTransformJob

CancelMLModelTransformJob 允許使用者取消 Neptune ML 模型轉換工作。

動作群組：讀寫。

必要資源：資料庫。

neptune-db:DeleteMLEndpoint

DeleteMLEndpoint 允許使用者刪除 Neptune ML 端點。

動作群組：讀寫。

必要資源：資料庫。

在 Neptune IAM 資料存取政策陳述式中指定資源

資料資源與資料動作一樣，也有 neptune-db: 字首。

在 Neptune 資料存取政策中，您可以使用下列格式，在 ARN 中指定要授與存取權的資料庫叢集：

```
arn:aws:neptune-db:region:account-id:cluster-resource-id/*
```

這樣的資源 ARN 包含以下幾個部分：

- *region*是 Amazon Neptune 資料庫叢集的 AWS 區域。
- *account-id* 是資料庫叢集的 AWS 帳戶號碼。
- *cluster-resource-id* 是資料庫叢集的資源 ID。

> **Important**
>
> cluster-resource-id 與叢集識別符不同。若要在 Neptune 中尋找叢集資源識別碼 AWS Management Console，請查看相關資料庫叢集的 [組態] 區段。

Neptune IAM 資料存取政策陳述式中可用的條件金鑰

使用條件金鑰，您可以在 IAM 政策陳述式中指定條件，以便陳述式只在條件成立時才生效。

您可以在 Neptune 資料存取政策陳述式中使用的條件金鑰分為下列類別：

- 全域條件索引鍵 — Neptune 在資料存取原則陳述式中支援的 AWS 全域條件金鑰子集如下所列。
- 服務特定條件金鑰 – 這些是 Neptune 定義的金鑰，專門用於資料存取政策陳述式。目前只有一個，neptune-db: QueryLanguage，它只有在使用特定的查詢語言時才授予訪問權限。

AWS Neptune 在資料存取原則陳述式中支援的全域條件內容金鑰

下表列出 Amazon Neptune 支援用於資料存取政策陳述式的 AWS 全域條件內容金鑰子集：

您可以在資料存取政策陳述式中使用的全域條件金鑰

條件金鑰	描述	Type
aws:CurrentTime	依請求的目前日期和時間篩選存取。	String
aws:EpochTime	按請求的日期和時間 (以 UNIX epoch 值表示) 篩選存取。	Numeric
aws:PrincipalAccount	依請求主體所屬帳戶篩選存取。	String
aws:PrincipalArn	依提出請求之主體的 ARN 篩選存取。	String
aws:PrincipalIsAWSService	只有當 AWS 服務主體直接進行呼叫時，才允許存取。	Boolean
aws:PrincipalOrgID	依據請求主參與者所屬「組織」中 AWS 組織的識別碼篩選存取。	String
aws:PrincipalOrgPaths	依照提出請求之主參與者的「Organ AWS izations」路徑篩選存取。	String
aws:PrincipalTag	依附加至提出請求之主體的標籤篩選存取。	String

條件金鑰	描述	Type
aws:PrincipalType	依提出請求之主體的類型篩選存取。	String
aws:RequestedRegion	依要求中呼叫的 AWS 區域篩選存取。	String
aws:SecureTransport	僅在使用 SSL 傳送請求時才允許存取。	Boolean
aws:SourceIp	依請求者的 IP 地址篩選存取。	String
aws:TokenIssueTime	依發出臨時安全憑證的日期/時間篩選存取。	String
aws:UserAgent	依請求者的用戶端應用程式來篩選存取權。	String
aws:userid	依請求者的主體識別符篩選存取。	String
aws:ViaAWSService	只有在 AWS 服務代表您提出要求時，才允許存取。	Boolean

Neptune 服務特定條件金鑰

Neptune 支援 IAM 政策的下列服務特定條件金鑰：

Neptune 服務特定條件金鑰

條件金鑰	描述	Type
neptune-db:QueryLanguage	依使用的查詢語言篩選資料存取。 有效值為：Gremlin、OpenCypher 和 Sparql。 支援的動作為 ReadDataViaQuery 、WriteDataViaQuery 、DeleteDataViaQuery 、GetQueryStatus 和 CancelQuery 。	String

Neptune IAM 資料存取政策的範例

下列範例說明如何建立自訂 IAM 政策，使用 Neptune 引擎 1.2.0.0 版中引進的資料平面 API 和動作的精細存取控制。

允許不受限制地存取 Neptune 資料庫叢集中資料的政策範例

以下範例政策可讓 IAM 使用者利用 IAM 資料庫身分驗證，連線至 Neptune 資料庫叢集，以及使用 "*" 字元比對所有可用的動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

上述範例包含資源 ARN，其格式為 Neptune IAM 身分驗證的專屬格式。若要建構 ARN，請參閱指定資料資源。請注意，用於 IAM 授權 Resource 的 ARN 不同於在建立時指派給叢集的 ARN。

允許對 Neptune 資料庫叢集進行唯讀存取的政策範例

以下政策會授與對 Neptune 資料庫叢集中資料進行完整唯讀存取的許可：

```
{
  "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/*"
    }
  ]
```

```
}
```

允許對 Neptune 資料庫叢集的所有存取的政策範例

預設 IAM 動作是拒絕存取資料庫叢集，除非已授與 Allow「效果」。但是，下列原則會拒絕特定 AWS 帳戶和區域對資料庫叢集的所有存取，然後該帳戶和區域的優先順序高於任何Allow影響。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

透過查詢授與讀取存取權的政策範例

以下政策只會授與使用查詢從 Neptune 資料庫叢集讀取的許可：

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:ReadDataViaQuery",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

只允許 Gremlin 查詢的政策範例

以下政策會使用 neptune-db:QueryLanguage 條件金鑰，授與僅使用 Gremlin 查詢語言查詢 Neptune 的許可：

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery",
        "neptune-db:DeleteDataViaQuery"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "neptune-db:QueryLanguage": "Gremlin"
        }
      }
    }
  ]
}
```

允許除了 Neptune ML 模型管理以外的所有存取的政策範例

以下政策會授與 Neptune 圖形操作的完整存取權，但 Neptune ML 模型管理功能除外：

```
{
  "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/*"
    }
  ]
}
```

允許存取 Neptune ML 模型管理的政策範例

此政策會授與 Neptune ML 模型管理功能的存取權：

```
{
  "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/*"
    }
  ]
}
```

授與完整查詢存取權的政策

以下政策會授與 Neptune 圖形查詢操作的完整存取權，但不會授與快速重設、串流、大量載入器、Neptune ML 模型管理等功能的完整存取權：

```
{
  "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/*"
    }
  ]
}
```

僅對 Gemlin 查詢授與完整存取權的政策範例

以下政策會授與使用 Gremlin 查詢語言完整存取 Neptune 圖形查詢操作的權限，但不會授權其他語言的查詢，也不會授權快速重設、串流、大量載入器、Neptune ML 模型管理等功能：

```
{
  "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"
          }
        }
      }
    ]
}
```

授與完整存取權 (快速重設除外) 的政策範例

以下政策會授與 Neptune 資料庫叢集的完整存取權，但使用快速重設除外：

```
{
  "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/*"
    }
  ]
}
```

使用 Neptune 的服務連結角色

Amazon Neptune 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Neptune 的一種獨特 IAM 角色類型。Neptune 會預先定義服務連結角色，並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

> **Important**
>
> 對於某些管理功能，Amazon Neptune 使用與 Amazon RDS 共用的操作技術。這包括服務連結角色和管理 API 許可。

服務連結角色可讓您更輕鬆使用 Neptune，因為您不需要手動新增必要許可。Neptune 定義其服務連結角色的許可，除非另有定義，否則僅有 Neptune 可以擔任其角色。定義的許可包括信任政策和許可政策，並且該許可政策不能附加到任何其他 IAM 實體。

您必須先刪除角色的相關資源，才能刪除角色。如此可保護您的 Neptune 資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱使用 IAM 的AWS 服務，並尋找服務連結角色欄顯示為是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Neptune 的服務連結角色許可

Neptune 使用AWSServiceRoleForRDS服務連結角色，允許 Neptune 和 Amazon RDS 代表您的資料庫執行個體呼叫 AWS 服務。AWSServiceRoleForRDS 服務連結角色信任 rds.amazonaws.com 服務來擔任該角色。

此角色許可政策允許 Neptune 對指定資源完成下列動作：

- 在 ec2 上的動作：
  - AssignPrivateIpAddresses
  - AuthorizeSecurityGroupIngress
  - CreateNetworkInterface
  - CreateSecurityGroup
  - DeleteNetworkInterface
  - DeleteSecurityGroup
  - DescribeAvailabilityZones
  - DescribeInternetGateways
  - DescribeSecurityGroups
  - DescribeSubnets
  - DescribeVpcAttribute
  - DescribeVpcs
  - ModifyNetworkInterfaceAttribute
  - RevokeSecurityGroupIngress
  - UnassignPrivateIpAddresses
- 在 sns 上的動作：

  - ListTopic
  - Publish
- 在 cloudwatch 上的動作：
  - PutMetricData
  - GetMetricData
  - CreateLogStream
  - PullLogEvents
  - DescribeLogStreams
  - CreateLogGroup

> **Note**
>
> 您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。您可能會遇到下列錯誤訊息：
> 無法建立資源。請確認您擁有建立服務連結角色的許可。否則請等待，然後再試一次。
> 如果您看到此訊息，請確認您已啟用以下許可：
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> 如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Neptune 的服務連結角色

您不需要手動建立一個服務連結角色。當您建立執行個體或叢集時，Neptune 會為您建立服務連結角色。

> **Important**
>
> 若要進一步了解，請參閱《IAM 使用者指南》中的我的 IAM 帳戶中出現新角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立執行個體或叢集時，Neptune 會再次為您建立服務連結角色。

編輯 Neptune 的服務連結角色

Neptune 不允許您編輯 AWSServiceRoleForRDS 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Neptune 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。不過，您必須先刪除您的所有執行個體和叢集，才能刪除關聯的服務連結角色。

刪除之前先清除服務連結角色

您必須先確認服務連結角色沒有作用中的工作階段，並移除該角色使用的資源，之後才能使用 IAM 將其刪除。

檢查服務連結角色是否於 IAM 主控台有作用中的工作階段

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色。然後選擇 AWSServiceRoleForRDS 角色的名稱 (而非核取方塊)。
3. 在所選角色的 Summary (摘要) 頁面中，選擇 Access Advisor (存取 Advisor) 分頁。
4. 在 Access Advisor (存取 Advisor) 分頁中，檢閱服務連結角色的近期活動。

> **Note**
>
> 如果您不確定 Neptune 是否正在使用 AWSServiceRoleForRDS 角色，可嘗試刪除該角色。如果服務正在使用該角色，則刪除會失敗，而您可以檢視正在使用該角色的 區域。如

果服務正在使用該角色，您必須先等到工作階段結束，才能刪除該角色。您無法撤銷服務連結角色的工作階段。

如果您想要移除 AWSServiceRoleForRDS 角色，則必須先刪除您的所有執行個體和叢集。

刪除您的所有執行個體

使用下列其中一個程序來刪除您的每一個執行個體。

刪除執行個體 (主控台)

1. 前往 https://console.aws.amazon.com/rds/，開啟 Amazon RDS 主控台。
2. 在導覽窗格中，選擇執行個體。
3. 在 Instances (執行個體) 清單中，選擇您要刪除的執行個體。
4. 選擇 Instance actions (執行個體動作)，然後選擇 Delete (刪除)。
5. 若出現 Create final Snapshot? (是否建立最終快照？) 提示訊息，請選擇 Yes (是) 或 No (否)。
6. 如果您在前一個步驟中選擇 Yes (是)，則對於 Final snapshot name (最終快照名稱)，輸入您的最終快照名稱。
7. 選擇刪除。

刪除執行個體 (AWS CLI)

請參閱 AWS CLI 命令參考中的 `delete-db-instance`。

刪除執行個體 (API)

請參閱`DeleteDBInstance`。

刪除您的所有叢集

使用以下其中一個程序來刪除單一叢集，然後為每個叢集重複此程序。

刪除叢集 (主控台)

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在 Clusters (叢集) 清單中，選擇您要刪除的叢集。

3. 選擇 Cluster Actions (叢集動作)，然後選擇 Delete (刪除)。
4. 選擇刪除。

刪除叢集 (CLI)

請參閱 AWS CLI 命令參考中的 `delete-db-cluster`。

刪除叢集 (API)

請參閱`DeleteDBCluster`

您可以使用 IAM 主控台、IAM CLI 或 IAM API 刪除 `AWSServiceRoleForRDS` 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

使用暫時登入資料的 IAM 身分驗證

Amazon Neptune 支援使用臨時憑證進行 IAM 身分驗證。

您可以使用假定的角色，並使用 IAM 身分驗證政策進行身分驗證，就像前面幾節的其中一個範例政策一樣。

如果您使用臨時的登入資料，除了指定 `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、`SERVICE_REGION`，還必須指定 `AWS_SESSION_TOKEN`。

> **Note**
>
> 臨時登入資料會在指定的時間後過期，包括工作階段字符。
> 當您請求新的登入資料時，必須更新工作階段字符。如需詳細資訊，請參閱使用臨時安全登入資料來要求 AWS 資源的存取權。

以下各節說明如何允許存取和擷取臨時登入資料。

使用暫時登入資料進行身分驗證

1. 建立具有 Neptune 叢集存取許可的 IAM 角色。如需建立此角色的詳細資訊，請參閱the section called “IAM 政策的類型”。
2. 為角色新增一個允許存取登入資料的信任關係。

擷取臨時登入資料，包括 AWS_ACCESS_KEY_ID、AWS_SECRET_ACCESS_KEY 和 AWS_SESSION_TOKEN。

3. 連線到 Neptune 叢集，並使用臨時憑證簽署請求。如需連線及簽署請求的詳細資訊，請參閱 the section called “連線與簽署”。

有多種方法可根據環境擷取臨時登入資料。

主題

- 使用 AWS CLI取得臨時登入資料
- 為 Neptune IAM 身分驗證設定 AWS Lambda
- 為 Neptune IAM 身分驗證設定 Amazon EC2

使用 AWS CLI取得臨時登入資料

若要使用 AWS Command Line Interface (AWS CLI) 取得認證，首先您需要新增信任關係，以授與將執行 AWS CLI 命令的使用 AWS 者承擔角色的權限。

將以下信任關係新增至 Neptune IAM 身分驗證角色。若您沒有 Neptune IAM 身分驗證角色，請參閱 the section called “IAM 政策的類型”。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如需將信任關係新增到角色的相關資訊，請參閱《AWS Directory Service 管理指南》中的編輯現有角色的信任關係。

若 Neptune 政策尚未附加到角色，請建立新角色。附加 Neptune IAM 身分驗證政策，然後新增信任政策。如需建立新角色的資訊，請參閱建立新角色。

> **Note**
>
> 以下各節假設您已安 AWS CLI 裝。

若要 AWS CLI 手動執行

1. 輸入以下命令，使用 AWS CLI請求登入資料。將角色 ARN、工作階段名稱、描述檔取代為您自己的值。

```
aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/NeptuneIAMAuthRole
 --role-session-name test --profile testprofile
```

2. 以下為此命令的範例輸出。Credentials 區段包含您需要的值。

> **Note**
>
> 記錄 Expiration 值，因為在這之後您將需要取得新的登入資料。

```
{
    "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. 使用傳回的登入資料設定環境變數。

```
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. 使用以下其中一個方法連接。

   - the section called “Gremlin 主控台”
   - the section called “Gremlin Java”
   - the section called “SPARQL Java (RDF4J 和 Jena)”
   - the section called “Python 範例”

使用指令碼取得登入資料

1. 執行以下命令來安裝 jq 命令。該腳本使用此命令來解析 AWS CLI 命令的輸出。

```
sudo yum -y install jq
```

2. 在文字編輯器中建立名為 `credentials.sh` 的檔案，並加入下列文字。將服務區域、角色 ARN、工作階段名稱、描述檔取代為您自己的值。

```
#!/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. 使用以下其中一個方法連接。

- the section called “Gremlin 主控台”
- the section called “Gremlin Java”
- the section called “SPARQL Java (RDF4J 和 Jena)”
- the section called “Python 範例”

為 Neptune IAM 身分驗證設定 AWS Lambda

AWS Lambda 每次執行 Lambda 函數時，都會自動包含認證。

首先，您需要新增信任關係，將擔任角色的許可授與 Lambda 服務。

將以下信任關係新增至 Neptune IAM 身分驗證角色。若您沒有 Neptune IAM 身分驗證角色，請參閱 the section called “IAM 政策的類型”。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如需將信任關係新增到角色的相關資訊，請參閱《AWS Directory Service管理指南》中的編輯現有角色的信任關係。

若 Neptune 政策尚未附加到角色，請建立新角色。附加 Neptune IAM 身分驗證政策，然後新增信任政策。如需建立新角色的相關資訊，請參閱《AWS Directory Service 管理指南》中的建立新角色。

從 Lambda 存取 Neptune

1. 請登入 AWS Management Console 並開啟 AWS Lambda 主控台，網址為 https://console.aws.amazon.com/lambda/。
2. 建立適用於 Python 3.6 版的新 Lambda 函數。
3. 將 `AWSLambdaVPCAccessExecutionRole` 角色指派給 Lambda 函數。需要此角色才能存取 Neptune 資源 (僅限 VPC)。
4. 將 Neptune 身分驗證 IAM 角色指派給 Lambda 函數。

   如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 AWS Lambda 許可。
5. 將 IAM 身分驗證 Python 範例複製到 Lambda 函數程式碼。

   如需範例及範本程式碼的詳細資訊，請參閱 the section called “Python 範例”。

為 Neptune IAM 身分驗證設定 Amazon EC2

Amazon EC2 可讓您使用執行個體描述檔來自動提供登入資料。如需詳細資訊，請參閱《IAM 使用者指南》中的使用執行個體設定檔。

首先，您需要新增信任關係，將擔任角色的許可授與 Amazon EC2 服務。

將以下信任關係新增至 Neptune IAM 身分驗證角色。若您沒有 Neptune IAM 身分驗證角色，請參閱 the section called “IAM 政策的類型”。

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

```
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如需將信任關係新增到角色的相關資訊，請參閱《AWS Directory Service 管理指南》中的編輯現有角色的信任關係。

若 Neptune 政策尚未附加到角色，請建立新角色。附加 Neptune IAM 身分驗證政策，然後新增信任政策。如需建立新角色的相關資訊，請參閱《AWS Directory Service 管理指南》中的建立新角色。

使用指令碼取得登入資料

1. 執行以下命令來安裝 jq 命令。指令碼會使用此命令來剖析 curl 命令的輸出。

   ```
   sudo yum -y install jq
   ```

2. 在文字編輯器中建立名為 `credentials.sh` 的檔案，並加入下列文字。將服務區域取代成您自己的值。

   ```
   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. 使用 source 命令在 bash shell 中執行指令碼：

```
source credentials.sh
```

更好的是將此腳本中的命令添加到 EC2 執行個體上的 .bashrc 檔案中，以便在您登入時自動呼叫它們，讓 Gremlin 主控台可使用臨時的登入資料。

4. 使用以下其中一個方法連接。

- the section called “Gremlin 主控台”
- the section called “Gremlin Java”
- the section called “SPARQL Java (RDF4J 和 Jena)”
- the section called “Python 範例”

記錄和監控 Amazon Neptune 資源

Amazon Neptune 支援各種監控效能和用量的方法。

- 叢集狀態 – 檢查 Neptune 叢集的圖形資料庫引擎的運作狀態。如需詳細資訊，請參閱 the section called “執行個體狀態”。
- Amazon CloudWatch — Neptune 自動將指標發送到警報， CloudWatch 並支持 CloudWatch 警報。如需詳細資訊，請參閱 the section called “使用 CloudWatch”。
- 稽核日誌檔 – 使用 Neptune 主控台檢視、下載或監看資料庫日誌檔。如需詳細資訊，請參閱 the section called “稽核日誌搭配 Neptune”。
- 將日誌發佈到 Amazon CloudWatch 日誌 — 您可以設定 Neptune 資料庫叢集，將稽核日誌資料發佈到 Amazon CloudWatch 日誌中的日誌群組。使用 CloudWatch 日誌，您可以對日誌數據進行實時分析，用 CloudWatch 於創建警報和查看指標，以及使用 CloudWatch 日誌將日誌記錄存儲在高度耐用的存儲中。如需詳細資訊，請參閱 Neptune CloudWatch 日誌。
- AWS CloudTrail— Neptune 支持使用 API 日誌記錄 CloudTrail. 如需詳細資訊，請參閱 the section called “使用記錄 Neptune API 呼叫 AWS CloudTrail”。
- 標記 – 使用標籤將中繼資料新增到 Neptune 資源，並根據標籤追蹤使用情形。如需詳細資訊，請參閱 the section called “標記 Neptune 資源”。

Amazon Neptune 的合規驗證

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您因應各種合規性需求，例如 PCI DSS，滿足特定合規性架構所規定的入侵偵測需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon Neptune 中的恢復能力

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

建立 Amazon Neptune 資料庫叢集的 Amazon VPC 必須在至少兩個可用區域內包含至少兩個子網路。將叢集執行個體分散到至少兩個可用區域，Neptune 有助於確保不幸情況下可用區域出錯時您的資料庫叢集內有執行個體可用。Neptune 資料庫叢集的叢集磁碟區一律橫跨三個可用區域，以提供具有高耐用性的儲存，減少資料遺失的可能性。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

將現有圖形遷移至 Amazon Neptune

有許多工具和技術可協助您將現有的圖形資料從另一個資料存放區遷移至 Amazon Neptune。

簡單的遷移工作流程涉及下列步驟：

- 將資料從其現有存放區匯出至 Amazon Simple Storage Service (Amazon S3)。
- 將其清除並格式化以進行匯入。
- 使用 Neptune 大量載入器 將其載入至 Neptune 資料庫叢集。
- 將您的 Gremlin 或 SPARQL 應用程式設定為使用 Neptune 提供的對應端點。

> **Note**
>
> Neptune 叢集必須在應用程式可存取的 VPC 中執行。

有多種方法可以簡化和自動化其中一些步驟，取決於資料的儲存位置：

主題

- 從 Neo4j 遷移到 Amazon Neptune
- 將現有的圖形從 Apache TinkerPop Gremlin 伺服器遷移到 Amazon Neptune
- 將現有圖形從 RDF 三元組存放區遷移到 Amazon Neptune
- 使用 AWS Database Migration Service (AWS DMS) 從關聯式資料庫或 NoSQL 資料庫遷移至 Amazon Neptune
- 從 Blazegraph 遷移到 Amazon Neptune

從 Neo4j 遷移到 Amazon Neptune

Neo4j 和 Amazon Neptune 都是圖形資料庫，專為支援標記屬性圖資料模型的線上交易圖形工作負載而設計。這些相似性使得 Neptune 成為試圖遷移其目前 Neo4j 應用程式之客戶的共同選擇。不過，這些遷移不只是隨即轉移，因為兩個資料庫之間在語言和功能支援、操作特性、伺服器架構和儲存體功能方面存在差異。

此頁面會制定遷移程序，並提出將 Neo4j 圖形應用程式遷移到 Neptune 之前需要考慮的事項。這些考量通常適用於任何 Neo4j 圖形應用程式，無論是由 Community、Enterprise 還是 Aura 提供都一樣。雖然每個解決方案都是獨一無二的，並且可能需要額外的程序，但所有遷移都會遵循相同的一般模式。

以下各節中描述的每個步驟都包含簡化遷移程序的考量和建議。此外，還有開放程式碼工具和描述該程序的部落格文章，以及功能相容性一節，其中具有建議的架構選項。

主題

有關從 Neo4j 遷移到 Neptune 的一般資訊

透過 Neptune 對 OpenCypher 查詢語言的支援，您可以將使用 Bolt 通訊協定或 HTTPS 的大多數 Neo4j 工作負載移至 Neptune。不過，OpenCypher 是一種開放程式碼規範，其中包含其他資料庫 (例如 Neo4j) 所支援功能的大部分但不是全部。

儘管在許多方面相容，Neptune 並不是 Neo4j 的直接替代品。Neptune 是一種全受管圖形資料庫服務，具有高可用性和高持久性等企業功能，在架構方面與 Neo4j 不同。Neptune 是以執行個體為基礎，具有單一主要寫入器執行個體和最多 15 個僅供讀取複本執行個體，可讓您水平擴展讀取容量。使用 Neptune Serverless，您可以自動向上擴展和縮減運算容量，取決於查詢磁碟區。這與 Neptune 儲存體無關，此儲存體會在您新增資料時自動擴展。

Neptune 支援開放原始碼 openCypher 標準規格第 9 版。在 AWS，我們相信開放原始碼對每個人都有好處，因此我們致力於將開放原始碼的價值帶給客戶，並將 AWS 的操作優勢帶給開放原始碼社群。

不過，在 Neo4j 上執行的許多應用程式也會使用不是開放原始碼且 Neptune 不支援的專屬功能。例如，Neptune 不支援 APOC 程序、某些 Cyphers 特定的子句和函數，以及 `Char`、`Date`、或 `Duration` 資料類型。Neptune 會自動將遺失的資料類型轉換為支援的資料類型。

除了 openCypher 之外，Neptune 還支援 Apache TinkerPop Gremlin 查詢語言用於屬性圖 (以及支援 SPARQL 用於 RDF 資料)。Gremlin 可以在同一個屬性圖上與 OpenCypher 交互操作，並且在許多情況下，您可以使用 Gremlin 來提供 OpenCypher 未提供的功能。以下是兩種語言的快速比較：

	openCypher	Gremlin
Style (樣式)	宣告式	命令式

語法	模式比對
	`Match p=(a)-[:route]->(d)`
	`WHERE a.code='ANC'`
	`RETURN p`
		周遊型
		`g.V().has('code', 'ANC').`
		`out('route').path().`
		`by(elementMap())`

易於使用	受 SQL 啟發，非程式設計人員可讀取	更陡峭的學習曲線，類似於 Java 等程式設計語言
彈性	低	高
查詢支援	字串型查詢	用戶端程式庫支援的字串型查詢或內嵌程式碼
用戶端	HTTPS 和 Bolt	HTTPS 和 Websocket

一般來說，不需要變更資料模型，即可從 Neo4j 遷移到 Neptune，因為 Neo4j 和 Neptune 都支援標記的屬性圖 (LPG) 資料。不過，Neptune 有一些架構和資料模型差異，您可以利用這些差異來最佳化效能：例如：

- Neptune ID 被視為頭等公民。
- Neptune 會使用 AWS Identity and Access Management(IAM) 政策，以靈活且精細的方式保護對圖形資料的存取。

- Neptune 會提供幾種方式，來使用 Jupyter 筆記本以執行查詢和視覺化結果。Neptune 也會使用第三方視覺化工具。
- > 雖然 Neptune 沒有 Neo4j 圖形資料科學 (GDS) 程式庫的直接替代品，但是 Neptune 如今會透過各種解決方案支援圖形分析。例如，數個範例筆記本示範如何在 Python 環境內利用 Neptune 與 AWS Pandas SDK 的整合，對圖形資料執行分析。

如果您有問題，請聯絡 AWS 支援或與您的 AWS 客戶團隊聯繫。我們會使用您的意見回饋，來排定符合您需求之新功能的優先順序。

準備從 Neo4j 遷移到 Neptune

遷移的方法

將 Neo4j 應用程式遷移到 Neptune 時，我們建議您使用以下兩種策略之一：平台轉換或重構/重新架構。如需有關遷移策略的詳細資訊，請參閱將應用程式遷移至雲端的 6 個策略，這是 Stephen Orban 所撰寫的部落格文章。

「平台轉換方法」有時稱為「捨棄再購買」(lift-tinker-and-shift)，其中涉及以下步驟：

- 識別您的應用程式想要滿足的使用案例。
- 修改現有的圖形資料模型和應用程式架構，以使用 Neptune 的功能充分處理這些工作負載需求。
- 決定如何將來源應用程式的資料、查詢和其他部分遷移至目標模型和架構。

如果這是一個全新的專案，這種回溯運作方法可讓您將應用程式遷移到您可能設計的 Neptune 解決方案。

相比之下，「重構方法」涉及：

- 識別現有實作的元件，包括基礎設施、資料、查詢和應用程式功能。
- 在 Neptune 中尋找可用來建置類似實作的等效項目。

這種向前運作方法試圖將一個實作交換為另一個實作。

在實務中，您可能會採用這兩種方法的混合。您可以從使用案例開始，設計目標 Neptune 架構，但然後轉至現有的 Neo4j 實作，來識別您必須維護的限制條件和不變量。例如，您可能必須繼續與其他外部系統整合，或繼續為圖形應用程式的消費者提供特定的 API。使用此資訊，您可以判斷已存在哪些資料要移至目標模型，以及哪些資料必須在其他位置取得。

在其他時候，您可能從分析 Neo4j 實作的特定部分開始，作為最佳資訊來源，取得應用程式打算執行之工作的相關資訊。現有應用程式中的這種考古方式可以協助定義一個使用案例，然後您可以使用 Neptune 的功能針對其進行設計。

無論您是使用 Neptune 建置新的應用程式還是從 Neo4j 遷移現有的應用程式，我們建議您透過使用案例的回溯運作來設計資料模型、一組查詢，以及處理您商務需求的應用程式架構。

Neptune 與 Neo4j 之間的架構差異

當客戶第一次考慮將應用程式從 Neo4j 遷移到 Neptune 時，通常很容易根據執行個體大小執行類似的比較。不過，Neo4j 和 Neptune 的架構有著根本上的差異。Neo4j 基於一體化方法，其中資料載入、資料 ETL、應用程式查詢、資料儲存以及管理操作都在同一組運算資源 (例如 EC2 執行個體) 中進行。

相比之下，Neptune 是以 OLTP 為中心的圖形資料庫，其中架構會分隔責任且資源會被分開，以便它們可以動態和獨立擴展。

從 Neo4j 遷移到 Neptune 時，請確定應用程式的資料持久性、可用性和可擴展性需求。Neptune 的叢集架構可簡化需要高持久性、可用性和可擴展性之應用程式的設計。了解 Neptune 的叢集架構後，您就可以設計 Neptune 叢集拓撲來滿足這些需求。

Neo4j 的叢集架構

許多生產應用程式會使用 Neo4j 的因果叢集來提供資料持久性、高可用性和可擴展性。Neo4j 的叢集架構會使用核心伺服器和僅供讀取複本執行個體：

- 核心伺服器透過使用 Rraft 通訊協定複寫資料，提供資料持久性和容錯能力。
- 僅供讀取複本會使用交易日誌傳送，以非同步方式複寫高讀取輸送量工作負載的資料。

叢集中的每個執行個體 (不論是核心伺服器還是僅供讀取複本) 都包含圖形資料的完整副本。

Neptune 的叢集架構

Neptune 叢集是由一個主要寫入器執行個體和最多 15 個僅供讀取複本執行個體組成。叢集中的所有執行個體共用與執行個體分開的相同基礎分散式儲存服務。

- 主要寫入器執行個體會協調資料庫的所有寫入操作，並可垂直擴展，以針對不同的寫入工作負載提供彈性支援。它還支援讀取操作。
- 僅供讀取複本執行個體支援基礎儲存磁碟區的讀取操作，並可讓您水平擴展以支援高讀取工作負載。它們也會做為主要執行個體的容錯移轉目標，以提供高可用性。

> **Note**
>
> 對於繁重的寫入工作負載，最好將僅供讀取複本執行個體擴展至與寫入器執行個體相同的大小，以確保讀取器可以與資料變更保持一致。

- 基礎儲存磁碟區會在資料庫中的資料增加時自動擴展儲存體容量，儲存體最多可為 128 TiB。

執行個體大小是動態且獨立的。每個執行個體都可以在叢集執行時調整大小，而且可以在叢集執行時新增或移除僅供讀取複本。

Neptune Serverless 功能可以隨著需求增加和降低自動擴展您的運算容量。這不僅可以降低您的管理負荷，還可以讓您設定資料庫來處理需求中的大量激增，而不會降低效能或要求您過度佈建。

您最多可以停止 Neptune 叢集 7 天。

Neptune 還支援自動擴展，以根據工作負載自動調整讀取器執行個體大小。

使用 Neptune 的全球資料庫功能，您最多可在 5 個其他區域鏡像叢集。

Neptune 也會特意設計為具備容錯能力：

- 將資料儲存提供給叢集中所有執行個體的叢集磁碟區跨越單一 AWS 區域 中的多個可用區域 (AZ)。每個 AZ 都包含叢集資料的完整副本。
- 如果主要執行個體無法使用，Neptune 會自動容錯移轉到現有的僅供讀取複本，資料不會遺失，通常在 30 秒內完成。如果叢集中沒有現有的僅供讀取複本，Neptune 會自動佈建新的主要執行個體 - 同樣，不會遺失資料。

這一切意味著，從 Neo4j 因果叢集遷移到 Neptune 時，您不必明確地架構叢集拓撲，以實現高資料持久性和高可用性。這使您只需透過幾種方式便可針對預期的讀取和寫入工作負載，以及任何增加的可用性需求調整叢集大小：

- 若要擴展讀取操作，請新增僅供讀取複本執行個體或啟用 Neptune Serverless 功能。
- 若要改善可用性，請將叢集中的主要執行個體和僅供讀取複本分佈在數個可用區域 (AZ) 上。
- 若要縮短任何容錯移轉時間，請至少佈建一個僅供讀取複本執行個體，做為主要執行個體的容錯移轉目標。您可以指派每個複本的優先順序，確定僅供讀取複本執行個體在失敗之後提升為主要執行個體的順序。最佳實務是確保容錯移轉目標具有能夠處理應用程式寫入工作負載的執行個體類別 (如果提升為主要執行個體的話)。

Neptune 與 Neo4j 之間的資料儲存差異

Neptune 會使用以原生四元組模型為基礎的圖形資料模型。將資料遷移至 Neptune 時，資料模型和儲存層的架構中有幾個差異，您應該注意這些差異，以充分利用 Neptune 提供的分散式和可擴展共用儲存體：

- Neptune 不會使用任何明確定義的結構描述或限制條件。它可讓您動態新增節點，邊緣和屬性，而無需提前定義結構描述。Neptune 不會限制所儲存資料的值和類型，但 Neptune 限制中提及的除

外。作為 Neptune 儲存體架構的一部分，資料也會以處理許多最常見存取模式的方式自動編製索引。這種儲存體架構移除了建立和管理資料庫結構描述和索引最佳化的操作負荷。

- Neptune 提供唯一的分散式和共用儲存體架構，此架構會隨著資料庫的儲存體需求成長自動以 10 GB 區塊擴展，最高可達 128 TiB。此儲存層可靠、耐用且容錯，資料複製 6 次，其中 3 個可用區域各複製 2 次，。根據預設，它為所有 Neptune 叢集提供高可用性和容錯資料儲存層。Neptune 的儲存體架構可降低成本，並免除佈建或過度佈建儲存體以處理未來資料成長的需求。

在將您的資料遷移到 Neptune 之前，最好先熟悉 Neptune 的屬性圖資料模型和交易語義。

Neptune 與 Neo4j 之間的操作差異

Neptune 是一個全受管服務，可以自動執行您在使用內部部署或自我管理的資料庫 (例如 Neo4j 企業版或社群版) 時必須執行的許多正常操作任務：

- 自動備份 – Neptune 會自動備份您的叢集磁碟區，並在您指定的保留期 (從 1 到 35 天) 保留備份。這些備份具有連續性和增量性，因此，您可以快速還原到保留期內的任何時間點。寫入備份資料時不會影響資料庫服務的效能或中斷服務。
- 手動快照 – Neptune 可讓您建立資料庫叢集的儲存磁碟區快照，來備份整個資料庫叢集。然後，這種快照可以用來還原資料庫、製作其副本，以及跨帳戶共用。
- 複製 – Neptune 支援複製功能，此功能可讓您快速建立具成本效益的資料庫複製。複製會使用寫入時複製通訊協定，以在建立它們之後只需要最少的額外空間。資料庫複製是試用新 Neptune 功能或升級的有效方法，而不會中斷原始叢集。
- 監控 – Neptune 提供各種方法來監控叢集的效能和使用情況，包括：
  - 執行個體
  - 與 Amazon CloudWatch 和 AWS CloudTrail 的整合
  - 稽核日誌功能
  - 事件通知
  - 標記
- 安全性 – Neptune 預設會提供安全環境。叢集位於私有 VPC 內，此 VPC 會提供與其他資源的網路隔離。所有流量都透過 SSL 加密，而所有資料都使用 AWS KMS 進行靜態加密。

  此外，Neptune 還與 AWS Identity and Access Management (IAM) 整合以提供身分驗證。透過指定 IAM 條件金鑰，您可以使用 IAM 政策，對資料動作提供更精細的存取控制。

Neptune 與 Neo4j 之間的工具和整合差異

Neptune 具有與 Neo4j 不同的整合和工具架構，這可能會影響應用程式的架構。Neptune 會使用叢集的運算資源來處理查詢，但會利用其他一流的 AWS 服務來實現全文檢索搜尋 (使用 OpenSearch)、ETL (使用 Glue) 等功能。如需這些整合的完整清單，請參閱 Neptune 整合。

從 Neo4j 遷移到 Neptune 時佈建基礎設施

Amazon Neptune 叢集是為了在三個維度中進行擴展而建置的：儲存體、寫入容量和讀取容量。以下各節討論遷移時要考慮的特定選項。

佈建儲存體

任何 Neptune 叢集的儲存體都是自動佈建的，您的部分沒有任何管理負荷。隨著叢集的儲存體需求增加，它會以 10 GB 區塊動態調整大小。因此，不需要估計和佈建或過度佈建儲存體來處理未來的資料成長。

佈建寫入容量

Neptune 會提供單一寫入器執行個體，其可以垂直擴展為 Neptune 定價頁面上提供的任何執行個體大小。讀取資料和寫入資料至寫入器執行個體時，所有交易都符合 ACID 規範，而資料隔離如 Neptune 中的交易隔離層級 中所定義。

為寫入器執行個體選擇最佳大小需要執行載入測試，以確定工作負載的最佳執行個體大小。您可以修改資料庫執行個體類別，隨時調整 Neptune 內的任何執行個體大小。您可以根據並行性和平均查詢延遲來估計起始執行個體大小，如下面 佈建叢集時估計最佳的執行個體大小 所述。

佈建讀取容量

建置 Neptune 是為了水平擴展（方法是在叢集內新增最多 15 個僅供讀取複本，或在 Neptune 全球資料庫中新增更多的僅供讀取複本) 僅供讀取複本執行個體，並將其垂直擴展至 Neptune 定價頁面上提供的任何執行個體大小。所有 Neptune 僅供讀取複本執行個體都會使用相同的基礎儲存磁碟區，從而以最小的延遲啟用資料的透明複寫。

除了啟用 Neptune 叢集內讀取請求的水平擴展之外，僅供讀取複本也可做為寫入器執行個體的容錯移轉目標，以啟用高可用性。如需如何確定叢集中僅供讀取複本的適當數量和放置位置的建議，請參閱 Amazon Neptune 基本操作準則。

對於無法預測其連線能力和工作負載的應用程式，Neptune 也支援自動擴展功能，其可根據您指定的準則自動調整 Neptune 複本的數量。

若要確定僅供讀取複本執行個體的最佳大小和數量，需要執行載入測試以確定其必須支援的讀取工作負載特性。您可以修改資料庫執行個體類別，隨時調整 Neptune 內的任何執行個體大小。您可以根據並行性和平均查詢延遲來估計起始執行個體大小，如下節所述。

使用 Neptune Serverless 視需要自動擴展讀取器和寫入器執行個體

雖然能夠估計預期工作負載所需的運算容量通常很有幫助，但您可以設定 Neptune Serverless 功能，以自動擴增和縮減讀取和寫入容量。這可協助您滿足尖峰需求，同時也會在需求減少時自動縮減回來。

佈建叢集時估計最佳的執行個體大小

估計最佳執行個體大小需要知道 Neptune 中的平均查詢延遲、工作負載執行時間，以及正在處理的並行查詢數目。執行個體大小的粗略估計值可以計算為平均查詢延遲乘以並行查詢數目。這會提供您處理工作負載所需之並行執行緒的平均數目。

Neptune 執行個體中的每個 vCPU 都可以支援兩個並行查詢執行緒，因此將執行緒除以 2 可提供所需的 vCPU 數目，然後這些 vCPU 可與 Neptune 定價頁面上的適當執行個體大小相互關聯。例如：

```
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
```

將此數目與執行個體中的 vCPU 數目相互關聯，我們發現我們得到了一個粗略的估計，即 `r5.4xlarge` 將是要為此工作負載嘗試的建議執行個體。這個估計是粗略的，只是為了提供執行個體大小選擇的初始指引。任何應用程式都應該進行調整適當大小的練習，以確定適合工作負載的執行個體數目和類型。

記憶體需求以及處理需求也應納入考慮。當查詢正在存取的資料在主記憶體緩衝集區快取中可用時，Neptune 的效能最佳。佈建足夠的記憶體也可大幅降低 I/O 成本。

您可以在 調整 Neptune 資料庫叢集中資料庫執行個體的大小 頁面上找到有關調整 Neptune 叢集中執行個體大小的其他詳細資訊和指引。

資料從 Neo4j 遷移到 Neptune

從 Neo4j 遷移到 Amazon Neptune 時，遷移資料是過程中的主要步驟。有多種方法遷移資料。正確的方法取決於應用程式的需求、資料大小和所需的遷移類型。不過，其中許多遷移都需要評估相同的考量，其中幾個會在下面反白顯示。

> **Note**
>
> 請參閱 AWS 資料庫部落格中的使用全自動化公用程式將 Neo4j 圖形資料庫遷移至 Neptune，以取得如何執行離線資料遷移之範例的完整逐步解說。

評估從 Neo4j 到 Neptune 的資料遷移

評估任何資料遷移時的第一步是確定如何遷移資料。這些選項取決於要遷移之應用程式的架構、資料大小，以及遷移期間的可用性需求。通常，遷移往往分為兩種類別之一：線上或離線。

離線遷移往往是最簡容易完成的，因為應用程式在遷移期間不接受讀取或寫入流量。在應用程式停止接受流量之後，可以匯出、最佳化、匯入資料，並在重新啟用應用程式之前測試應用程式。

線上遷移較為複雜，因為在遷移資料時，應用程式仍然需要接受讀取和寫入流量。每個線上遷移的確切需求可能會有所不同，但一般架構通常會與下列架構類似：

- 資料庫的持續變更摘要需要在 Neo4j 中啟用，方法是設定 Neo4j 串流作為 Kafka 叢集的來源。
- 一旦完成此操作，就可以遵循 遷移到 Neptune 時從 Neo4j 匯出資料 中的指示，以及稍後與 Kafka 主題相互關聯時提到的時間取得執行中系統的匯出。
- 然後，匯出的資料會遵循 遷移到 Neptune 時從 Neo4j 匯入資料 中的指示匯入至 Neptune。
- 來自 Kafka 串流的變更資料接著可以複製到 Neptune 叢集，方法是使用與從 Amazon Kinesis Data Streams 寫入至 Amazon Neptune 中所述架構類似的架構。請注意，變更複寫可以並行執行，以驗證新的應用程式架構和效能。
- 在驗證資料遷移之後，應用程式流量可被重新導向至 Neptune 叢集，且 Neo4j 執行個體可被解除委任。

從 Neo4j 遷移到 Neptune 的資料模型最佳化

Neptune 和 Neo4j 都支援標示的屬性圖 (LPG)。不過，Neptune 有一些架構和資料模型差異，您可以利用這些差異來最佳化效能：

最佳化節點和邊緣 ID

Neo4j 會自動產生數值長 ID。使用 Cypher，您可以按 ID 引用節點，但通常不鼓勵這樣做，而是傾向於透過編製索引的屬性查閱節點。

Neptune 可讓您為頂點和邊緣提供您自己的字串型 ID。如果您未提供自己的 ID，Neptune 會自動為新邊緣和頂點產生 UUID 的字串表示法。

如果您透過從 Neo4j 匯出資料，然後將其大量匯入至 Neptune，以將資料從 Neo4j 遷移至 Neptune，則您可以保留 Neo4j 的 ID。由 Neo4j 產生的數值可在匯入至 Neptune 時作為使用者提供的 ID，其中它們是以字串表示，而不是數值。

不過，在某些情況下，您可能想要提升頂點屬性以成為頂點 ID。就像使用編製索引的屬性查閱節點是在 Neo4j 中尋找節點的最快方法一樣，按 ID 查閱頂點是在 Neptune 中尋找頂點的最快方法。因此，如果您可以識別包含唯一值的適當頂點屬性，則應考慮在大量載入 CSV 檔案中以指定的屬性值取代頂點 ~id。如果這樣做，您還必須在 CSV 文件中重寫任何對應的 ~from 和 ~to 邊緣值。

將資料從 Neo4j 遷移到 Neptune 時的結構描述限制條件

在 Neptune 內，唯一可用的結構描述限制條件是節點或邊緣 ID 的唯一性。鼓勵需要利用唯一性限制條件的應用程式來查看此方法，透過指定節點或邊緣 ID 來實現唯一性限制條件。如果應用程式使用了多個資料行做為唯一性限制條件，則 ID 可能會設定為這些值的組合。例如，`id=123, code='SEA'` 可以表示為 `ID='123_SEA'`) 來實現複雜的唯一性限制條件。

將資料從 Neo4j 遷移到 Neptune 時的邊緣方向最佳化

當節點、邊緣或屬性新增至 Neptune 時，它們會自動以三種不同的方式編製索引，搭配選用的第四個索引。由於 Neptune 建置和使用索引的方式，因此遵循傳出邊緣的查詢比使用傳入邊緣的查詢更有效率。根據 Neptune 的圖形資料儲存模型，這些是使用 SPOG 索引的主題型搜尋。

如果在將資料模型和查詢遷移至 Neptune 時，您發現最重要的查詢依賴於周遊高度散發的傳入邊緣，您可能想要考慮更改模型，以便這些周遊遵循傳出邊緣，尤其是在您無法指定要周遊的邊緣時。若要這麼做，請反轉相關邊緣的方向，並更新邊緣標籤以反映此方向變更的語義。例如，您可能會變更：

```
person_A – parent_of – person_B
   to:
person_B – child_of – person_A
```

若要在大量載入邊緣 CSV 檔案中進行此變更，只要交換 `~from` 和 `~to` 資料行標頭，然後更新 `~label` 資料行的值即可。

作為反轉邊緣方向的替代方法，您可以啟用第四個 Neptune 索引，即 OSGP 索引，這使得周遊傳入邊緣或物件型搜索更有效。但是，啟用此第四個索引將降低插入率，並且需要更多的儲存體。

將資料從 Neo4j 遷移到 Neptune 時篩選最佳化

Neptune 會進行最佳化，以在將屬性篩選為最具選擇性的可用屬性時發揮最佳效果。使用多個篩選器時，會為每個篩選器找到相符的項目集，然後計算所有相符集的重疊部分。如果可能，將多個屬性合併為單一屬性，這可將索引查詢的次數降至最低，並減少查詢的延遲。

例如，此查詢會使用兩個索引查詢和一個聯結：

```
MATCH (n) WHERE n.first_name='John' AND n.last_name='Doe' RETURN n
```

這個查詢會使用單一索引查詢擷取相同的資訊：

```
MATCH (n) WHERE n.name='John Doe' RETURN n
```

Neptune 支援與 Neo4j 不同的資料類型。

Neo4j 的資料類型對應到 Neptune 支援的資料類型

- 邏輯：`Boolean`

  在 Neptune 中將此項對應到 `Bool` 或 `Boolean`。
- 數值：`Number`

  在 Neptune 中將此項對應到以下 Neptune openCypher 類型中最窄的一個，此類型可以支援有問題的數值屬性的所有值：

  ```
  Byte
  Short
  Integer
  Long
  Float
  Double
  ```
- 文字：`String`

  在 Neptune 中將此項對應到 `String`。

- 時間點：

```
Date
Time
LocalTime
DateTime
LocalDateTime
```

  使用 Neptune 支援的下列其中一種 ISO-8601 格式，在 Neptune 中將這些項目 `Date` 對應為 UTC：

```
yyyy-MM-dd
yyyy-MM-ddTHH:mm
yyyy-MM-ddTHH:mm:ss
yyyy-MM-ddTHH:mm:ssZ
```

- 持續時間：`Duration`

  如有必要，請在 Neptune 中將此項對應到日期算術的數值。

- 空間：`Point`

  在 Neptune 中將此項對應到元件數值，每個數值接著都會變成個別的屬性，或者表示為要由用戶端應用程式解譯的字串值。請注意，Neptune 使用 OpenSearch 進行全文檢索搜尋整合時，可讓您編製地理位置屬性的索引。

將多值屬性從 Neo4j 遷移到 Neptune

Neo4j 允許簡單類型的異質清單同時儲存為節點和邊緣的屬性。這些清單可以包含重複值。

不過，在屬性資料圖中，Neptune 對於頂點屬性只允許集合或單一基數，而對於邊緣屬性，則只允許單一基數。因此，無法將包含重複值的 Neo4j 節點清單屬性直接遷移到 Neptune 頂點屬性，也無法將 Neo4j 關係清單屬性直接遷移到 Neptune 邊緣屬性。

將具有重複值的 Neo4j 多值節點屬性遷移到 Neptune 的一些可能策略如下：

- 捨棄重複值並將多值 Neo4j 節點屬性轉換為集合基數 Neptune 頂點屬性。請注意，Neptune 集合可能不會接著反映原始 Neo4j 多值屬性中項目的順序。
- 將多值 Neo4j 節點屬性轉換為 Neptune 頂點字串屬性中 JSON 格式清單的字串表示法。
- 將每個多值屬性值擷取至具有值屬性的個別頂點，然後使用以屬性名稱標記的邊緣將這些頂點連線到父頂點。

同樣地，將 Neo4j 多值關係屬性遷移到 Neptune 的可能策略如下：

- 將多值 Neo4j 關係屬性轉換為 JSON 格式清單的字串表示法，並將其儲存為 Neptune 邊緣字串屬性。
- 將 Neo4j 關係重構為附加到中間頂點的傳入和傳出 Neptune 邊緣。將每個多值關係屬性值擷取至具有值屬性的個別頂點，然後使用以屬性名稱標記的邊緣將這些頂點連線到此中繼頂點。

請注意，JSON 格式清單的字串表示法對 OpenCypher 查詢語言來說是不透明的，雖然 OpenCypher 包含允許在字串值內進行簡單搜尋的 `CONTAINS` 述詞。

遷移到 Neptune 時從 Neo4j 匯出資料

從 Neo4j 匯出資料時，使用 APOC 程序匯出到 CSV 或 GraphML。雖然可以匯出到其他格式，但有開放原始碼工具可將從 Neo4j 匯出的 CSV 資料轉換為 Neptune 大量載入格式，並且也有開放原始碼工具可將從 Neo4j 匯出的 GraphML 資料轉換為 Neptune 大量載入格式。

您也可以使用各種 APOC 程序將資料直接匯出到 Amazon S3。匯出至 Amazon S3 儲存貯體預設為停用，但可以使用 Neo4j APOC 文件中匯出至 Amazon S3 中反白的程序來啟用此儲存貯體。

遷移到 Neptune 時從 Neo4j 匯入資料

您可以使用 Neptune 大量載入器，或在支援的查詢語言 (例如 openCypher) 中使用應用程式邏輯，將資料匯入至 Neptune。

Neptune 大量載入器是匯入大量資料的偏好方法，因為如果您遵循最佳實務，它可提供最佳化的匯入效能。大量載入器支援兩種不同的 CSV 格式，您可以使用上面 匯出資料 一節中提到的開放原始碼公用程式，將從 Neo4j 匯出的資料轉換為這些格式。

您還可以使用 OpenCypher，搭配用於剖析、轉換和匯入的自訂邏輯來匯入資料。您可以透過 HTTPS 端點 (建議使用) 或使用 Bolt 驅動程式來提交 OpenCypher 查詢。

應用程式從 Neo4j 遷移到 Neptune

在您已將資料從 Neo4j 遷移到 Neptune 之後，下一步是遷移應用程式本身。與資料一樣，根據您使用的工具、需求、架構差異等，有多種遷移應用程式的方法。下面概述了您通常需要在此過程中考慮的事項。

從 Neo4j 移到 Neptune 時遷移連線

如果您目前未使用 Bolt 驅動程式，或想要使用替代驅動程式，您可以連線至 HTTPS 端點，該端點可提供對所傳回資料的完整存取權。

如果您確實具有使用 Bolt 通訊協定的應用程式，則可以將這些連線遷移到 Neptune，並讓您的應用程式可以使用與您在 Neo4j 中使用的同一驅動程式進行連線。若要連線到 Neptune，您可能需要對應用程式進行下列一或多個變更：

- 必須更新 URL 和連接埠，才能使用叢集端點和叢集連接埠 (預設值為 8182)。
- Neptune 會要求所有連線都使用 SSL，因此您需要為每個連線指定加密。
- Neptune 透過指派 IAM 政策和角色來管理身分驗證。IAM 政策和角色在應用程式內提供極為靈活的使用者管理層級，因此在設定叢集之前，務必先閱讀並了解 IAM 概觀中的資訊。
- Bolt 連線在 Neptune 中的行為與在 Neo4j 中的行為會在幾個方面有所不同，如 Neptune 中的 Bolt 連線行為 中所述。
- 您可以在 使用 openCypher 和 Bolt 的 Neptune 最佳實務 中找到更多的資訊和建議。

在 使用 Bolt 通訊協定，對 Neptune 進行 openCypher 查詢 中，有一些常用語言 (例如 Java、Python、.NET 和 NodeJS) 的程式碼範例，以及連線案例 (例如使用 IAM 身分驗證) 的程式碼範例。

從 Neo4j 移到 Neptune 時將查詢路由到叢集執行個體

Neo4j 的用戶端應用程式會使用路由驅動程式，並指定存取模式，以將讀取和寫入請求路由至因果叢集中適當的伺服器。

將用戶端應用程式遷移至 Neptune 時，請使用 Neptune 端點將查詢有效率地路由至叢集中適當的執行個體：

- Neptune 的所有連線都應在 URL 中使用 `bolt://`，而不是 `bolt+routing://` 或 `neo4j://`。
- 叢集端點會連線至叢集中目前的主要執行個體。使用叢集端點將寫入請求路由至主要執行個體。

- 讀取器端點會將連線分發到叢集中的僅供讀取複本執行個體。如果您具有的單一執行個體叢集沒有僅供讀取複本執行個體，則讀取器端點會連線到支援寫入操作的主要執行個體。如果叢集確實包含一或多個僅供讀取複本執行個體，則將寫入請求傳送至讀取器端點會產生例外狀況。
- 叢集中的每個執行個體也可以具有自己的執行個體端點。如果您的用戶端應用程式需要將請求傳送至叢集中的特定執行個體，請使用執行個體端點。

如需更多詳細資訊，請參閱 Neptune 端點考量。

Neptune 中的資料一致性

使用 Neo4j 因果叢集時，僅供讀取複本最終會與核心伺服器一致，但用戶端應用程式可以使用因果鏈來確保因果一致性。因果鏈需要在交易之間傳遞書籤，這允許用戶端應用程式寫入核心伺服器，然後從僅供讀取複本讀取自己的寫入。

在 Neptune 中，僅供讀取複本執行個體最終會與寫入器一致，複本延遲通常少於 100 毫秒。不過，在複寫變更之前，複本執行個體上看不到對現有邊緣和頂點的更新，以及新增的邊緣和頂點。因此，如果您的應用程式需要透過讀取每個寫入以在 Neptune 上立即保持一致性，請使用叢集端點進行先讀後寫操作。這是使用叢集端點進行讀取操作的唯一時機。在所有其他情況下，請使用讀取器端點進行讀取。

將查詢從 Neo4j 遷移到 Neptune

儘管 Neptune 對 OpenCypher 的支援大幅地減少了從 Neo4j 遷移查詢所需的工作量，但仍有一些在遷移時要評估的差異：

- 如上面 資料模型最佳化 中所討論，您可能需要對資料模型進行修改，以便為 Neptune 建立最佳化的圖形資料模型，因而需要對您的查詢和測試進行變更。
- Neo4j 提供了各種 Cypher 特定的語言延伸模組，這些延伸模組不包括在由 Neptune 實作的 OpenCypher 規格中。根據使用案例和使用的功能，OpenCypher 語言內可能有因應措施，或使用 Gremlin 語言，或透過其他機制 (如 重寫 Cypher 查詢以在 Neptune 上的 OpenCpher 中執行 中所述)。
- 應用程式通常會使用其他中介軟體元件與資料庫互動，而不是 Bolt 驅動程式本身。請檢查 Neptune 與 Neo4j 的相容性 以查看是否支援您正在使用的工具或中介軟體。
- 在容錯移轉的情況下，Bolt 驅動程式可能會繼續連線到先前的寫入器或讀取器執行個體，因為提供給連線的叢集端點已解析為 IP 地址。您應用程式中的適當錯誤處理應該處理這個問題，如 在容錯移轉之後建立新連線 中所述。

- 交易由於無法解決的衝突或鎖定等待逾時而取消時，Neptune 會以 `ConcurrentModificationException` 回應。如需更多詳細資訊，請參閱 引擎錯誤代碼。根據最佳實務，用戶端應一律截獲和處理這些例外狀況。

  當多個執行緒或多個應用程式同時寫入系統時，偶爾會發生 `ConcurrentModificationException`。由於交易隔離層級，因此這些衝突有時可能是不可避免的。
- Neptune 支援在相同的資料上執行 Gremlin 和 OpenCypher 查詢。這表示在某些情況下，您可能需要考慮使用查詢功能更強大的 Gemlin，來執行查詢的某些功能。

如上面 佈建基礎設施 所討論，每個應用程式都應該經過調整適當大小的練習，以確保執行個體數量、執行個體大小和叢集拓撲都針對應用程式的特定工作負載進行最佳化。

這裡針對遷移應用程式討論的考量是最常見的，但這不是一個詳盡的清單。每個應用程式都是唯一的。如果您還有其他問題，請聯絡 AWS 支援或與您的客戶團隊聯繫。

遷移 Neo4j 特定功能和工具

Neo4j 具有各種自訂功能和附加元件，其中具有您的應用程式可能依賴的功能。在評估是否需要遷移此功能時，調查 AWS 內是否有更好的方法來實現相同的目標，通常很有幫助。考慮到 Neo4j 與 Neptune 之間的架構差異，您通常可以找到利用其他 AWS 服務或整合的有效替代方案。

如需 Neo4J 特定功能和所建議因應措施的清單，請參閱 Neptune 與 Neo4j 的相容性。

Neptune 與 Neo4j 的相容性

Neo4j 具有一體化架構方法，其中資料載入、資料 ETL、應用程式查詢、資料儲存以及管理操作都在同一組運算資源 (例如 EC2 執行個體) 中進行。Amazon Neptune 是以 OLTP 為中心的開放規格圖形資料庫，其中架構會分隔操作並分離資源，以便它們可以動態擴展。

Neo4j 中有各種功能和工具，包括第三方工具，這些工具不是 OpenCypher 規格的一部分、與 OpenCypher 不相容，或與 Neptune 的 OpenCypher 實作不相容。下方列出一些最常見的功能和工具。

Neptune 中不存在的 NEO4J 特定功能

- **`LOAD CSV`** – Neptune 具有與 Neo4j 不同的架構方法來載入資料。為了允許更好的擴展和成本最佳化，Neptune 實作了涉及資源的分隔，並建議您使用其中一個 AWS 服務整合 (例如 AWS Glue)，來執行必要的 ETL 程序，以 Neptune 大量載入器所支援的格式準備資料。

  另一個選項是使用在 AWS 運算資源 (例如 Amazon EC2 執行個體、Lambda 函數、Amazon Elastic Container Service、AWS Batch 工作等) 上執行的應用程式程式碼，執行相同的操作。此程式碼可以使用 Neptune 的 HTTPS 端點或 Bolt 端點。
- 精細的存取控制 – Neptune 支援使用 IAM 條件金鑰，對資料存取動作進行精細的存取控制。可以在應用程式層實現其他精細的存取控制。
- Neo4j Fabric – Neptune 確實支援使用 SPARQL `SERVICE` 關鍵字，對 RDF 工作負載進行跨資料庫的查詢聯合。由於目前沒有針對屬性圖工作負載的查詢聯合開放標準或規格，因此需要在應用程式層實作該功能。
- 角色型存取控制 (RBAC) – Neptune 透過指派 IAM 政策和角色來管理身分驗證。IAM 政策和角色在應用程式內提供極為靈活的使用者管理層級，因此值得在設定叢集之前，先閱讀並了解 IAM 概觀中的資訊。

- 書籤 – Neptune 叢集由單一寫入器執行個體和最多 15 個僅供讀取複本執行個體組成。寫入至寫入器執行個體的資料符合 ACID 規範，並會在後續讀取時提供強大的一致性保證。僅供讀取複本會使用與寫入器執行個體相同的儲存磁碟區，並且最終保持一致，通常在寫入資料後不到 100 毫秒內。如果您的使用案例立即需要保證新寫入的讀取一致性，則這些讀取應導向至叢集端點，而不是讀取器端點。
- APOC 程序 – 由於 APOC 程序不包含在 OpenCypher 規格中，因此 Neptune 不提供外部程序的直接支援。相反，Neptune 依賴與其他 AWS 服務的整合，以可擴展、安全且穩健的方式實現類似的最終使用者功能。有時候 APOC 程序可以在 OpenCypher 或 Gremlin 中重寫，有些與 Neptune 應用程式無關。

一般而言，APOC 程序分為以下類別：

- 匯入 – Neptune 支援使用查詢語言、Neptune 大量載入器或作為 AWS Database Migration Service 的目標，搭配各種格式匯入資料。您可以使用 AWS Glue 和 `neptune-python-utils` 開放原始碼套件，對資料執行 ETL 操作。
- 匯出 – Neptune 支援使用 `neptune-export` 公用程式匯出資料，該公用程式支援各種常見的匯出格式和方法。
- 資料庫整合 – Neptune 支援使用 ETL 工具 (例如 AWS Glue) 或遷移工具 (例如 AWS Database Migration Service) 與其他資料庫整合。
- 圖形更新 – Neptune 透過對 OpenCypher 和 Gremlin 查詢語言的支援，支援一組豐富的功能更新屬性圖資料。如需重寫常用程序的範例，請參閱 Cypher 重寫。
- 資料結構 – Neptune 透過對 OpenCypher 和 Gremlin 查詢語言的支援，支援一組豐富的功能更新屬性圖資料。如需重寫常用程序的範例，請參閱 Cypher 重寫。
- 暫時 (日期時間) – Neptune 透過對 OpenCypher 和 Gremlin 查詢語言的支援，支援一組豐富的功能更新屬性圖資料。如需重寫常用程序的範例，請參閱 Cypher 重寫。
- 數學 – Neptune 透過對 OpenCypher 和 Gremlin 查詢語言的支援，支援一組豐富的功能更新屬性圖資料。如需重寫常用程序的範例，請參閱 Cypher 重寫。
- 進階圖形查詢 – Neptune 透過對 OpenCypher 和 Gremlin 查詢語言的支援，支援一組豐富的功能更新屬性圖資料。如需重寫常用程序的範例，請參閱 Cypher 重寫。
- 比較圖形 – Neptune 透過對 OpenCypher 和 Gremlin 查詢語言的支援，支援一組豐富的功能更新屬性圖資料。如需重寫常用程序的範例，請參閱 Cypher 重寫。
- Cypher 執行 – Neptune 透過對 OpenCypher 和 Gremlin 查詢語言的支援，支援一組豐富的功能更新屬性圖資料。如需重寫常用程序的範例，請參閱 Cypher 重寫。

- 自訂程序 – Neptune 不支援使用者建立的自訂程序。此功能必須在應用程式層實作。
- 地理空間 – 雖然 Neptune 不為地理空間功能提供原生支援，但可以透過與其他 AWS 服務整合來實現類似的功能，如這篇部落格文章所示：合併 Amazon Neptune 和 Amazon OpenSearch Service 進行地理空間查詢，作者為 Ross Gabay 和 Abhilash Vinod (2022 年 2 月 1 日)。
- 圖形資料科學 — 現今 Neptune 透過 Neptune Analytics 支援圖形分析，這是一種支援圖形分析演算法程式庫的記憶體最佳化引擎。

  Neptune 也提供了與 AWSPandas SDK 的整合以及幾個範例筆記本，這些筆記本範例展示如何在 Python 環境內利用此整合，對圖形資料進行分析。

- 結構描述限制條件 – 在 Neptune 內，唯一可用的結構描述限制條件是節點或邊緣 ID 的唯一性。沒有功能可以指定任何其他結構描述限制條件，或指定圖形中元素上的任何其他唯一性或值限制條件。Neptune 中的 ID 值是字串，而且可以使用 Grimlin 進行設定，如下所示：

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

  鼓勵需要利用 ID 做為唯一性限制條件的應用程式來試用此方法，實現唯一性限制條件。如果應用程式使用了多個資料行做為唯一性限制條件，則 ID 可能會設定為這些值的組合。例如，`id=123, code='SEA'` 可以表示為 `ID='123_SEA'`，以實現複雜的唯一性限制條件。

- 多重租用 – Neptune 僅支援每個叢集單一圖形。若要使用 Neptune 建置多租用戶系統，請使用多個叢集，或在單一圖形內以邏輯方式分割租用戶，並使用應用程式端邏輯強制執行分隔。例如，新增屬性 `tenantId` 並將其包含在每個查詢中，如下所示：

  ```
  MATCH p=(n {tenantId:1})-[]->({tenantId:1}) RETURN p LIMIT 5)
  ```

  Neptune Serverless 可讓使用多個資料庫叢集實作多重租用相對容易，每個資料庫叢集都會視需要獨立且自動擴展。

Neptune 對 Neo4j 工具的支援

Neptune 提供了 Neo4j 工具的下列替代方案：

- Neo4j Browser – Neptune 提供開放原始碼圖形筆記本，其中提供以開發人員為中心的 IDE 來執行查詢和視覺化結果。
- Neo4j Bloom – Neptune 支援使用第三方視覺化解決方案 (例如 Graph-explorer、Tom Sawyer、Cambridge Intelligence、Graphistry、metaphacts 和 G.V()) 進行豐富的圖形視覺化。
- GraphQL – Neptune 目前透過自訂 AWS AppSync 整合支援 GraphQL。請參閱使用 Amazon Neptune 和 AWS Amplify 建置圖形應用程式部落格文章，以及使用 AWS AppSync 和 Amazon Neptune 建置 Serverless Calorie 追蹤器應用程式範例專案。
- NeoSemantics – Neptune 原本支援 RDF 資料模型，因此建議想要執行 RDF 工作負載的客戶使用 Neptune 的 RDF 模型支援。
- Arrows.app – 使用匯出命令匯出模型時建立的 Cypher 與 Neptune 相容。
- Linkurious Ogma – 這裡提供與 Linkurious Ogma 的範例整合。
- Spring Data Neo4j – 這目前與 Neptune 不相容。

- Neo4j Spark Connector – Neo4j Spark Connector 可在 Spark 工作內用來使用 OpenCypher 連線到 Neptune。以下是一些範例程式碼和應用程式組態：

  範本程式碼：

```
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();
```

  應用程式組態

```
<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>
```

這裡未列出的 Neo4j 功能和工具

如果您使用未在這裡列出的工具或功能，我們不確定其與 AWS 內 Neptune 或其他服務的相容性。如果您還有其他問題，請聯絡 AWS 支援或與您的客戶團隊聯繫。

重寫 Cypher 查詢以在 Neptune 上的 OpenCpher 中執行

OpenCypher 語言是屬性圖的宣告式查詢語言，最初由 Neo4j 開發，然後在 2015 年成為開放原始碼，並在 Apache 2 開放原始碼授權下投入 OpenCypher 專案。在 AWS，我們相信開放原始碼對每個人都有好處，因此我們致力於將開放原始碼的價值帶給客戶，並將 AWS 的操作優勢帶給開放原始碼社群。

OpenCypher 語法記載於 Cypher 查詢語言參考第 9 版。

由於 OpenCypher 包含 Cypher 查詢語言的語法和功能子集，因此某些遷移案例需要以 OpenCypherr 相容的格式重寫查詢，或檢查替代方法以實現所需的功能。

本節包含處理常見差異的建議，但它們絕不是全面的。您應該使用這些重寫徹底測試任何應用程式，以確保結果就是您預期的。

重寫 `None`、`All` 和 `Any` 述詞函數

這些函數不是 openCypher 規格的一部分。可比較的結果可以在 OpenCypher 中使用清單理解來實現。

例如，尋找從節點 `Start` 到節點 `End` 的所有路徑，但不允許任何行程通過類別屬性為 `D` 的節點：

```
# 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
```

清單理解可以實現這些結果，如下所示：

```
all  => size(list_comprehension(list)) = size(list)
any  => size(list_comprehension(list)) >= 1
none => size(list_comprehension(list)) = 0
```

在 openCypher 中重寫 Cypher `reduce()` 函數

`reduce()` 函數不是 openCypher 規格的一部分。它通常用來從清單內的元素建立資料的彙總。在許多情況下，您可以使用清單理解和 `UNWIND` 子句的組合來實現類似的結果。

例如，在安克雷奇 (ANC) 與奧斯汀 (AUS) 之間具有一到三個停靠站的路徑上，下列 Cypher 查詢會找出所有位於這些路徑上的機場，並傳回每條路徑的總距離：

```
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
```

您可以在 OpenCypher 為 Neptune 編寫相同的查詢，如下所示：

```
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
```

在 openCypher 中重寫 Cypher FOREACH 子句

FOREACH 子句不是 openCypher 規格的一部分。它通常用來在查詢過程中更新資料，通常是從路徑內的彙總或元素中更新資料。

作為一個路徑範例，在安克雷奇 (ANC) 與奧斯汀 (AUS) 之間不超過兩個停靠站的路徑上，找出所有位於該路徑的機場，並對每個機場設定已訪問屬性：

```
# 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
```

另一個範例是：

```
# 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
```

在 Neptune 中重寫 NEO4j APOC 程序

下面範例會使用 openCypher 來取代一些最常用的 APOC 程序。這些範例僅供參考，旨在提供一些有關如何處理常見案例的建議。實際上，每個應用程序都有所不同，您必須制定自己的策略來提供您需要的所有功能。

重寫 **apoc.export** 程序

Neptune 會使用 neptune-export 公用程式，為各種輸出格式 (例如 CSV 和 JSON) 的完整圖形和查詢型匯出提供一系列選項 (請參閱 從 Neptune 資料庫叢集匯出資料)。

重寫 **apoc.schema** 程序

Neptune 沒有明確定義的結構描述、索引或限制條件，因此不再需要許多 apoc.schema 程序。範例如下：

- 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

Neptune OpenCypher 確實支援擷取的值與程序擷取的值類似，如下所示，但可能會在較大的圖形上遇到效能問題，因為這樣做需要掃描圖形的大部分才能返回答案。

```
# 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)
```

apoc.do 程序的替代方案

這些程序是用來提供條件式查詢執行，很容易使用其他 OpenCypher 子句來實作。在 Neptune 中，至少有兩種方法可以實現類似的行為：

- 一種方法是將 OpenCypher 的清單理解功能與 UNWIND 子句結合。
- 另一種方法是在 Gremlin 中使用 choose() 和 coalesce() 步驟。

這些方法的範例如下所示。

apoc.do.when 的替代方案

```
# 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())
```

apoc.do.case 的替代方案

```
# 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())
```

清單型屬性的替代方案

Neptune 目前不支援儲存清單型屬性。不過，您可以將清單值儲存為逗號分隔字串，然後使用 `join()` 和 `split()` 函數來建構和解構清單屬性，以取得類似的結果。

例如，如果想要將標籤清單儲存為屬性，則可以使用重寫範例，其中展示如何擷取逗號分隔屬性，然後使用 `split()` 和 `join()` 函數搭配清單理解來實現可比較的結果：

```
# 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
```

從 Neo4j 遷移到 Neptune 的資源

Neptune 提供數個可協助遷移程序的工具和資源。

協助從 Neo4j 遷移到 Neptune 的工具

- openCypher CheatSheet。
- neo4j-to-neptune – 用於將資料從 Neo4j 遷移到 Neptune 的命令列公用程式。
- fully-automated-neo4j-to-neptune – 一種 AWS CDK 應用程式，其會為您展示如何將簡單的 Neo4j 資料庫遷移到 Amazon Neptune。
- csv-to-neptune-bulk-format – 此工具會採取組態型方法，將一個或多個 CSV 文件重新格式化為支援的 Neptune 大量載入格式。

部落格文章

- 使用 Amazon Managed Streaming for Apache Kafka 將資料擷取從 Neo4j 變更為 Amazon Neptune，作者為 Sanjeet Sahay (2020 年 6 月 22 日)
- 使用全自動公用程式將 Neo4j 圖形資料庫遷移到 Amazon Neptune，作者為 Sanjeet Sahay (2020 年 4 月 13 日)。

將現有的圖形從 Apache TinkerPop Gremlin 伺服器遷移到 Amazon Neptune

如果您在 Apache TinkerPop Girmlin 伺服器中具有您想要遷移到 Amazon Neptune 的圖形資料，您將會採取以下步驟：

1. 將資料從 Gremlin 伺服器匯出至 Amazon Simple Storage Service (Amazon S3)。
2. 將匯出的資料轉換為 Neptune 大量載入器可以匯入的 CSV 格式。
3. 使用 Neptune 大量載入器，將資料匯入至您已準備好的 Neptune 資料庫叢集。
4. 修改您現有的應用程式以連線至 Neptune 的 Gemlin 端點，並進行必要的變更以符合 Neptune Gremlin 實作差異。

將現有圖形從 RDF 三元組存放區遷移到 Amazon Neptune

如果您在 RDF/SPARQL 中具有要遷移至 Amazon Neptune 的圖形資料，則將會採取以下步驟：

1. 從您的 RDF 三元組存放區匯出資料。
2. 將匯出的資料轉換為 Neptune 大量載入器可以匯入的格式。
3. 將要匯入的資料儲存在 Amazon Simple Storage Service (Amazon S3) 中。
4. 使用 Neptune 大量載入器，將資料從 Amazon S3 匯入至您已準備好的 Neptune 資料庫叢集。
5. 修改現有的應用程式以連線到 Neptune 的 SPARQL 端點。

如果您想要嘗試將屬性圖 CSV 資料遷移至 RDF，則可以使用 Amazon Neptune CSV 至 RDF 轉換器。

使用 AWS Database Migration Service (AWS DMS) 從關聯式資料庫或 NoSQL 資料庫遷移至 Amazon Neptune

AWS Database Migration Service (AWS DMS) 是一項雲端服務，可讓您輕鬆遷移關聯式資料庫、NoSQL 資料庫、資料倉儲和其他類型的資料存放區。如果您將圖形資料儲存在其中一個 AWS DMS 支援的關聯式資料庫或 NoSQL 資料庫中，AWS DMS 可以協助您快速安全地遷移至 Neptune，而不需要從目前的資料庫停機。如需詳細資訊，請參閱 用 AWS Database Migration Service 於從不同的資料存放區將資料載入 Amazon Neptune。

使用 AWS DMS 的遷移資料流程如下：

- 建立 AWS DMS 資料表對應物件。此 JSON 物件指定應依何種順序從來源資料庫讀取哪些資料表，以及如何命名資料行。它也可以篩選所複製的資料列，並提供簡單的值轉換，如轉換成小寫或四捨五入。
- 建立一個 Neptune `GraphMappingConfig`，以指定擷取自來源資料庫的資料應如何載入至 Neptune。
  - 對於 RDF 資料 (使用 SPARQL 查詢得來)，`GraphMappingConfig` 是採用 W3 的標準 R2RML 對應語言編寫。
  - 對於屬性圖形資料 (使用 Gremlin 查詢得來)，`GraphMappingConfig` 是 JSON 物件，如 GraphMappingConfig 屬性圖形/格林資料的佈局 中所述。
- 在與 Neptune 資料庫叢集相同的 VPC 中建立 AWS DMS 複寫執行個體，以執行遷移。
- 建立 Amazon S3 儲存貯體，用作暫存要遷移之資料的中介儲存體。
- 執行 AWS DMS 遷移任務。

如需詳細資訊，請參閱 用 AWS Database Migration Service 於從不同的資料存放區將資料載入 Amazon Neptune，且亦請參閱 Chris Smith 的四段部落格文章：「使用 AWS Database Migration Service (DMS) 從關聯式資料庫將您的圖形填入 Amazon Neptune 中：」

- 第 1 部分：設定階段
- 第 2 部分：設計屬性圖模型
- 第 3 部分：設計 RDF 模型
- 第 4 部分：將它們放在一起

從 Blazegraph 遷移到 Amazon Neptune

如果您在開放原始碼 Blazegraph RDF 三元組存放區中具有圖形，則可以使用下列步驟，將圖形資料遷移到 Amazon Neptune：

- 佈建 AWS 基礎設施。首先，使用 AWS CloudFormation 範本佈建所需的 Neptune 基礎設施 (請參閱建立資料庫叢集)。
- 從 Blazegraph 匯出資料。從 Blazegraph 匯出資料有兩種主要方法，即使用 SPARQL CONSTRUCT 查詢或使用 Blazegraph 匯出公用程式。
- 將資料匯入至 Neptune。然後，您可以使用 Neptune 工作台和 Neptune 大量載入器，將匯出的資料檔案載入至 Neptune。

這種方法通常也適用於從其他 RDF 三元組存放區資料庫遷移。

Blazegraph 與 Neptune 相容性

在將圖形資料遷移到 Neptune 之前，Blazegraph 與 Neptune 之間有數個您應該注意的顯著差異。這些差異可能需要變更查詢、應用程式架構或兩者，甚至使遷移變得不切實際：

- **`Full-text search`** – 在 Blazegraph 中，您可以透過與 Apache Solr 的整合來使用內部全文檢索搜尋或外部全文檢索搜尋功能。如果您使用其中任一功能，請隨時了解 Neptune 支援的全文檢索搜尋功能的最新更新。請參閱 Neptune 全文檢索搜尋。
- **`Query hints`** – Blazegraph 和 Neptune 都會使用查詢提示的概念來延伸 SPARQL。在遷移期間，您需要遷移您使用的任何查詢提示。如需 Neptune 支援的最新查詢提示的相關資訊，請參閱 SPARQL 查詢提示。
- 推論 – Blazegraph 支援在三元組模式下當作可設定選項推論，但在四元組模式下不支援。Neptune 尚未支援推論。
- 地理空間搜索 – Blazegraph 支援啟用地理空間支援之命名空間的組態。此功能尚未在 Neptune 中提供。
- 多重租用 – Blazegraph 支援單一資料庫內的多重租用。在 Neptune 中，支援多重租用的方式為將資料儲存在具名圖形中，並針對 SPARQL 查詢使用 USING NAMED 子句，或為每個租用戶建立個別的資料庫叢集。
- 聯合 – Neptune 目前支援 SPARQL 1.1 聯合至可供 Neptune 執行個體存取的位置，例如在私有 VPC 內、跨 VPC 或外部網際網路端點。根據特定設定和必要的聯合端點，您可能需要一些額外的網路組態。

- Blazegraph 標準延伸模組 — Blazegraph 包含 SPARQL 和 REST API 標準的多個延伸模組，而 Neptune 僅與標準規格本身相容。這可能需要變更您的應用程式，否則會使遷移變得困難。

為 Neptune 佈建 AWS 基礎設施

雖然您可以透過 AWS Management Console 或 AWS CLI 手動建構必要的 AWS 基礎設施，但改用 CloudFormation 範本通常更為便利，如下所述：

使用 CloudFormation 範本佈建 Neptune：

1. 導覽至 使用 AWS CloudFormation 堆疊建立 Neptune 資料庫叢集。
2. 在您偏好的區域中選擇啟動堆疊。
3. 設定必要的參數 (堆疊名稱和 `EC2SSHKeyPairName`)。也會設定下列選用參數，讓遷移程序可以輕鬆地執行：
   - 將 `AttachBulkloadIAMRoleToNeptuneCluster` 設定為 true。此參數允許建立適當的 IAM 角色並將其附加到您的叢集，以允許大量載入資料。
   - 將 `NotebookInstanceType` 設定為您偏好的執行個體類型。此參數會建立 Neptune 工作簿，您會用來將大量載入執行至 Neptune 並驗證遷移。
4. 選擇 Next (下一步)。
5. 設定您想要的任何其他堆疊選項。
6. 選擇 Next (下一步)。
7. 檢閱您的選項並選取這兩個核取方塊，以確認 AWS CloudFormation 可能需要其他功能。
8. 選擇 Create Stack (建立堆疊)。

堆疊建立程序需要幾分鐘的時間。

從 Blazegraph 匯出資料

下一步是以與 Neptune 大量載入器相容的格式從 Blazegraph 中匯出資料。

根據資料在 Blazegraph 中的儲存方式 (三元組或四元組) 以及使用中的命名圖形數量，Blazegraph 可能會要求您多次執行匯出程序並產生多個資料檔案：

- 如果資料儲存為三元組，則您需要針對每個具名圖形執行一次匯出。
- 如果資料儲存為四元組，您可以選擇以 N-Quads 格式匯出資料，或以三元組格式匯出每個具名圖形。

下面我們假設您將單一命名空間匯出為 N-Quads，但是您可以針對其他命名空間或所需的匯出格式重複此程序。

如果您需要 Blazegraph 在遷移期間上線且可用，請使用 SPARQL CONSTRUCT 查詢。這需要您安裝、設定和執行 Blazegraph 執行個體，具中具有可存取的 SPARQL 端點。

如果您不需要 Blazegraph 上線，請使用 BlazeGraph 匯出公用程式。若要這樣做，您必須下載 Blazegraph，並且資料檔案和組態檔案必須是可存取的，但伺服器不需要執行中。

使用 SPARQL CONSTRUCT 從 Blazegraph 匯出資料

SPARQL CONSTRUCT 是 SPARQL 的一個功能，其會傳回一個符合所指定查詢範本的 RDF 圖形。對於此使用案例，您可以使用它，透過使用如下查詢一次一個命名空間匯出資料：

```
CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }
```

雖然存在其他 RDF 工具來匯出此資料，但執行此查詢的最簡便方法是使用 Blazegraph 提供的 REST API 端點。以下指令碼示範如何使用 Python (3.6+) 指令碼，將資料匯出為 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()
```

如果資料儲存為三元組，您必須變更 `Accept` 標頭參數，才能使用 Blazegraph GitHub 存放庫上指定的值，以適當的格式 (N-Triples、RDF/XML 或 Turtle) 匯出資料。

使用 Blazegraph 匯出公用程式匯出資料

Blazegraph 包含一個公用程式方法來匯出資料，即 `ExportKB` 類別。`ExportKB` 有助於從 Blazegraph 匯出資料，但與以前的方法不同，其會要求在匯出執行時伺服器處於離線狀態。這使得它成為在下列情況使用的理想方法：您可以在遷移期間使 Blazegraph 離線，或者從資料備份進行遷移。

在已安裝 Blazegraph 但其未執行的電腦上，您會從 Java 命令列執行公用程式。執行此命令的最簡便方法是下載位於 GitHub 的最新 blazegraph.jar 版本。執行此命令需要數個參數：

- **`log4j.primary.configuration`** – log4j 屬性檔案的位置。
- **`log4j.configuration`** – log4j 屬性檔案的位置。
- **`output`** – 所匯出資料的輸出目錄。檔案是以 `tar.gz` 形式位於依知識庫中所記載方式命名的子目錄。
- **`format`** – 所需的輸出格式，後面跟著 `RWStore.properties` 檔案的位置。如果您正在使用三元組，則需要將 `-format` 參數變更為 `N-Triples`、`Turtle` 或 `RDF/XML`。

例如，如果您有 Blazegraph 日誌檔案和屬性檔案，請使用下列程式碼，將資料匯出為 N-Quads：

```
java -cp blazegraph.jar \
    com.bigdata.rdf.sail.ExportKB \
    -outdir ~/temp/ \
    -format N-Quads \
    ./RWStore.properties
```

如果匯出成功，您會看到如下輸出：

```
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
```

建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體，並將匯出的資料複製到其中

一旦從 Blazegraph 匯出了您的資料，請在與目標 Neptune 資料庫叢集相同的區域中建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體，以供 Neptune 大量載入器用來從中匯入資料。

如需如何建立 Amazon S3 儲存貯體的指示，請參閱 Amazon Simple Storage Service 使用者指南https://docs.aws.amazon.com/AmazonS3/latest/user-guide/中的如何建立 S3 儲存貯體？，以及 Amazon Simple Storage Service 使用者指南https://docs.aws.amazon.com/AmazonS3/latest/dev/中的建立儲存貯體的範例。

如需如何將您已匯出的資料複製到新 Amazon S3 儲存貯體的相關指示，請參閱 Amazon Simple Storage Service 使用者指南中的將物件上傳到儲存貯體，或參閱搭配 AWS CLI 使用高階 (s3) 命令。您也可以使用如下的 Python 程式碼逐一複製檔案：

```
import boto3

region = 'region name'
bucket_name = 'bucket name'
s3 = boto3.resource('s3')
s3.meta.client.upload_file('export.nq', bucket_name, 'export.nq')
```

使用 Neptune 大量載入器將資料匯入至 Neptune

在從 Blazegraph 匯出您的資料並將其複製到 Amazon S3 儲存貯體之後，您就可以將資料匯入至 Neptune。與使用 SPARQL 執行負載作業相比，Neptune 具有可以更快地載入資料且負荷更少的大量載入器。大量載入器程序會透過呼叫載入器端點 API 來啟動，以將儲存在所識別 S3 儲存貯體的資料載入至 Neptune。

雖然您可以直接呼叫載入器 REST 端點來執行此操作，但您必須對目標 Neptune 執行個體執行所在的私有 VPC 具有存取權。您可以設定堡壘主機、透過 SSH 進入該機器，然後執行 cURL 命令，但使用 Neptune 工作台更容易。

Neptune 工作台是一個預先設定的 Jupyter 筆記本，以 Amazon SageMaker 筆記本形式執行，其中安裝了數個 Neptune 特定的筆記本魔法。這些魔法可簡化常見的 Neptune 操作，例如檢查叢集狀態、執行 SPARQL 和 Gremlin 周遊，以及執行大量載入操作。

若要啟動大量載入程序，請使用 `%load` 魔法，這會提供一個執行 Neptune 載入器命令 的介面：

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 選取 aws-neptune-blazegraph-to-neptune。
3. 選擇開啟筆記本。
4. 在 Jupyter 的執行中執行個體中，請選取現有的筆記本，或使用 Python 3 核心建立新的筆記本。
5. 在您的記事本中，開啟儲存格、輸入 `%load`，然後執行儲存格。

6. 設定大量載入器的參數：

   a. 針對來源，輸入要匯入的來源檔案位置：`s3://{bucket_name}/{file_name}`。

   b. 針對格式，選擇適當的格式，在此範例中為 `nquads`。

   c. 針對載入 ARN，輸入 `IAMBulkLoad` 角色的 ARN (此資訊位於 IAM 主控台的角色下)。

7. 選擇 Submit (提交)。

結果包含請求的狀態。大量載入通常是長時間執行的過程，因此回應並不表示載入已完成，只是它已開始。此狀態資訊會定期更新，直到其報告工作完成為止。

> **Note**
>
> 此資訊也可以在部落格文章移至雲端：將 Blazegraph 遷移至 Amazon Neptune 中取得。

將資料載入至 Amazon Neptune

有幾種方式可以將圖形資料載入至 Amazon Neptune：

- 如果您只須載入相對少量的資料，您可以使用查詢，例如 SPARQL `INSERT` 陳述式或 Gremlin `addV` 和 `addE` 步驟。
- 您可以利用 Neptune 大量載入器 來擷取位於外部檔案中的大量資料。大量載入器命令的速度較快，而且其額外負荷比查詢語言命令少。它已針對大型資料集最佳化，且支援 RDF (資源描述架構) 資料和 Gremlin 資料。
- 您可以使用 AWS Database Migration Service (AWS DMS) 從其他資料倉庫匯入資料（ 用 AWS Database Migration Service 於從不同的資料存放區將資料載入 Amazon Neptune請參閱和使AWS Database Migration Service 用指南 ）。
- 最後，您可以使用 Gremlin 的 `g.io(URL).read()` 步驟，來讀取 GraphML (XML 格式)、GraphSON (JSON 格式) 和其他格式的資料檔案。詳情請參閱TinkerPop文件。

主題

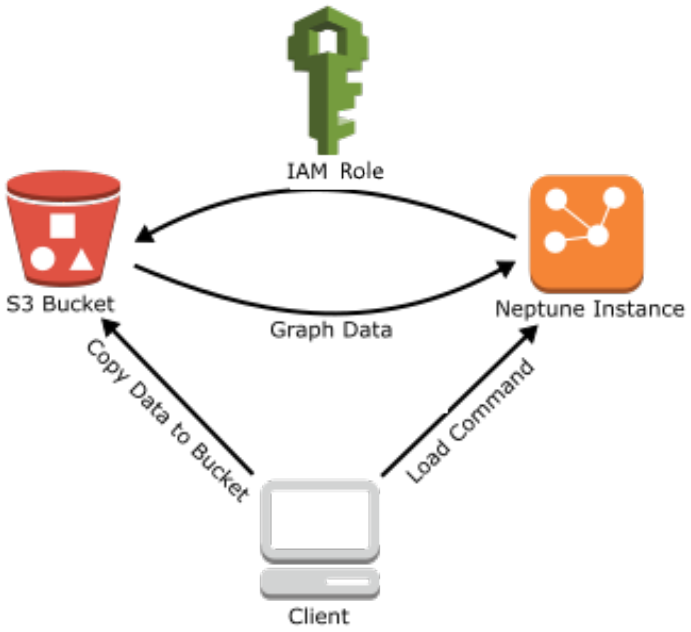
- 使用 Amazon Neptune 大量載入器擷取資料
- 用 AWS Database Migration Service 於從不同的資料存放區將資料載入 Amazon Neptune

使用 Amazon Neptune 大量載入器擷取資料

Amazon Neptune 提供 `Loader` 命令，直接從外部檔案將資料載入至 Neptune 資料庫叢集。您可以使用此命令，而非執行大量 `INSERT` 陳述式、`addV` 和 `addE` 步驟，或其他 API 呼叫。

Neptune Loader 命令速度更快且負荷較低、已針對大型資料集進行最佳化，且同時支援 Gremlin 資料和 SPARQL 使用的 RDF (資源描述架構) 資料。

下圖顯示載入程序的概觀：



以下是載入程序的步驟：

1. 將資料檔案複製到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。
2. 建立具有讀取和列出儲存貯體的存取權的 IAM 角色。
3. 建立 Amazon S3 VPC 端點。
4. 透過 HTTP 將請求傳送至 Neptune 資料庫執行個體，來啟動 Neptune 載入器。
5. Neptune 資料庫執行個體擔任 IAM 角色，從儲存貯體載入資料。

> **Note**
>
> 您可以從 Amazon S3 載入加密的資料 (如果它是使用 Amazon S3 `SSE-S3` 或 `SSE-KMS` 模式加密的)，前提是您用於大量載入的角色可以存取 Amazon S3 物件，而且在 SSE-KMS 的情況下，也可以存取 `kms:decrypt`。然後，Neptune 可以模擬您的憑證，代表您發出 `s3:getObject` 呼叫。
> 不過，Neptune 目前不支援載入使用 `SSE-C` 模式加密的資料。

以下章節提供準備資料並將其載入至 Neptune 的指示。

主題

必要條件：IAM 角色和 Amazon S3 存取

從 Amazon Simple Storage Service (Amazon S3) 貯體載入資料需要具有存取儲存貯體的 AWS Identity and Access Management (IAM) 角色。Amazon Neptune 會擔任此角色來載入資料。

> **Note**
>
> 您可以從 Amazon S3 載入加密的資料 (如果它是使用 Amazon S3 SSE-S3 模式加密的)。在這種情況下，Neptune 可以模擬您的憑證，並代表您發出 `s3:getObject` 呼叫。
> 您也可以從 Amazon S3 載入使用 SSE-KMS 模式加密的加密資料，只要您的 IAM 角色包含存取 AWS KMS的必要許可。如果沒有適當的 AWS KMS 權限，大量載入作業就會失敗並傳回回`LOAD_FAILED`應。
> Neptune 目前不支援載入使用 SSE-C 模式加密的 Amazon S3 資料。

以下章節說明如何使用受管 IAM 政策，建立 IAM 角色以存取 Amazon S3 資源，然後將該角色附加至您的 Neptune 叢集。

主題

> **Note**
>
> 這些指示要求您具有 IAM 主控台的存取權，以及管理 IAM 角色和政策的許可。如需詳細資訊，請參閱 IAM 使用者指南中的 AWS 管理主控台工作的許可。
> Amazon Neptune 主控台要求使用者具有下列 IAM 許可，以將角色附加至 Neptune 叢集：

```
iam:GetAccountSummary on resource: *
iam:ListAccountAliases on resource: *
iam:PassRole on resource: * with iam:PassedToService restricted to
 rds.amazonaws.com
```

建立 IAM 角色以允許 Amazon Neptune 存取 Amazon S3 資源

使用 AmazonS3ReadOnlyAccess 受管 IAM 政策建立新的 IAM 角色，其會允許 Amazon Neptune 存取 Amazon S3 資源。

建立允許 Neptune 存取 Amazon S3 的新 IAM 角色

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。
3. 選擇 Create Role (建立角色)。
4. 在 AWS 服務下，選擇 RDS。
5. 選擇下一步：許可。
6. 使用篩選方塊按 S3 一詞進行篩選，並勾選 AmazonS3 ReadOnly 存取旁邊的方塊。

   > **Note**
   >
   > 此政策授予對所有儲存貯體的 s3:Get* 和 s3:List* 許可。稍後的步驟會使用信任政策來限制角色的存取權。
   > 載入器只需要載入來源儲存貯體的 s3:Get* 和 s3:List* 許可，因此您也可以按 Amazon S3 資源限制這些許可。
   > 如果您的 S3 儲存貯體已加密，則必須添加 kms:Decrypt 許可

7. 選擇下一步：檢閱。
8. 將角色名稱設為您的 IAM 角色名稱，例如：NeptuneLoadFromS3。您也可以新增選用的角色描述值，例如：「允許 Neptune 代表您存取 Amazon S3 資源」。
9. 選擇建立角色。
10. 在導覽窗格中，選擇角色。
11. 在 Search (搜尋) 欄位中，輸入您建立的角色名稱，然後在該角色出現於清單時選擇它。
12. 在 Trust Relationships (信任關係) 索引標籤上選擇 Edit trust relationship (編輯信任關係)。

13. 在文字欄位中，貼上以下信任政策。

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

14. 選擇更新信任政策。
15. 完成「將 IAM 角色新增至 Amazon Neptune 叢集」中的步驟。

將 IAM 角色新增至 Amazon Neptune 叢集

使用主控台將 IAM 角色新增至 Amazon Neptune 叢集。這會允許叢集中的任何 Neptune 資料庫執行個體擔任該角色，並從 Amazon S3 載入。

> **Note**
>
> Amazon Neptune 主控台要求使用者具有下列 IAM 許可，以將角色附加至 Neptune 叢集：
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

將 IAM 角色新增至 Amazon Neptune 叢集

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。

2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇您要修改之叢集的叢集識別碼。
4. 選擇 [連線與安全性] 索引標籤。
5. 在「IAM 角色」區段中，選擇您在上一節中建立的角色。
6. 選擇 Add role (新增角色)。
7. 等到 IAM 角色可供叢集存取，再使用它。

建立 Amazon S3 VPC 端點

Neptune 載入器需要 Amazon S3 的 VPC 端點 (類型為閘道)。

設定 Amazon S3 的存取

1. 登入 AWS Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
2. 在導覽窗格中選擇端點。
3. 選擇建立端點。
4. 為閘道類型端點選擇服務名稱 `com.amazonaws.region.s3`。

   > **Note**
   >
   > 如果此處的區域不正確，請務必確保主控台區域正確。

5. 選擇包含 Neptune 資料庫執行個體的 VPC (在 Neptune 主控台中，它是針對您的資料庫執行個體而列出的)。
6. 選取與您叢集相關子網路關聯的路由表旁邊的核取方塊。如果您只有一個路由表，您必須選擇此方塊。
7. 選擇建立端點。

如需建立端點的相關資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 端點。如需 VPC 端點各項限制的相關資訊，請參閱 Amazon S3 的 VPC 端點。

後續步驟

現在您已授與 Amazon S3 儲存貯體的存取權，您可以準備載入資料。如需有關支援格式的資訊，請參閱 載入資料格式。

在 Amazon Neptune 中鏈結 IAM 角色

> **Important**
>
> 在引擎版本 1.2.1.0.R3 中引進的新大量載入跨帳戶功能，其會利用鏈結 IAM 角色，但在某些情況下可能會導致您觀察到大量載入效能降低。因此，已暫時暫停升級至支援此功能的引擎版本，直到此問題得到解決為止。

當您將角色附加至叢集時，您的叢集可以擔任該角色，代表您存取 Amazon S3 中儲存的資料。從引擎版本 1.2.1.0.R3 開始，如果該角色無法存取您需要的所有資源，您可以鏈結您的叢集可以擔任的一或多個其他角色，以取得其他資源的存取權。鏈結中的每個角色都會擔任鏈結中的下一個角色，直到您的叢集已擔任鏈結尾端的角色為止。

若要鏈結這些角色，請在它們之間建立信任關係。例如，若要將 `RoleB` 鏈接至 `RoleA`，`RoleA` 必須具有允許它擔任 `RoleB` 的許可政策，而且 `RoleB` 必須具有允許其將其許可傳回 `RoleA` 的信任政策。如需詳細資訊，請參閱使用 IAM 角色。

鏈結中的第一個角色必須附加至正在載入資料的叢集。

第一個角色以及擔任鏈結中隨後角色的每個後續角色必須具有：

- 包含對 `sts:AssumeRole` 動作具有 `Allow` 效果之特定陳述式的政策。
- `Resource` 元素中下一個角色的 Amazon Resource Name (ARN)。

> **Note**
>
> 目標 Amazon S3 儲存貯體必須與叢集位於相同的 AWS 區域。

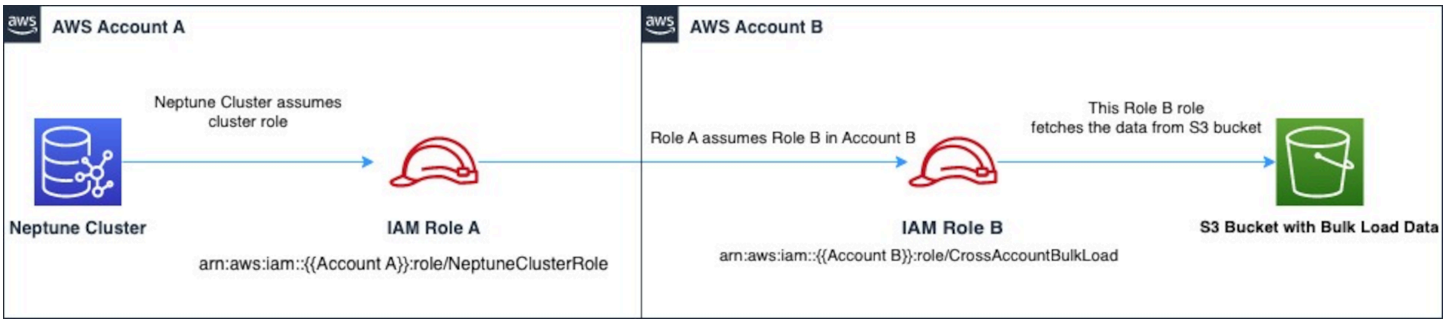
使用鏈結的角色進行跨帳戶存取

您可以鏈結屬於另一個帳戶的一個或多個角色來授與跨帳戶存取權。當您的叢集暫時擔任屬於另一個帳戶的角色時，它可以取得該處資源的存取權。

例如，假設帳戶 A 想要存取屬於帳戶 B 的 Amazon S3 儲存貯體中的資料：

- 帳戶 A 會為名為的 Neptune 建立 AWS 服務角色，`RoleA`並將其附加至叢集。
- 帳戶 B 會建立一個名為 `RoleB` 的角色，其獲授權可以存取帳戶 B 儲存貯體中的資料。

- 帳戶 A 會將許可政策附加至 RoleA，允許其擔任 RoleB。
- 帳戶 B 會將信任策略附加至 RoleB，允許其將其許可傳回 RoleA。
- 若要存取帳戶 B 儲存貯體中的資料，帳戶 A 會使用鏈結 RoleA 和 RoleB 的 iamRoleArn 參數執行載入器命令。在載入器操作期間，RoleA 接著會暫時擔任 RoleB，以存取帳戶 B 中的 Amazon S3 儲存貯體。



例如，RoleA 會有一個與 Neptune 建立信任關係的信任政策：

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

RoleA 還會有一個允許它擔任 RoleB 的許可政策，此角色是帳戶 B 擁有的：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1487639602000",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
```

```
      "Resource": "arn:aws:iam::(Account B ID):role/RoleB"
    }
  ]
}
```

相反地，RoleB 會有一個信任政策，與 RoleA 建立信任關係：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::(Account A ID):role/RoleA"
      }
    }
  ]
}
```

RoleB 還需要許可，才能存取位於帳戶 B 的 Amazon S3 儲存貯體中的資料。

建立 AWS Security Token Service (STS) VPC 端點

當您連結 IAM 角色以透過私有 IP 位址私有存取 AWS STS API AWS STS 時，Neptune 載入程式需要 VPC 端點。您可以以安全且可擴展的方式，直接從 Amazon VPC 連線到 AWS STS VPC 端點。當您使用介面 VPC 端點時，它會提供更安全的狀態，因為您不需要開啟輸出流量防火牆。它還會提供使用 Amazon VPC 端點的其他好處。

使用 VPC 端點時，流量 AWS STS 不會透過網際網路傳輸，也不會離開 Amazon 網路。您的 VPC 人雲端會安全地連線至網路流量， AWS STS 不會造成可用性風險或頻寬限制。如需詳細資訊，請參閱使用 AWS STS 介面 VPC 端點。

若要設定 AWS Security Token Service (STS) 的存取權

1. 登入 AWS Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
2. 在導覽窗格中選擇端點。
3. 選擇建立端點。
4. 為介面類型端點選擇服務名稱：com.amazonaws.region.sts。

5. 選擇包含您的 Neptune 資料庫執行個體和 EC2 執行個體的 VPC。
6. 選取 EC2 執行個體所在子網路旁邊的核取方塊。您無法在相同的可用區域內選取多個子網路。
7. 針對 IP address type (IP 地址類型)，從下列選項中選擇：
   - IPv4 - 將 IPv4 地址指派給您的端點網路介面。只有當所有選取的子網都具有 IPv4 地址範圍時，才支援此選項。
   - IPv6 - 將 IPv6 地址指派給您的端點網路介面。只有當所有選取的子網路都是僅限 IPv6 子網路，才支援此選項。
   - Dualstack - 將 IPv4 和 IPv6 地址指派給您的端點網路介面。只有當所有選取的子網都具有 IPv4 和 IPv6 地址範圍時，才支援此選項。
8. 對於 安全群組，選取要與 VPC 端點的端點網路界面建立關聯的安全群組。您需要選取所有附加到 Neptune 資料庫執行個體和 EC2 執行個體的安全群組。
9. 對於 Policy (政策)，選取 Full access (完整存取)，以允許 VPC 端點上所有資源的所有主體進行所有操作。否則，選取 Custom (自訂) 以連接 VPC 端點政策，該政策控制主體在 VPC 端點上對資源執行動作時所具有的許可。只有服務支援 VPC 端點政策時，此選項才可用。如需詳細資訊，請參閱端點政策。
10. (選用) 若要新增標籤，請選擇新增標籤，然後輸入您想要的標籤金鑰和標籤值。
11. 選擇建立端點。

如需建立端點的相關資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 端點。請注意，Amazon STS VPC 端點是 IAM 角色鏈結的必要條件。

現在您已授與 AWS STS 端點的存取權，您可以準備載入資料。如需有關支援格式的相關資訊，請參閱載入資料格式。

在載入器命令內鏈結角色

您可以在執行載入器命令時指定角色鏈結，方法是在 `iamRoleArn` 參數中包含逗號分隔的角色 ARN 清單。

儘管您大多只需要在鏈結中具有兩個角色，但當然可以同時鏈結三個以上的角色。例如，此載入器命令會鏈結三個角色：

```
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"
     }'
```

載入資料格式

Amazon Neptune Load API 支援載入各種格式的資料。

屬性圖載入格式

然後，可以同時使用 Gremlin 和 OpenCypher，來查詢以下列其中一個屬性圖格式載入的資料：

- Gremlin 載入資料格式 (`csv`)：逗號分隔值 (CSV) 格式。
- OpenCypher 資料載入格式 (`opencypher`)：逗號分隔值 (CSV) 格式。

RDF 載入格式

若要載入您使用 SPARQL 查詢的資源描述架構 (RDF) 資料，您可以使用下列其中一個標準格式，這些格式是由全球資訊網協會 (W3C) 指定：

- N-Triples (`ntriples`) 取自 https://www.w3.org/TR/n-triples/ 的規格。
- N-Quads (`nquads`) 取自 https://www.w3.org/TR/n-quads/ 的規格。
- RDF/XML (`rdfxml`) 取自 https://www.w3.org/TR/rdf-syntax-grammar/ 的規格。
- Turtle (`turtle`) 取自 https://www.w3.org/TR/turtle/ 的規格。

載入資料必須使用 UTF-8 編碼

> **Important**
>
> 所有載入資料都必須以 UTF-8 形式編碼。如果檔案不是以 UTF-8 編碼，Neptune 仍會嘗試以 UTF-8 形式載入它。

在包含 Unicode 的 N-Quads 與 N-triples 資料中，支援 \u*xxxxx* 逸出序列。不過，Neptune 不支援標準化。如果存在需要標準化的值，則 byte-to-byte 在查詢期間將不匹配。如需有關正規化的詳細資訊，請參閱 Unicode.org 中的正規化頁面。

如果您的資料不是支援的格式，您必須先轉換資料，然後再將資料載入 。

將 GraphML 轉換為 Neptune CSV 格式的工具可在上的專案中使用。GitHub

支援壓縮載入資料檔案

Neptune 支援壓縮 `gzip` 或 `bzip2` 格式的個別檔案。

壓縮檔案必須具有 `.gz` 或 `.bz2` 副檔名，且必須是以 UTF-8 格式編碼的單一文字檔案。您可以載入多個檔案，但每一個都必須是個別的 `.gz`、`.bz2` 或未壓縮的文字檔案。不支援封存檔案具有 `.tar`、`.tar.gz` 和 `.tgz` 之類的副檔名。

以下章節將更詳細說明格式。

主題

- Gremlin 載入資料格式
- openCypher 資料的載入格式
- RDF 載入資料格式

Gremlin 載入資料格式

若要使用 CSV 格式載入 Apache 格 TinkerPop 林資料，您必須在不同的檔案中指定頂點和邊緣。

載入器可在單一載入工作中，從多點和多邊檔案進行載入。

對於每個載入命令，要載入的檔案集在 Amazon S3 儲存貯體中必須位於同一資料夾，而且您會為 `source` 參數指定資料夾名稱。檔案名稱和副檔名不重要。

Amazon Neptune CSV 格式遵循 RFC 4180 CSV 規格。如需詳細資訊，請參閱 Internet Engineering Task Force (IETF) 網站上的 Common Format and MIME Type for CSV Files (CSV 檔案的常見格式與 MIME 類型)。

> **Note**
>
> 所有檔案都必須以 UTF-8 格式編碼。

每個檔案都有以逗號分隔的標頭資料列。此標頭資料列包含系統欄標題和屬性欄標頭。

系統欄標題

頂點檔案和邊緣檔案的必要和允許的系統欄標頭是不同的。

每個系統欄只能在標頭中出現一次。

所有標籤皆區分大小寫。

頂點標頭

- `~id` - 必要

  頂點的 ID。
- `~label`

  頂點的標籤。允許多個標籤值，以分號 (`;`) 分隔。

  如果`~label`不存在，則提 TinkerPop 供帶有值的標籤`vertex`，因為每個頂點必須至少有一個標籤。

邊緣標頭

- `~id` - 必要

  邊緣的 ID。
- `~from` - 必要

  從頂點的頂點 ID。
- `~to` - 必要

  至頂點的頂點 ID。
- `~label`

  邊緣的標籤。邊緣只能有一個標籤。

  如果`~label`不存在，則提 TinkerPop 供帶有值的標籤`edge`，因為每個邊緣都必須有一個標籤。

屬性欄標題

您可以使用以下語法指定屬性欄 (`:`)。類型名稱不區分大小寫。不過，請注意，如果屬性名稱內出現冒號，則必須在其前面加上反斜線來逸出冒號：`\:`。

```
propertyname:type
```

> **Note**
>
> 欄標題中不允許使用空格、逗號、換行符號和換行字元，因此屬性名稱不能包含這些字元。

您可以透過將 [] 新增至類型，來為陣列類型指定一個欄：

```
propertyname:type[]
```

> **Note**
>
> 邊緣屬性只能有一個值，如果指定了陣列類型或指定了第二個值，會導致錯誤。

以下範例顯示名為 age 的 Int 類型屬性的欄標頭。

```
age:Int
```

檔案中的每個資料列在該位置都必須有一個整數或保留空白。

允許字串陣列，但陣列中的字串不能包含分號 (;) 字元，除非使用反斜線將其逸出 (像這樣：\;)。

指定欄的基數

從 版本 1.0.1.0.200366.0 (2019 年 7 月 26 日) 開始，欄標頭可用來指定依欄識別之屬性的「基數」。這可讓大量載入器遵守 Gremlin 查詢執行的基數相似性。

您指定的欄基數類似這樣：

```
propertyname:type(cardinality)
```

##值可以是 single 或 set。預設值假設為 set，這表示欄可接受多個值。如果是邊緣檔案，基數一律為一個，指定任何其他基數會導致載入器擲出例外狀況。

如果基數是 single，則當載入一或多個值時，若已有舊值，載入器會拋出錯誤。您可以覆寫這種行為，使用 updateSingleCardinalityProperties 標記，讓載入的新值取代現有的值。請參閱載入器命令。

您可以使用基數設定與陣列類型，但這通常沒必要。以下是可能的組合：

- `name:type` – 基數為 `set`，且內容為單一值。
- `name:type[]` – 基數為 `set`，且內容為多值。
- `name:type(single)` – 基數為 `single`，且內容為單一值。
- `name:type(set)` – 基數為 `set`，與預設值相同，且內容為單一值。
- `name:type(set)[]` – 基數為 `set`，且內容為多值。
- `name:type(single)[]` – 這是矛盾的且會導致擲回錯誤。

下節列出所有可用的 Gremlin 資料類型。

Gremlin 資料類型

這是一個允許的屬性類型清單及每個類型的描述。

布林 (或布林值)

表示布林值欄位。允許的值：`false`、`true`

> **Note**
>
> 所有 `true` 以外的值會被視為 false。

整數類型

超出定義範圍的值將導致錯誤。

Type	範圍
位元組	-128 到 127
Short	-32768 到 32767
Int	-2^31 到 2^31-1
Long	-2^63 到 2^63-1

小數類型

支援小數符號或科學符號。此外，允許符號，例如 (+/-) Infinity 或 NaN。不支援 INF。

Type	範圍
Float	32 位元 IEEE 754 浮點
Double	64 位元 IEEE 754 浮點

浮點數和雙精確度值過長，將四捨五入到最接近 24 位元 (float) 和 53 位元 (double) 精確度的值並載入。位元層級的剩餘數中段值將四捨五入為 0。

字串

引號是選用的。逗號、新行及換行字元如果包含在由雙引號 (") 包圍的字串中，將會自動逸出。範例：`"Hello, World"`

若要在引號字串中包含引號，您可以在同一列使用兩個引號以逸出引號，例如：`"Hello ""World"""`

允許字串陣列，但陣列中的字串不能包含分號 (`;`) 字元，除非使用反斜線將其逸出 (像這樣：`\;`)。

如果您要使用引號包圍陣列中的字串，您必須以一組引號包圍整個陣列。範例：`"String one; String 2; String 3"`

日期

ISO-8601 格式的 Java 資料。支援格式如下：`yyyy-MM-dd`、`yyyy-MM-ddTHH:mm`、`yyyy-MM-ddTHH:mm:ss`、`yyyy-MM-ddTHH:mm:ssZ`

Gremlin 列格式

分隔符號

列中的欄位以逗號分隔。記錄以新行或新行後接換行符號加以分隔。

空白欄位

非必要欄允許使用空白欄位 (例如，使用者定義的屬性)。空白欄位仍然需要逗號分隔符號。必填欄位上的空白欄位會導致剖析錯誤。空字串值會解譯為欄位的空字串值，而不是空白欄位。下一節的範例在每個案例頂點中皆有空白欄位。

頂點 ID

~id 值必須是每個頂點檔案的所有頂點中的唯一值。具有相同 ~id 值的多個頂點列將套用至圖形中的單一頂點。空字符串（""）是一個有效的 ID，頂點創建一個空字符串作為 id。

邊緣 ID

此外，~id 值必須是每個邊緣檔案的所有邊緣中的唯一值。具有相同 ~id 值的多個邊緣列將套用至圖形中的單一邊緣。空字符串（""）是一個有效的 ID，並創建一個空字符串作為 id 的邊緣。

標籤

標籤區分大小寫，不能為空。的值""將導致錯誤。

字串值

引號是選用的。逗號、新行及換行字元如果包含在由雙引號 (") 包圍的字串中，將會自動逸出。空字串值("")會解譯為欄位的空字串值，而非空白欄位。

CSV 格式規格

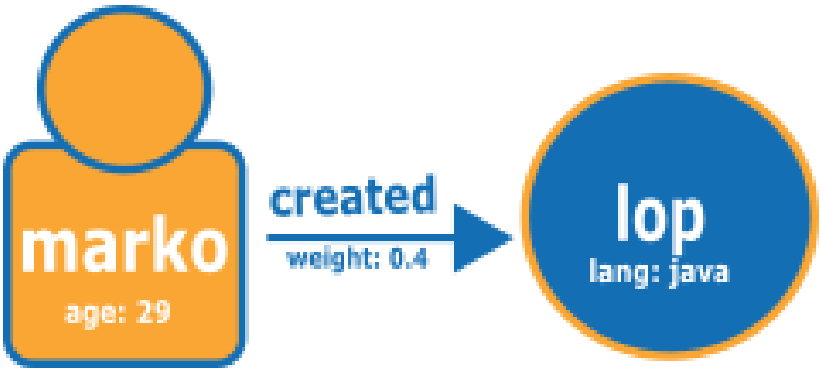
Neptune CSV 格式遵循 RFC 4180 CSV 規格，包括以下需求。

- 支援 Unix 和 Windows 樣式的行尾 (\n 或 \r\n)。
- 任何欄位皆可使用引號 (使用雙引號)。
- 內含分行符號、雙引號或逗號的欄位必須加上引號。(否則，將立即中止載入。)
- 欄位中的雙引號字元 (") 必須由兩個 (雙) 引號字元呈現。例如，Hello "World" 字串在資料中必須以 "Hello ""World""" 呈現。
- 分隔符號前後的空格將被忽略。如果資料列顯示為value1, value2，則會將其儲存為"value1"和"value2"。
- 任何其他逸出字元將逐字儲存。例如，"data1\tdata2" 將儲存為 "data1\tdata2"。這些字元只要括在引號中，就不需要逸出。
- 允許空白欄位。空白欄位會視為空的值。
- 欄位中的多個值可在值之間使用分號 (;) 加以指定。

如需詳細資訊，請參閱 Internet Engineering Task Force (IETF) 網站上的 Common Format and MIME Type for CSV Files (CSV 檔案的常見格式與 MIME 類型)。

Gremlin 範例

下圖顯示了兩個頂點和從 TinkerPop 現代圖採取的邊緣的例子。



以下是 Neptune CSV 載入格式的圖形。

頂點檔案：

```
~id,name:String,age:Int,lang:String,interests:String[],~label
v1,"marko",29,,"sailing;graphs",person
v2,"lop",,"java",,software
```

頂點檔案的表格式檢視：

~id	name:String	age:Int	lang:String	興趣:字串 []	~label
v1	"marko"	29		[「帆船」，「圖表」]	person
v2	"lop"		"java"		software

邊緣檔案：

```
~id,~from,~to,~label,weight:Double
e1,v1,v2,created,0.4
```

邊緣檔案的表格式檢視：

~id	~from	~to	~label	weight:Double
e1	v1	v2	已建立	0.4

後續步驟

現在您對於載入格式已有更深入的了解，請參閱 範例：將資料載入至 Neptune 資料庫執行個體。

openCypher 資料的載入格式

若要使用 OpenCypher CSV 格式載入 openCypher 資料，您必須在個別的檔案中指定節點和關係。載入器可以在單一載入工作中從其中多個節點檔案和關係檔案載入。

對於每個載入命令，要載入的檔案集必須在 Amazon Simple Storage Service 儲存貯體中具有相同的路徑字首。您可以在 source 參數中指定該字首。實際檔案名稱和副檔名並不重要。

在 Amazon Neptune 中，openCypher CSV 格式遵守 RFC 4180 CSV 規格。如需詳細資訊，請參閱 Internet Engineering Task Force (IETF) 網站上的 CSV 檔案的常見格式與 MIME 類型 (https://tools.ietf.org/html/rfc4180)。

> **Note**
>
> 這些檔案必須以 UTF-8 格式編碼。

每個檔案都有一個以逗號分隔的標頭列，其中包含系統欄標頭和屬性欄標頭。

openCypher 資料載入檔案中的系統欄標頭

一個給定系統欄只能在每個檔案中出現一次。所有系統欄標頭標籤都會區分大小寫。

對於 OpenCypher 節點載入檔案和關係載入檔案，必要和允許的系統欄標頭有所不同：

節點檔案中的系統欄標頭

- **`:ID`** – (必要) 節點的 ID。

  選用的 ID 空間可以新增至節點 `:ID` 欄標頭，如下所示：`:ID(`*`ID Space`*`)`。例如，`:ID(movies)`。

  載入連線此檔案中節點的關係時，請在關係檔案 `:START_ID` 和/或 `:END_ID` 欄中使用相同的 ID 空間。

  節點 `:ID` 欄可以選擇性地儲存為屬性，格式為 *`property name`*`:ID`。例如，`name:ID`。

在目前和先前載入的所有節點檔案中，節點 ID 應該是唯一的。如果使用 ID 空間，則在目前和之前載入中使用相同 ID 空間的所有節點檔案中，節點 ID 應該是唯一的。

- **:LABEL** – 節點的標籤。

  允許多個標籤值，以分號 (;) 分隔。

關係檔案中的系統欄標頭

- **:ID** – 關係的 ID。當 `userProvidedEdgeIds` 為 true (預設值) 時，這是必要的，但在 `userProvidedEdgeIds` 為 `false` 時無效。

  在目前和先前載入的所有關係檔案中，關係 ID 應該是唯一的。
- **:START_ID** – (必要) 此關係起始之節點的節點 ID。

  或者，ID 空間可與格式為 `:START_ID(`*`ID Space`*`)` 的起始 ID 欄相關聯。指派給起始節點 ID 的 ID 空間應與指派給節點檔案中節點的 ID 空間相符。
- **:END_ID** – (必要) 此關係結束之節點的節點 ID。

  或者，ID 空間可與格式為 `:END_ID(`*`ID Space`*`)` 的結束 ID 欄相關聯。指派給結束節點 ID 的 ID 空間應與指派給節點檔案中節點的 ID 空間相符。
- **:TYPE** – 關係的類型。關係只能具有單一類型。

> **Note**
>
> 如需大量載入程序如何處理重複節點或關係 ID 的相關資訊，請參閱 載入 openCypher 資料。

openCypher 資料載入檔案中的屬性欄標頭

您可以使用下列格式的屬性欄標頭，指定資料欄保留特定屬性的值：

```
propertyname:type
```

欄標題中不允許使用空格、逗號、換行符號和換行字元，因此屬性名稱不能包含這些字元。以下是名為 `age` 且類型為 `Int` 之屬性的欄標頭範例：

```
age:Int
```

以 age:Int 作為欄標頭的資料欄接著必須在每一列中包含整數或空值。

Neptune openCypher 資料載入檔案中的資料類型

- **Bool** 或 **Boolean** – 布林值欄位。允許的值為 true 和 false。

  除 true 以外的任何值都會視為 false。
- **Byte** – 範圍 -128 至 127 內的整數。
- **Short** – 範圍 -32,768 至 32,767 內的整數。
- **Int** – 範圍 -2^31 至 2^31 - 1 內的整數。
- **Long** – 範圍 -2^63 至 2^63 - 1 內的整數。
- **Float** – 32 位元 IEEE 754 浮點數。同時支援十進位符號和科學符號。Infinity、-Infinity 和 NaN 全都得到認可，但 INF 未得到認可。

  位數太多而無法容納的值會四捨五入為最接近的值 (對於位元層級的最後一個剩餘數字，中間值會四捨五入為 0)。
- **Double** – 64 位元 IEEE 754 浮點數。同時支援十進位符號和科學符號。Infinity、-Infinity 和 NaN 全都得到認可，但 INF 未得到認可。

  位數太多而無法容納的值會四捨五入為最接近的值 (對於位元層級的最後一個剩餘數字，中間值會四捨五入為 0)。
- **String** – 引號是選用的。逗號、新行及換行字元若包含在由雙引號 (") 包圍的字串中，將會自動逸出，例如 "Hello, World"。

  您可以在引號括住的字串中包含引號，方法是連續使用兩個，例如 "Hello ""World"""。
- **DateTime** – 下列其中一種 ISO-8601 格式的 Java 日期：
  - yyyy-MM-dd
  - yyyy-MM-ddTHH:mm
  - yyyy-MM-ddTHH:mm:ss
  - yyyy-MM-ddTHH:mm:ssZ

Neptune openCypher 資料載入檔案中的自動轉換資料類型

提供自動轉換資料類型旨在載入 Neptune 目前原本不支援的資料類型。這類欄中的資料會逐字儲存為字串，不對其預期格式進行驗證。允許下列自動轉換資料類型：

- **Char** – Char 欄位。儲存為字串。

- **`Date`、`LocalDate` 和 `LocalDateTime`** – 請參閱 Neo4j Temporal Instants，以取得`date`、`localdate` 和 `localdatetime` 類型的說明。這些值會逐字載入為字串，無需驗證。
- **`Duration`** – 請參閱 Neo4j Duration 格式。這些值會逐字載入為字串，無需驗證。
- Point – 用於儲存空間資料的點欄位。請參閱 Spatial instants。這些值會逐字載入為字串，無需驗證。

openCypher 載入格式的範例

從 TinkerPop 現代圖表取下圖顯示了兩個節點和關係的一個例子：

以下是一般 Neptune openCypher 載入格式的圖形。

節點檔案：

```
:ID,name:String,age:Int,lang:String,:LABEL
v1,"marko",29,,person
v2,"lop",,"java",software
```

關係檔案：

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,v1,v2,created,0.4
```

或者，您可以使用 ID 空間和 ID 做為屬性，如下所示：

第一個節點檔案：

```
name:ID(person),age:Int,lang:String,:LABEL
"marko",29,,person
```

第二個節點檔案：

```
name:ID(software),age:Int,lang:String,:LABEL
"lop",,"java",software
```

關係檔案：

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,"marko","lop",created,0.4
```

RDF 載入資料格式

若要載入資源描述架構 (RDF) 資料，您可以使用以下其中一個標準格式，如全球資訊網協會 (W3C) 的規定：

- N-Triples (`ntriples`) 取自 https://www.w3.org/TR/n-triples/ 的規格
- N-Quads (`nquads`) 取自 https://www.w3.org/TR/n-quads/ 的規格
- RDF/XML (`rdfxml`) 取自 https://www.w3.org/TR/rdf-syntax-grammar/ 的規格
- Turtle (`turtle`) 取自 https://www.w3.org/TR/turtle/ 的規格

> **Important**
>
> 所有檔案都必須以 UTF-8 格式編碼。
> 在包含 Unicode 的 N-Quads 與 N-triples 資料中，支援 \u*xxxxx* 逸出序列。不過，Neptune 不支援標準化。如果存在需要標準化的值，則 byte-to-byte 在查詢期間將不匹配。如需有關正規化的詳細資訊，請參閱 Unicode.org 中的正規化頁面。

後續步驟

現在您對於載入格式已有更深入的了解，請參閱 範例：將資料載入至 Neptune 資料庫執行個體。

範例：將資料載入至 Neptune 資料庫執行個體

此範例說明如何將資料載入至 Amazon Neptune。除非另有說明，否則您必須在與 Neptune 資料庫執行個體相同的 Amazon Virtual Private Cloud (VPC) 中遵循這些來自 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體中的步驟。

資料載入範例的先決條件

開始之前，您必須準備好以下項目：

- Neptune 資料庫執行個體。

  如需啟動 Neptune 資料庫執行個體的相關資訊，請參閱 建立新的 Neptune 資料庫叢集。

- 要在其中放置資料檔案的 Amazon Simple Storage Service (Amazon S3) 儲存貯體。

  您可以使用現有的儲存貯體。如果您沒有 S3 儲存貯體，請參閱《Amazon S3 入門指南》中的建立儲存貯體。

- 以 Neptune 載入器支援的其中一種格式載入的圖形資料：

  如果您使用 Grinmlin 來查詢圖形，Neptune 可以載入 comma-separated-values (CSV) 格式的資料，如中所述。Gremlin 載入資料格式

  如果您使用 OpenCypher 來查詢您的圖形，Neptune 也可以載入 openCypher 特定 CSV 格式的資料，如 openCypher 資料的載入格式 中所述。

  如果您使用的是 SPARQL，Neptune 可以使用多種 RDF 格式載入資料，如 RDF 載入資料格式 中所述。

- Neptune 資料庫執行個體要擔任的 IAM 角色，其具有 IAM 政策，允許存取 S3 儲存貯體中的資料檔案。此政策必須授予讀取與列出許可。

  如需建立可存取 Amazon S3 的角色，然後將其與 Neptune 叢集建立關聯的相關資訊，請參閱 必要條件：IAM 角色和 Amazon S3 存取。

  > **Note**
  >
  > Neptune Load API 僅需要資料檔案的讀取存取權。IAM 政策不需要允許整個儲存貯體的寫入存取權或存取權。

- Amazon S3 VPC 端點。如需詳細資訊，請參閱 建立 Amazon S3 VPC 端點 一節。

建立 Amazon S3 VPC 端點

Neptune 載入器需要 Amazon S3 的 VPC 端點。

設定 Amazon S3 的存取

1. 登入 AWS Management Console 並開啟 Amazon VPC 主控台，網址為 https://console.aws.amazon.com/vpc/。
2. 在左側導覽窗格中選擇 Endpoints (端點)。
3. 選擇建立端點。
4. 選擇 Service Name (服務名稱) `com.amazonaws.region.s3`。

   > **Note**
   >
   > 如果此處的區域不正確，請務必確保主控台區域正確。

5. 選擇包含 Neptune 資料庫執行個體的 VPC。
6. 選取與您叢集相關子網路關聯的路由表旁邊的核取方塊。如果您只有一個路由表，您必須選擇此方塊。
7. 選擇建立端點。

如需建立端點的相關資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 端點。如需 VPC 端點各項限制的相關資訊，請參閱 Amazon S3 的 VPC 端點。

將資料載入至 Neptune 資料庫執行個體

1. 將資料檔複製至 Amazon S3 儲存貯體。S3 儲存貯體必須與載入資料的叢集位於相同的 AWS 區域。

   您可以使用下列 AWS CLI 指令將檔案複製到值區。

   > **Note**
   >
   > 此命令不需要從 Amazon EC2 執行個體執行。

   ```
   aws s3 cp data-file-name s3://bucket-name/object-key-name
   ```

   > **Note**
   >
   > 在Amazon S3 中，物件金鑰名稱是檔案的完整路徑，包括檔案名稱。

範例：在命令 `aws s3 cp datafile.txt s3://examplebucket/mydirectory/datafile.txt` 中，物件鍵名稱為 **`mydirectory/datafile.txt`**。

或者，您可以使用 AWS Management Console 將檔案上傳到 S3 儲存貯體。在 https://console.aws.amazon.com/s3/ 開啟 Amazon S3 主控台，然後選擇儲存貯體。在左上角，選擇 Upload (上傳) 以上傳檔案。

2. 從命令列視窗中，輸入以下命令，並使用端點、Amazon S3 路徑、格式和 IAM 角色 ARN 的正確值，來執行 Neptune 載入器。

   `format` 參數可以是下列任何一個值：`csv` (適用於 Gremlin)、`opencypher` (適用於 openCypher) 或 `ntriples`、`nquads`、`turtle` 和 `rdfxml` (適用於 RDF)。如需有關其他參數的資訊，請參閱 Neptune 載入器命令。

   如需尋找 Neptune 資料庫執行個體主機名稱的相關資訊，請參閱 連線至 Amazon Neptune 端點一節。

   區域參數必須符合叢集和 S3 儲存貯體的區域。

   Amazon Neptune 在以下 AWS 區域提供：

   - 美國東部 (維吉尼亞北部)：`us-east-1`
   - 美國東部 (俄亥俄)：`us-east-2`
   - 美國西部 (加利佛尼亞北部)：`us-west-1`
   - 美國西部 (奧勒岡)：`us-west-2`
   - 加拿大 (中部)：`ca-central-1`
   - 南美洲 (聖保羅)：`sa-east-1`
   - 歐洲 (斯德哥爾摩)：`eu-north-1`
   - 歐洲 (愛爾蘭)：`eu-west-1`
   - 歐洲 (倫敦)：`eu-west-2`
   - 歐洲 (巴黎)：`eu-west-3`
   - 歐洲 (法蘭克福)：`eu-central-1`
   - 中東 (巴林)：`me-south-1`
   - 中東 (阿拉伯聯合大公國)：`me-central-1`
   - 以色列 (特拉維夫)：`il-central-1`

- 非洲 (開普敦) : af-south-1
- 亞太區域 (香港) : ap-east-1
- 亞太區域 (東京) : ap-northeast-1
- 亞太區域 (首爾) : ap-northeast-2
- 亞太區域 (大阪): ap-northeast-3
- 亞太區域 (新加坡) : ap-southeast-1
- 亞太區域 (雪梨) : ap-southeast-2
- 亞太區域 (孟買) : ap-south-1
- 中國 (北京) : cn-north-1
- 中國 (寧夏) : cn-northwest-1
- AWS GovCloud (美國西部): us-gov-west-1
- AWS GovCloud (美國東部): 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"]
    }'
```

如需建立 IAM 角色並將其與 Neptune 叢集建立關聯的相關資訊，請參閱 必要條件：IAM 角色和 Amazon S3 存取。

> **Note**
>
> 如需載入請求參數的詳細資訊，請參閱 Neptune 載入器請求參數)。簡而言之：source 參數接受指向單一檔案或資料夾的 Amazon S3 URI。如果您指定資料夾，Neptune 會將每個資料檔案載入該資料夾。

資料夾可包含多個頂點檔案和多個邊緣檔案。
URI 可以是下列任何格式。

- `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`

`format` 參數可為以下任一項：

- 用於 Gremlin 屬性圖的 Gremlin CSV 格式 (`csv`)
- 用於 openCypher 屬性圖的 openCypher CSV 格式 (`opencypher`)
- N -Triples (`ntriples`) 格式用於 RDF/SPARQL
- N-Quads (`nquads`) 格式用於 RDF/SPARQL
- RDF/XML (`rdfxml`) 格式用於 RDF/SPARQL
- Turtle (`turtle`) 格式用於 RDF/SPARQL

選用的 `parallelism` 參數可讓您限制大量載入程序所用的執行緒數目。它可以設定為 `LOW`、`MEDIUM`、`HIGH` 或 `OVERSUBSCRIBE`。
當 `updateSingleCardinalityProperties` 設為 `"FALSE"` 時，如果針對邊緣或單一基數頂點屬性所載入的來源檔案中提供了多個值，則載入器會傳回錯誤。
如果已有執行中的載入任務，則將 `queueRequest` 設為 `"TRUE"` 會造成載入請求置於佇列。
`dependencies` 參數會在已置於佇列的一或多個載入任務順利完成時執行載入請求。

3. Neptune 載入器會傳回一個工作 `id`，可讓您檢查狀態或取消載入程序，例如：

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

4. 輸入以下命令，從步驟 3 取得具有 `loadId` 之載入的狀態：

```
curl -G 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

如果載入狀態列出錯誤，您可以請求更詳細的狀態和錯誤清單。如需詳細資訊和範例，請參閱 Neptune 載入器 Get-Status API。

5. (選用) 取消 Load 工作。

   輸入以下命令，從步驟 3 中 Delete 具有工作 id 的載入器工作：

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

   成功取消時，DELETE 命令將傳回 HTTP 程式碼 200 OK。

   來自已完成載入的載入工作的檔案資料不會還原。資料仍會保留在 Neptune 資料庫執行個體中。

最佳化 Amazon Neptune 大量載入

使用下列策略，讓 Neptune 大量載入保持最短的載入時間：

- 清除您的資料：
  - 載入之前，請務必將您的資料轉換為支援的資料格式。
  - 刪除任何重複內容或已知錯誤。
  - 盡可能減少唯一述詞 (例如邊緣和頂點的屬性) 的數目。
- 最佳化您的檔案：
  - 如果您從 Amazon S3 儲存貯體載入大型檔案 (例如 CSV 檔案)，載入器會透過將這些大型檔案剖析為其可以平行載入的區塊，來為您管理並行。使用大量小檔案可能會減慢此程序的速度。
  - 如果您從 Amazon S3 資料夾載入多個檔案，載入器會先自動載入頂點檔案，然後再載入邊緣檔案。
  - 壓縮檔案可減少傳輸時間。載入器支援來源檔案的 gzip 壓縮。
- 檢查您的載入器設定：
  - 如果您不需要在載入期間執行任何其他操作，請使用 OVERSUBSCRIBEparallelism 參數。此參數設定會導致大量載入器在執行時使用所有可用的 CPU 資源。通常需要 60%-70% 的 CPU 容量，才能將操作的執行速度保持與 I/O 限制允許的速度一樣快。

> **Note**
>
> 當 `parallelism` 設定為 `OVERSUBSCRIBE` 或 `HIGH` (預設設定) 時，載入 OpenCypher 資料時存在執行緒可能會遇到競爭條件和死結的風險，進而導致 `LOAD_DATA_DEADLOCK` 錯誤。在此情況下，請將 `parallelism` 設定為較低的設定，然後重試載入。

- 如果您的載入工作將包含多個載入請求，請使用 `queueRequest` 參數。將 `queueRequest` 設定為 `TRUE` 可讓 Neptune 將您的請求排入佇列，這樣您就不必等待一個請求完成後，再發出另一個請求。
- 如果您的載入請求正要排入佇列，您可以使用 `dependencies` 參數設定相依性層級，以便一個工作若失敗就會導致相依工作失敗。這樣可以防止載入的資料中出現不一致的情況。
- 如果載入工作將涉及更新先前載入的值，請務必將 `updateSingleCardinalityProperties` 參數設定為 `TRUE`。如果不這樣做，載入器會將嘗試更新現有的單一基數值視為錯誤。對於 Gremlin 資料，也會在屬性欄標頭中指定基數 (請參閱 屬性欄標題)。

> **Note**
>
> 此`updateSingleCardinalityProperties`參數不適用於資源描述架構 (RDF) 資料。

- 您可以使用 `failOnError` 參數來決定發生錯誤時，大量載入操作應該失敗還是繼續。此外，您可以使用 `mode` 參數，來確定載入工作會從前一個工作失敗的那一點繼續載入，而不是重新載入已載入的資料。

- 縱向擴展 – 在大量載入之前，將資料庫叢集的寫入器執行個體設定為大小上限。請注意，如果這樣做，您必須同時縱向擴展資料庫叢集中的任何僅供讀取複本執行個體，或將其移除，直到完成載入資料為止。

  當大量載入完成時，請務必再次縮減寫入器執行個體。

> **Important**
>
> 如果您由於大量載入期間的複寫延遲而遭遇重複僅供讀取複本重新啟動的週期，則您的複本可能無法跟上資料庫叢集中的寫入器。將讀取器擴展為大於寫入器，或在大量載入期間暫時將其移除，然後在完成後重新建立這些讀取器。

如需設定載入器請求參數的詳細資訊，請參閱 請求參數。

Neptune 載入器參考

本節描述適用於 Amazon Neptune 的 `Loader` API，其可從 Neptune 資料庫執行個體的 HTTP 端點取得。

> **Note**
>
> 如需載入器在發生錯誤時傳回的錯誤和饋送訊息清單，請參閱 Neptune 載入器的錯誤和饋送訊息。

內容

Neptune 載入器命令

將資料從 Amazon S3 儲存貯體載入至 Neptune 資料庫執行個體。

若要載入資料，您必須將 HTTP POST 請求傳送至 `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader` 端點。您可以用 POST 本文或以 URL 編碼參數來傳送 `loader` 請求的參數。

> **Important**
>
> MIME 類型必須是 `application/json`。

S3 儲存貯體必須與叢集位於相同的 AWS 區域。

> **Note**
>
> 您可以從 Amazon S3 載入加密的資料 (如果它是使用 Amazon S3 SSE-S3 模式加密的)。在這種情況下，Neptune 可以模擬您的憑證，並代表您發出 `s3:getObject` 呼叫。
> 您也可以從 Amazon S3 載入使用 SSE-KMS 模式加密的加密資料，只要您的 IAM 角色包含存取 AWS KMS的必要許可。如果沒有適當的 AWS KMS 權限，大量載入作業就會失敗並傳回回`LOAD_FAILED`應。
> Neptune 目前不支援載入使用 SSE-C 模式加密的 Amazon S3 資料。

您不必等待一個載入工作完成，然後再啟動另一個工作。Neptune 最多可以一次將 64 個工作排入佇列，前提是其 `queueRequest` 參數全都設為 `"TRUE"`。作業的佇列順序為 first-in-first-out (FIFO)。另

一方面，若不希望載入工作排入佇列，則可將其 queueRequest 參數設為 "FALSE" (預設值)，如此一來，一項載入工作進行時，便無法啟動另一項載入工作。

針對只在且須在佇列中指定的先前任務順利完成後才能執行的任務，您可以使用 dependencies 參數將這些任務排入佇列。若這麼做，且任一指定的任務失敗的話，則您的任務將不會執行，且其狀態將設為 LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED。

Neptune 載入器請求語法

```
{
  "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"]
}
```

Neptune 載入器請求參數

- **`source`** – Amazon S3 URI。

  SOURCE 參數接受可識別單一檔案、多個檔案、一個資料夾或多個資料夾的 Amazon S3 URI。Neptune 會載入任何所指定資料夾中的每個資料檔案。

  URI 可以是下列任何格式。

  - `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`

  URI 的`object-key-name`元素相當於 Amazon S3 ListObjectsAPI 呼叫中的前置詞參數。它會識別所指定 Amazon S3 儲存貯體中其名稱以該字首開頭的所有物件。這可以是單一檔案或資料夾，也可以是多個檔案和/或資料夾。

一個或多個指定的資料夾可以包含多個頂點檔案和多個邊緣檔案。

例如，如果您在名為的 Amazon S3 儲存貯體中具有下列資料夾結構和檔案`bucket-name`：

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
s3://bucket-name/bcd
```

如果 source 參數指定為`s3://bucket-name/a`，則會載入前三個檔案。

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
```

- **`format`** – 資料的格式。如需有關 Neptune `Loader` 命令的資料格式詳細資訊，請參閱 使用 Amazon Neptune 大量載入器擷取資料。

  允許的值

  - **`csv`** 表示 Gremlin CSV 資料格式。
  - **`opencypher`** 表示 openCypher CSV 資料格式。
  - **`ntriples`** 表示 N-Triples RDF 資料格式。
  - **`nquads`** 表示 N-Quads RDF 資料格式。
  - **`rdfxml`** 表示 RDF\XML RDF 資料格式。
  - **`turtle`** 表示 Turtle RDF 資料格式。

- **`iamRoleArn`** – Neptune 資料庫執行個體為了存取 S3 儲存貯體 而擔任之 IAM 角色的 Amazon Resource Name (ARN)。如需建立可存取 Amazon S3 的角色，然後將其與 Neptune 叢集建立關聯的相關資訊，請參閱 必要條件：IAM 角色和 Amazon S3 存取。

  從引擎版本 1.2.1.0.R3 開始，如果 Neptune 資料庫執行個體和 Amazon S3 儲存貯體位於不同的帳戶中，您也可以鏈結多個 IAM 角色。 AWS 在此情況下，`iamRoleArn` 會包含逗號分隔的角色 ARN 清單，如 在 Amazon Neptune 中鏈結 IAM 角色 中所述。例如：

```
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**— region 參數必須與叢集的 AWS 區域和 S3 儲存貯體相符。

  Amazon Neptune 可在下列 區域使用：

  - 美國東部 (維吉尼亞北部)：us-east-1
  - 美國東部 (俄亥俄)：us-east-2
  - 美國西部 (加利佛尼亞北部)：us-west-1
  - 美國西部 (奧勒岡)：us-west-2
  - 加拿大 (中部)：ca-central-1
  - 南美洲 (聖保羅)：sa-east-1
  - 歐洲 (斯德哥爾摩)：eu-north-1
  - 歐洲 (愛爾蘭)：eu-west-1
  - 歐洲 (倫敦)：eu-west-2
  - 歐洲 (巴黎)：eu-west-3
  - 歐洲 (法蘭克福)：eu-central-1
  - 中東 (巴林)：me-south-1
  - 中東 (阿拉伯聯合大公國)：me-central-1
  - 以色列 (特拉維夫)：il-central-1
  - 非洲 (開普敦)：af-south-1
  - 亞太區域 (香港)：ap-east-1
  - 亞太區域 (東京)：ap-northeast-1
  - 亞太區域 (首爾)：ap-northeast-2
  - 亞太區域 (大阪): ap-northeast-3
  - 亞太區域 (新加坡)：ap-southeast-1
  - 亞太區域 (雪梨)：ap-southeast-2
  - 亞太區域 (孟買)：ap-south-1
  - 中國 (北京)：cn-north-1

- 中國 (寧夏)：cn-northwest-1
- AWS GovCloud (美國西部): us-gov-west-1
- AWS GovCloud (美國東部): us-gov-east-1

- **mode** – 載入工作模式。

  允許的值：RESUME、NEW、AUTO。

  預設值：AUTO

  - RESUME – 在 RESUME 模式下，載入器會從此來源尋找先前的載入，若找到某載入工作，則會繼續該工作。如果找不到先前的載入任務，載入器便會停止。

    載入程式可避免重新載入已順利載入先前任務的檔案。它只會嘗試處理失敗的檔案。如果從 Neptune 叢集捨棄先前載入的資料，則不會在此模式下重新載入該資料。如果先前的載入工作已從相同來源順利載入所有檔案，則不會重新載入任何工作，且載入器會傳回成功。
  - NEW – 在 NEW 模式下，建立新的載入請求，無論先前有任何的載入。您可以使用此模式，在捨棄 Neptune 叢集先前載入的資料之後，從來源重新載入所有資料，或者載入同一來源的可用新資料。
  - AUTO – 在 AUTO 模式下，載入器會從相同來源尋找先前的載入任務，若找到一項載入任務，則會繼續該任務，如同 RESUME 模式一般。

    若載入器未能從相同來源找到先前的載入任務，則會從該來源載入所有資料，如同 NEW 模式那樣。
- **failOnError** – 一種旗標，用來切換錯誤時完全停止。

  允許的值："TRUE"、"FALSE"。

  預設值："TRUE"。

  此參數設為 "FALSE" 時，載入器會嘗試載入指定位置中的所有資料，並略過任何有錯誤的項目。

  此參數設為 "TRUE" 時，載入器會在遇到錯誤時立即停止。到該時間點載入的資料仍然存在。
- **parallelism** – 這是選用參數，可設定以減少大量載入程序所用的執行緒數量。

  允許的值：

  - LOW – 使用的執行緒數目是可用 vCPU 除以 8 後的數字。
  - MEDIUM – 使用的執行緒數目是可用 vCPU 除以 2 後的數字。

- HIGH – 使用的執行緒數目與可用 vCPU 的數目相同。
- OVERSUBSCRIBE – 使用的執行緒數目是可用 vCPU 乘以 2 後的數字。如果使用此值，大量載入器會佔用所有可用的資源。

  不過，這並不表示 OVERSUBSCRIBE 設定會產生 100% CPU 使用率。由於載入操作受 I/O 限制，因此預期的最高 CPU 使用率在 60% 到 70% 的範圍內。

預設值：HIGH

載入 OpenCypher 資料時，parallelism 設定有時可能會導致執行緒之間發生死結。發生這種情況時，Neptune 會傳回 LOAD_DATA_DEADLOCK 錯誤。通常，您可以透過將 parallelism 設定為較低的設定，並重試載入命令來修正此問題。

- **parserConfiguration** – 包含額外剖析器組態值的選用物件。也可選用每個子參數：

名稱	範例值	描述
namedGraphUri	*http://aws.amazon.com/neptune/vocab/v01/ DefaultNamed ##*	未指定圖形時，所有 RDF 格式的預設圖形 (適用於非 quads 格式及無圖形的 NQUAD 項目)。預設為 http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph 。
baseUri	*http://aws.amazon.com/neptune/default*	RDF/XML 和 Turtle 格式的基本 URI。預設值為 http://aws.amazon.com/neptune/default 。
allowEmptyStrings	*true*	載入 CSV 資料時，Gemlin 使用者必須能夠傳遞空字串值 ("") 做為節點和邊緣屬性。如果 allowEmptyStrings 設定為 false (預設值)，則這類空字串會被視為 null 而不會將其載入。

如果 allowEmptyStrings 設定為 true，載入器會將空字串視為有效的屬性值，並相應地載入它們。

如需詳細資訊，請參閱 SPARQL 預設圖形和具名圖形。

- **updateSingleCardinalityProperties** – 這是選用參數，可控制大量載入器如何處理單一基數頂點或邊緣屬性的新值。不支援將此用於載入 openCypher 資料 (請參閱 載入 openCypher 資料)。

  允許的值："TRUE"、"FALSE"。

  預設值："FALSE"。

  根據預設，或當 updateSingleCardinalityProperties 明確設定為 "FALSE" 時，載入器會將新值視為錯誤，因為它違反單一基數。

  另一方面，當 updateSingleCardinalityProperties 設為 "TRUE" 時，大量載入器會以新的值取代現有的值。如果在要載入的來源檔案中提供多個邊緣或單一基數頂點屬性值，則大量載入結尾的最終值可以是這些新值的其中之一。載入器只會保證現有的值已由其中一個新值取代。

- **queueRequest** – 此為選用標記參數，用以指示載入請求能否排入佇列。

  您不必等到載入工作完成才能發出另一個載入工作，因為 Neptune 最多可以一次將 64 個工作排入佇列，前提是其 queueRequest 參數都設為 "TRUE"。作業的佇列順序為 first-in-first-out (FIFO)。

  如果 queueRequest 參數遭省略或設為 "FALSE"，且已有其他執行中的載入任務，則此載入請求將失敗。

  允許的值："TRUE"、"FALSE"。

  預設值："FALSE"。

- **dependencies** – 這是選用參數，可在一或多個佇列中先前工作順利完成時，發出佇列載入請求。

  Neptune 最多可以一次將 64 個載入請求排入佇列，前提是其 queueRequest 參數設為 "TRUE"。dependencies 參數可讓您在一或多個佇列中指定的先前請求順利完成時，執行這類佇列請求。

  例如，如果載入 Job-A 和 Job-B 彼此獨立，但載入 Job-C 須先完成 Job-A 和 Job-B 才能開始，請依照指示執行：

1. 以任何順序一個接著一個提交 `load-job-A` 和 `load-job-B`，並儲存其載入 ID。
2. 使用兩任務其 `dependencies` 欄位中的載入 ID 提交 `load-job-C`：

```
"dependencies" : ["job_A_load_id", "job_B_load_id"]
```

由於 `dependencies` 參數的關係，大量載入器在 `Job-A` 和 `Job-B` 順利完成之前，將不會啟動 `Job-C`。如果其中任何一個失敗，則不執行 Job-C，且其狀態將設為 `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`。

您可以使用這種方式設定多個相依性層級，如此一來，一項任務若失敗，將導致直間接仰賴該任務的所有請求遭取消。

- **`userProvidedEdgeIds`** – 只有在載入包含關係 ID 的 OpenCypher 資料時，才需要這個參數。在載入資料中明確提供 OpenCypher 關係 ID 時，必須包含它並設定為 `True` (建議使用)。

  如果 `userProvidedEdgeIds` 不存在或設定為 `True`，則載入中的每個關係檔案中都必須存在一個 `:ID` 欄。

  當 `userProvidedEdgeIds` 存在且設定為 `False` 時，載入中的關係檔案不得包含 `:ID` 欄。相反地，Neptune 載入器會自動為每個關係產生一個 ID。

  明確提供關係 ID 很有用，如此一來，載入器就可以在 CSV 資料中的錯誤完成修正之後繼續載入，而不必重新載入任何已載入的關係。如果尚未明確指派關係 ID，則載入器無法繼續失敗的載入 (如果任何關係檔案必須更正的話)，反而必須重新載入所有關係。
- `accessKey` – 已棄用 可存取 S3 儲存貯體和資料檔案的 IAM 角色的存取金鑰 ID。

  建議改用 `iamRoleArn` 參數。如需建立可存取 Amazon S3 的角色，然後將其與 Neptune 叢集建立關聯的相關資訊，請參閱 必要條件：IAM 角色和 Amazon S3 存取。

  如需詳細資訊，請參閱存取金鑰 (存取金鑰 ID 和私密存取金鑰)。
- `secretKey` – 已棄用 建議改用 `iamRoleArn` 參數。如需建立可存取 Amazon S3 的角色，然後將其與 Neptune 叢集建立關聯的相關資訊，請參閱 必要條件：IAM 角色和 Amazon S3 存取。

  如需詳細資訊，請參閱存取金鑰 (存取金鑰 ID 和私密存取金鑰)。

載入 openCypher 資料的特殊考量

- 以 CSV 格式載入 openCypher 資料時，必須將格式參數設定為 `opencypher`。

- 不支援 `updateSingleCardinalityProperties` 參數用於 openCypher 載入，因為所有 openCypher 屬性都有單一基數。OpenCypher 載入格式不支援陣列，而且如果 ID 值出現多次，則系統會將其視為重複或插入錯誤 (請參閱下文)。
- Neptune 載入器會處理它在 OpenCypher 資料中遇到的重複項目，如下所示：
  - 如果載入器遇到多個具有相同節點 ID 的資料列，則會使用下列規則合併它們：
    - 資料列行中的所有標籤現在會新增至節點。
    - 對於每個屬性，只會載入其中一個屬性值。選取要載入哪一個是不具確定性的。
  - 如果載入器遇到多個具有相同關係 ID 的資料列，則只會載入其中一個資料列。選取要載入哪一個是不具確定性的。
  - 如果載入器遇到具有現有節點或關係 ID 的載入資料，則它永遠不會更新資料庫中現有節點或關係的屬性值。不過，它確實會載入現有節點或關係中不存在的節點標籤和屬性。
- 雖然您不必將 ID 指派給關係，但是這樣做通常是個好主意 (請參閱上面的 `userProvidedEdgeIds` 參數)。若沒有明確的關係 ID，載入器必須在關係檔案中發生錯誤時重新載入所有關係，而不是從失敗處繼續載入。

  此外，如果載入資料未包含明確的關係 ID，則載入器無法偵測重複的關係。

以下是 openCypher 載入命令的範例：

```
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",
     }'
```

載入器回應與正常回應相同。例如：

```
{
  "status" : "200 OK",
  "payload" : {
```

```
    "loadId" : "guid_as_string"
  }
}
```

Neptune 載入器回應語法

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "guid_as_string"
    }
}
```

200 OK

成功開始的載入任務會傳回 200 程式碼。

Neptune 載入器錯誤

發生錯誤時，回應的 BODY 中會傳回 JSON 物件。message 物件包含此錯誤的描述。

錯誤類別

- Error 400 – 語法錯誤會傳回 HTTP 400 錯誤的請求錯誤。描述錯誤的訊息。
- Error 500 – 無法處理的有效請求會傳回 HTTP 500 內部伺服器錯誤。描述錯誤的訊息。

以下是來自載入器的可能錯誤訊息及錯誤的描述。

載入器錯誤訊息

- Couldn't find the AWS credential for iam_role_arn (HTTP 400)

  找不到登入資料。對照 IAM 主控台或 AWS CLI 輸出驗證提供的登入資料。確定您已將 iamRoleArn 中指定的 IAM 角色新增至叢集。
- S3 bucket not found for source (HTTP 400)

  S3 儲存貯體不存在。檢查儲存貯體的名稱。
- The source *source-uri* does not exist/not reachable (HTTP 400)

  在 S3 儲存貯體找不到相符的檔案。

- `Unable to connect to S3 endpoint. Provided source = source-uri and region = aws-region` (HTTP 500)

  無法連線至 Amazon S3。區域必須符合叢集區域。確定您擁有 VPC 端點。如需有關建立 VPC 端點的資訊，請參閱 建立 Amazon S3 VPC 端點。

- `Bucket is not in provided Region (aws-region)` (HTTP 400)

  儲存貯體必須與 Neptune 資料庫執行個體位於相同的 AWS 區域。

- `Unable to perform S3 list operation` (HTTP 400)

  提供的 IAM 使用者或角色沒有儲存貯體或資料夾的 `List` 許可。檢查儲存貯體的政策或存取控制清單 (ACL)。

- `Start new load operation not permitted on a read replica instance` (HTTP 405)

  載入是一種寫入操作。在讀取/寫入叢集端點上重試載入。

- `Failed to start load because of unknown error from S3` (HTTP 500)

  Amazon S3 傳回未知的錯誤。請聯絡 AWS Support。

- `Invalid S3 access key` (HTTP 400)

  存取金鑰無效。檢查提供的登入資料。

- `Invalid S3 secret key` (HTTP 400)

  私密金鑰無效。檢查提供的登入資料。

- `Max concurrent load limit breached` (HTTP 400)

  如不使用 `"queueRequest" : "TRUE"` 提交載入請求 ，且某載入任務正在執行中，則此請求將失敗，並顯示此錯誤。

- `Failed to start new load for the source "source name". Max load task queue size limit breached. Limit is 64` (HTTP 400)

  Neptune 最多支援一次將 64 個載入器工作排入佇列。若在佇列已包含 64 個任務時，再繼續提交其他載入請求到該佇列，則請求會失敗並顯示此訊息。

Neptune 載入器範例

Example 請求

以下是使用 `curl` 命令透過 HTTP POST 傳送的請求。它會以 Neptune CSV 格式載入一個檔案。如需詳細資訊，請參閱 Gremlin 載入資料格式。

```
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 回應

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

Neptune 載入器 Get-Status API

取得 `loader` 工作的狀態。

若要取得載入狀態，您必須將 HTTP `GET` 請求傳送至 `https://your-neptune-endpoint:port/loader` 端點。若要取得特定載入請求的狀態，您必須包含 `loadId` 做為 URL 參數，或將 `loadId` 連接到 URL 路徑。

Neptune 只會追蹤最近 1,024 個大量載入任務，並且只會存放每個任務的最後 10,000 個錯誤詳細資訊。

如需載入器在發生錯誤時傳回的錯誤和饋送訊息清單，請參閱 Neptune 載入器的錯誤和饋送訊息。

內容

Neptune 載入器 Get-Status 請求

載入器 Get-Status 請求語法

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

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

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

Neptune 載入器 Get-Status 請求參數

- **`loadId`** – 載入工作的 ID。如果您未指定 `loadId`，會傳回載入 ID 的清單。
- **`details`** – 除了整體狀態之外也包含詳細資訊。

  允許的值：`TRUE`、`FALSE`。

預設值：FALSE。

- **errors** – 包括錯誤清單。

  允許的值：TRUE、FALSE。

  預設值：FALSE。

  錯誤清單會分頁。page 和 errorsPerPage 參數可讓您翻閱所有錯誤頁面。

- **page** – 錯誤頁面號碼。僅適用於 errors 參數設為 TRUE 時。

  允許的值：正整數。

  預設值：1。

- **errorsPerPage** – 每個頁面的錯誤數。僅適用於 errors 參數設為 TRUE 時。

  允許的值：正整數。

  預設值：10。

- **limit** – 要列出的載入 ID 數目。僅適用於藉由傳送 GET 請求且未指定 loadId 以請求載入 ID 清單時。

  允許值：1 到 100 的正整數。

  預設值：100。

- **includeQueuedLoads** – 請求載入 ID 清單時，選用參數可用來排除佇列中載入請求的載入 ID。

  > **Note**
  >
  > 此參數從 Neptune 引擎 1.0.3.0 版開始可用。

  依預設，所有狀態為 LOAD_IN_QUEUE 的載入任務的載入 ID 都包含在這類清單中。它們會出現在其他任務的載入 ID 之前，依加入佇列的時間，從最近到最早進行排序。

  允許的值：TRUE、FALSE。

  預設值：TRUE。

Neptune 載入器 Get-Status 回應

Neptune 載入器 Get-Status 回應 JSON 配置

載入器狀態回應的一般配置如下：

```
{
    "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" : [ ]
        }
    }
}
```

Neptune 載入器 Get-Status **overallStatus** 和 **failedFeeds** 回應物件

針對每個失敗饋送傳回的可能回應 (包括錯誤描述) 與 Get-Status 回應中的 overallStatus 物件相同。

以下欄位會出現在所有載入的 overallStatus 物件中，以及每個失敗饋送的 failedFeeds 物件中。

- **fullUri** – 要載入的一個或多個檔案的 URI。

  類型：字串

  格式：s3://*bucket*/*key*。
- **runNumber** – 此載入或饋送的執行數量。它會在載入重新啟動時遞增。

  類型：unsigned long。
- **retryNumber** – 此載入或饋送的重試次數。它會在載入器自動重試饋送或載入時遞增。

  類型：unsigned long。
- **status** – 載入或饋送傳回的狀態。LOAD_COMPLETED 表示成功載入，沒有發生問題。如需其他 load-status 訊息的清單，請參閱 Neptune 載入器的錯誤和饋送訊息。

  類型：字串。
- **totalTimeSpent** – 載入或饋送時用於剖析及插入資料的時間 (以秒為單位)。這不包含擷取原始檔案清單所花費的時間。

  類型：unsigned long。
- **totalRecords** – 已載入或嘗試載入的記錄總數。

  類型：unsigned long。

  請注意，從 CSV 檔案載入時，記錄計數並不是指載入的行數，而是指這些行中個別記錄的數目。例如，採取一個像這樣的小型 CSV 檔案：

```
~id,~label,name,team
'P-1','Player','Stokes','England'
```

Neptune 會認為這個檔案包含 3 筆記錄，即：

```
P-1  label Player
P-1  name  Stokes
P-1  team  England
```

- **`totalDuplicates`** – 遇到的重複記錄數目。

  類型：unsigned long。

  與 `totalRecords` 計數的情況一樣，此值包含 CSV 檔案中個別重複記錄的數目，而不是重複行的數目。採取這個小型 CSV 檔案，例如：

```
~id,~label,name,team
P-2,Player,Kohli,India
P-2,Player,Kohli,India
```

  載入它之後傳回的狀態看起來像這樣，總共報告了 6 筆記錄，其中 3 筆是重複的：

```
{
  "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
    }
  }
}
```

對於 openCypher 載入，發生以下情況時計為重複：

- 載入器偵測到節點檔案中的資料列具有一個未含 ID 空間的 ID，其與另一個未含 ID 空間的 ID 值相同，不是在另一個資料列中，就是屬於現有節點。
- 載入器偵測到節點檔案中的資料列具有一個含 ID 空間的 ID，其與另一個含 ID 空間的 ID 值相同，不是在另一個資料列中，就是屬於現有節點。

請參閱載入 openCypher 資料的特殊考量。

- **`parsingErrors`** – 遇到的剖析錯誤數目。

  類型：unsigned long。

- **`datatypeMismatchErrors`** – 資料類型與給定資料不符的記錄數目。

  類型：unsigned long。

- **`insertErrors`** – 由於錯誤而無法插入的記錄數目。

  類型：unsigned long。

Neptune 載入器 Get-Status **`errors`** 回應物件

錯誤分為以下類別：

- **`Error 400`** – 無效的 `loadId` 會傳回 HTTP `400` 錯誤的請求錯誤。描述錯誤的訊息。
- **`Error 500`** – 無法處理的有效請求會傳回 HTTP `500` 內部伺服器錯誤。描述錯誤的訊息。

如需載入器在發生錯誤時傳回的錯誤和饋送訊息清單，請參閱 Neptune 載入器的錯誤和饋送訊息。

發生錯誤時，回應的 `BODY` 中會傳回 JSON `errors` 物件，具有下列欄位：

- **`startIndex`** – 第一個包含的錯誤的索引。

  類型：unsigned long。

- **`endIndex`** – 最後一個包含的錯誤的索引。

類型：unsigned long。

- **loadId** – 載入的 ID。您可以使用此 ID 並將 errors 參數設為 TRUE 以列印載入的錯誤。

  類型：字串。

- **errorLogs** – 錯誤清單。

  類型：清單。

Neptune 載入器 Get-Status **errorLogs** 回應物件

載入器 Get-Status 回應中 errors 下的 errorLogs 物件包含使用下列欄位描述每個錯誤的物件：

- **errorCode** – 識別錯誤的本質。

  它可以採取下列其中一個值：

  - 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 (當大量載入器無法確定錯誤類型時傳回)。

- **errorMessage** – 描述錯誤的訊息。

  這可以是與錯誤代碼相關聯的一般訊息，或包含詳細資訊的特定訊息，例如關於遺失來源/目標頂點或關於剖析錯誤。

- **fileName** – 饋送的名稱。

- **recordNum** – 在發生剖析錯誤的情況下，這是無法剖析的記錄在檔案中的記錄編號。如果記錄編號不適用於錯誤，或無法確定，則其設定為零。

例如，如果大量載入器在 RDF nquads 檔案中遇到如下的錯誤資料列，則會產生剖析錯誤：

```
<http://base#subject> |http://base#predicate> <http://base#true> .
```

如您所見，上面資料列中的第二個 http 應該在其前加上 < 而不是 |。狀態回應中 errorLogs 下產生的錯誤物件看起來像這樣：

```
{
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Expected '<', found: |",
    "fileName" : "s3://bucket/key",
    "recordNum" : 12345
},
```

Neptune 載入器 Get-Status 範例

載入狀態的請求範例

以下是使用 curl 命令透過 HTTP GET 傳送的請求。

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)'
```

Example 回應

```
{
    "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
        }
    }
}
```

載入 ID 的請求範例

以下是使用 curl 命令透過 HTTP GET 傳送的請求。

```
curl -X GET 'https://your-neptune-endpoint:port/loader?limit=3'
```

Example 回應

```
{
    "status" : "200 OK",
    "payload" : {
         "loadIds" : [
            "a2c0ce44-a44b-4517-8cd4-1dc144a8e5b5",
            "09683a01-6f37-4774-bb1b-5620d87f1931",
            "58085eb8-ceb4-4029-a3dc-3840969826b9"
        ]
    }
}
```

詳細狀態的請求範例

以下是使用 curl 命令透過 HTTP GET 傳送的請求。

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true'
```

Example 回應

```
{
    "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
        }
    }
}
```

Neptune 載入器 Get-Status **`errorLogs`** 範例

發生錯誤時的詳細狀態回應範例

這是使用 `curl` 透過 HTTP `GET` 傳送的請求：

```
curl -X GET 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3'
```

Example 發生錯誤時的詳細回應

這是您可能從上面查詢中取得的回應範例，其中 `errorLogs` 物件列出了遇到的載入錯誤：

```
{
    "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
        }
    }
}
```

Data prefetch task interrupted 錯誤的範例

有時候當您取得 LOAD_FAILED 狀態並接著請求更多詳細資訊時，這時傳回的錯誤可能為 PARSING_ERROR 並搭配 Data prefetch task interrupted 訊息，如下所示：

```
"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
    }
]
```

當資料載入作業發生通常原因不在於您的請求或資料的暫時中斷時，就會出現這個錯誤。通常只要再次執行大量上傳請求，就能解決這個問題。如果您使用的是預設設定，即 "mode":"AUTO" 和 "failOnError":"TRUE"，這時載入器會略過其已成功載入的檔案，並恢復其在中斷發生時尚未載入的檔案載入。

Neptune 載入器取消工作

取消載入工作。

若要取消工作，則必須傳送 HTTP DELETE請求至 https://*your-neptune-endpoint*:*port*/loader 端點。loadId 可以附加到 /loader URL 路徑，或包含它以做為 URL 中的變數。

取消工作請求語法

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

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

取消任務請求參數

loadId

載入工作的 ID。

取消任務回應語法

```
no response body
```

200 OK

成功刪除的載入任務會傳回 200 程式碼。

取消任務錯誤

發生錯誤時，回應的 BODY 中會傳回 JSON 物件。message 物件包含此錯誤的描述。

錯誤類別

- **Error 400** – 無效的 loadId 會傳回 HTTP 400 錯誤的請求錯誤。描述錯誤的訊息。
- **Error 500** – 無法處理的有效請求會傳回 HTTP 500 內部伺服器錯誤。描述錯誤的訊息。

取消任務錯誤訊息

以下是來自取消 API 的可能錯誤訊息及錯誤的描述。

- The load with id = *load_id* does not exist or not active (HTTP 404) – 找不到載入。檢查 id 參數值。
- Load cancellation is not permitted on a read replica instance. (HTTP 405) – 載入是一種寫入操作。在讀取/寫入叢集端點上重試載入。

取消任務範例

Example 請求

以下是使用 curl 命令透過 HTTP DELETE 傳送的請求。

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802'
```

用 AWS Database Migration Service 於從不同的資料存放區將資料載入 Amazon Neptune

AWS Database Migration Service (AWS DMS) 可以快速安全地將資料從支援的來源資料庫載入 Neptune。來源資料庫在遷移期間仍然能夠維持完全正常運作，讓倚賴它的應用程式可以將停機時間縮到最短。

您可以在AWS Database Migration Service 用戶指南和 AWS Database Migration Service API 參考 AWS DMS中找到有關的詳細信息。尤其，您可以在使用 Amazon Neptune 做為 AWS Database Migration Service的目標中了解如何將 Neptune 叢集設定為遷移目標。

以下是使用 AWS DMS將資料匯入至 Neptune 的一些先決條件：

- 您將需要創建一個 AWS DMS 表映射對象來定義應如何從源數據庫中提取數據（有關詳細信息，請參閱 AWS DMS UserGuide 中的使用 JSON 通過表映射指定表選擇和轉換）。此資料表對應組態物件指定應依何種順序讀取哪些資料表，以及如何命名資料欄。它也可以篩選所複製的資料列，並提供簡單的值轉換，如轉換成小寫或四捨五入。
- 您必須建立一個 Neptune `GraphMappingConfig`，以指定擷取自來源資料庫的資料應如何載入至 Neptune。對於 RDF 資料 (使用 SPARQL 查詢得來)，`GraphMappingConfig` 是採用 W3 的標準 R2RML 對應語言編寫。對於屬性圖資料 (使用 Gremlin 查詢得來)，`GraphMappingConfig` 是 GraphMappingConfig 屬性圖形/格林資料的佈局 中所述的 JSON 物件。
- 您必須使 AWS DMS 用在與 Neptune 資料庫叢集相同的 VPC 中建立複寫執行個體，以調解資料的傳輸。
- 您還需要一個 Amazon S3 儲存貯體做為暫存遷移資料的中繼儲存體。

建立 Neptune GraphMappingConfig

您建立的 `GraphMappingConfig` 指定應如何將擷取自來源資料存放區的資料載入至 Neptune 資料庫叢集。它的格式取決於它是用於載入 RDF 資料還是用於載入屬性圖資料。

若是 RDF 資料，您可以使用 W3 R2RML 語言將關聯式資料對應到 RDF。

如果您載入的是即將使用 Gremlin 查詢的屬性圖形資料，請為 `GraphMappingConfig` 建立 JSON 物件。

GraphMappingConfig RDF/SPARQL 資料的配置

如果您是載入要使用 SPARQL 查詢的 RDF 資料，則可以使用 R2RML 撰寫 GraphMappingConfig。R2RML 是用於將關聯式資料映射到 RDF 的標準 W3 語言。以下是一個範例：

```
@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" ];
    ] .
```

以下是另一個範例：

```
@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" ]
    ] .
```

R2RML: RDB to RDF Mapping Language 的 W3 建議中提供了語言的詳細資訊。

GraphMappingConfig 屬性圖形/格林資料的佈局

適用於屬性圖形資料的比較 GraphMappingConfig 是一個 JSON 物件，為即將根據來源資料產生的每個圖形實體提供對應規則。下列範本顯示此物件中每個規則的外觀：

```
{
  "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)"
            }
          ]
        }
      ]
    }
  ]
}
```

請注意，出現頂點標籤表示頂點是在此處建立，沒有頂點標籤則表示頂點是由不同來源建立的，而這項定義只是新增頂點屬性。

以下是員工記錄的規則範例：

```
{
  "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"
            }
          ]
        }
      ]
    }
  ]
}
```

以 Neptune 為目標建立 AWS DMS 複寫工作

一旦建立了資料表映射和圖形映射組態，就會使用下列程序，將資料從來源存放區載入至 Neptune。如需有關相關 AWS DMS API 的更多詳細資訊，請參閱文件。

步驟 1：建立 AWS DMS 複製執行個體

在執行 Neptune DB 叢集的 VPC 中建立 AWS DMS 複寫執行個體 (請參閱使 AWS DMS 用者指南中的使用 AWS DMS 複寫執行個CreateReplication體和執行個體)。您可以使用類似下面的 AWS CLI 命令來執行此操作：

```
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)
```

步驟 2. 建立來源資料庫的 AWS DMS 端點

下一步是為來源資料存放區建立 AWS DMS 端點。你可以 AWS CLI 像這樣使用 AWS DMS CreateEndpointAPI：

```
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)
```

步驟 3。為 Neptune 設定要用於暫存資料的 Amazon S3 儲存貯體

如果您沒有可用於暫存資料的 Amazon S3 儲存貯體，請按照《 Amazon S3 入門指南》中的建立儲存貯體，或《主控台使用者指南》中的如何建立 S3 儲存貯體？所述建立儲存貯體。

如果您還沒有 IAM 政策，您必須建立一個 IAM 政策，將 `GetObject`、`PutObject`、`DeleteObject` 和 `ListObject` 許可授與儲存貯體：

```
{
  "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)/*"
      ]
    }
  ]
}
```

如果您的 Neptune 資料庫叢集已啟用 IAM 身分驗證，您還必須包含下列政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "(the ARN of your Neptune DB cluster resource)"
    }
  ]
}
```

建立 IAM 角色做為要附加政策的信任文件：

```
{
  "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"
    }
  ]
}
```

在將政策附加到角色之後，將此角色附加到您的 Neptune 資料庫叢集。這將允許使 AWS DMS 用存儲桶來暫存正在加載的數據。

步驟 4. 在 Amazon VPC 中建立 Amazon S3 端點

現在，在 Neptune 叢集所在的 VPC 中為您的中繼 Amazon S3 儲存貯體建立一個 VPC 閘道端點。您可以使用 AWS Management Console 或 AWS CLI 來執行此操作，如建立閘道端點中所述。

步驟 5. 建立 Neptune 的 AWS DMS 目標端點

為目標 Neptune 資料庫叢集建立 AWS DMS 端點。您可以將 AWS DMS CreateEndpointAPI 與`NeptuneSettings`參數一起使用，如下所示：

```
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) }'
```

傳遞至其NeptuneSettings參數中 AWS DMS CreateEndpoint API 的 JSON 物件具有下列欄位：

- **ServiceAccessRoleArn** – (必要) IAM 角色的 ARN，此角色允許精細存取 S3 儲存貯體，此儲存貯體用來暫存遷移至 Neptune 的資料。此角色也應具有存取 Neptune 資料庫叢集的許可 (如果該叢集已啟用 IAM 授權)。
- **S3BucketName** – (必要) 若是完整載入遷移，複寫執行個體會將所有 RDS 資料轉換為 CSV、四元組檔案，並將它們上傳到 S3 中的這個暫存儲存貯體，然後再大量載入至 Neptune。
- **S3BucketFolder** – (必要) 要在 S3 暫存儲存貯體中使用的資料夾。
- **ErrorRetryDuration** – (選用) Neptune 請求失敗後，再次提出重試請求之前要等待的毫秒數。預設值為 250。
- **MaxRetryCount**— (選用) 可重試失敗後， AWS DMS 應發出的重試要求數目上限。預設值為 5。
- **MaxFileSize** – (選用) 遷移期間儲存至 S3 的每個暫存檔案的大小上限 (以位元組為單位)。預設值為 1,048,576 KB (1 GB)。
- **IsIAMAuthEnabled** – 選用) 如果已在 Neptune 資料庫叢集上啟用 IAM 驗證，則設為 `true`，否則設為 `false`。預設值為 `false`。

步驟 6. 測試與新端點的連線

您可以使用 AWS DMS TestConnectionAPI 測試與每個新端點的連接，如下所示：

```
aws dms test-connection \
    --replication-instance-arn (the ARN of the replication instance) \
    --endpoint-arn (the ARN of the endpoint you are testing)
```

步驟 7. 建立 AWS DMS 複製任務

成功完成上述步驟後，請使用如下所示的工作 API，建立將來源資料存放區中的資料移轉至 Neptune 的複寫 AWS DMS CreateReplication工作：

```
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')
```

TaskData 參數提供 GraphMappingConfig，其會指定所複製資料應該如何儲存在 Neptune 中。

步驟 8. 啟動 AWS DMS 複製任務

現在您可以啟動複寫任務：

```
aws dms start-replication-task
    --replication-task-arn (ARN of the replication task started in the previous step)
    --start-replication-task-type start-replication
```

查詢 Neptune 圖形

Neptune 支援下列圖形查詢語言來存取圖形：

- 小鬼，由 Apache 定義用 TinkerPop於創建和查詢屬性圖。

  Grimlin 中的查詢是由離散步驟組成的周遊，每個步驟都沿著一個邊緣到一個節點。

  請參閱 使用 Gremlin 存取 Neptune 圖形 以了解如何在 Neptune 中使用 Grimlin，以及參閱 Amazon Neptune 中的 Gremlin 標準合規，找出有關 Neptune 實作 Gremlin 的具體詳細資訊。
- openCypher 是屬性圖的宣告式查詢語言，最初由 Neo4j 開發，然後在 2015 年成為開放原始碼，並在 Apache 2 開放原始碼授權下投入 OpenCypher 專案。它的語法記載於 openCypher 規格中。
- SPARQL 是一種基於圖形模式比對的宣告式語言，用於查詢 RDF 資料。它受到全球資訊網協會支援。

  請參閱 使用 SPARQL 存取 Neptune 圖形 以了解如何在 Neptune 中使用 SPARQL，以及參閱 Amazon Neptune 中的 SPARQL 標準合規，找出有關 Neptune 實作 SPARQL 的具體詳細資訊。

> **Note**
>
> Grimlin 和 OpenCypher 都可以用來查詢 Neptune 中儲存的任何屬性圖資料，而不管其載入方式。

主題

Amazon Neptune 中的查詢佇列

開發和調校圖形應用程式時，了解資料庫將查詢排入佇列的方式會很有幫助。在 Amazon Neptune 中，查詢佇列發生方式如下：

- 無論執行個體大小為何，每個執行個體可以在佇列中等候的查詢數目上限為 8,192 個。超過該數目的任何查詢都會被拒絕，並且會失敗而產生 `ThrottlingException`。
- 一次可執行的查詢數目上限取決於指派的工作者執行緒數目，通常該數目設定為可用虛擬 CPU 核心 (vCPU) 數目的兩倍。
- 查詢延遲包含查詢花費在佇列中的時間，以及網路往返時間和實際執行時間。

判定佇列在指定時刻有多少個查詢

此`MainRequestQueuePendingRequests` CloudWatch 測量結果會以五分鐘的精細度記錄輸入佇列中等待的要求數目 (請參閱Neptune CloudWatch 度量)。

若是 Gremlin，您可以使用 Gremlin 查詢狀態 API 傳回的 `acceptedQueryCount` 值來取得佇列中目前的查詢計數。但請注意，SPARQL 查詢狀態 API 傳回的 `acceptedQueryCount` 值包含從伺服器啟動以來接受的所有查詢，包括完成的查詢。

查詢佇列如何影響逾時時間

如上所述，查詢延遲包含查詢花費在佇列中的時間，以及執行所花費的時間。

由於查詢的逾時期間通常是從查詢進入佇列起算，因此移動緩慢的佇列可能會使許多查詢一移出佇列就立即逾時。這個情況顯然不好，所以應避免將大量查詢排入佇列，除非它們可以快速執行。

使用 Gremlin 存取 Neptune 圖形

Amazon Neptune 與阿帕奇 TinkerPop 3 和小鬼兼容。這表示您可以連接到 Neptune 資料庫執行個體，並使用 Gremlin 遍歷語言來查詢圖形 (請參閱 Apache TinkerPop 3 文件中的圖形)。如需 Neptune 實作 Gremlin 時的差異，請參閱 Gremlin 標準合規。

不同的 Neptune 引擎版本支援不同的 Gremlin 版本。檢查您正在執行的 Neptune 版本的引擎版本頁面，以確定它支援哪個 Grinlin 版本。

Gremlin 中的「周遊」是一系列的鏈結步驟。它從頂點 (或邊緣) 開始。它先沿著每個頂點的外緣，再沿著這些頂點的外緣導出圖形。每個步驟皆是周遊的操作。如需詳細資訊，請參閱 TinkerPop 3 文件中的遍歷。

有 Gremlin 語言變體和各種程式設計語言中的 Gremlin 存取支援。如需詳細資訊，請參閱 TinkerPop 3 文件中的關於 Gremlin 語言變體。

本文件描述如何使用下列變體和程式設計語言存取 Neptune。

如 傳輸中加密：使用 SSL/HTTPS 連線至 Neptune 中所述，當連線至 Neptune 時，您必須在所有的 AWS 區域中使用 Transport Layer Security/Secure Sockets Layer (TLS/SSL)。

Gremlin-Groovy

本節中的 Gremlin 主控台和 HTTP REST 範例將使用 Gremlin-Groovy 變體。如需 Gremlin 主控台和 Amazon Neptune 的詳細資訊，請參閱《快速入門》的 the section called “使用 Gremlin” 一節。

Gremlin-Java

Java 示例是使用官方的 TinkerPop 3 Java 實現編寫的，並使用小鬼 Java 變體。

Gremlin-Python

Python 示例是使用官方 TinkerPop 3 Python 實現編寫的，並使用小鬼 Python 變體。

以下各節將逐步引導您使用 Gremlin 主控台、REST over HTTPS 和各種程式設計語言連線到 Neptune 資料庫執行個體。

開始之前，您必須準備好以下事項：

- Neptune 資料庫執行個體。如需建立 Neptune 資料庫執行個體的相關資訊，請參閱 建立新的 Neptune 資料庫叢集。
- 與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體。

如需有關將資料載入至 Neptune 的詳細資訊，包括先決條件、載入格式及載入參數，請參閱 將資料載入至 Amazon Neptune。

主題

- 設定 Gremlin 主控台來連線至 Neptune 資料庫執行個體
- 使用 HTTP REST 端點連線至 Neptune 資料庫執行個體
- 要與 Amazon Neptune 搭配使用的 Java 型 Gremlin 用戶端
- 使用 Python 連線至 Neptune 資料庫執行個體
- 使用 .NET 連線至 Neptune 資料庫執行個體
- 使用 Node.js 連線至 Neptune 資料庫執行個體
- 使用 Go 連線至 Neptune 資料庫執行個體
- Gremlin 查詢提示
- Gremlin 查詢狀態 API

- Gremlin 查詢取消
- 支援 Gremlin 指令碼型工作階段
- Neptune 中的 Gremlin 交易
- 搭配 Amazon Neptune 使用 Gremlin API
- 在 Amazon Neptune Gremlin 中快取查詢結果
- 使用 Gremlin mergeV() 和 mergeE() 步驟進行有效的 upsert
- 使用 fold()/coalesce()/unfold() 進行有效的 Gremlin upsert
- 使用 Gremlin explain 分析 Neptune 查詢執行
- 搭配 Neptune DFE 查詢引擎使用 Gremlin

設定 Gremlin 主控台來連線至 Neptune 資料庫執行個體

Grinlin 控制台允許您在 REPL (read-eval-print 循環) 環境中嘗試 TinkerPop 圖形和查詢。

安裝 Gimlin 主控台並以尋常的方式連線至其中

您可以使用 Gremlin 主控台連接到遠端圖形資料庫。下節會引導您安裝與設定 Gremlin 主控台，從遠端連線至 Neptune 資料庫執行個體。您必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

如需使用 SSL/TLS (這是必要的) 連線至 Neptune 的說明，請參閱 SSL/TLS 組態。

> **Note**
>
> 如果您已在 Neptune 資料庫叢集上啟用 IAM 身分驗證，請遵循 搭配 Signature 第 4 版簽署，使用 Gremlin 主控台連線到 Neptune 中的指示來安裝 Gemlin 主控台，而不是此處的指示。

安裝 Gremlin 主控台並連線至 Neptune

1. Gremlin 主控台二進位檔需要 Java 8 或 Java 11。這些指示假設使用 Java 11。您可以在 EC2 執行個體上安裝 Java 11，如下所示：

   - 如果您使用的是 Amazon Linux 2 (AL2)：

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - 如果您使用的是 Amazon Linux 2023 (AL2023)：

```
sudo yum install java-11-amazon-corretto-devel
```

- 對於其他發行版，請使用以下任一適當的方式：

```
sudo yum install java-11-openjdk-devel
```

或：

```
sudo apt-get install openjdk-11-jdk
```

2. 輸入以下命令將 Java 11 設定為 EC2 執行個體上預設的執行期。

```
sudo /usr/sbin/alternatives --config java
```

出現提示時，輸入 Java 11 的編號。

3. 從 Apache 網站下載適當版本的 Gremlin 主控台。您可以檢查您目前正在執行之 Neptune 版本的引擎版本頁面，以確定它支援哪個 Grinlin 版本。例如，對於 3.6.5 版，您可以從 Apache Tinkerpop3 網站將 Gremlin主控台下載到您的 EC2 執行個體，如下所示：

```
wget https://archive.apache.org/dist/tinkerpop/3.6.5/apache-tinkerpop-gremlin-
console-3.6.5-bin.zip
```

4. 解壓縮 Gremlin 主控台 zip 檔。

```
unzip apache-tinkerpop-gremlin-console-3.6.5-bin.zip
```

5. 將目錄變更為解壓縮的目錄。

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

6. 在解壓縮目錄的 `conf` 根資料夾中，使用下列文字建立名為 `neptune-remote.yaml` 的檔案。將 *`your-neptune-endpoint`* 取代為 Neptune 資料庫執行個體的主機名稱或 IP 地址。需使用方括號 (`[ ]`)。

> **Note**
>
> 如需尋找 Neptune 資料庫執行個體主機名稱的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

```
hosts: [your-neptune-endpoint]
port: 8182
connectionPool: { enableSsl: true }
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: true }}
```

> **Note**
>
> 序列化程式已從gremlin-driver模組移至 3.7.0 版中的新gremlin-util模組。該軟件包從組織改變了修補程序。

7. 在終端機上，導覽至 Gremlin 主控台目錄 (apache-tinkerpop-gremlin-console-3.6.5)，然後輸入下列命令來執行 Gremlin 主控台。

```
bin/gremlin.sh
```

您應該會看到下列輸出：

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

您現在進入 gremlin> 提示。您將在這個提示下輸入其餘的步驟。

8. 在 gremlin> 提示下，輸入以下命令以連線到 Neptune 資料庫執行個體。

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

9. 在 gremlin> 提示下，輸入以下內容以切換為遠端模式。這會傳送所有 Gremlin 查詢到遠端連線。

```
:remote console
```

10. 輸入以下內容以將查詢傳送到 Gremlin 圖形。

```
g.V().limit(1)
```

11. 完成後，輸入以下內容以退出 Gremlin 主控台。

```
:exit
```

> **Note**
>
> 使用分號 (`;`) 或換行符號字元 (\n) 來分隔每個陳述式。
> 最終周遊之前的每個周遊節尾必須為 `next()`，才能執行。但只有最後的周遊資料會傳回。

如需 Neptune 實作 Gremlin 的詳細資訊，請參閱 the section called “Gremlin 標準合規”。

連線至 Gremlin 主控台的替代方式

正常連線方法的缺點

連線到 Gemlin 主控台的最常用方法是上面說明的方法，在 `gremlin>` 提示處使用如下的命令：

```
gremlin> :remote connect tinkerpop.server conf/(file name).yaml
gremlin> :remote console
```

其效果很好，並且可讓您將查詢傳送至 Neptune。不過，它需要將 Groovy 指令碼引擎帶出迴圈，因此 Neptune 會將所有查詢視為純粹的 Grimlin。這表示下列查詢形式會失敗：

```
gremlin> 1 + 1
gremlin> x = g.V().count()
```

以這種方式連線時，最接近使用變數的方法是使用主控台維護的 `result` 變數，並使用 `:>` 傳送查詢，如下所示：

```
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]
```

不同的連線方式

您也可以透過不同的方式連線到 Gemlin 主控台，從而可能發現更好的方式，如下所示：

```
gremlin> g = traversal().withRemote('conf/neptune.properties')
```

這裡的 `neptune.properties` 會採取這種形式：

```
gremlin.remote.remoteConnectionClass=org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteCon
gremlin.remote.driver.clusterFile=conf/my-cluster.yaml
gremlin.remote.driver.sourceName=g
```

`my-cluster.yaml` 檔案應該看起來像這樣：

```
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**
>
> 序列化程式已從`gremlin-driver`模組移至 3.7.0 版中的新`gremlin-util`模組。該軟件包從組織改變了修補程序。

像這樣設定 Gimlin 主控台連線可讓您成功地進行以下類型的查詢：

```
gremlin> 1+1
==>2

gremlin> x=g.V().count().next()
==>4249

gremlin> println("The answer was ${x}")
The answer was 4249
```

您可以避免顯示結果，如下所示：

```
gremlin> x=g.V().count().next();[]
gremlin> println(x)
4249
```

所有尋常的查詢方式 (沒有終端步驟) 會繼續運作。例如：

```
gremlin> g.V().count()
==>4249
```

您甚至可以使用 `g.io().read()` 步驟，搭配這種類型的連線載入檔案。

使用 HTTP REST 端點連線至 Neptune 資料庫執行個體

Amazon Neptune 會提供 HTTP 端點進行 Gremlin 查詢。REST 介面與您的資料庫叢集正在使用的任何 Gremlin 版本相容 (請參閱您正在使用之 Neptune 引擎版本的引擎版本頁面，以確定其支援哪個 Gemlin 版本)。

> **Note**
>
> 如 傳輸中加密：使用 SSL/HTTPS 連線至 Neptune 中所討論，Neptune 現在要求您使用 HTTPS 而不是 HTTP 進行連線。

以下說明引導您使用 `curl` 命令和 HTTPS 連接到 Gremlin 端點。您必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

對 Neptune 資料庫執行個體進行 Gremlin 查詢時所用的 HTTPS 端點為 `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`。

> **Note**
>
> 如需尋找 Neptune 資料庫執行個體主機名稱的相關資訊，請參閱 連線至 Amazon Neptune 端點。

使用 HTTP REST 端點連線至 Neptune

以下範例使用 curl 透過 HTTP POST 提交 Gremlin 查詢。查詢是以 POST 本文中的 JSON 格式做為 `gremlin` 屬性提交。

```
curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://your-neptune-endpoint:port/
gremlin
```

此範例會使用 `g.V().limit(1)` 周遊傳回圖形中的第一個頂點。您可以查詢其他項目，方法是將其取代為另一個 Gremlin 周遊。

> **Important**
>
> 根據預設，REST 端點會以單一 JSON 結果集傳回所有結果。如果此結果集太大，Neptune 資料庫執行個體上可能會發生 `OutOfMemoryError` 例外狀況。
> 您可以啟用區塊回應 (以一系列個別回應傳回的結果) 來避免這種情況。請參閱使用選用的 HTTP 結尾標頭來啟用多部分 Gremlin 回應。

雖然建議使用 HTTP POST 請求來傳送 Gremlin 查詢，但也可以使用 HTTP GET 請求：

```
curl -G "https://your-neptune-endpoint:port?gremlin=g.V().count()"
```

> **Note**
>
> Neptune 不支援 `bindings` 屬性。

使用選用的 HTTP 結尾標頭來啟用多部分 Gremlin 回應

根據預設，對 Gremlin 查詢的 HTTP 回應會以單一 JSON 結果集傳回。如果結果集非常大，這可能會導致資料庫執行個體上發生 `OutOfMemoryError` 例外狀況。

不過，您可以啟用「區塊」回應 (以多個個別部分傳回的回應)。您可以在請求中包含轉移編碼 (TE) 尾端標頭 (`te: trailers`) 來執行此操作。如需 TE 標頭的詳細資訊，請參閱有關 TE 請求標頭的 MDN 頁面。

當回應以多個部分傳回時，很難診斷在收到第一個部分之後發生的問題，因為第一部分送達時，HTTP 狀態碼為 200 (OK)。後續失敗通常會產生包含損毀回應的訊息內文，而 Neptune 會在此內文結尾處附加錯誤訊息。

為了更輕鬆地偵測和診斷此類失敗，Neptune 還會在每個回應區塊的結尾標頭內包含兩個新的標頭欄位：

- `X-Neptune-Status` – 包含回應碼，後面接著簡短名稱。例如，若成功，結尾標頭將是：`X-Neptune-Status: 200 OK`。若失敗，回應代碼將是其中一個 Neptune 引擎錯誤代碼，例如 `X-Neptune-Status: 500 TimeLimitExceededException`。
- `X-Neptune-Detail` – 對於成功的請求而言是空的。若發生錯誤，它會包含 JSON 錯誤訊息。由於 HTTP 標頭值中只允許 ASCII 字元，因此 JSON 字串會進行 URL 編碼。

> **Note**
>
> Neptune 目前不支援對區塊回應進行 `gzip` 壓縮。如果用戶端同時要求區塊編碼和壓縮，Neptune 會略過壓縮。

要與 Amazon Neptune 搭配使用的 Java 型 Gremlin 用戶端

您可以使用兩個開放原始碼 Java 的 Grimlin 用戶端中的任何一個搭配 Amazon Neptune：Apache TinkerPop Java Grimlin 用戶端或 Amazon Neptune 的 Grimlin 用戶端。

阿帕奇 TinkerPop Java 小鬼客戶端

如果可以的話，請始終使用引擎版本支持的最新版本的 Apache TinkerPop Java Grinlin 客戶端。較新的版本包含許多錯誤修正，其可以提高用戶端的穩定性、效能和可用性。

下表列出不同 Neptune 引擎版本所支援的最早和最新版本的 TinkerPop 用戶端：

Neptune 引擎版本	最低 TinkerPop 版本	最大 TinkerPop 版本
`1.3.2.0`	`3.6.2`	`3.7.1`

Neptune 引擎版本	最低 TinkerPop 版本	最大 TinkerPop 版本
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 和更舊版本	(已棄用)	(已棄用)

TinkerPop 客戶端通常在一個系列中向後兼容（3.3.x例如，或3.4.x）。在特殊情況下，向後相容性必須中斷，因此最好在TinkerPop升級到新的用戶端版本之前檢查升級建議。

用戶端可能無法使用伺服器所支援版本以後的版本中引進的新步驟或新功能，但除非升級建議指出重大變更，否則您可以預期現有的查詢和功能可以運作。

> **Note**
>
> 從 Neptune 引擎 TinkerPop 版本 1.1.1.0 開始，請勿使用低於. 3.5.2
> Python 使用者應避免使用 TinkerPop 版本，3.4.9因為需要直接設定的預設逾時設定 (請參閱 TINKERPOP-2505)。

適用於 Amazon Neptune 的 Gremlin Java 用戶端

Amazon Neptune 的 Gremlin 客戶端是一個基於 Java 的開源 Gemlin 客戶端，可作為標準 Java 客戶端的插入替代品。 TinkerPop

Neptune Gimlin 用戶端已針對 Neptune 叢集進行了最佳化。它可讓您管理叢集中多個執行個體的流量分佈，並在您新增或移除複本時因應叢集拓撲中的變更。您甚至可以設定用戶端，根據角色、執行個體類型、可用區域 (AZ) 或與執行個體相關聯的標籤，跨叢集中的執行個體子集分配請求。

Neptune Gremlin Java 用戶端的最新版本可在 Maven Central 中取得。

如需有關 Neptune Gremlin Java 用戶端的詳細資訊，請參閱此部落格文章。有關代碼示例和演示，請查看客戶的 GitHub 項目。

使用 Java 連線至 Neptune 資料庫執行個體

以下章節將逐步引導您完成連線到 Neptune 資料庫執行個體並使用 Apache Grimlin 用戶端執行 Gemlin 周遊的完整 Java 範例的執行過程。 TinkerPop

必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

使用 Java 連線至 Neptune

1. 在 EC2 執行個體上安裝 Apache Maven。首先，輸入以下內容，以新增包含 Maven 套件的儲存庫：

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   輸入以下內容以設定套件的版本編號：

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   接著使用 yum 安裝 Maven：

   ```
   sudo yum install -y apache-maven
   ```

2. 安裝 Java。Gremlin 程式庫需要 Java 8 或 11。您可以安裝 Java 11，如下所示：

   - 如果您使用的是 Amazon Linux 2 (AL2)：

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - 如果您使用的是 Amazon Linux 2023 (AL2023)：

```
sudo yum install java-11-amazon-corretto-devel
```

- 對於其他發行版，請使用以下任一適當的方式：

```
sudo yum install java-11-openjdk-devel
```

或：

```
sudo apt-get install openjdk-11-jdk
```

3. 將 Java 11 設定為 EC2 執行個體上的預設執行期：輸入以下命令，將 Java 8 設定為 EC2 執行個體上的預設執行期：

```
sudo /usr/sbin/alternatives --config java
```

出現提示時，輸入 Java 11 的編號。

4. 建立名為 **gremlinjava** 的新目錄：

```
mkdir gremlinjava
cd gremlinjava
```

5. 在 gremlinjava 目錄中，建立一個 pom.xml 檔案，然後在文字編輯器中開啟檔案：

```
nano pom.xml
```

6. 將以下內容 pom.xml 檔案，然後儲存檔案：

```
<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**
>
> 如果您要修改現有的 Maven 專案，所需的相依性將在先前的程式碼中反白顯示。

7. 若要為原始程式碼範例 (`src/main/java/com/amazonaws/`) 建立子目錄，在命令列輸入下列命令：

```
mkdir -p src/main/java/com/amazonaws/
```

8. 在 `src/main/java/com/amazonaws/` 目錄中，建立一個名為 `App.java` 的檔案，然後在文字編輯器中開啟檔案。

```
nano src/main/java/com/amazonaws/App.java
```

9. 將以下內容複製到 `App.java` 檔案。將 *`your-neptune-endpoint`* 取代為 Neptune 資料庫執行個體的地址。不要在 `addContactPoint` 方法中包括 `https://` 字首。

> **Note**
>
> 如需尋找 Neptune 資料庫執行個體主機名稱的相關資訊，請參閱 連線至 Amazon Neptune 端點。

```
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();
  }
}
```

如需使用 SSL/TLS (這是必要的) 連線至 Neptune 的說明，請參閱 SSL/TLS 組態。

10. 使用下列 Maven 命令編譯並執行範例：

```
mvn compile exec:exec
```

上述範例使用 `g.V().limit(3).elementMap()` 周遊傳回圖形中前兩個頂點的金鑰和每個屬性值的對應。若要查詢其他內容，將其換成其他使用其中一個適當之結束方法的 Gremlin 周遊。

> **Note**
>
> Gremlin 查詢最後的部分 `.toList()` 用來提交周遊至伺服器，以供進行評估。如果您未包含該方法或其他同等方法，該查詢不會提交到 Neptune 資料庫執行個體。
> 您也必須在新增頂點或邊緣時附加適當結尾，例如當您使用 `addV( )` 步驟時。

以下方法會查詢提交至 Neptune 資料庫執行個體：

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

Gremlin Java 用戶端的 SSL/TLS 組態

Neptune 需要預設啟用 SSL/TLS。一般而言，如果 Java 驅動程式是使用 `enableSsl(true)` 設定的，它可以連線至 Neptune，而不必設定 `trustStore()` 或 `keyStore()`，其中具有憑證的本機副本。早期版本的 TinkerPop 鼓勵使用`keyCertChainFile()`來配置本地存儲的`.pem`文件，但該文件已被棄用，並且在 3.5.x 之後不再可用。如果您是使用該設定，搭配使用 `SFSRootCAG2.pem` 的公有憑證，您現在可以移除本機副本。

不過，如果您連線的執行個體沒有透過其驗證公有憑證的網際網路連線，或者如果您使用的憑證不是公有的，則您可以採取下列步驟來設定本機憑證副本：

設定本機憑證副本以啟用 SSL/TLS

1. 從 Oracle 下載並安裝 keytool。這將使設定本機金鑰存放區更加容易。
2. 下載 `SFSRootCAG2.pem` CA 憑證 (Gremlin Java SDK 需要憑證來驗證遠端憑證)。

```
wget https://www.amazontrust.com/repository/SFSRootCAG2.pem
```

3. 以 JKS 或 PKCS12 格式建立金鑰存放區。此範例使用 JKS。根據提示回答後面的問題。稍後將需要您在此處建立的密碼：

```
keytool -genkey -alias (host name) -keyalg RSA -keystore server.jks
```

4. 將您下載的 SFSRootCAG2.pem 檔案匯入至新建立的金鑰存放區：

```
keytool -import -keystore server.jks -file .pem
```

5. 以程式設計方式設定 Cluster 物件：

```
Cluster cluster = Cluster.build("(your neptune endpoint)")
                         .port(8182)
                         .enableSSL(true)
                         .keyStore('server.jks')
                         .keyStorePassword("(the password from step 2)")
                         .create();
```

如果想要的話，您可以在組態檔案中執行相同的操作，就像您使用 Gemlin 主控台所做一樣：

```
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 }}
```

使用重新連線邏輯來連線至 Neptune 資料庫執行個體的 Java 範例

以下 Java 範例示範如何使用重新連線邏輯來連線至 Gemlin 用戶端，以從非預期的中斷連線回復。

它具有下列相依性：

```
<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>
```

以下是範例程式碼：

```
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;
  };
}
```

使用 Python 連線至 Neptune 資料庫執行個體

如果可以的話，請始終使用您的引擎版本支持的最新版本的 Apache TinkerPop Python 小鬼客戶端。較新的版本包含許多錯誤修正，其可以改善用戶端的穩定性、效能和可用性。要使用的gremlinpython版 TinkerPop本通常會與 Java Gemlin 客戶端的表格中描述的版本保持一致。

> **Note**
>
> 只要您在編寫的小鬼查詢中僅使用 TinkerPop 3.4.x 功能，3. gremlinpython 5.x 版本與 3.4.x 版本兼容。

下節引導您逐步執行 Python 範例，其會連線至 Amazon Neptune 資料庫執行個體，並執行 Gremlin 周遊。

您必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

開始之前，請執行以下動作：

- 從 Python.org 網站下載並安裝 Python 3.6 或更新版本。
- 確認已安裝 pip。如果您沒有 pip 或不確定，請參閱 pip 文件中的我是否需要安裝 pip？。
- 如果 Python 安裝未提供 pip，請如下下載 `futures`：`pip install futures`

使用 Python 連線至 Neptune

1. 輸入以下內容以安裝 `gremlinpython` 套件：

   ```
   pip install --user gremlinpython
   ```

2. 建立名為 `gremlinexample.py` 的檔案，並在文字編輯器中開啟。

3. 將以下內容複製到 `gremlinexample.py` 檔案。將 *`your-neptune-endpoint`* 取代為 Neptune 資料庫執行個體的地址。

   如需尋找 Neptune 資料庫執行個體地址的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

```
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. 輸入下列命令以執行範例：

```
python gremlinexample.py
```

   本範例結尾的 Gremlin 查詢將以清單傳回頂點 (`g.V().limit(2)`)。接著此清單將以標準 Python `print` 函數列印。

> **Note**
>
> Gremlin 查詢最後的部分 `toList()` 用來提交周遊至伺服器，以供進行評估。如果您未包含該方法或其他同等方法，該查詢不會提交到 Neptune 資料庫執行個體。

   以下方法會查詢提交至 Neptune 資料庫執行個體：

- `toList()`
- `toSet()`
- `next()`

- `nextTraverser()`
- `iterate()`

上述範例使用 `g.V().limit(2).toList()` 周遊傳回圖形中的前兩個頂點。若要查詢其他內容，將其換成其他使用其中一個適當之結束方法的 Gremlin 周遊。

使用 .NET 連線至 Neptune 資料庫執行個體

如果可以的話，請始終使用您的引擎版本支持的最新版本的 Apache TinkerPop NET Grim lin 客戶端。較新的版本包含許多錯誤修正，其可以改善用戶端的穩定性、效能和可用性。要使用的`Gremlin.Net`版 TinkerPop本通常會與 Java Gemlin 客戶端的表格中描述的版本保持一致。

下節包含以 C# 撰寫的程式碼範例，其會連線至 Neptune 資料庫執行個體並執行 Gremlin 周遊。

必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體連線至 Amazon Neptune。此範本程式碼已在執行 Ubuntu 的 Amazon EC2 執行個體上測試。

開始之前，請執行以下動作：

- 在 Amazon EC2 執行個體上安裝 .NET。如需在包括 Windows、Linux 和 macOS 等多重作業系統上安裝 .NET 的指示，請參閱開始使用 .NET。
- 請執行套件的 `dotnet add package gremlin.net`以安裝 Gremlin.NET。如需詳細資訊，請參閱說明文件中的 Gemlin.net。 TinkerPop

使用 Gremlin.NET 連線至 Neptune

1. 建立新的 .NET 專案。

```
dotnet new console -o gremlinExample
```

2. 將目錄變更為新的專案目錄。

```
cd gremlinExample
```

3. 將以下內容複製到 `Program.cs` 檔案。將 *`your-neptune-endpoint`* 取代為 Neptune 資料庫執行個體的地址。

如需尋找 Neptune 資料庫執行個體地址的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

```
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. 輸入下列命令以執行範例：

```
dotnet run
```

本範例結尾的 Gremlin 查詢會傳回單一頂點的數量，以進行測試。接著會列印至主控台。

> **Note**
>
> Gremlin 查詢最後的部分 `Next()` 用來提交周遊至伺服器，以供進行評估。如果您未包含該方法或其他同等方法，該查詢不會提交到 Neptune 資料庫執行個體。

以下方法會查詢提交至 Neptune 資料庫執行個體：

- `ToList()`
- `ToSet()`
- `Next()`
- `NextTraverser()`
- `Iterate()`

如果您需要序列化並傳回查詢結果，請使用 `Next()`，或者如果不需要，則使用 `Iterate()`。

上述範例會使用 `g.V().Limit(3).ToList()` 周遊傳回清單。若要查詢其他內容，將其換成其他使用其中一個適當之結束方法的 Gremlin 周遊。

使用 Node.js 連線至 Neptune 資料庫執行個體

如果可以的話，請始終使用引擎版本支持的最新版本的 Apache Gri TinkerPop JavaScript mlin 客戶端 gremlin。較新的版本包含許多錯誤修正，其可以改善用戶端的穩定性、效能和可用性。`gremlin`要使用的版 TinkerPop 本通常會與 Java Gemlin 客戶端的表格中描述的版本保持一致。

下節引導您逐步執行 Node.js 範例，其會連線至 Amazon Neptune 資料庫執行個體，並執行 Gremlin 周遊。

您必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

開始之前，請執行以下動作：

- 確認已安裝 Node.js 8.11 或更高版本。如果沒有，請從 Nodejs.org 網站下載並安裝 Node.js。

使用 Node.js 連線至 Neptune

1. 輸入以下內容以安裝 `gremlin-javascript` 套件：

```
npm install gremlin
```

2. 建立名為 `gremlinexample.js` 的檔案，並在文字編輯器中開啟。
3. 將以下內容複製到 `gremlinexample.js` 檔案。將 *`your-neptune-endpoint`* 取代為 Neptune 資料庫執行個體的地址。

   如需尋找 Neptune 資料庫執行個體地址的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

```
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. 輸入下列命令以執行範例：

```
node gremlinexample.js
```

前面的範例回傳使用 `g.V().limit(1).count().next()`周遊回傳圖表中的單一頂點計數。若要查詢其他內容，將其換成其他使用其中一個適當之結束方法的 Gremlin 周遊。

> **Note**
>
> Gremlin 查詢最後的部分 `next()` 用來提交周遊至伺服器，以供進行評估。如果您未包含該方法或其他同等方法，該查詢不會提交到 Neptune 資料庫執行個體。

以下方法會查詢提交至 Neptune 資料庫執行個體：

- `toList()`
- `toSet()`
- `next()`
- `nextTraverser()`
- `iterate()`

如果您需要序列化並傳回查詢結果，請使用 `next()`，或者如果不需要，則使用 `iterate()`。

> **Important**
>
> 這是一個獨立的 Node.js 範例。如果您打算在 AWS Lambda 函數中執行類似這樣的程式碼，請參閱以Lambda 函數範例取得有關在 Neptune Lambda 函數中 JavaScript 有效使用的詳細資訊。

使用 Go 連線至 Neptune 資料庫執行個體

如果可以的話，請始終使用最新版本的 Apache TinkerPop Go 小鬼客戶端，gremlingo，您的引擎版本支持。較新的版本包含許多錯誤修正，其可以改善用戶端的穩定性、效能和可用性。

要使用的`gremlingo`版 TinkerPop本通常會與 Java Gemlin 客戶端的表格中描述的版本保持一致。

> **Note**
>
> 只要您在編寫的小鬼查詢中僅使用 3. TinkerPop 4.x 功能，gremlingo 3.5.x 版本就可以向後兼容 3.4.x 版本。

下節引導您逐步執行 Go 範例，其會連線至 Amazon Neptune 資料庫執行個體，並執行 Gremlin 周遊。

您必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

開始之前，請執行以下動作：

- 從 go.dev 網站下載並安裝 Go 1.17 或更新版本。

使用 Go 連線至 Neptune

1. 從空目錄開始，初始化一個新的 Go 模組：

```
go mod init example.com/gremlinExample
```

2. 將 gremlin-go 新增為新模組的相依性：

```
go get github.com/apache/tinkerpop/gremlin-go/v3/driver
```

3. 建立名為 `gremlinExample.go` 的檔案，然後在文字編輯器中開啟。
4. 將下列內容複製到 `gremlinExample.go` 檔案，以 Neptune 資料庫執行個體的地址取代 *(your neptune endpoint)*：

```
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**
>
> 搭配 macOS 的 Go 1.18+ 目前不支援 Neptune TLS 憑證格式，因此在嘗試啟動連線時可能會出現 509 錯誤。對於本機測試，可以跳過此情況，方法是將 "crypto/tls" 新增至匯入，並修改 `DriverRemoteConnection` 設定，如下所示：
>
> ```
> // 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. 輸入下列命令以執行範例：

```
go run gremlinExample.go
```

本範例結尾的 Gremlin 查詢會以切片傳回頂點 (`g.V().Limit(2)`)。然後，此切片會使用標準 `fmt.Println` 函數進行迭代和列印。

> **Note**
>
> Gremlin 查詢最後的部分 `ToList()` 用來提交周遊至伺服器，以供進行評估。如果您未包含該方法或其他同等方法，該查詢不會提交到 Neptune 資料庫執行個體。

以下方法會查詢提交至 Neptune 資料庫執行個體：

- `ToList()`
- `ToSet()`
- `Next()`
- `GetResultSet()`
- `Iterate()`

上述範例使用 `g.V().Limit(2).ToList()` 周遊傳回圖形中的前兩個頂點。若要查詢其他內容，將其換成其他使用其中一個適當之結束方法的 Gremlin 周遊。

Gremlin 查詢提示

您可以在 Amazon Neptune 中使用查詢提示，來指定特定 Gremlin 查詢的最佳化和評估策略。

使用以下語法在查詢中新增 `withSideEffect` 步驟，即可指定查詢提示。

```
g.withSideEffect(hint, value)
```

- 提示 - 識別要套用的提示類型。
- 值 - 確定正在考慮的系統方面的行為。

例如，以下示範如何在 Gremlin 周遊中包含 `repeatMode` 提示。

> **Note**
>
> 所有 Gremlin 查詢提示的副作用皆加上 `Neptune#` 字首。

```
g.withSideEffect('Neptune#repeatMode',
 'DFS').V("3").repeat(out()).times(10).limit(1).path()
```

上述查詢會指示 Neptune 引擎周遊圖形時「深度優先」(DFS)，而非預設 Neptune「廣度優先」(BFS)。

以下章節提供可用查詢提示及其用法的詳細資訊。

主題

- Gremlin repeatMode 查詢提示
- Gremlin noReordering 查詢提示
- Gremlin typePromotion 查詢提示
- Gremlin useDFE 查詢提示
- 使用結果快取的 Gremlin 查詢提示

Gremlin repeatMode 查詢提示

Neptune `repeatMode` 查詢提示指定 Neptune 引擎如何在 Gremlin 周遊中評估 `repeat()` 步驟：廣度優先、深度優先或區塊深度優先。

當 `repeat()` 步驟的評估模式用於尋找或遵循路徑，而不只是在有限時間內重複單一步驟時，此模式很重要。

語法

將 `withSideEffect` 步驟加入查詢中，即可指定 `repeatMode` 查詢提示。

```
g.withSideEffect('Neptune#repeatMode', 'mode').gremlin-traversal
```

> **Note**
>
> 所有 Gremlin 查詢提示的副作用皆加上 `Neptune#` 字首。

可用模式

- BFS

廣度優先搜尋

`repeat()` 步驟的預設執行模式。會先取得所有同級節點，再進入更深的路徑。

這個模式會佔用非常大量記憶體，領域可能變得非常龐大。查詢用盡記憶體而被 Neptune 引擎取消的風險更高。這最符合其他 Gremlin 實作。

- DFS

  深度優先搜尋

  依循每個路徑到最大的深度，再移到下一個解決方案。

  這會使用較少的記憶體。在要從多個躍點中開始尋找單一路徑的情況下，這可以提供更好的效能。

- CHUNKED_DFS

  區塊深度優先搜尋

  一種混合的方法，會在 1,000 個節點的區塊中進行深度優先的圖形探索，而不是在1 個節點 (DFS) 或所有節點 (BFS)) 中探索。

  Neptune 引擎會在每個層級收集節點到最多 1,000 個節點，再進入更深的路徑。

  這是在速度和記憶體使用量之間達到平衡的做法。

  如果您想要使用 BFS，但查詢使用過多記憶體，這很實用。

範例

以下區段說明 Gremlin 周遊重複模式的作用。

在 Neptune 中，`repeat()` 步驟的預設模式是在所有周遊中執行廣度優先 (BFS) 執行策略。

在大多數情況下， TinkerGraph 實現使用相同的執行策略，但在某些情況下，它會改變遍歷的執行。

例如， TinkerGraph 實作會修改下列查詢。

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

此周遊的 `repeat()` 步驟會「展開」為以下周遊，導致深度優先 (DFS) 策略。

```
g.V(<id>).out().out().out().out().out().out().out().out().out().out().limit(1).path()
```

> **Important**
>
> Neptune 查詢引擎不會自動執行此操作。

廣度優先（BFS）是默認的執行策略， TinkerGraph 在大多數情況下類似。不過，有些案例則偏好使用深度優先 (DFS) 策略。

BFS (預設值)

廣度優先 (BFS) 是 `repeat()` 運算子的預設執行策略。

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Neptune 引擎會先探索完前九個躍點的領域，再尋找十個躍點的解決方案。在許多情況下，這非常有效，例如短路徑查詢。

不過，在上述範例中，使用 `repeat()` 運算子的深度優先 (DFS) 模式周遊會更快。

DFS

以下查詢的 `repeat()` 運算子會使用深度優先 (DFS) 模式。

```
g.withSideEffect("Neptune#repeatMode", "DFS").V("3").repeat(out()).times(10).limit(1)
```

這會依循每個個別解決方案到最大的深度，再探索下一個解決方案。

Gremlin noReordering 查詢提示

在提交 Gremlin 周遊時，Neptune 查詢引擎會調查周遊的結構並重新排序部分查詢，嘗試將評估所需工作量和查詢回應時間減到最低。例如，有多個 `has()` 步驟等諸多限制的周遊，通常不會依指定的順序加以評估。它會在以靜態分析檢查查詢後重新排序。

Neptune 查詢引擎會嘗試找出哪一個限制有更高選擇性，並先執行它。這通常可提高效能，但 Neptune 選擇評估查詢的順序，不一定是最佳的。

如果您知道資料的確切特性，並希望手動指示查詢執行的順序，您可以使用 Neptune noReordering 查詢提示，指定依給定的順序評估周遊。

語法

將 withSideEffect 步驟加入查詢中，即可指定 noReordering 查詢提示。

```
g.withSideEffect('Neptune#noReordering', true or false).gremlin-traversal
```

> **Note**
>
> 所有 Gremlin 查詢提示的副作用皆加上 Neptune# 字首。

可用值

- true
- false

Gremlin typePromotion 查詢提示

當您提交篩選數值或範圍的 Gemlin 周遊時，Neptune 查詢引擎通常必須在執行查詢時使用類型提升。這表示它必須檢查可以保留您正在篩選之值的每種類型的值。

例如，如果您要篩選等於 55 的值，則引擎必須尋找等於 55 的整數、等於 55L 的長整數、等於 55.0 的浮點數等等。每個類型提升都需要對儲存體進行額外查詢，這可能會導致明顯簡單的查詢需要非預期的長時間才能完成。

假設您正在搜尋客戶年齡屬性大於 5 的所有頂點：

```
g.V().has('customerAge', gt(5))
```

若要徹底執行該周遊，Neptune 必須擴大查詢，以檢查您要查詢的值可以提升至的每個數值類型。在此情況下，必須針對任何超過 5 的整數、任何超過 5L 的長整數、任何超過 5.0 的浮點數，以及任何超過 5.0 的 double 整數套用 gt 篩選條件。由於其中每一個類型提升都需要對儲存體進行額外的查詢，因此當您針對此查詢執行 Gremlin profile API 時，您會看到每個數值篩選條件有多個篩選條件，而且完成所花費的時間會比您可能預期的時間要長很多。

通常類型提升不是必要的，因為您事先知道只需要找到一個特定類型的值即可。若是這種情況，您可以使用 typePromotion 查詢提示來關閉類型提升，以大幅加快查詢速度。

語法

將 withSideEffect 步驟加入查詢中，即可指定 typePromotion 查詢提示。

```
g.withSideEffect('Neptune#typePromotion', true or false).gremlin-traversal
```

> **Note**
>
> 所有 Gremlin 查詢提示的副作用皆加上 Neptune# 字首。

可用值

- true
- false

若要關閉上述查詢的類型提升，請使用：

```
g.withSideEffect('Neptune#typePromotion', false).V().has('customerAge', gt(5))
```

Gremlin useDFE 查詢提示

使用此查詢提示來啟用使用 DFE 執行查詢。根據預設，Neptune 不會在未將此查詢提示設定為 true 的情況下使用 DFE，因為 neptune_dfe_query_engine 執行個體參數預設為 viaQueryHint。如果將該執行個體參數設定為 enabled，則除了 useDFE 查詢提示設定為 false 的查詢以外，所有查詢都會使用 DFE 引擎。

針對查詢啟用 DFE 的範例：

```
g.withSideEffect('Neptune#useDFE', true).V().out()
```

使用結果快取的 Gremlin 查詢提示

啟用查詢結果快取時，可以使用下列查詢提示。

Gremlin **enableResultCache** 查詢提示

值為 `true` 的 `enableResultCache` 查詢提示會導致從快取傳回這些結果 (如果已快取它們的話)。如果沒有，它會傳回新的結果並快取它們，直到它們從快取中清除。例如：

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

稍後，您可以再次發出完全相同的查詢來存取快取的結果。

如果此查詢提示的值為 `false`，或者如果它不存在，則不會快取查詢結果。不過，將其設定為 `false` 不會清除現有的快取結果。若要清除快取的結果，請使用 `invalidateResultCache` 或 `invalidateResultCachekey` 提示。

Gremlin **enableResultCacheWithTTL** 查詢提示

`enableResultCacheWithTTL` 查詢提示也會傳回快取的結果 (如果有的話)，而不會影響已在快取中的結果的 TTL。如果目前沒有快取的結果，查詢會傳回新的結果，並快取這些結果，存留時間為 `enableResultCacheWithTTL` 查詢提示所指定的存留時間 (TTL)。該存留時間是以秒為單位指定的。例如，下列查詢會指定 60 秒的存留時間：

```
g.with('Neptune#enableResultCacheWithTTL', 60)
 .V().has('genre','drama').in('likes')
```

在 60 秒 time-to-live 結束之前，您可以使用與`enableResultCache`或查詢提示相同的`enableResultCacheWithTTL`查詢 (此處`g.V().has('genre','drama').in('likes')`) 來存取快取的結果。

> **Note**
>
> 使用 `enableResultCacheWithTTL` 指定的存留時間不會影響已快取的結果。
>
> - 如果之前已使用 `enableResultCache` 快取結果，則必須先明確地清除此快取，然後 `enableResultCacheWithTTL` 才能產生新結果，並快取它們，存留時間為其指定的 TTL。
> - 如果之前已使用 `enableResultCachewithTTL` 快取結果，則先前的 TTL 必須先過期，然後 `enableResultCacheWithTTL` 才能產生新結果，並快取它們，存留時間為其指定的 TTL。

在存留時間過後，會清除查詢的快取結果，然後相同查詢的後續執行個體會傳回新的結果。如果 enableResultCacheWithTTL 附加至該後續查詢，則會快取新結果，存留時間為其指定的 TTL 。

Gremlin **invalidateResultCacheKey** 查詢提示

invalidateResultCacheKey 查詢提示可以採取 true 或 false 值。true 值會導致清除附加 invalidateResultCacheKey 的查詢的快取結果。例如，下列範例會導致清除針對查詢金鑰 g.V().has('genre','drama').in('likes') 快取的結果：

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

上述範例查詢不會導致其新的結果進行快取。如果您想要在清除現有的快取結果之後快取新結果，則可以在同一查詢中包含 enableResultCache (或 enableResultCacheWithTTL)：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Gremlin **invalidateResultCache** 查詢提示

invalidateResultCache 查詢提示可以採取 true 或 false 值。true 值會導致清除結果快取中的所有結果。例如：

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

上述範例查詢不會導致其結果進行快取。如果您想要在完全清除現有的快取之後快取新結果，則可以在同一查詢中包含 enableResultCache (或 enableResultCacheWithTTL)：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Gremlin **numResultsCached** 查詢提示

numResultsCached 查詢提示只能與包含 iterate() 的查詢搭配使用，而且它會針對其所附加的查詢指定要快取的結果數目上限。請注意，numResultsCached 存在時快取的結果不會傳回，只會快取。

例如，下列查詢會指定應快取其最多 100 個結果，但不會傳回任何快取的結果：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').iterate()
```

然後，您可以使用如下的查詢來擷取一系列快取的結果 (此處為前十個)：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Gremlin **noCacheExceptions** 查詢提示

noCacheExceptions 查詢提示可以採取 true 或 false 值。true 值會導致隱藏與結果快取相關的任何例外狀況。例如：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

尤其，這會隱藏 QueryLimitExceededException，如果查詢的結果太大而無法容納在結果快取中，就會引發此例外狀況。

Gremlin 查詢狀態 API

若要取得 Gremlin 查詢的狀態，請使用 HTTP GET 或 POST 向 https://*your-neptune-endpoint*:*port*/gremlin/status 端點提出請求。

Gremlin 查詢狀態請求參數

- queryId (選用) – 執行中 Gremlin 查詢的 ID。只顯示指定查詢的狀態。
- includeWaiting (選用) – 傳回所有等待中查詢的狀態。

  通常，回應中只包含執行中查詢，但是當指定 includeWaiting 參數時，也會傳回所有等待中查詢的狀態。

Gremlin 查詢狀態回應語法

```
{
  "acceptedQueryCount": integer,
```

```
  "runningQueryCount": integer,
  "queries": [
    {
      "queryId":"guid",
      "queryEvalStats":
        {
          "waited": integer,
          "elapsed": integer,
          "cancelled": boolean
        },
      "queryString": "string"
    }
  ]
}
```

Gremlin 查詢狀態回應值

- 接受 QueryCount — 已接受但尚未完成的查詢數目，包括佇列中的查詢。
- 執行中 QueryCount — 目前正在執行的 Grinlin 查詢數目。
- queries – 目前的 Gremlin 查詢清單。
- queryId – 查詢的 GUID ID。Neptune 會自動將此 ID 值指派給每個查詢，或者您也可以指派自己的 ID (請參閱 將自訂 ID 注入至 Neptune Gremlin 或 SPARQL 查詢)。
- query EvalStats — 此查詢的統計資料。
- subqueries – 此查詢中的子查詢數目。
- elapsed – 到目前為止查詢已執行的毫秒數。
- cancelled – True 表示查詢已取消。
- queryString – 已提交的查詢。如果超過 1024 個字元即予截斷。
- waited – 指出查詢已等待多長時間 (以毫秒為單位)。

Gremlin 查詢狀態範例

下面是使用 `curl` 和 HTTP `GET` 的狀態命令範例。

```
curl https://your-neptune-endpoint:port/gremlin/status
```

此輸出會顯示單一執行中查詢。

```
{
```

```
  "acceptedQueryCount":9,
  "runningQueryCount":1,
  "queries": [
    {
      "queryId":"fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
      "queryEvalStats":
        {
          "waited": 0,
          "elapsed": 23,
          "cancelled": false
        },
      "queryString": "g.V().out().count()"
    }
  ]
}
```

Gremlin 查詢取消

若要取得 Gremlin 查詢的狀態，請使用 HTTP GET 或 POST 向 `https://`*your-neptune-endpoint*`:`*port*`/gremlin/status` 端點提出請求。

Gremlin 查詢取消請求參數

- cancelQuery – 取消的必要項目。參數沒有對應的值。
- queryId – 要取消之執行中 Gremlin 查詢的 ID。

Gremlin 查詢取消範例

以下為取消查詢的 `curl` 命令範例。

```
curl https://your-neptune-endpoint:port/gremlin/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

成功取消會傳回 HTTP `200` OK。

支援 Gremlin 指令碼型工作階段

您可以在 Amazon Neptune 中使用 Gremlin 工作階段搭配隱含交易。如需有關 Gemlin 工作階段的資訊，請參閱 Apache TinkerPop 文件中的考量工作階段。下面章節描述如何使用 Gremlin 工作階段搭配 Java。

> **Note**
>
> 此功能從 Neptune 引擎版本 1.0.1.0.200463.0開始可用。
> 從 Neptune 引擎版本 1.1.1.0 和 TinkerPop 版本 3.5.2 開始，您也可以使用。Gremlin 交易

> **Important**
>
> 目前 Neptune 可保持指令碼型工作階段開啟的最長時間是 10 分鐘。如果不在此時前關閉工作階段，工作階段會逾時，並轉返其中所有內容。

主題

- Gremlin 主控台上的 Gremlin 工作階段
- Gremlin 語言變體中的 Gremlin 工作階段

Gremlin 主控台上的 Gremlin 工作階段

如果您在 Gremlin 主控台建立遠端連線，而沒有 `session` 參數，則遠端連線會以「無工作階段」模式建立。在此模式中，每個提交至伺服器的請求本身都會視為完整交易，而且請求之間不會儲存狀態。如果請求失敗，只會轉返該請求。

如果您建立確實會使用 `session` 參數的遠端連線，則您建立的指令碼型工作階段會持續到您關閉遠端連線為止。每個工作階段都由一個主控台產生並傳回給您的唯一 UUID 識別。

以下是建立工作階段的主控台呼叫範例。提交查詢之後，另一個呼叫會關閉工作階段並遞交查詢。

> **Note**
>
> Gemlin 用戶端必須始終關閉才能釋出伺服器端資源。

```
gremlin> :remote connect tinkerpop.server conf/neptune-remote.yaml session
  . . .
  . . .
gremlin> :remote close
```

如需詳細資訊和範例，請參閱 TinkerPop 文件中的工作階段。

您在工作階段期間執行的所有查詢都會形成單一交易，直到所有查詢成功且關閉遠端連線才會遞交。如有某個查詢失敗，或者您未在 Neptune 支援的最長工作階段生命週期內關閉連線，則工作階段交易不會遞交，並轉返其中所有查詢。

Gremlin 語言變體中的 Gremlin 工作階段

在 Gremlin 語言變體 (GLV) 中，您需要建立 `SessionedClient` 物件，以單一交易發出多個查詢，如下列範例所示。

```
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();
}
```

上述範例中的第 3 行根據針對有問題之叢集所設定的組態選項，建立 `SessionedClient` 物件。您傳送到連線方法的 *`sessionName`* 字串會成為此工作階段的唯一名稱。若要避免衝突，名稱請使用 UUID。

用戶端會在初始化時啟動工作階段交易。只有在您呼叫 `client.close( )` 時，您在工作階段表單期間執行的所有查詢才會遞交。同樣地，如有單一查詢失敗，或者您未在 Neptune 支援的最長工作階段生命週期內關閉連線，則工作階段交易會失敗，並轉返其中所有查詢。

> **Note**
>
> Gemlin 用戶端必須始終關閉才能釋出伺服器端資源。

```
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();
    }
}
```

Neptune 中的 Gremlin 交易

有數個內容，Gremlin 交易會在其中執行。使用 Gremlin 時，務必了解您正在其中運作的內容以及其含義：

- **`Script-based`** – 請求是使用文字型 Gremlin 字串提出的，如下所示：
  - 使用 Java 驅動程式和 `Client.submit(string)`。
  - 使用 Gremlin 主控台和 `:remote connect`。
  - 使用 HTTP API。
- **`Bytecode-based`** – 請求是使用 Girmlin 語言變體 (GLV) 典型的序列化 Gremlin 位元碼提出的。

  例如，使用 Java 驅動程式 (`g = traversal().withRemote(...)`)。

對於上述任一內容，有額外的請求內容，被當作無工作階段形式或繫結至工作階段的方式傳送。

> **Note**
>
> 必須始終遞交或復原 Gremlin 交易，才能釋出伺服器端資源。

無工作階段請求

無工作階段時，請求等同於單一交易。

對於指令碼，其含義是單一請求中傳送的一個或多個 Gremlin 陳述式將做為單一交易進行遞交或復原。例如：

```
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();
```

對於位元碼，針對從 g 產生和執行的每個周遊提出無工作階段請求：

```
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();
```

繫結至工作階段的請求

當繫結至工作階段時，可在單一交易的內容中套用多個請求。

對於指令碼，含義是不需要將所有圖形操作串連成單一內嵌字串值：

```
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();
}
```

對於字節碼，之後 TinkerPop `3.5.x`，可以明確控制事務並透明地管理會話。Gremlin 語言變體 (GLV) 支援 `commit()` 或 `rollback()` 交易的 Gremlin `tx()` 語法，如下所示：

```
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();
    }
}
```

雖然上述範例子是以 Java 撰寫的，您也可以在 Python，JavaScript 和 .NET 中使用這個 `tx()` 語法。

> **Warning**
>
> 無工作階段唯讀查詢是在 SNAPSHOT 隔離下執行，但是在明確交易內執行的唯讀查詢是在 SERIALIZABLE 隔離下執行。在 SERIALIZABLE 隔離下執行的唯讀查詢會產生更高的負荷，並且可能會封鎖並行寫入或遭其封鎖，這與在 SNAPSHOT 隔離下執行的唯讀查詢不同。

搭配 Amazon Neptune 使用 Gremlin API

> **Note**
>
> Amazon Neptune 不支援 `bindings` 屬性。

Gremlin HTTPS 請求全部使用單一端點：`https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`。所有 Neptune 連線都必須使用 HTTPS。

您可以通過直接將 Girmlin 控制台連接到 Neptune 圖形。 WebSockets

如需連接到 Gremlin 端點的詳細資訊，請參閱 使用 Gremlin 存取 Neptune 圖形。

Gremlin 的 Amazon Neptune 實作有您需考慮的特定詳細資訊和差異。如需詳細資訊，請參閱 Amazon Neptune 中的 Gremlin 標準合規。

如需有關 Grinlin 語言和遍歷的詳細資訊，請參閱 Apache 文件中的穿越。 TinkerPop

在 Amazon Neptune Gremlin 中快取查詢結果

從引擎 1.0.5.1 版開始，Amazon Neptune 支援 Gremlin 查詢的結果快取。

您可以啟用查詢結果快取，然後使用查詢提示來快取 Gremlin 唯讀查詢的結果。

然後，只要快取的結果仍在快取中，重新執行查詢就會擷取這些結果，延遲低且沒有 I/O 成本。這適用於在 HTTP 端點上和使用 Websockets 提交的查詢，無論是做為位元碼或以字串形式。

> **Note**
>
> 即使啟用查詢快取，也不會快取傳送至設定檔端點的查詢。

您可以透過數種方式控制 Neptune 查詢結果快取的行為。例如：

- 您可以透過區塊取得分頁的快取結果。
- 您可以為指定的查詢指定 time-to-live (TTL)。
- 您可以清除所指定查詢的快取。
- 您可以清除整個快取。
- 您可以設定，若結果超過快取大小，則會收到通知。

快取是使用 least-recently-used (LRU) 原則來維護，這表示一旦配置給快取的空間已滿，結果就會在快取新 least-recently-used 結果時移除以騰出空間。

> **Important**
>
> `t3.medium` 或 `t4.medium` 執行個體類型上無法使用查詢結果快取。

在 Neptune 中啟用查詢結果快取

若要在 Neptune 中啟用查詢結果快取，請使用主控台將 `neptune_result_cache` 資料庫執行個體參數設定為 1 (已啟用)。

一旦啟用了結果快取，Neptune 就會留出目前記憶體的一部分來快取查詢結果。您使用的執行個體類型越大且可用的記憶體越多，Neptune 為快取留出的記憶體就越多。

如果結果快取記憶體已滿，Neptune 會自動捨棄 least-recently-used (LRU) 快取的結果，讓位給新的結果。

您可以使用 執行個體狀態 命令檢查結果快取的目前狀態。

使用提示快取查詢結果

一旦啟用了查詢結果快取，就會使用查詢提示來控制查詢快取。以下所有範例都適用於相同的查詢周遊，即：

```
g.V().has('genre','drama').in('likes')
```

使用 **enableResultCache**

啟用查詢結果快取後，您可以使用 enableResultCache 查詢提示，快取 Grinlin 查詢的結果，如下所示：

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Neptune 接著會將查詢結果傳回給您，同時也會快取它們。稍後，您可以再次發出完全相同的查詢來存取快取的結果：

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

識別快取結果的快取金鑰是查詢字串本身，即：

```
g.V().has('genre','drama').in('likes')
```

使用 **enableResultCacheWithTTL**

您可以使用查詢提示，指定應快取 enableResultCacheWithTTL 查詢結果多長時間。例如，下列查詢會指定查詢結果應在 120 秒後到期：

```
g.with('Neptune#enableResultCacheWithTTL', 120)
 .V().has('genre','drama').in('likes')
```

同樣地，識別快取結果的快取金鑰是基礎查詢字串：

```
g.V().has('genre','drama').in('likes')
```

您可以再次搭配 enableResultCache 查詢提示使用該查詢字串，存取快取的結果：

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

如果自緩存結果以來已經過了 120 秒或更多秒，則該查詢將返回新的結果，並緩存它們，而不會有任何結果 time-to-live。

您也可以使用 enableResultCacheWithTTL 查詢提示再次發出相同的查詢來存取快取的結果。例如：

```
g.with('Neptune#enableResultCacheWithTTL', 140)
 .V().has('genre','drama').in('likes')
```

直到過了 120 秒後 (也就是目前有效的 TTL)，這個使用 enableResultCacheWithTTL 查詢提示的新查詢就會傳回快取的結果。120 秒後，它將返回新結果並緩存它們 140 秒。 time-to-live

> **Note**
>
> 如果查詢索引鍵的結果已快取，則使用的相同查詢索引鍵enableResultCacheWithTTL不會產生新的結果，也不會影響目前快取 time-to-live 的結果。
>
> - 如果之前已使用 enableResultCache 快取結果，則必須先清除此快取，然後 enableResultCacheWithTTL 才能產生新結果，並快取它們，存留時間為其指定的 TTL。
> - 如果之前已使用 enableResultCachewithTTL 快取結果，則先前的 TTL 必須先過期，然後 enableResultCacheWithTTL 才能產生新結果，並快取它們，存留時間為其指定的 TTL。

使用 **invalidateResultCacheKey**

您可以使用 invalidateResultCacheKey 查詢提示來清除某個特定查詢的快取結果。例如：

```
g.with('Neptune#invalidateResultCacheKey', true)
```

```
  .V().has('genre','drama').in('likes')
```

該查詢會清除查詢金鑰 g.V().has('genre','drama').in('likes') 的快取，並傳回該查詢的新結果。

您也可以結合 invalidateResultCacheKey 與 enableResultCache 或 enableResultCacheWithTTL。例如，下列查詢會清除目前快取的結果、快取新結果，然後傳回它們：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

使用 **invalidateResultCache**

您可以使用 invalidateResultCache 查詢提示來清除查詢結果快取中的所有快取結果。例如：

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

該查詢會清除整個結果快取，並傳回查詢的新結果。

您也可以結合 invalidateResultCache 與 enableResultCache 或 enableResultCacheWithTTL。例如，下列查詢會清除整個結果快取、快取此查詢的新結果，然後傳回它們：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

對快取的查詢結果進行分頁

假設您已快取大量的結果，如下所示：

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

現在假設您發出以下範圍查詢：

```
g.with('Neptune#enableResultCache', true)
```

```
  .V().has('genre','drama').in('likes').range(0,10)
```

Neptune 首先尋找完整的快取金鑰，即 `g.V().has('genre','drama').in('likes').range(0,10)`。如果該金鑰不存在，Neptune 接下來會查看該查詢字串是否有金鑰沒有範圍 (即 `g.V().has('genre','drama').in('likes')`)。當它找到該金鑰時，Neptune 接著會從其快取中擷取前十個結果，如範圍所指定。

> **Note**
>
> 如果您搭配結尾有範圍的查詢使用 `invalidateResultCacheKey` 查詢提示，Neptune 會在找不到與具有範圍的查詢完全相符的項目時，Neptune 會清除沒有範圍的查詢快取。

搭配使用 **numResultsCached** 與 **.iterate()**

使用 `numResultsCached` 查詢提示，您可以填入結果快取，而不會傳回所有快取的結果，這在您偏好對大量結果進行分頁時很有用。

`numResultsCached` 查詢提示僅會使用結尾為 `iterate()` 的查詢。

例如，如果您想要快取範例查詢的前 50 個結果：

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

在此情況下，快取中的查詢金鑰是：`g.with("Neptune#numResultsCached", 50).V().has('genre','drama').in('likes')`。您現在可以使用此查詢，擷取前十個快取結果：

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(0, 10)
```

您也可以從查詢中擷取接下來的十個結果，如下所示：

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
```

```
 .V().has('genre','drama').in('likes').range(10, 20)
```

請不要忘記包含 numResultsCached 提示！它是查詢金鑰的必要部分，因此必須存在才能存取快取的結果。

使用 **numResultsCached** 時要謹記的一些事項：

- 您使用 **numResultsCached** 提供的數目會在查詢結束時套用。 例如，這表示下列查詢實際上會快取範圍 (1000, 1500) 內的結果：

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

- 您使用 **numResultsCached** 提供的數目會指定要快取的結果數目上限。 例如，這表示下列查詢實際上會快取範圍 (1000, 2000) 內的結果：

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 100000)
 .V().range(1000, 2000).iterate()
```

- 結尾為 **.range().iterate()** 的查詢所快取的結果具有自己的範圍。 例如，假設您使用如下的查詢來快取結果：

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

  若要從快取中擷取前 100 個結果，您將撰寫如下的查詢：

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).range(0, 100)
```

  這百個結果將等同於來自範圍 (1000, 1100) 中基礎查詢的結果。

用來尋找快取結果的查詢快取金鑰

在已快取查詢結果之後，具有相同「查詢快取金鑰」的後續查詢會從快取擷取結果，而不是產生新的結果。查詢的查詢快取金鑰會進行如下的評估：

1. 所有快取相關的查詢提示都會被忽略，但 numResultsCached 除外。
2. 最後一個 iterate() 步驟會被忽略。
3. 查詢的其餘部分會根據其位元碼表示法進行排序。

產生的字串會與快取中已存在的查詢結果索引進行比對，以判斷查詢是否有快取命中。

例如，採取以下查詢：

```
g.withSideEffect('Neptune#typePromotion', false).with("Neptune#enableResultCache",
 true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

它將儲存為如下的位元碼版本：

```
g.withSideEffect('Neptune#typePromotion', false)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes')
```

與結果快取相關的例外狀況

如果即使在移除先前快取的所有項目之後，您嘗試快取的查詢結果仍太大而無法容納於快取記憶體中，Neptune 就會引發 QueryLimitExceededException 錯誤。系統不會傳回任何結果，且例外狀況會產生下列錯誤訊息：

```
The result size is larger than the allocated cache,
      please refer to results cache best practices for options to rerun the query.
```

您可以使用 noCacheExceptions 查詢提示來隱藏此訊息，如下所示：

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

使用 Gremlin **mergeV()** 和 **mergeE()** 步驟進行有效的 upsert

upsert (或條件式插入) 會重複使用頂點或邊緣 (如果它已經存在)，或者如果不存在，則建立它。有效的 upsert 可以在 Gremlin 查詢的效能上產生顯著的差異。

Upsert 可讓您撰寫等冪性插入操作：無論您執行多少次這類操作，整體結果都是一樣的。這在高度並行寫入案例中非常有用，其中對圖形的相同部分進行並行修改可以強制一或多個交易使用 `ConcurrentModificationException` 進行復原，從而需要重試。

例如，以下查詢會使用提供的 `Map` upsert 頂點，以首先嘗試尋找 `T.id` 為 `"v-1"` 的頂點。如果找到該頂點，則會傳回它。如果沒有找到，則會透過 `onCreate` 子句建立具有該 `id` 和屬性的頂點。

```
g.mergeV([(id):'v-1']).
  option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org'])
```

批次 upsert 以改善輸送量

對於高輸送量寫入案例，您可以將 `mergeV()` 和 `mergeE()` 步驟鏈結在一起，以批次方式 upsert 頂點和邊緣。批次處理可減少 upsert 大量頂點和邊緣的交易負荷。然後，您可以使用多個用戶端平行 upsert 批次請求，進一步改善輸送量。

根據經驗法則，我們建議每個批次請求 upsert 大約 200 筆記錄。記錄是單一頂點或邊緣標籤或屬性。例如，具有單一標籤和 4 個屬性的頂點會建立 5 筆記錄。具有一個標籤和單一屬性的邊緣會建立 2 筆記錄。如果您想要 upsert 頂點批次，每個頂點都有單一標籤和 4 個屬性，您應該從 40 的批次大小開始，因為 `200 / (1 + 4) = 40`。

您可以嘗試批次大小。每個批次 200 筆記錄是一個很好的起點，但理想的批次大小可能會更高或更低，取決於您的工作負載。不過，請注意，Neptune 可能會限制每個請求的 Girmlin 步驟總數。此限制沒有明文記載，但為了安全起見，請嘗試確保您的請求包含不超過 1,500 個 Gemlin 步驟。Neptune 可能會拒絕超過 1,500 個步驟的大型批次請求。

若要增加輸送量，您可以使用多個用戶端平行 upsert 批次 (請參閱 建立有效率的多執行緒 Gremlin 寫入)。用戶端數目應與 Neptune 寫入器執行個體上的工作者執行緒數目相同，通常是伺服器上 vCPU 數目的 2 倍。例如，一個 `r5.8xlarge` 執行個體具有 32 個 vCPU 和 64 個工作者執行緒。對於使用 `r5.8xlarge` 的高輸送量寫入案例，您將會使用 64 個將批次 upsert 平行寫入 Neptune 的用戶端。

每個用戶端都應提交批次請求，並等待請求完成，然後再提交另一個請求。儘管多個用戶端平行執行，但每個個別的用戶端以都序列方式提交請求。這可確保為伺服器提供穩定的請求串流，這些請求會佔用所有工作者執行緒，而不會大量湧入伺服器端要求佇列 (請參閱 調整 Neptune 資料庫叢集中資料庫執行個體的大小)。

嘗試避免產生多個周遊器的步驟

當一個 Gemlin 步驟執行時，它需要一個傳入的周遊器，並發出一個或多個輸出周遊器。由一個步驟發出的周遊器數目會確定下一個步驟的執行次數。

通常，在執行批次操作時，您想要每個操作 (例如 upsert 頂點 A) 執行一次，以便操作序列看起來像這樣：upsert 頂點 A、接著 upsert 頂點 B，然後 upsert 頂點 C，依此類推。只要一個步驟僅建立或修改一個元素，它就只會發出一個周遊器，而代表下一個操作的步驟只會執行一次。另一方面，如果一個操作建立或修改多個元素，它會發出多個周遊器，這又會導致後續步驟執行多次，每個發出的周遊器一次。這可能會導致資料庫執行不必要的額外工作，並且在某些情況下可能會導致建立不需要的其他頂點、邊緣或屬性值。

一個可能出錯的範例是類似 `g.V().addV()` 的查詢。這個簡單的查詢會為圖形中找到的每個頂點新增一個頂點，因為 `V()` 會為圖形中的每個頂點發出一個周遊器，而且其中每個周遊器都會觸發對 `addV()` 的呼叫。

如需處理可以發出多個周遊器之操作的方法，請參閱 混合 upsert 和插入。

Upsert 頂點

`mergeV()` 步驟是專門針對 upsert 頂點而設計的。它會以引數形式採取 `Map`，其代表要針對圖形中現有頂點進行比對的元素，而且如果找不到一個元素，則會使用該 `Map` 建立一個新的頂點。此步驟還可讓您在建立或比對的情況下變更行為，其中 `option()` 調幅器可與 `Merge.onCreate` 和 `Merge.onMatch` 權杖一起套用，來控制這些各自的行為。如需有關如何使用此步驟的進一步資訊，請 TinkerPop 參閱參考文件。

您可以使用頂點 ID 來判斷特定頂點是否存在。這是偏好的方法，因為 Neptune 會針對 ID 周圍的高度並行使用案例最佳化 upsert。舉例來說，以下查詢會建立具有給定頂點 ID 的頂點 (如果尚未存在)，或重複使用它 (如果存在)：

```
g.mergeV([(T.id): 'v-1']).
    option(onCreate, [(T.label): 'PERSON', email: 'person-1@example.org', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

請注意，此查詢以 `id()` 步驟結尾。雖然基於 upsert 頂點的目的並不是絕對必要的，但 upsert 查詢結尾的 `id()` 步驟可確保伺服器不會將所有頂點屬性序列化回用戶端，這有助於降低查詢的鎖定成本。

或者，您可以使用頂點屬性來識別頂點：

```
g.mergeV([email: 'person-1@example.org']).
    option(onCreate, [(T.label): 'PERSON', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

如果可能的話，請使用您自己使用者提供的 ID 來建立頂點，並使用這些 ID 來判斷在 upsert 操作期間頂點是否存在。這可讓 Neptune 最佳化 upsert。當並行修改很常見時，ID 型 upsert 可能明顯比屬性型 upsert 更有效率。

鏈結頂點 upsert

您可以將頂點 upsert 鏈結在一起，以批次方式插入它們：

```
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()
```

或者，您也可以使用下列 `mergeV()` 語法：

```
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'])
```

不過，因為這種形式的查詢包含了搜尋條件中對於基本查詢 (依 `id`) 而言是多餘的元素，所以效率不如先前的查詢。

Upsert 邊緣

`mergeE()` 步驟是專門針對 upsert 邊緣而設計的。它會以引數形式採取 `Map`，其代表要針對圖形中現有邊緣進行比對的元素，而且如果找不到一個元素，則會使用該 `Map` 建立一個新的邊緣。此步驟還可讓您在建立或比對的情況下變更行為，其中 `option()` 調幅器可與 `Merge.onCreate` 和 `Merge.onMatch` 權杖一起套用，來控制這些各自的行為。如需有關如何使用此步驟的進一步資訊，請 TinkerPop 參閱參考文件。

您可以使用邊緣 ID，以與您使用自訂頂點 ID upsert 頂點的相同方式來 upsert 邊緣。同樣地，這是偏好的方法，因為它允許 Neptune 最佳化查詢。例如，以下查詢會根據邊緣 ID 建立邊緣 (如果不存在)，或者如果存在，則會重複使用它。如果此查詢需要建立一個新的邊緣，它也會使用 `Direction.from` 和 `Direction.to` 頂點的 ID：

```
g.mergeE([(T.id): 'e-1']).
    option(onCreate, [(from): 'v-1', (to): 'v-2', weight: 1.0]).
    option(onMatch, [weight: 0.5]).
  id()
```

請注意，此查詢以 `id()` 步驟結尾。雖然基於 upsert 邊緣的目的並不是絕對必要的，但將 `id()` 步驟新增至查詢結尾可確保伺服器不會將所有邊緣屬性序列化回用戶端，這有助於降低查詢的鎖定成本。

許多應用程式會使用自訂頂點 ID，但會讓 Neptune 產生邊緣 ID。如果您不知道邊緣的 ID，但確實知道 `from` 和 `to` 頂點 ID，則可以使用這種查詢來 upsert 邊緣：

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  id()
```

`mergeE()` 參考的所有頂點都必須存在，步驟才能建立邊緣。

鏈結邊緣 upsert

與頂點 upsert 一樣，將批次請求的 `mergeE()` 步驟鏈結在一起很簡單：

```
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()
```

結合頂點和邊緣 upsert

有時您可能想要 upsert 頂點和連線它們的邊緣。您可以混合此處呈現的批次範例。以下範例會 upsert 3 個頂點和 2 個邊緣：

```
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()
```

混合 upsert 和插入

有時您可能想要 upsert 頂點和連線它們的邊緣。您可以混合此處呈現的批次範例。以下範例會 upsert 3 個頂點和 2 個邊緣：

Upsert 通常一次處理一個元素。如果您堅持使用此處呈現的 upsert 模式，則每個 upsert 操作都會發出單一周遊器，這會導致後續操作僅執行一次。

不過，有時您可能想要混合 upsert 與插入。例如，如果您使用邊緣來代表動作或事件的執行個體，則可能會發生這種情況。請求可能會使用 upsert 來確保所有必要的頂點都存在，然後使用插入來新增邊緣。對於這種請求，請注意從每個操作發出的潛在周遊器數目。

考慮以下範例，它混合了 upsert 和插入，以將代表事件的邊緣新增至圖形：

```
// 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()
```

查詢應該插入 5 個邊緣：2 FOLLOWED 邊緣和 3 VISITED 邊緣。不過，查詢在撰寫時會插入 8 個邊緣：2 個 FOLLOWED 和 6 個 VISITED。這樣做的原因是，插入 2 個 FOLLOWED 邊緣的操作會發出 2 個周遊器，從而導致後續的插入操作 (插入 3 個邊緣) 執行兩次。

修正方法是在每個可能發出多個周遊器的操作之後新增一個 `fold()` 步驟：

```
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()
```

在這裡，我們已在插入 FOLLOWED 邊緣的操作後面插入 fold() 步驟。這會產生單一周遊器，然後導致後續操作僅執行一次。

這種方法的缺點是查詢現在未完全最佳化，因為 fold() 未最佳化。接在 fold() 後面的插入操作現在也不會最佳化。

如果您需要使用 fold()，代表後續步驟減少周遊器的數目，請嘗試排序您的操作，以便最便宜的操作佔用查詢的非最佳化部分。

設定基數

在 Neptune 頂點屬性的默認基數被設置，這意味著使用 MergeV（ ）在地圖中提供的值時都將被賦予該基數。若要使用單一基數，您必須明確使用它。從 TinkerPop 3.7.0 開始，有一個新的語法，允許提供基數作為地圖的一部分，如下列範例所示：

```
g.mergeV([(T.id): 1234]).
  option(onMatch, ['age': single(20), 'name': single('alice'), 'city': set('miami')])
```

或者，您可以將基數設置為默認值，如下option所示：

```
// 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)
```

在 3.7.0 mergeV() 之前的版本中，設定基數的選項較少。一般的做法是回到property()步驟，如下所示：

```
g.mergeV([(T.id): '1234']).
  option(onMatch, sideEffect(property(single,'age', 20).
  property(set,'city','miami')).constant([:]))
```

> **Note**
>
> 這種方法只有在與開始步驟一起使用mergeV()時才能使用。因此，您將無法在單個遍歷mergeV()內進行鏈接，因為如果傳入的遍歷器是圖形元素，則使用此語法的開始步驟mergeV()之後的第一個步驟將產生錯誤。在這種情況下，您需要將mergeV()呼叫分解為多個請求，其中每個請求都可以是開始步驟。

使用 **fold()/coalesce()/unfold()** 進行有效的 Gremlin upsert

upsert (或條件式插入) 會重複使用頂點或邊緣 (如果它已經存在)，或者如果不存在，則建立它。有效的 upsert 可以在 Gremlin 查詢的效能上產生顯著的差異。

本頁面說明如何使用 `fold()/coalesce()/unfold()` Grimlin 模式進行有效的 upsert。但是，隨著引擎 TinkerPop 版本 1.2.1.0 中 Neptune 中引入的 3.6.x 版本，在大多數情況下，新的mergeV()和mergeE()步驟更可取。此處描述的 `fold()/coalesce()/unfold()` 模式在某些複雜的情況下仍然很有用，但通常會使用 `mergeV()` 和 `mergeE()` (如果可以的話)，如 使用 Gremlin mergeV() 和 mergeE() 步驟進行有效的 upsert 中所述。

Upsert 可讓您撰寫等冪性插入操作：無論您執行多少次這類操作，整體結果都是一樣的。這在高度並行寫入案例中非常有用，其中對圖形的相同部分進行並行修改可以強制一或多個交易使用 `ConcurrentModificationException` 進行復原，從而需要重試。

例如，以下查詢會 upsert 頂點，方法是首先尋找資料集中的指定頂點，然後將結果摺疊成清單。在提供給 `coalesce()` 步驟的第一個周遊中，查詢接著會展開此清單。如果展開的清單不是空的，則會從 `coalesce()` 發出結果。不過，如果 `unfold()` 由於頂點目前不存在而傳回空集合，則 `coalesce()` 繼續評估與其一起提供的第二個周遊，並在此第二個周遊中，查詢會建立缺少的頂點。

```
g.V('v-1').fold()
          .coalesce(
             unfold(),
             addV('Person').property(id, 'v-1')
                           .property('email', 'person-1@example.org')
           )
```

使用最佳化形式的 **coalesce()** 進行 upsert

Neptune 可以最佳化 `fold().coalesce(unfold(), ...)` 慣用語以進行高輸送量更新，但只有在 `coalesce()` 的這兩個部分都傳回頂點或邊緣，但未傳回其他項目時，此最佳化才有效。如果您嘗試

從 `coalesce()` 的任何部分傳回不同的項目 (例如屬性)，則 Neptune 最佳化不會發生。查詢可能會成功，但它的執行效能不會與最佳化的版本一樣好，特別是針對大型資料庫。

因為未最佳化的 upsert 查詢會增加執行時間並減少輸送量，所以值得您使用 Gremlin `explain` 端點來判斷 upsert 查詢是否已完全最佳化。檢閱 `explain` 計劃時，請找出哪些行以 `WARNING: >>` 和 `+ not converted into Neptune steps` 開頭。例如：

```
+ not converted into Neptune steps: [FoldStep, CoalesceStep([[UnfoldStep],
 [AddEdgeSte...
WARNING: >> FoldStep << is not supported natively yet
```

這些警告可協助您識別查詢中阻止其完全最佳化的部分。

有時候不可能完全最佳化查詢。在這些情況下，您應該嘗試在查詢結尾處放置無法最佳化的步驟，從而允許引擎最佳化盡可能多的步驟。此技術用於某些批次 upsert 範例，其中會先執行一組頂點或邊緣的所有最佳化 upsert，然後再將任何其他可能未最佳化的修改套用至相同的頂點或邊緣。

批次 upsert 以改善輸送量

對於高輸送量寫入案例，您可以將 upsert 步驟鏈結在一起，以批次方式 upsert 頂點和邊緣。批次處理可減少 upsert 大量頂點和邊緣的交易負荷。然後，您可以使用多個用戶端平行 upsert 批次請求，進一步改善輸送量。

根據經驗法則，我們建議每個批次請求 upsert 大約 200 筆記錄。記錄是單一頂點或邊緣標籤或屬性。例如，具有單一標籤和 4 個屬性的頂點會建立 5 筆記錄。具有一個標籤和單一屬性的邊緣會建立 2 筆記錄。如果您想要 upsert 頂點批次，每個頂點都有單一標籤和 4 個屬性，您應該從 40 的批次大小開始，因為 `200 / (1 + 4) = 40`。

您可以嘗試批次大小。每個批次 200 筆記錄是一個很好的起點，但理想的批次大小可能會更高或更低，取決於您的工作負載。不過，請注意，Neptune 可能會限制每個請求的 Girmlin 步驟總數。此限制沒有明文記載，但為了安全起見，請嘗試確保您的請求包含不超過 1500 個 Gemlin 步驟。Neptune 可能會拒絕超過 1500 個步驟的大型批次請求。

若要增加輸送量，您可以使用多個用戶端平行 upsert 批次 (請參閱 建立有效率的多執行緒 Gremlin 寫入)。用戶端數目應與 Neptune 寫入器執行個體上的工作者執行緒數目相同，通常是伺服器上 vCPU 數目的 2 倍。例如，一個 `r5.8xlarge` 執行個體具有 32 個 vCPU 和 64 個工作者執行緒。對於使用 `r5.8xlarge` 的高輸送量寫入案例，您將會使用 64 個將批次 upsert 平行寫入 Neptune 的用戶端。

每個用戶端都應提交批次請求，並等待請求完成，然後再提交另一個請求。儘管多個用戶端平行執行，但每個個別的用戶端以都序列方式提交請求。這可確保為伺服器提供穩定的請求串流，這些請求會佔用

所有工作者執行緒，而不會大量湧入伺服器端要求佇列 (請參閱 調整 Neptune 資料庫叢集中資料庫執行個體的大小)。

嘗試避免產生多個周遊器的步驟

當一個 Gemlin 步驟執行時，它需要一個傳入的周遊器，並發出一個或多個輸出周遊器。由一個步驟發出的周遊器數目會確定下一個步驟的執行次數。

通常，在執行批次操作時，您想要每個操作 (例如 upsert 頂點 A) 執行一次，以便操作序列看起來像這樣：upsert 頂點 A、接著 upsert 頂點 B，然後 upsert 頂點 C，依此類推。只要一個步驟僅建立或修改一個元素，它就只會發出一個周遊器，而代表下一個操作的步驟只會執行一次。另一方面，如果一個操作建立或修改多個元素，它會發出多個周遊器，這又會導致後續步驟執行多次，每個發出的周遊器一次。這可能會導致資料庫執行不必要的額外工作，並且在某些情況下可能會導致建立不需要的其他頂點、邊緣或屬性值。

一個可能出錯的範例是類似 `g.V().addV()` 的查詢。這個簡單的查詢會為圖形中找到的每個頂點新增一個頂點，因為 `V()` 會為圖形中的每個頂點發出一個周遊器，而且其中每個周遊器都會觸發對 `addV()` 的呼叫。

如需處理可以發出多個周遊器之操作的方法，請參閱 混合 upsert 和插入。

Upsert 頂點

您可以使用頂點 ID 來判斷對應頂點是否存在。這是偏好的方法，因為 Neptune 會針對 ID 周圍的高度並行使用案例最佳化 upsert。舉例來說，以下查詢會建立具有給定頂點 ID 的頂點 (如果尚未存在)，或重複使用它 (如果存在)：

```
g.V('v-1')
 .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .id()
```

請注意，此查詢以 `id()` 步驟結尾。雖然基於 upsert 頂點的目的並不是絕對必要的，但將 `id()` 步驟新增至查詢結尾可確保伺服器不會將所有頂點屬性序列化回用戶端，這有助於降低查詢的鎖定成本。

或者，您可以使用頂點屬性來判斷頂點是否存在：

```
g.V()
```

```
  .hasLabel('Person')
  .has('email', 'person-1@example.org')
  .fold()
  .coalesce(unfold(),
            addV('Person').property('email', 'person-1@example.org'))
  .id()
```

如果可能的話，請使用您自己使用者提供的 ID 來建立頂點，並使用這些 ID 來判斷在 upsert 操作期間頂點是否存在。這可讓 Neptune 圍繞 ID 最佳化 upsert。在高度並行修改案例中，ID 型 upsert 可能明顯比屬性型 upsert 更有效率。

鏈結頂點 upsert

您可以將頂點 upsert 鏈結在一起，以批次方式插入它們：

```
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()
```

Upsert 邊緣

您可以使用邊緣 ID，以與您使用自訂頂點 ID upsert 頂點的相同方式來 upsert 邊緣。同樣地，這是偏好的方法，因為它允許 Neptune 最佳化查詢。例如，以下查詢會根據邊緣 ID 建立邊緣 (如果不存在)，或者如果存在，則會重複使用它。如果此查詢需要建立一個新的邊緣，它也會使用 `from` 和 `to` 頂點的 ID。

```
g.E('e-1')
  .fold()
  .coalesce(unfold(),
```

```
          addE('KNOWS').from(V('v-1'))
                       .to(V('v-2'))
                       .property(id, 'e-1'))
 .id()
```

許多應用程式會使用自訂頂點 ID，但會讓 Neptune 產生邊緣 ID。如果您不知道邊緣的 ID，但確實知道 `from` 和 `to` 頂點 ID，則可以使用此資訊來 upsert 邊緣：

```
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()
```

請注意，`where()` 子句中的頂點步驟應該是 `inV()` (或者，如果您曾經使用 `inE()` 來尋找邊緣，則為 `outV()`)，而不是 `otherV()`。不要在這裡使用 `otherV()`，否則查詢將不會進行最佳優化，且效能將受到影響。例如，Neptune 不會最佳化下列查詢：

```
// 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()
```

如果您事先不知道邊緣或頂點 ID，則可以使用頂點屬性進行 upsert：

```
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()
```

與頂點 upsert 一樣，最好使用 ID 型邊緣 upsert (其會使用邊緣 ID 或 `from` 和 `to` 頂點 ID)，而不是使用屬性型 upsert，以便 Neptune 可以完全最佳化 upsert。

檢查 **`from`** 和 **`to`** 頂點是否存在

請注意，建立新邊緣之步驟的建構方式：`addE().from().to()`。這種建構方式可確保查詢檢查 `from` 和 `to` 頂點是否存在。如果其中一個不存在，則查詢會傳回錯誤，如下所示：

```
{
  "detailedMessage": "Encountered a traverser that does not map to a value for child...
  "code": "IllegalArgumentException",
  "requestId": "..."
}
```

如果 `from` 或 `to` 頂點可能不存在，則您應在它們之間 upsert 邊緣之前嘗試 upsert 它們。請參閱結合頂點和邊緣 upsert。

有一個替代的建構方式，可以建立您不應該使用的邊緣：`V().addE().to()`。如果 `from` 頂點存在，它只會新增一個邊緣。如果 `to` 頂點不存在，則查詢會產生錯誤，如先前所述，但是如果 `from` 頂點不存在，則它會無訊息地無法插入邊緣，而不會產生任何錯誤。例如，如果 `from` 頂點不存在，則下列 upsert 會完成，而不會 upsert 邊緣：

```
// 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()
```

鏈結邊緣 upsert

如果您想要將邊緣 upsert 鏈結在一起以建立批次請求，則必須使用頂點查詢開始每個 upsert，即使您已經知道邊緣 ID 也是如此。

如果您已經知道要 upsert 的邊緣 ID，以及 `from` 和 `to` 頂點的 ID，則可以使用以下公式：

```
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()
```

也許最常見的批次邊緣 upsert 案例是您知道 `from` 和 `to` 頂點 ID，但不知道要 upsert 的邊緣 ID。在此情況下，請使用以下公式：

```
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()
```

如果您知道要 upsert 的邊緣 ID，但不知道 `from` 和 `to` 頂點的 ID (這是預設值)，則可以使用以下公式：

```
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()
```

結合頂點和邊緣 upsert

有時您可能想要 upsert 頂點和連線它們的邊緣。您可以混合此處呈現的批次範例。以下範例會 upsert 3 個頂點和 2 個邊緣：

```
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()
```

混合 upsert 和插入

有時您可能想要 upsert 頂點和連線它們的邊緣。您可以混合此處呈現的批次範例。以下範例會 upsert 3 個頂點和 2 個邊緣：

Upsert 通常一次處理一個元素。如果您堅持使用此處呈現的 upsert 模式，則每個 upsert 操作都會發出單一周遊器，這會導致後續操作僅執行一次。

不過，有時您可能想要混合 upsert 與插入。例如，如果您使用邊緣來代表動作或事件的執行個體，則可能會發生這種情況。請求可能會使用 upsert 來確保所有必要的頂點都存在，然後使用插入來新增邊緣。對於這種請求，請注意從每個操作發出的潛在周遊器數目。

考慮以下範例，它混合了 upsert 和插入，以將代表事件的邊緣新增至圖形：

```
// 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()
```

查詢應該插入 5 個邊緣：2 FOLLOWED 邊緣和 3 VISITED 邊緣。不過，查詢在撰寫時會插入 8 個邊緣：2 個 FOLLOWED 和 6 個 VISITED。這樣做的原因是，插入 2 個 FOLLOWED 邊緣的操作會發出 2 個周遊器，從而導致後續的插入操作 (插入 3 個邊緣) 執行兩次。

修正方法是在每個可能發出多個周遊器的操作之後新增一個 `fold()` 步驟：

```
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()
```

在這裡，我們已在插入 FOLLOWED 邊緣的操作後面插入 `fold()` 步驟。這會產生單一周遊器，然後導致後續操作僅執行一次。

這種方法的缺點是查詢現在未完全最佳化，因為 fold() 未最佳化。接在 fold() 後面的插入操作現在不會最佳化。

如果您需要使用 fold()，代表後續步驟減少周遊器的數目，請嘗試排序您的操作，以便最便宜的操作佔用查詢的非最佳化部分。

修改現有頂點和邊緣的 Upsert

有時，您想要建立頂點或邊緣 (如果不存在)，然後將屬性新增至其中或更新其屬性，而不管它是新的還是現有的頂點或邊緣。

若要新增或修改屬性，請使用 property() 步驟。在 coalesce() 步驟之外使用此步驟。如果您嘗試修改 coalesce() 步驟內現有頂點或邊緣的屬性，Neptune 查詢引擎可能不會最佳化查詢。

以下查詢會在每個 upsert 的頂點上新增或更新計數器屬性。每個 property() 步驟都有單一基數，以確保新值會取代任何現有值，而不是新增至一組現有值。

```
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()
```

如果您有適用於所有 upsert 元素的屬性值 (例如 lastUpdated 時間戳記值)，則可以在查詢結束時新增或更新它：

```
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()
```

如果有其他條件，判斷是否應進一步修改頂點或邊緣，則您可以使用 has() 步驟，篩選將套用修改的元素。以下範例會使用 has() 步驟，根據其 version 屬性的值篩選 upsert 的頂點。然後，查詢會將其 version 小於 3 的任何頂點的 version 更新為 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()
```

使用 Gremlin **explain** 分析 Neptune 查詢執行

Amazon Neptune 已新增名為 explain 的 Gremlin 功能。這個功能是自助式工具，用於了解 Neptune 引擎採取的執行方法。您透過將 `explain` 參數新增到提交 Gremlin 查詢的 HTTP 呼叫，來進行叫用。

此 `explain` 功能可提供查詢執行計劃邏輯結構的相關資訊。您可以使用此資訊來識別潛在的評估和執行瓶頸，並調校您的查詢，如 調校 Gremlin 查詢 中所述。您也可以使用查詢提示，以改善查詢執行計劃。

> **Note**
>
> 從 版本 1.0.1.0.200463.0 (2019 年 10 月 15 日) 開始就可以使用這項功能。

主題

- 了解 Gremlin 查詢如何在 Neptune 中運作
- 在 Neptune 中使用 Gremlin explain API
- Neptune 中的 Gremlin profile API
- 使用 explain 和 profile 調校 Gremlin 查詢
- Amazon Neptune 的原生 Gremlin 步驟支援

了解 Gremlin 查詢如何在 Neptune 中運作

若要充分利用 Amazon Neptune 中的 Gremlin `explain` 和 `profile` 報告，了解有關 Gremlin 查詢的一些背景資訊很有幫助。

主題

- Neptune 中的 Gremlin 陳述式
- Neptune 如何使用陳述式索引來處理 Gremlin 查詢
- 如何在 Neptune 中處理 Gremlin 查詢

Neptune 中的 Gremlin 陳述式

Amazon Neptune 中的屬性圖資料由四個位置 (四元組) 陳述式組成。其中每個陳述式都代表屬性圖表資料的個別原子單位。如需詳細資訊，請參閱 Neptune 圖形資料模型。。與資源描述架構 (RDF) 資料模型類似，這四個位置如下：

- `subject (S)`
- `predicate (P)`
- `object (O)`
- `graph (G)`

每個陳述式都是有關一或多個資源的宣告。例如，陳述式可以宣告兩個資源之間的關係，或者將屬性 (鍵/值組) 連接到一些資源。

您可以將述詞視為陳述式的動詞，描述關係或屬性的類型。物件是關係的目標，或是屬性的值。圖形位置是選用的，可用於多種不同的方法。對於 Neptune 屬性圖 (PG) 資料，它是未使用的 (Null 圖)，或用來代表邊緣的識別符。具有共用資源識別符的一組陳述式可建立圖形。

Neptune 屬性圖資料模型中有三種類別的陳述式：

主題

- Gremlin 頂點標籤陳述式
- Gremlin 邊緣陳述式
- Gremlin 屬性陳述式

Gremlin 頂點標籤陳述式

Neptune 中的頂點標籤陳述式有兩個用途：

- 它們會追蹤頂點的標籤。
- 其中至少有一個陳述式暗示圖形中存在特定頂點。

這些陳述式的主題是頂點識別符，而物件是標籤，兩者都由使用者指定。您可以針對這些陳述式使用特殊的固定述詞 (顯示為 <~label>)，以及預設圖形識別符 (Null 圖形) (顯示為 <~>)。

例如，請考量下列 addV 周遊。

```
g.addV("Person").property(id, "v1")
```

此周遊會導致以下陳述式新增至圖形。

```
StatementEvent[Added(<v1> <~label> <Person> <~>) .]
```

Gremlin 邊緣陳述式

Gremlin 邊緣陳述式暗示 Neptune 圖形中兩個頂點之間存在邊緣。邊緣陳述式的主旨 (S) 是來源 `from` 頂點。述詞 (P) 是使用者提供的邊緣標籤。物件 (O) 是目標 `to` 頂點。圖形 (G) 是使用者提供的邊緣識別符。

例如，請考量下列 `addE` 周遊。

```
g.addE("knows").from(V("v1")).to(V("v2")).property(id, "e1")
```

此周遊會導致以下陳述式新增至圖形。

```
StatementEvent[Added(<v1> <knows> <v2> <e1>) .]
```

Gremlin 屬性陳述式

Neptune 中的 Gremlin 屬性陳述式會宣告頂點或邊緣的個別屬性值。主旨是使用者提供的頂點或邊緣識別符。述詞是屬性名稱 (索引鍵)，而物件是個別屬性值。圖形 (G) 再次是預設圖形識別符 (Null 圖形)，顯示為 <~>。

請考量下列範例。

```
g.V("v1").property("name", "John")
```

此陳述式會產生下列結果。

```
StatementEvent[Added(<v1> <name> "John" <~>) .]
```

屬性陳述式與其他屬性不同，其物件是基本值
(`string`、`date`、`byte`、`short`、`int`、`long`、`float` 或 `double`)。其物件不是可做為另一個宣告之主旨的資源識別符。

對於多重屬性，集合中的每個個別屬性值都會收到自己的陳述式。

```
g.V("v1").property(set, "phone", "956-424-2563").property(set, "phone", "956-354-3692
 (tel:9563543692)")
```

成果如下所示。

```
StatementEvent[Added(<v1> <phone> "956-424-2563" <~>) .]
StatementEvent[Added(<v1> <phone> "956-354-3692" <~>) .]
```

Neptune 如何使用陳述式索引來處理 Gremlin 查詢

陳述式會在 Amazon Neptune 中透過三個陳述式索引的方式存取，如 陳述式在 Neptune 中如何編製索引 中所述。Neptune 從 Gremlin 查詢中擷取陳述式「模式」，其中某些位置已知，其餘位置則留給索引搜尋進行探索。

Neptune 假設屬性圖結構描述的大小不大。這表示不同邊緣標籤和屬性名稱的數量相當低，導致不同的述詞總數很低。Neptune 會在個別索引中追蹤不同的述詞。它會使用此述詞快取來執行的 `{ all P x POGS }` 聯合掃描，而不是使用 OSGP 索引。避免需要反向周遊 OSGP 索引，可同時節省儲存空間和載入輸送量。

Neptune Gremlin Explain/設定檔 API 可讓您在圖形中取得述詞計數。然後，您可以判斷您的應用程式是否會使屬性圖結構描述很小的 Neptune 假設失效。

以下範例可協助說明 Neptune 如何使用索引來處理 Gremlin 查詢。

問：什麼是頂點 **v1** 的標籤？

```
Gremlin code:      g.V('v1').label()
Pattern:           (<v1>, <~label>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<~label>:*
```

問：什麼是頂點 **v1** 的「已知」外邊緣？

```
Gremlin code:      g.V('v1').out('knows')
Pattern:           (<v1>, <knows>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<knows>:*
```

問：哪些頂點有 **Person** 頂點標籤？

```
Gremlin code:      g.V().hasLabel('Person')
Pattern:           (?, <~label>, <Person>, <~>)
Known positions:   POG
Lookup positions:  S
Index:             POGS
Key range:         <~label>:<Person>:<~>:*
```

問：什麼是指定邊緣 **e1** 的來源/目標頂點？

```
Gremlin code:      g.E('e1').bothV()
Pattern:           (?, ?, ?, <e1>)
Known positions:   G
Lookup positions:  SPO
Index:             GPSO
Key range:         <e1>:*
```

Neptune 沒有的陳述式索引是反向周遊 OSGP 索引。此索引可用來收集所有邊緣標籤的所有傳入邊緣，如下列範例所示。

問：什麼是傳入的相鄰頂點**v1**？

```
Gremlin code:      g.V('v1').in()
Pattern:           (?, ?, <v1>, ?)
Known positions:   O
Lookup positions:  SPG
Index:             OSGP  // <-- Index does not exist
```

如何在 Neptune 中處理 Gremlin 查詢

在 Amazon Neptune 中，更複雜的周遊可由一系列模式表示，這些模式根據可跨模式共用的具名變數定義來建立關係，以建立聯結。如以下範例所示。

問：什麼是頂點 **v1** 的雙躍點鄰近項？

```
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
```

透過在兩個模式之間共用 ?2 變數 (在第一個模式中的 O 位置和第二個模式的 S 位置)，您可以建立從第一個躍點鄰近項到第二個躍點鄰近項的聯結。每個 Neptune 解決方案都有三個命名變量的綁定，這些變量可用於重新創建TinkerPop遍歷器（包括路徑信息）。

在 Gemlin 查詢處理的第一步是將查詢解析為一個 TinkerPop 遍歷對象，由一系列步驟組成。TinkerPop 這些步驟屬於開放原始碼 Apache TinkerPop 專案的一部分，是在參考實作中組成 Gemlin 遍歷的邏輯和實體運算子。它們都是用來代表查詢的模型。它們是可執行運算子，可根據所代表運算子的語意產生解決方案。例如，`.V()`由表示和執行 TinkerPop GraphStep。

因為這些 off-the-shelf TinkerPop 步驟是可執行的，所以這樣的 TinkerPop 遍歷可以執行任何 Gemlin 查詢並產生正確的答案。但是，當對大圖執行時， TinkerPop 步驟有時可能非常低效和緩慢。Neptune 不會使用它們，而是會嘗試將周遊轉換為由模式群組組成的宣告形式，如先前所述。

Neptune 目前在其原生查詢引擎中不支援所有 Gremlin 運算子 (步驟)。因此，它會嘗試盡可能將多個步驟摺疊成單一 `NeptuneGraphQueryStep`，其中包含所有已轉換步驟的宣告式邏輯查詢計畫。理想情況下，所有步驟都會進行轉換。但是，當遇到無法轉換的步驟時，Neptune 會中斷原生執行，並將所有查詢執行從該點推遲到步 TinkerPop 驟。它不會嘗試將原生執行編入和編出。

在步驟轉換為邏輯查詢計劃之後，Neptune 會執行一系列查詢最佳化工具，根據靜態分析和估計基數來重寫查詢計畫。這些最佳化工具會執行如下動作：根據範圍計數重新排序運算子、刪除不必要或冗餘運算子、重新安排篩選條件、將運算子推送至不同的群組等等。

在產生最佳化的查詢計畫之後，Neptune 會建立實體運算子的管道，以執行查詢的工作。這包括從陳述式索引讀取資料、執行各種類型的聯結、篩選、排序等等。管線會產生解決方案資料流，然後再轉換回 TinkerPop Traverser 物件串流。

查詢結果的序列化

Amazon Neptune 目前依賴 TinkerPop 回應訊息序列化器將查詢結果 (TinkerPop 遍歷器) 轉換為序列化資料，以透過線路傳送回用戶端。這些序列化格式往往相當冗長。

例如，若要序列化頂點查詢的結果 (例如 `g.V().limit(1)`)，Neptune 查詢引擎必須執行單一搜尋來產生查詢結果。不過，`GraphSON` 序列化程式會執行大量的額外搜尋，將頂點封裝為序列化格式。它將必須執行一次搜尋以取得標籤、執行一次搜尋以取得屬性金鑰，以及針對頂點的每個屬性金鑰執行一次搜尋，以取得每個金鑰的所有值。

某些序列化格式更有效率，但都需要額外的搜尋。此外， TinkerPop 序列化器不會嘗試避免重複的搜索，通常導致許多搜索不必要地重複。

這使得編寫查詢非常重要，以便它們只特別要求其所需的資訊。例如，`g.V().limit(1).id()` 只會傳回頂點 ID 並消除所有其他序列化程式搜尋。Neptune 中的 Gremlin `profile` API 可讓您查看在查詢執行期間和序列化期間進行多少次搜尋呼叫。

在 Neptune 中使用 Gremlin **`explain`** API

Amazon Neptune `explain` API 會傳回在所指定查詢執行時將執行的查詢計畫。因為 API 並未實際執行查詢，所以幾乎會立即傳回計劃。

它不同於 TinkerPop .explain () 步驟，以便能夠報告特定於 Neptune 引擎的資訊。

Gremlin **`explain`** 報告中包含的資訊

`explain` 報告包含下列資訊：

- 如請求的查詢字串。
- 原始周遊。這是通過將查詢字符串解析成步驟生成 TinkerPop 的 TinkerPop 遍歷對象。它等同於針對查詢執行所產生`.explain()`的原始查詢 TinkerPop TinkerGraph。
- 轉換後的周遊。這是通過將遍歷轉換為 Neptune 邏輯查詢計劃表示產生的 Neptune TinkerPop 遍歷。在許多情況下，整個 TinkerPop 遍歷轉換為兩個 Neptune 步驟：一個執行整個查詢 (`NeptuneGraphQueryStep`) 和一個將 Neptune 查詢引擎輸出轉換回 TinkerPop Traversers ()。`NeptuneTraverserConverterStep`
- 最佳化的周遊。這是 Neptune 查詢計畫的最佳化版本，其最佳化過程是透過一系列靜態減少工作最佳化工具來執行此查詢計畫，並根據靜態分析和估計基數來重寫查詢。這些最佳化工具會執行如下動作：根據範圍計數重新排序運算子、刪除不必要或冗餘運算子、重新安排篩選條件、將運算子推送至不同的群組等等。
- 述詞計數。由於先前所述的 Neptune 索引策略，具有大量不同的述詞可能會產生效能問題。這對於沒有邊緣標籤 (`.in` 或 `.both`) 的反向周遊運算子的查詢尤其適用。如果使用這類運算子且述詞計數夠高，則 `explain` 報告會顯示警告訊息。
- DEF 資訊。啟用 DFE 替代引擎時，下列周遊元件可能會顯示在最佳化的周遊中：
  - **`DFEStep`** – 周遊中 Neptune 最佳化的 DFE 步驟，其中包含子 `DFENode`。`DFEStep` 代表查詢計畫在 DFE 引擎中執行的一部分。
  - **`DFENode`** – 包含做為一個或多個子 `DFEJoinGroupNodes` 的中繼表示法。
  - **`DFEJoinGroupNode`** – 代表一個或多個 `DFENode` 或 `DFEJoinGroupNode` 元素的聯結。
  - **`NeptuneInterleavingStep`** – 周遊中 Neptune 最佳化的 DFE 步驟，其中包含子 `DFEStep`。

也包含一個包含周遊相關資訊的 stepInfo 元素，例如邊界元素、使用的路徑元素等。此資訊用來處理子 DFEStep。

了解 DFE 是否正在評估查詢的一個簡單方法，就是檢查 explain 輸出是否包含 DFEStep。不屬於周遊的任何部分DFEStep將不會由 DFE 執行，並且將由引擎執行。 TinkerPop

如需範例報告，請參閱 啟用 DFE 的範例。

Gremlin **explain** 語法

explain API 的語法與用於查詢的 HTTP API 語法相同，差別在於它使用 /gremlin/explain 做為端點，而非 /gremlin，如下列範例所示。

```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().limit(1)"}'
```

上述查詢會產生下列輸出。

```
*******************************************************
                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
```

未轉換的 TinkerPop 步驟

理想情況下，遍歷中的所有 TinkerPop 步驟都具有原生 Neptune 操作員覆蓋範圍。如果不是這種情況，Neptune 會退回其操作員覆蓋範圍中的差距的 TinkerPop 步驟執行。如果周遊使用的步驟，Neptune 還沒有其原生涵蓋範圍，則 `explain` 報告會顯示警告，其中顯示差距發生的位置。

遇到沒有對應原生 Neptune 運算子的步驟時，即使後續步驟確實具有原生 Neptune 運算子，也會使用步 TinkerPop 驟執行從該點開始前進的整個遍歷。

例外狀況是在呼叫 Neptune 全文搜尋時。 NeptuneSearchStep 實作不含原生對等項目的步驟做為全文檢索搜尋步驟。

查詢中所有步驟都具有原生對等項目的 **`explain`** 輸出範例

以下是查詢的範例 `explain` 報告，其中所有步驟都有原生對等項目：

```
*******************************************************
                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
```

查詢中的部分步驟沒有原生對等項目的範例

Neptune 同時原生處理 `GraphStep` 和 `VertexStep`，但如果您引入 `FoldStep` 和 `UnfoldStep`，則產生的 `explain` 輸出就會不同：

```
*******************************************************
                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
```

在此情況下，FoldStep 會使您從原生執行中脫離。但即使由於後續 VertexStep 似乎在 Fold/Unfold 步驟的下游，而不再原生處理它也一樣。

為了節省效能和成本，請務必嘗試規劃周遊，以便在 Neptune 查詢引擎內部以原生方式完成可能的最大工作量，而不是由步驟實作。 TinkerPop

使用 Neptune 的查詢範例 full-text-search

以下查詢使用 Neptune 全文搜尋：

```
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*")
```

`.has("name", "Neptune#fts mark*")` 部分將搜尋限制為具有 name 的頂點，而 `.has("Person", "name", "Neptune#fts mark*")` 將搜尋限制為具有 name 和標籤 Person 的頂點。這會在 explain 報告中產生以下周遊：

```
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}
    }
}]
```

DFE 啟用時使用 **explain** 的範例

以下是 DFE 替代查詢引擎啟用時的 explain 報告範例：

```
********************************************************
                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=92233720368547758O7}]
        }, {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=92233720368547758O7}],
            DFEPatternNode[(?12, ?11, ?9, ?13) . project ALL[?9, ?12] graphFilters=(!=
 defaultGraph[526] . ), {rangeCountEstimate=92233720368547758O7}]
          }, {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
```

如需報告中 DFE 特定部份的描述，請參閱 explain 中的資訊。

Neptune 中的 Gremlin **profile** API

Neptune Gremlin profile API 會執行指定的 Gremlin 周遊、收集各種有關執行的指標，以及產生設定檔報告做為輸出。

> **Note**
>
> 從 版本 1.0.1.0.200463.0 (2019 年 10 月 15 日) 開始就可以使用這項功能。

它不同於 TinkerPop .profile () 步驟，以便能夠報告 Neptune 引擎特定的資訊。

描述檔報告包含下列查詢計畫的相關資訊：

- 實體運算子管道
- 查詢執行和序列化的索引操作
- 結果的大小

profile API 使用 HTTP API 語法的延伸版本進行查詢，以 /gremlin/profile 做為端點，而不是 /gremlin。

Neptune Gremlin **profile** 的特定參數

- profile.results – `boolean`，允許的值：`TRUE` 和 `FALSE`，預設值：`TRUE`。

  若為 true，則會收集查詢結果，並將其顯示為 `profile` 報告的一部分。若為 false，則只會顯示結果計數。

- profile.chop – `int`，預設值：250。

  若為非零，則會在該字元數處截斷結果字串。這不會防止擷取所有結果。它只會限制描述檔報告中字串的大小。若設定為零，則字串會包含所有結果。

- profile.serializer – `string`，預設值：`<null>`。

  若為非 null，則會依此參數指定的格式，以序列化回應訊息傳回所收集的結果。這時會報告產生該回應訊息所需的索引操作數，以及要傳送到用戶端的大小 (以位元組為單位)。

  允許的值是`<null>`或任何有效的 MIME 類型或 TinkerPop 驅動程序「序列化器」枚舉值。

  ```
  "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`，允許的值：`TRUE` 和 `FALSE`，預設值：`FALSE`。

  若為 true，則會顯示查詢執行和序列化期間所發生之所有索引操作的詳細報告。警告：此報告可能冗長。

Neptune Grmlin **profile** 的範例輸出

以下是範例 `profile` 回應。

```
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"}'
```

在來自以下部落格文章的航線範例圖上執行時，此查詢會產生以下 profile 報告：讓我為您繪製圖形 – 第 1 部分 - 航線。

```
*******************************************************
                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
```

除了呼叫傳回給 Neptune `explain` 的查詢計畫之外，`profile` 結果還包括查詢執行時的執行期統計資料。每個聯結操作都會標記執行聯結所花費的時間，以及傳遞它的實際解決方案數量。

`profile` 輸出包含核心查詢執行階段，以及序列化階段 (如果指定 `profile.serializer` 選項) 所花費的時間。

每個階段期間執行的索引操作明細也會包含在 `profile` 輸出底部。

請注意，連續執行相同查詢可能會由於快取而在執行時間和索引操作時顯示不同的結果。

對於使用 `repeat()` 步驟的查詢，如果 `repeat()` 步驟做為 `NeptuneGraphQueryStep` 的一部分下推，則每次反覆運算的邊界明細可供其使用。

DFE 啟用時 **`profile`** 報告中的差異

Neptune DFE 替代查詢引擎啟用時，`profile` 輸出會有點不同：

最佳化的周遊：此區段類似於 `explain` 輸出中的區段，但包含其他資訊。這包括在規劃中考慮的 DFE 運算子類型，以及相關聯的最壞案例和最佳案例成本估算。

實體管道：此區段會擷取用來執行查詢的運算子。`DFESubQuery` 元素會提取 DFE 用來執行其所負責之計劃部份的實體計畫。這些 `DFESubQuery` 元素會在以下列出 DFE 統計資料的區段中展開。

DFE QueryEngine 統計值：此段落只有在 DFE 至少執行部分查詢時才會顯示。它概述了 DFE 特定的各種執行期統計資料，並包含查詢執行的各個部分所花費時間的詳細明細 (按 `DFESubQuery`)。

在此區段中，不同 `DFESubQuery` 元素中的巢狀子查詢會扁平化，而且唯一識別符會以開頭為 `subQuery=` 的標頭標記。

周遊指標：此區段顯示步驟層級周遊指標，並在 DFE 引擎執行全部或部分查詢時，顯示 `DFEStep` 和/或 `NeptuneInterleavingStep` 的指標。請參閱使用 `explain` 和 `profile` 調校 Gremlin 查詢。

> **Note**
>
> DFE 是在實驗室模式下發行的一項實驗功能，因此 `profile` 輸出的確切格式仍然可能會有所變更。

Neptune 資料流程引擎 (DFE) 啟用時的範例 **`profile`** 輸出

當 DFE 引擎用來執行 Grimlin 查詢時，Gremlin `profile` API 的輸出會進行格式化，如下面範例所示。

查詢：

```
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=92233720368547758O7})],
              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**
>
> 因為 DFE 引擎是在實驗室模式下發行的一項實驗功能，所以 `profile` 輸出的確切格式可能會有所變更。

使用 **`explain`** 和 **`profile`** 調校 Gremlin 查詢

您通常可以在 Amazon Neptune 中調校 Gremlin 查詢，以取得更好的效能，方法是使用從 Neptune explain 和 profile API 取得的報告中可供您使用的資訊 這樣做有助於了解 Neptune 如何處理 Girmlin 周遊。

> **Important**
>
> 在 TinkerPop 版本 3.4.11 中進行了更改，可提高查詢處理方式的正確性，但目前有時會嚴重影響查詢性能。
> 例如，此類查詢的執行速度可能會明顯變慢：

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

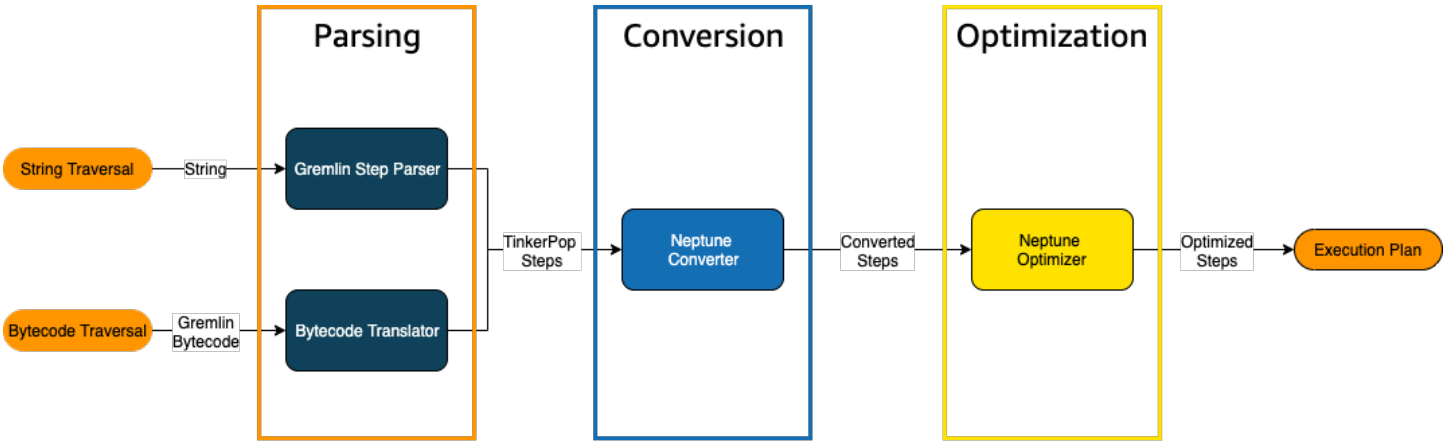
極限步驟後的頂點現在會以 3.4.11 變更的非最佳方式擷取。 TinkerPop 若要避免這種情況，您可以在 `order().by()` 之後的任何點新增 barrier() 步驟來修改查詢。例如：

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 在 Neptune 引擎版本 1.0.5.0 中已啟用 3.4.11。

了解 Neptune 中處理的 Gremlin 周遊

將 Grimlin 周遊傳送至 Neptune 時，有三個主要程序可將周遊轉換為基礎執行計畫，供引擎執行。這些是剖析、轉換和最佳化：



周遊剖析程序

處理周遊的第一步是將其剖析為通用語言。在 Neptune 中，該通用語言是屬於 TinkerPopAPI 一部分的一組 TinkerPop 步驟。這些步驟的每一個都代表周遊內的計算單位。

您可以將 Girmlin 周遊傳送至 Neptune 做為字串或位元碼。REST 端點和 Java 用戶端驅動程式 `submit()` 方法會以字串形式傳送周遊，如以下範例所示：

```
client.submit("g.V()")
```

使用 Gremlin 語言變體 (GLV) 的應用程式和語言驅動程式以位元碼形式傳送周遊。

周遊轉換程序

處理遍歷的第二個步驟是將其步 TinkerPop 驟轉換為一組轉換和未轉換的 Neptune 步驟。Apache TinkerPop Gramlin 查詢語言中的大多數步驟都會轉換為 Neptune 特定的步驟，這些步驟經過最佳化，以便在底層 Neptune 引擎上執行。在遍歷中遇到沒有 Neptune 對等的步 TinkerPop 驟時，查詢引擎會處理該步驟和遍歷中的所有後續步驟。 TinkerPop

如需在哪些情況下可轉換哪些步驟的詳細資訊，請參閱 Gremlin 步驟支援。

周遊最佳化程序

周遊處理中的最後一步是透過最佳化工具執行一系列已轉換和未轉換的步驟，以嘗試確定最佳執行計畫。此最佳化的輸出是 Neptune 引擎所處理的執行計畫。

使用 Neptune Gremlin **`explain`** API 來調校查詢

Neptune Explain API 與 Gremlin `explain()` 步驟不同。它會傳回 Neptune 引擎在執行查詢時將處理的最終執行計畫。因為它不會執行任何處理，所以不論使用哪些參數，都會傳回相同的計畫，而且其輸出不包含任何有關實際執行的統計資料。

考慮以下簡單的周遊，尋找安克拉治的所有機場頂點：

```
g.V().has('code','ANC')
```

有兩種方法，您可以透過 Neptune `explain` API 執行此周遊。第一種方法是對 Explain 端點進行 REST 呼叫，如下所示：

```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

第二種方法是搭配 `explain` 參數使用 Neptune 工作台的 %%gremlin 儲存格魔法。這會將儲存格本文中包含的周遊傳遞至 Neptune `explain` API，然後在您執行儲存格時顯示產生的輸出：

```
%%gremlin explain

g.V().has('code','ANC')
```

產生的 `explain` API 輸出描述了 Neptune 針對周遊的執行計畫。如您在下圖中所見，該計畫包含了處理管道中 3 個步驟中的每個步驟：

```
Explain
```

```
*******************************************************
               Neptune Gremlin Explain
*******************************************************

Query String
============

g.V().has('code','ANC')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)])]

Converted Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <~label>, ?2, <~>) . project distinct ?1 .]
            PatternNode[(?1, <code>, "ANC", ?) . project ask .]
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1}
        }, annotations={path=[Vertex(?1):GraphStep], maxVarId=3}
    },
    NeptuneTraverserConverterStep
]

Predicates
==========
# of predicates: 22
```

Parsing

Conversion

Optimization

透過查看未轉換的步驟來調校周遊

要在 Neptune `explain` API 輸出中尋找的其中一個首要項目是未轉換為 Neptune 原生步驟的 Gremlin 步驟。在查詢計畫中，當遇到無法轉換為 Neptune 原生步驟的步驟時，它和計畫中的所有後續步驟都會由 Gramlin 伺服器處理。

在上面的範例中，已轉換周遊中的所有步驟。讓我們檢查這個周遊的 `explain` API 輸出：

```
g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not
 an airport'))
```

如您在下圖中所見，Neptune 無法轉換 `choose()` 步驟：

```
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
```

您可以做幾件事來調校周遊的效能。第一件事是以這樣的方式重寫它，以消除無法轉換的步驟。另一件事是將步驟移至周遊尾端，以便所有其他步驟都可以轉換為原生步驟。

具有未轉換之步驟的查詢計畫並不一定需要調校。如果無法轉換的步驟位於周遊尾端，並且與輸出的格式化方式相關，而不是圖形的周遊方式，它們可能對效能沒有什麼影響。

在檢查來自 Neptune `explain` API 的輸出時要尋找的另一件事物是未使用索引的步驟。以下周遊會尋找有航班降落在安克拉治的所有機場：

```
g.V().has('code','ANC').in().values('code')
```

來自 Explain API 針對此周遊的輸出是：

```
*******************************************************
                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
```

因為無法使用 Neptune 維護的 3 個索引之一，處理周遊中的 `in()` 步驟，所以輸出底部出現 `WARNING` 訊息 (請參閱 陳述式在 Neptune 中如何編製索引 和 Neptune 中的 Gremlin 陳述式)。因為 `in()` 步驟沒有包含任何邊緣篩選條件，所以無法使用 SPOG、POGS 或 GPSO 索引來剖析它。相反地，Neptune 必須執行聯合掃描以尋找請求的頂點，效率要低得多。

在此情況下，有兩種方法可以調校周遊。第一種方法是將一或多個篩選條件新增至 `in()` 步驟，以便可以使用已編製索引的查詢來剖析查詢。對於上面範例，這可能是：

```
g.V().has('code','ANC').in('route').values('code')
```

來自 Neptune `explain` API 針對已修訂周遊的輸出不再包含 `WARNING` 訊息：

```
*******************************************************
               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
```

如果您正在執行許多此類周遊，則另一個選項是在已啟用選用 OSGP 索引的 Neptune 資料庫叢集中執行它們 (請參閱 啟用 OSGP 索引)。啟用 OSGP 索引有缺點：

- 在載入任何資料之前，必須在資料庫叢集中啟用它。
- 頂點和邊緣的插入速率可能會降低多達 23%。
- 儲存體用量將增加 20% 左右。
- 將請求分散到所有索引的讀取查詢可能會增加延遲。

有一個 OSGP 索引對於一組受限制的查詢模式很有意義，但除非您經常執行這些模式，否則通常最好嘗試確保您撰寫的周遊可以使用三個主要索引來剖析。

使用大量述詞

Neptune 會將圖形中的每個邊緣標籤和每個不同的頂點或邊屬性名稱視為述詞，並且根據預設，設計為使用相當少量的不同述詞。當您的圖形資料中有幾千個以上的述詞時，效能可能會降低。

若是這種情況，Neptune `explain` 輸出將會警告您：

```
Predicates
==========
# of predicates: 9549
WARNING: high predicate count (# of distinct property names and edge labels)
```

如果不方便重新設計資料模型以減少標籤和屬性的數目，從而減少述詞的數目，則調校周遊的最佳方法是在已啟用 OSGP 索引的資料庫叢集中執行它們，如上所述。

使用 Neptune Gremlin **`profile`** API 來調校周遊

Neptune `profile` API 與 Gremlin `profile()` 步驟有很大不同。與 `explain` API 一樣，其輸出包含 Neptune 引擎在執行周遊時使用的查詢計畫。此外，鑑於其參數的設定方式，`profile` 輸出還包括周遊的實際執行統計資料。

再次採取簡單的周遊，尋找安克拉治的所有機場頂點：

```
g.V().has('code','ANC')
```

與 `explain` API 一樣，您可以使用 REST 呼叫來調用 `profile` API：

```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

您也可以搭配 `profile` 參數使用 Neptune 工作台的 %%gremlin 儲存格魔法。這會將儲存格本文中包含的周遊傳遞至 Neptune `profile` API，然後在您執行儲存格時顯示產生的輸出：

```
%%gremlin profile

g.V().has('code','ANC')
```

產生的 `profile` API 輸出包含 Neptune 對周遊的執行計畫，以及有關計劃執行的統計資料，如您在下圖所見：

```
Profile
```

```
************************************************************
                Neptune Gremlin Profile
************************************************************

Query String
==================

g.V().has('code','ANC')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)])]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1, indexTime=0, joinTime=0, numSear
        }, annotations={path=[Vertex(?1):GraphStep], joinStats=true, optimizationTime=1, maxVarId=3, executionTime=3}
    },
    NeptuneTraverserConverterStep
]
```

Execution Plan

```
Physical Pipeline
=================
NeptuneGraphQueryStep
    |-- StartOp
    |-- JoinGroupOp
        |-- SpoolerOp(1000)
        |-- DynamicJoinOp(PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1})

Runtime (ms)
============
Query Execution: 5.096
```

Pipeline

```
Traversal Metrics
=================
Step                                                               Count  Traversers       Time (ms)    % Dur
-------------------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                          1           1           0.956    90.62
NeptuneTraverserConverterStep                                          1           1           0.099     9.38
                                            >TOTAL                     -           -           1.055        -

Predicates
==========
# of predicates: 26

Results
=======
Count: 1
Output: [v[2]]


Index Operations
================
Query execution:
    # of statement index ops: 1
    # of unique statement index ops: 1
    Duplication ratio: 1.0
    # of terms materialized: 0
```

Statistics and Results

在 `profile` 輸出中，執行計畫區段僅包含周遊的最終執行計畫，而不包含中繼步驟。管道區段包含已執行的實體管道操作，以及周遊執行所花費的實際時間 (以毫秒為單位)。執行期指標在比較兩個不同版本的周遊在您最佳化它們時所花費的時間非常有用。

> **Note**
>
> 周遊的初始執行期通常比後續執行期長，因為第一個執行期會導致快取相關資料。

`profile` 輸出的第三個區段包含執行統計資料和周遊的結果。若要了解此資訊在調校周遊時如何有用，請考慮以下周遊，它會尋找名稱以 "Anchora" 開頭的每個機場，以及從這些機場二趟短航程即可到達的所有機場，同時傳回機場代碼、航線和距離：

```
%%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'))
```

Neptune **`profile`** API 輸出中的周遊指標

所有 `profile` 輸出中可用的第一組指標是周遊指標。這些類似於 Gemlin `profile()` 步驟指標，但有一些差異：

```
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        -
```

周遊指標資料表的第一個資料行列出周遊所執行的步驟。前兩個步驟通常是 Neptune 特定的步驟，即 `NeptuneGraphQueryStep` 和 `NeptuneTraverserConverterStep`。

NeptuneGraphQueryStep 代表可由 Neptune 引擎以原生方式轉換和執行之周遊整個部分的執行時間。

NeptuneTraverserConverterStep表示將那些轉換步驟的輸出轉換為 TinkerPop 遍歷器的過程，這些步驟允許無法被轉換的步驟（如果有的話）被處理，或以兼容的格式返回結果。 TinkerPop

在上面的例子中，我們有幾個未轉換的步驟，因此我們看到每個 TinkerPop 步驟（ProjectStep，PathStep）然後在表中顯示為一行。

表格中的第二欄會報告通過步驟的代表遍歷數目，而第三欄則報告通過該步驟的遊覽者數目，如設定檔步驟文件中TinkerPop所述。CountTraversers

在我們的範例中，有 3,856 個頂點和 3,856 個由 NeptuneGraphQueryStep 傳回的周遊，並且這些數目在整個剩餘處理過程中保持不變，因為 ProjectStep 和 PathStep 正在格式化結果，而不是篩選它們。

> **Note**
>
> 不同於 TinkerPop，Neptune 引擎不會通過在其NeptuneGraphQueryStep和NeptuneTraverserConverterStep步驟中膨脹來優化性能。膨脹是將遍歷器結合在同一頂點以減少 TinkerPop操作開銷的操作，這就是導致Count和Traversers數字不同的原因。因為膨脹只發生在 Neptune 委派的步驟中 TinkerPop，而不是 Neptune 原生處理的步驟中，所以Count和Traverser資料行很少不同。

時間資料行會報告步驟所花費的毫秒數，而 % Dur 資料行則報告步驟所花費的總處理時間百分比。這些指標會透過顯示花費最多時間的步驟，告訴您將調校工作集中在哪裡。

Neptune **profile** API 輸出中的索引操作指標

Neptune 設定檔 API 輸出中的另一組指標是索引操作：

```
Index Operations
================
Query execution:
    # of statement index ops: 23191
    # of unique statement index ops: 5960
    Duplication ratio: 3.89
    # of terms materialized: 0
```

這些報告：

- 索引查閱的總數。
- 已執行的唯一索引查詢數目。
- 索引查詢總數與唯一索引查詢數目的比率。較低的比率指示較少的冗餘。
- 從字詞字典具體化的字詞數目。

Neptune **profile** API 輸出中的重複指標

如果您的周遊使用上述範例中的 `repeat()` 步驟，則輸出中會出現包含重複指標的區段：`profile`

```
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
```

這些報告：

- 資料列的迴圈計數 (`Iteration` 資料行)。
- 迴圈造訪的元素數目 (`Visited` 資料行)。
- 迴圈輸出的元素數目 (`Output` 資料行)。
- 迴圈輸出的最後一個元素 (`Until` 資料行)。
- 迴圈發出的元素數目 (`Emit` 資料行)。
- 從迴圈傳遞至後續迴圈的元素數目 (`Next` 資料行)。

這些重複指標對於理解周遊的分支因素非常有幫助，如此可讓您開始了解資料庫正在完成多少工作。您可以使用這些數字來診斷效能問題，尤其是當相同的周遊搭配不同的參數執行有很大不同時。

Neptune **profile** API 輸出中的全文搜尋指標

當周遊使用全文搜尋查詢 (如上述範例所示) 時，包含全文搜尋 (FTS) 指標的區段會出現在 `profile` 輸出中。

```
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
```

這會顯示傳送至 ElasticSearch (ES) 叢集的查詢，並報告與之互動有關的數個度量， ElasticSearch 可協助您找出與全文檢索搜尋相關的效能問題：

- 有關 ElasticSearch 索引調用的摘要信息：
  - 所有 RemoteCall 滿足查詢所需的毫秒總數 (`total`)。
  - 在 remoteCall 中花費的平均毫秒數 (`avg`)。
  - 在 remoteCall 中花費的毫秒數下限 (`min`)。
  - 在 remoteCall 中花費的毫秒數上限 (`max`)。
- 遠端呼叫 ElasticSearch (`remoteCallTime`) 使用的總時間。
- 對 ElasticSearch (`remoteCalls`) 所做的遠端呼叫數目。
- 結果 () 聯 ElasticSearch 結所花費的毫秒`joinTime`數。
- 進行索引查詢所花費的毫秒數 (`indexTime`)。
- () 傳回的結果 ElasticSearch 總`remoteResults`數。

Amazon Neptune 的原生 Gremlin 步驟支援

Amazon Neptune 引擎目前對所有 Gremlin 步驟沒有完整的原生支援。如 調校 Gremlin 查詢 中所述。目前的支援分為四個類別：

- 一律可以轉換為原生 Neptune 引擎操作的 Gremlin 步驟
- 在某些情況下可以轉換為原生 Neptune 引擎操作的 Gremlin 步驟
- 永不轉換為原生 Neptune 引擎操作的 Gremlin 步驟
- Neptune 完全不支援的 Gremlin 步驟

一律可以轉換為原生 Neptune 引擎操作的 Gremlin 步驟

許多 Gemlin 步驟只要符合下列條件就可以轉換為原生 Neptune 引擎操作：

- 它們不會在查詢前面加上無法轉換的步驟。
- 它們的父步驟 (如果有的話) 都可以進行轉換。
- 它們所有的子周遊 (如果有的話) 都可以進行轉換。

下列 Gemlin 步驟只要符合這些條件，一律轉換為原生 Neptune 引擎操作：

- 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( )

在某些情況下可以轉換為原生 Neptune 引擎操作的 Gremlin 步驟

在某些情況下，有些 Gemlin 步驟可以轉換為原生 Neptune 引擎操作，但在其他情況下則不能：

- addE( ) – `addE( )` 步驟通常可以轉換為原生 Neptune 引擎操作，除非它緊接著包含周遊做為索引鍵的 `property( )` 步驟。
- addV( ) – `addV( )` 步驟通常可以轉換為原生 Neptune 引擎操作，除非它緊接著包含周遊做為索引鍵的 `property( )` 步驟，或除非指派了多個標籤。
- aggregate( ) – `aggregate( )` 步驟通常可以轉換為原生 Neptune 引擎操作，除非該步驟用於子周遊或次周遊，或者除非要儲存的值是頂點、邊緣、ID、標籤或屬性值以外的值。

  在下面範例中，不會轉換 `aggregate( )`，因為正在子周遊中使用它：

  ```
  g.V().has('code','ANC').as('a')
       .project('flights').by(select('a')
       .outE().aggregate('x'))
  ```

  在此範例中，不會轉換 aggregate()，因為儲存的是值的 `min( )`：

  ```
  g.V().has('code','ANC').outE().aggregate('x').by(values('dist').min())
  ```

- barrier( ) – `barrier( )` 步驟通常可以轉換為原生 Neptune 引擎操作，除非其後的步驟未轉換。
- cap( ) – 轉換 `cap( )` 步驟的唯一情況是在其與 `unfold( )` 步驟結合，以傳回頂點、邊緣、ID 或屬性之彙總的展開版本時。在這個範例中，將轉換 `cap( )`，因為它後面是 `.unfold( )`：

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport').unfold()
  ```

  不過，如果您移除 `.unfold( )`，將不會轉換 `cap( )`：

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport')
  ```

- coalesce () -轉換`coalesce( )`步驟的唯一情況是遵循配方頁面上建議的 Upsert 模式。TinkerPop 不允許其他合併 coalesce() 模式。轉換限於可以轉換所有子周遊的情況，它們都會產生與輸出

相同的類型 (頂點、邊緣、ID、值、金鑰或標籤)、它們都會周遊到一個新元素，並且它們不包含 `repeat()` 步驟。

- constant( ) – 目前，僅在周遊的 `sack().by()` 部分內使用 constant() 步驟，以指派常數值時，才會轉換此步驟，如下所示：

```
g.V().has('code','ANC').sack(assign).by(constant(10)).out().limit(2)
```

- cyclicPath( ) – `cyclicPath()` 步驟通常可以轉換為原生 Neptune 引擎操作，除非該步驟與 `by()`、`from()` 或 `to()` 調幅器搭配使用。例如，在以下查詢中，不會轉換 `cyclicPath()`：

```
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( ) – `drop()` 步驟通常可以轉換為原生 Neptune 引擎操作，除非該步驟是在 `sideEffect()` 或 `optional()` 內部使用。
- fold ()-只有兩種情況下 fold () 步驟可以轉換，即在TinkerPop 食譜頁面上建議的 Upsert 模式中使用它時，以及在如下所示的上下`group().by()`文中使用時：

```
g.V().has('code','ANC').out().group().by().by(values('code', 'city').fold())
```

- id( ) – 除非在屬性上使用 `id()` 步驟，否則會轉換此步驟，如下所示：

```
g.V().has('code','ANC').properties('code').id()
```

- order( ) – `order()` 步驟通常可以轉換為原生 Neptune 引擎操作，除非下列其中一個條件成立：
  - `order()` 步驟是在巢狀子周遊內，如下所示：

```
g.V().has('code','ANC').where(V().out().order().by(id))
```

  - 例如，本機排序正在與 `order(local)` 搭配使用。
  - 正在排序依據的 `by()` 調幅中使用自訂比較器。使用 `sack()` 的範例如下：

```
g.withSack(0).
  V().has('code','ANC').
      repeat(outE().sack(sum).by('dist').inV()).times(2).limit(10).
      order().by(sack())
```

  - 在同一個元素上有多個排序。

- project( ) – `project()` 步驟通常可以轉換為原生 Neptune 引擎操作，除非 `project()` 後面的 `by()` 陳述式數目與指定的標籤數目不符，如下所示：

```
g.V().has('code','ANC').project('x', 'y').by(id)
```

- range( ) – 僅在有問題範圍的下限為零 (例如，`range(0,3)`) 時，才會轉換 `range()` 步驟。
- repeat( ) – `repeat()` 步驟通常可以轉換為原生 Neptune 引擎操作，除非它在另一個 `repeat()` 步驟內形成巢狀，如下所示：

```
g.V().has('code','ANC').repeat(out().repeat(out()).times(2)).times(2)
```

- sack( ) – `sack()` 步驟通常可以轉換為原生 Neptune 引擎操作，但在下列情況下除外：
  - 如果使用非數字 sack 運算子。
  - 如果使用 `+`、`-`、`mult`、`div`、`min` 和 `max` 以外的數字 sack 運算子。
  - 如果在 `where()` 步驟內使用 `sack()`，根據 sack 值進行篩選，如下所示：

```
g.V().has('code','ANC').sack(assign).by(values('code')).where(sack().is('ANC'))
```

- sum( ) – `sum()` 步驟通常可以轉換為原生 Neptune 引擎操作，但在用來計算全域求和時不能轉換，如下所示：

```
g.V().has('code','ANC').outE('routes').values('dist').sum()
```

- union( ) – `union()` 步驟可以轉換為原生 Neptune 引擎操作，只要它是查詢中的最後一個步驟，但終端步驟除外。
- 展開()-`unfold()` 步驟只有在TinkerPop配方頁面上建議的 Upsert 模式中使用時，才能將步驟轉換為原生 Neptune 引擎操作，並且與這`cap()`樣一起使用時：

```
g.V().has('airport','country','IE').aggregate('airport').limit(2)
     .cap('airport').unfold()
```

- where( ) – `where()` 步驟通常可以轉換為原生 Neptune 引擎操作，但在下列情況下除外：
  - 使用 by() 調幅時，如下所示：

```
g.V().hasLabel('airport').as('a')
     .where(gt('a')).by('runways')
```

  - 使用 `eq`、`neq`、`within` 和 `without` 以外的比較運算子時。

- 利用使用者提供的彙總時。

永不轉換為原生 Neptune 引擎操作的 Gremlin 步驟

Neptune 中支援下列 Gemlin 步驟，但這些步驟永遠不會轉換為原生 Neptune 引擎操作。相反地，它們是由 Gemlin 伺服器執行。

- choose( )
- coin( )
- inject( )
- match( )
- math( )
- max( )
- mean( )
- min( )
- option( )
- optional( )
- path( )
- propertyMap( )
- sample( )
- skip( )
- tail( )
- timeLimit( )
- tree( )

Neptune 完全不支援的 Gremlin 步驟

Neptune 中完全不支援下列 Gremlin 步驟。在大多數情況下，這是因為它們需要一個 `GraphComputer`，但 Neptune 目前不支援它。

- connectedComponent( )
- io( )
- shortestPath( )

- withComputer( )
- pageRank( )
- peerPressure( )
- program( )

`io()` 步驟實際上是部分受到支援，因為它可以用來從 URL 進行 `read()`，但不能進行 `write()`。

搭配 Neptune DFE 查詢引擎使用 Gremlin

如果您在實驗室模式中完全啟用稱為 DFE 的 Neptune 替代查詢引擎 (方法是將 `neptune_lab_mode` 資料庫叢集參數設定為 `DFEQueryEngine=enabled`)，Neptune 會將唯讀的 Gremlin 查詢/周遊轉換為中繼邏輯表示法，並盡可能在 DFE 引擎上執行它們。

不過，DFE 尚未支援所有的 Gremlin 步驟。當步驟無法在 DFE 上以原生方式執行時，Neptune 會回頭執 TinkerPop 行步驟。`explain` 和 `profile` 報告包含此情況發生時的警告。

> **Note**
>
> 從引擎 1.0.5.0 版開始，在沒有原生支援的情況下處理 Gemlin 步驟的預設 DFE 行為已變更。凡以前 DFE 發動機倒回 Neptune Grinlin 發動機，現在它回落在香草發動機。 TinkerPop

DFE 引擎原生支援的 Gremlin 步驟

- **`GraphStep`**
- **`VertexStep`**
- **`EdgeVertexStep`**
- **`IdStep`**
- **`TraversalFilterStep`**
- **`PropertiesStep`**
- 支援對屬性和 ID 以及標籤上的頂點和邊緣進行 **`HasStep`** 篩選，但文字和 `Without` 述詞除外。
- **`WherePredicateStep`** (含 `Path` 範圍篩選條件)，但沒有 `ByModulation`、`SideEffect` 或 `Map` 查詢支援
- **`DedupGlobalStep`**，但 `ByModulation`、`SideEffect` 和 `Map` 查詢支援除外。

查詢規劃交錯

當轉換程序遇到沒有對應的原生 DFE 運算子的 Gremlin 步驟時，在退回使用 Tinkerpop 之前，它會嘗試尋找其他可以在 DFE 引擎上以原生方式執行的中繼查詢組件。它透過將交錯邏輯應用到頂層周遊來執行此操作。結果是盡可能使用支援的步驟。

任何此類中繼、非字首查詢轉換都會在 `explain` 和 `profile` 輸出中使用 `NeptuneInterleavingStep` 來表示。

為了進行效能比較，您可能想要關閉查詢中的交錯，同時仍使用 DFE 引擎來執行字首部分。或者，您可能只希望使用引 TinkerPop 擎執行非前綴查詢。您可以使用 `disableInterleaving` 查詢提示來做到這一點。

正如值為 `false` 的 useDFE 查詢提示可防止完全不能在 DFE 上執行查詢一樣，值為 `true` 的 `disableInterleaving` 查詢提示會關閉 DFE 交錯以進行查詢轉換。例如：

```
g.with('Neptune#disableInterleaving', true)
 .V().has('genre','drama').in('likes')
```

已更新 Gremlin **`explain`** 和 **`profile`** 輸出

Gemlin explain 會提供有關 Neptune 用來執行查詢的最佳化周遊的詳細資訊。請參閱範例 DFE `explain` 輸出，以取得 DFE 引擎啟用時 `explain` 輸出的外觀範例。

Gremlin `profile` API 會執行指定的 Gemlin 周遊、收集有關執行的各種指標，以及產生設定檔報告，其中包含有關最佳化的查詢計畫和各種運算子的執行期統計資料的詳細資訊。請參閱範例 DFE `profile` 輸出，以取得 DFE 引擎啟用時 `profile` 輸出的外觀範例。

> **Note**
>
> 因為 DFE 引擎是在實驗室模式下發行的一項實驗功能，所以 `explain` 和 `profile` 輸出的確切格式可能會有所變更。

使用 openCypher 存取 Neptune 圖形

Neptune 支援使用 OpenCypher 建置圖形，OpenCypher 是使用圖形資料庫的開發人員目前最熱門的其中一個查詢語言。開發人員、商務分析師和資料科學家都愛用 OpenCypher 的 SQL 啟發語法，因為它提供了熟悉結構來編寫屬性應用程式的查詢。

OpenCypher 是屬性圖的宣告式查詢語言，最初由 Neo4j 開發，然後在 2015 年成為開放原始碼，並在 Apache 2 開放原始碼授權下投入 OpenCypher 專案。其語法記載於 Cypher 查詢語言參考第 9 版。

如需 Neptune 支援 OpenCypher 規格的限制和差異，請參閱 Amazon Neptune 中的開放密碼規範合規。

> **Note**
>
> Cypher 查詢語言的目前 Neo4j 實作已某些方面與 OpenCypher 規格有所分歧。如果您要將目前 Neo4j Cypher 程式碼遷移至 Neptune，請參閱 Neptune 與 Neo4j 的相容性 和 重寫 Cypher 查詢以在 Neptune 上的 OpenCpher 中執行 以取得協助。

從引擎 1.1.1.0 版開始，openCypher 適用於 Neptune 中的生產用途。

Gremlin 與 openCypher：相似性與差異

Gremlin 和 OpenCypher 都是屬性圖查詢語言，並且它們在許多方面彼此互補。

Gemlin 旨在吸引程式設計人員並無縫融入程式碼中。因此，Gremlin 的設計是命令式，而對於具有 SQL 或 SPARQL 經驗的人員， OpenCypher 的宣告式語法可能會感覺更熟悉。對於在 Jupyter 筆記本中使用 Python 的資料科學家來說，Gremlin 似乎更自然，而對於具有某些 SQL 背景的商務使用者來說，openCypher 似乎更直觀。

好處是，您不必在 Neptune 中的 Gremlin 與 openCypher 之間進行選擇。無論使用這兩種語言的哪一種輸入資料，任一種語言的查詢都可以在相同的圖形上操作。您可能會發現將 Gremlin 用於某些事情更方便，而對於其他事情，使用 OpenCypher 則更方便，取決於您正在做的事情。

Gremlin 使用命令式語法，可讓您透過一系列步驟來控制您在圖形中移動的方式，每個步驟都會包含資料串流、對其執行某些動作 (使用篩選條件、映射等)，然後將結果輸出至下一個步驟。Grimlin 查詢通常會採取格式 `g.V()`，後面跟著額外的步驟。

在 OpenCypher 中，您可以使用受 SQL 啟發的宣告式語法，其會指定節點和關係的模式，以使用 motif 語法 (例如 `()-[]->()`) 在圖形中尋找。OpenCypher 查詢通常以 `MATCH` 子句開頭、後面跟著其他子句，例如 `WHERE`、`WITH` 和 `RETURN`。

開始使用 openCypher

您可以使用 OpenCypher 查詢 Neptune 中的屬性圖資料，而不管它的載入方式，但是您不能使用 OpenCypher 來查詢以 RDF 載入的資料。

Neptune 大量載入器接受 Gremlin CSV 格式的屬性圖資料，以及 openCypher CSV 格式的屬性圖資料。另外，當然，您可以使用 Gremlin 和/或 OpenCypher 查詢，將屬性資料新增至圖形。

有許多在線上教學課程可用於學習 Cypher 查詢語言。在這裡，OpenCypher 查詢的幾個簡單範例可以協助您了解該語言，但到目前為止，開始使用 OpenCypher 查詢 Neptune 圖形的最好且最簡單的方法是透過在 Neptune 工作台中使用 OpenCypher 筆記本。該工作台是開源的，並託管 GitHub 在 https://github.com/aws-samples/amazon-neptune-samples 上。

您可以在 GitHub Neptune 圖形筆記本存儲庫中找到 OpenCypher 筆記本電腦。尤其，查看 openCypher 的 Air-routes visualization 和 English Premier Teams 筆記本。

由 OpenCypher 處理的資料採取無排序的一系列鍵/值映射形式。優化、操作和擴大這些映射的主要方法是使用子句，執行模式比對、插入、更新和刪除鍵值對等任務。

在 OpenCypher 中有幾個子句用於尋找圖形中的資料模式，其中 MATCH 是最常見的。MATCH 可讓您指定要在圖形中尋找之節點、關係和篩選條件的模式。例如：

- 取得所有節點

```
MATCH (n) RETURN n
```

- 尋找連線的節點

```
MATCH (n)-[r]->(d) RETURN n, r, d
```

- 尋找路徑

```
MATCH p=(n)-[r]->(d) RETURN p
```

- 取得具有標籤的所有節點

```
MATCH (n:airport) RETURN n
```

請注意，上面的第一個查詢會傳回圖形中的每個單一節點，而接下來的兩個查詢會傳回每個具有一個關係的節點，但通常不建議這樣做！在幾乎所有情況下，您都想要縮小傳回的資料範圍，您可以透過指定節點或關係標籤和屬性來做到這一點，如第四個範例所示。

您可以在 Neptune github 範例儲存庫中找到 OpenCypher 語法的便捷備忘單。

Neptune openCypher servlet 和狀態端點

OpenCypher 狀態端點可讓您存取目前正在伺服器上執行或等待執行之查詢的相關資訊。它也可讓您取消這些查詢。端點為：

```
https://(the server):(the port number)/openCypher/status
```

您可以使用 HTTP GET 和 POST 方法，從伺服器取得目前狀態，或者取消查詢。您也可以使用 DELETE 方法，取消執行中或等待中的查詢。

狀態請求的參數

狀態查詢參數

- **includeWaiting** (true 或 false) – 當設定為 true 且其他參數不存在時，這會導致傳回等待中查詢以及執行中查詢的狀態資訊。
- **cancelQuery** – 僅與 GET 和 POST 方法搭配使用，以指示這是取消請求。DELETE 方法不需要此參數。

  不會使用 cancelQuery 參數的值，但當 cancelQuery 存在時，需要 queryId 參數，以識別要取消的查詢。
- **queryId** – 包含特定查詢的 ID。

  如果與 GET 或 POST 方法搭配使用，且 cancelQuery 參數不存在 時，queryId 會針對其識別的特定查詢傳回狀態資訊。如果 cancelQuery 參數存在，則會取消 queryId 識別的特定查詢。

  與 DELETE 方法搭配使用時，queryId 一律指示要取消的特定查詢。
- **silent** – 僅在取消查詢時使用。如果設定為 true，則會導致取消以無訊息方式發生。

狀態請求回應欄位

狀態回應欄位 (如果未提供特定查詢的 ID)

- 接受 QueryCount — 已接受但尚未完成的查詢數目，包括佇列中的查詢。
- 執行中 QueryCount — 目前正在執行的開放式密碼查詢數目。
- queries – 目前的 openCypher 查詢清單。

特定查詢的狀態回應欄位

- queryId – 查詢的 GUID ID。Neptune 會自動將此 ID 值指派給每個查詢，或者您也可以指派自己的 ID (請參閱 將自訂 ID 注入至 Neptune Gremlin 或 SPARQL 查詢)。
- queryString – 已提交的查詢。如果超過 1024 個字元即予截斷。
- 查詢 EvalStats — 此查詢的統計資料：
  - waited – 指出查詢已等待多長時間 (以毫秒為單位)。
  - elapsed – 到目前為止查詢已執行的毫秒數。
  - cancelled – `True` 指示查詢已取消或 `False` 指示尚未取消。

狀態請求和回應的範例

- 請求所有查詢的狀態，包括等待中的查詢：

```
curl https://server:port/openCypher/status \
  --data-urlencode "includeWaiting=true"
```

回應：

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- 請求執行中查詢的狀態，不包括等待中的查詢：

```
curl https://server:port/openCypher/status
```

回應：

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- 請求單一查詢的狀態：

```
curl https://server:port/openCypher/status \
 --data-urlencode "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

回應：

```
{
  "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
  }
}
```

- 請求取消查詢

  1. 使用 POST：

```
curl -X POST https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

  回應：

```
{
  "status" : "200 OK",
  "payload" : true
}
```

  2. 使用 GET：

```
curl -X GET https://server:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=588af350-cfde-4222-bee6-b9cedc87180d"
```

  回應：

```
{
  "status" : "200 OK",
  "payload" : true
}
```

3. 使用 DELETE：

```
curl -X DELETE \
  -s "https://server:port/openCypher/status?queryId=b9a516d1-d25c-4301-
bb80-10b2743ecf0e"
```

回應：

```
{
  "status" : "200 OK",
  "payload" : true
}
```

The Amazon Neptune openCypher HTTPS 端點

主題

- HTTPS 端點上的 openCypher 讀取和寫入查詢
- 預設的 openCypher JSON 結果格式

HTTPS 端點上的 openCypher 讀取和寫入查詢

OpenCypher HTTPS 端點支援使用 GET 和 POST 方法進行讀取和更新查詢。不支援 DELETE 和 PUT 方法。

以下指示會逐步引導您使用 curl 命令和 HTTPS 連線到 openCypher 端點。您必須從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

語法是：

```
HTTPS://(the server):(the port number)/openCypher
```

以下是範例讀取查詢，一個使用 POST，一個使用 GET：

1. 使用 POST：

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n1) RETURN n1;"
```

2. 使用 GET (查詢字串是 URL 編碼的)：

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n1)%20RETURN%20n1"
```

以下是範例寫入/更新查詢，一個使用 POST，一個使用 GET：

1. 使用 POST：

```
curl HTTPS://server:port/openCypher \
  -d "query=CREATE (n:Person { age: 25 })"
```

2. 使用 GET (查詢字串是 URL 編碼的)：

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=CREATE%20(n%3APerson%20%7B%20age%3A
%2025%20%7D)"
```

預設的 openCypher JSON 結果格式

預設會傳回下列 JSON 格式，或將請求標頭明確設定為 `Accept: application/json` 來傳回此格式。這種格式旨在使用大多數程式庫的原生語言功能輕鬆地剖析為物件。

傳回的 JSON 文件包含一個欄位 `results`，其中包含查詢傳回值。下面範例顯示常見值的 JSON 格式。

值回應範例：

```
{
  "results": [
    {
      "count(a)": 121
    }
  ]
}
```

節點回應範例：

```
{
  "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
        }
      }
    }
  ]
}
```

關係回應範例：

```
{
  "results": [
    {
      "r": {
        "~id": "7389",
        "~entityType": "relationship",
        "~start": "22",
        "~end": "151",
        "~type": "route",
        "~properties": {
          "dist": 956
```

```
        }
      }
    }
  ]
}
```

路徑回應範例：

```
{
  "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
          }
        }
      ]
    }
  ]
}
```

使用 Bolt 通訊協定，對 Neptune 進行 openCypher 查詢

Bolt 是最初由 Neo4j 開發的面向語句的客戶端/服務器協議，並根據知識共享 3.0 署名許可許可。ShareAlike它是用戶端驅動的，這表示用戶端始終啟動訊息交換。

若要使用 Neo4j 的 Bolt 驅動程式連線至 Neptune，只需使用 `bolt` URI 方案將 URL 和連接埠號碼取代為叢集端點即可。如果您有一個正在執行的單一 Neptune 執行個體，請使用 read_write 端點。如果有多個執行個體正在執行，則建議兩個驅動程式，一個用於寫入器，另一個用於所有僅供讀取複本。如果您只有預設的兩個端點，一個 read_write 和一個 read_only 驅動程式就足夠了，但如果您也有自訂端點，請考慮為每個端點建立一個驅動程式執行個體。

> **Note**
>
> 儘管螺栓規範指出，Bolt 可以使用 TCP 進行連接 WebSockets，或者 Neptune 僅支持 Bolt 的 TCP 連接。

Neptune 允許高達 1000 個並行 Bolt 連線。

如需使用 Bolt 驅動程式之各種語言的 OpenCypher 查詢範例，請參閱 Neo4j 驅動程式與語言指南文件。

> **Important**
>
> 適用於 Python，.NET 和 Golang 的 Neo4j 螺栓驅動程序最初不支持 AWS 簽名 v4 身份驗證令牌的自動更新。 JavaScript這表示在簽章過期後 (通常在 5 分鐘內)，驅動程式便無法進行身分驗證，而且後續請求失敗。下面的 Python JavaScript，.NET 和 Go 示例都受到此問題的影響。
>
> 有關更多信息，請參閱 Neo4j 的 Python 驅動程序問題 #834，Neo4j 的 .NET 問題 #664，Neo4j 的 JavaScript驅動程序問題 #993 和 Neo4j 的黃金驅動程序問題 #429。
>
> 從驅動程序 5.8.0 版開始，Go 驅動程式已發行新的預覽重新身分驗證 API (請參閱 v5.8.0 – 重新身分驗證時所需的回饋)。

搭配 Java 使用 Bolt 連線至 Neptune

您可以從 Maven MVN 儲存庫下載要使用的任何版本的驅動程式，也可以將此相依性新增至您的專案：

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
  <version>4.3.3</version>
</dependency>
```

然後，若要在 Java 中使用這些 Bolt 驅動程式之一連線到 Neptune，請使用如下的程式碼，為叢集中的主要/寫入器執行個體建立驅動程式執行個體：

```
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());
```

如果您有一個或多個讀取器複本，則可以使用如下的程式碼，以類似的方式為它們建立驅動程式執行個體：

```
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());
```

或者，搭配逾時：

```
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());
```

如果您具有自訂端點，則也可能值得您為每個端點建立一個驅動程式執行個體。

使用 Bolt 的 Python openCypher 查詢範例

以下是如何在 Python 中使用 Bolt 進行 openCypher 查詢：

```
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)
```

請注意，會忽略 auth 參數。

使用 Bolt 的 .NET openCypher 查詢範例

要使用螺栓在. NET 中的 OpenCypher 查詢，第一步是使用安裝 Neo4j 的驅動程序。 NuHet若要進行同步呼叫，請使用 .Simple 版本，如下所示：

```
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();
      }
    }
  }
```

```
}
```

搭配 IAM 身分驗證使用 Bolt 的 Java openCypher 查詢範例

下面的 Java 程式碼說明如何搭配 IAM 身分驗證使用 Bolt，在 Java 中進行 openCypher 查詢。該 JavaDoc 評論描述了它的用法。一旦驅動程式執行個體可用，您就可以使用它，提出多個已經過身分驗證的請求。

```
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);
  }
}
```

搭配 IAM 身分驗證使用 Bolt 的 Python openCypher 查詢範例

下面的 Python 類別可讓您搭配 IAM 身分驗證使用 Bolt，在 Python 中進行 openCypher 查詢：

```
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)
```

您可以使用這個類別來建立一個驅動程式，如下所示：

```
  authToken = NeptuneAuthToken(creds, REGION, URL)
  driver = GraphDatabase.driver(URL, auth=authToken, encrypted=True)
```

使用 IAM 身分驗證和 Bolt 的 Node.js 範例

下面的 Node.js 代碼使用 JavaScript 版本 3 和 ES6 語法的 AWS SDK 來創建一個驗證請求的驅動程序：

```
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);
});
```

搭配 IAM 身分驗證使用 Bolt 的 .NET openCypher 查詢範例

若要在 .NET 中啟用 IAM 身分驗證，您需要在建立連線時簽署請求。下面的範例展示如何建立一個 NeptuneAuthToken 協助程式，來產生一個身分驗證權杖：

```
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);
    }
  }
}
```

以下是如何搭配 IAM 身分驗證使用 Bolt，在 .NET 中進行 OpenCypher 查詢。下面的範例會使用 NeptuneAuthToken 協助程式：

```
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);
      }
    }
  }
}
```

您可以在具有下列套件的 .NET 6 或 .NET 7 上執行下面的程式碼來啟動此範例：

- **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();
    }
  }
}
```

搭配 IAM 身分驗證使用 Bolt 的 Golang openCypher 查詢範例

下面的 Golang 套件說明如何搭配 IAM 身分驗證使用 Bolt，以 Go 語言進行 OpenCypher 查詢：

```
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)
}
```

Neptune 中的 Bolt 連線行為

以下是一些要謹記的 Neptune Bolt 連線相關事項：

- 因為 Bolt 連線是在 TCP 層建立的，所以您無法像使用 HTTP 端點一樣，在它們前面使用 Application Load Balancer。
- Neptune 用於 Bolt 連線的連接埠是您資料庫叢集的連接埠。
- Neptune 伺服器會根據傳遞給它的 Bolt 前導碼，選取最高的 Bolt 版本 (1、2、3 或 4.0)。
- 用戶端可以在任何時間點開啟的 Neptune 伺服器連線數目上限為 1,000。
- 如果用戶端在查詢之後沒有關閉連線，則該連線可以用來執行下一個查詢。
- 不過，如果連線閒置 20 分鐘，伺服器會自動將其關閉。
- 如果未啟用 IAM 身分驗證，您可以使用 `AuthTokens.none()`，而不是提供虛擬使用者名稱和密碼。例如，在 Java 中：

```
GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
 AuthTokens.none(),

 Config.builder().withEncryption().withTrustStrategy(TrustStrategy.trustSystemCertificates())
```

- 啟用 IAM 身分驗證時，如果 Bolt 連線由於其他一些原因而尚未關閉，則 Bolt 連線一律會在建立 10 天之後的幾分鐘內中斷連線。

- 如果用戶端傳送查詢以透過連線執行，而未取用先前查詢的結果，則會捨棄新查詢。若要改為捨棄先前的結果，用戶端必須透過連線傳送重設訊息。
- 在給定連線上一次只能建立一個交易。
- 如果在交易期間發生例外狀況，Neptune 伺服器會復原交易並關閉連線。在此情況下，驅動程式會為下一個查詢建立一個新連線。
- 請注意，工作階段不是執行緒安全的工作階段。多個平行操作必須使用多個個別的工作階段。

openCypher 參數化查詢的範例

Neptune 支援參數化的 openCypher 查詢。這可讓您搭配不同引數使用相同的查詢結構多次。由於查詢結構不會變更，因此 Neptune 可以快取其抽象語法樹 (AST)，而不必多次剖析它。

使用 HTTPS 端點的 openCypher 參數化查詢範例

以下是搭配 Neptune openCypher HTTPS 端點使用參數化查詢的範例。查詢為：

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

參數定義如下：

```
parameters={"name": "john", "age": 20}
```

使用 GET，您可以提交參數化查詢，如下所示：

```
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"
```

或者，您可以使用 POST：

```
curl -k \
  https://localhost:8182/openCypher \
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d "parameters={\"name\": \"john\", \"age\": 20}"
```

或者，使用 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"
```

使用 Bolt 進行 openCypher 參數化查詢的範例

以下是使用 Bolt 通訊協定進行 OpenCypher 參數化查詢的 Python 範例：

```
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()
```

以下是使用 Bolt 通訊協定進行 OpenCypher 參數化查詢的 Java 範例：

```
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);
```

openCypher 資料模型

Neptune OpenCypher 引擎建置在與 Gremlin 相同的屬性圖模型上。尤其是：

- 每個節點都有一個或多個標籤。如果插入沒有標籤的節點，則會附加名為 `vertex` 的預設標籤。如果嘗試刪除節點的所有標籤，則會擲回錯誤。
- 關係是只有一種關係類型的實體，而且其會在兩個節點之間形成單向連線 (也就是說，「從」其中一個節點「到」另一個節點)。
- 節點和關係都可以具有屬性，但不必如此。Neptune 支援沒有屬性的節點和關係。

- Neptune 不支援中繼屬性，而這些中繼屬性也不包括在 OpenCypher 規格中。
- 如果您圖形中的屬性是使用 Gremlin 建立的，則它們可以是多值的。也就是說，一個節點或關係屬性可有一組不同的值，而不是只有一個值。Neptune 已擴展 OpenCypher 語義，來正常地處理多值屬性。

支援的資料類型記載於 openCypher 資料格式。不過，我們目前不建議將 `Array` 屬性值插入至 OpenCypher 圖形。雖然可以使用大量載入器插入陣列屬性值，但目前的 Neptune OpenCypher 版本會將它視為一組多值屬性，而非單一清單值。

以下是此版本支援的資料類型清單：

- `Bool`
- `Byte`
- `Short`
- `Int`
- `Long`
- `Float` (包括正負 Infinity 和 NaN，但不包括 INF)
- `Double` (包括正負 Infinity 和 NaN，但不包括 INF)
- `DateTime`
- `String`

openCypher **`explain`** 功能

openCypher `explain` 功能是 Amazon Neptune 中的自助式工具，可協助您了解 Neptune 引擎採取的執行方法。若要調用 Explain，您可以使用 `explain=`*`mode`* 將參數傳遞至 OpenCypher HTTPS 請求，其中 *`mode`* 值可以是下列其中一個：

- **`static`** – 在 `static` 模式中，`explain` 只會列印查詢計畫的靜態結構。它實際上並不會執行查詢。
- **`dynamic`** – 在 `dynamic` 模式中，`explain` 也會執行查詢，並包含查詢計畫的動態層面。這些層面可能包含經由運算子流動的中繼繫結數和傳入繫結與傳出繫結的比率，以及每個運算子所花費的時間。
- **`details`** – 在 `details` 模式中，`explain` 會列印動態模式中顯示的資訊及其他詳細資訊，例如，實際 openCypher 查詢字串，以及構成聯結運算子基礎之模式的預估範圍計數。

例如，使用 POST：

```
curl HTTPS://server:port/openCypher \
  -d "query=MATCH (n) RETURN n LIMIT 1;" \
  -d "explain=dynamic"
```

或者，使用 GET：

```
curl -X GET \
  "HTTPS://server:port/openCypher?query=MATCH%20(n)%20RETURN%20n%20LIMIT
%201&explain=dynamic"
```

Neptune 中 openCypher **explain** 的限制

OpenCypher Explain 的目前版本具有下列限制：

- Explain 計畫目前僅適用於執行唯讀操作的查詢。不支援執行任何類型變動的查詢，例如 CREATE、DELETE、MERGE、SET 等等。
- 特定計畫的運算子和輸出可能會在未來版本中變更。

openCypher **explain** 輸出中的 DFE 運算子

若要使用 OpenCypher explain 功能提供的資訊，您必須了解一些有關 DFE 查詢引擎運作方式的詳細資料 (DFE 是 Neptune 用來處理 OpenCypher 查詢的引擎)。

DEF 引擎會將每個查詢轉換為運算子的管道。從第一個運算子開始，中繼解決方案會透過此運算子管道從一個運算子流至下一個運算子。Explain 資料表中的每一列都代表評估時間點之前的結果。

可以出現在 DFE 查詢計畫中的運算子如下：

DFEApply – 針對儲存在指定變數中的值，執行引數區段中指定的函數

DFE BindRelation-將具有指定名稱的變量綁定在一起

DFE ChunkLocal SubQuery-這是一個非阻塞操作，充當正在執行的子查詢的包裝。

DFE DistinctColumn — 根據指定的變數傳回輸入值的不同子集。

DFE DistinctRelation — 根據指定的變數傳回輸入解法的不同子集。

DFEDrain – 出現在子查詢結尾處，做為該子查詢的終止步驟。解決方案的數目會記錄為 `Units In`。`Units Out` 一律為零。

DFE ForwardValue — 將所有輸入區塊直接複製為輸出區塊，以傳遞給其下游運算子。

DFE GroupBy HashIndex — 根據先前計算的雜湊索引 (使用作業)，針對輸入解決方案執行群組依據作業。`DFEHashIndexBuild`作為輸出，指定的輸入會從包含每個輸入解決方案的群組索引鍵的資料欄延伸。

DFE 構HashIndex建-構建一個哈希索引在一組變量作為副作用。此雜湊索引通常會在稍後的作業中重複使用。請參閱 `DFEHashIndexJoin` 或 `DFEGroupByHashIndex` 以了解可能使用此雜湊索引的位置。

DFE HashIndex Join — 針對先前建立的雜湊索引，針對傳入的解決方案執行聯結。請參閱 `DFEHashIndexBuild` 以了解此雜湊索引的建置位置。

DFE JoinExists — 取左右輸入關係，並保留來自左側關係的值，該關係在右關係中具有由給定聯結變數所定義的對應值。

– 這是一種非封鎖操作，其會做為子查詢的包裝函數，允許它重複執行以在迴圈中使用。

DFE MergeChunks-這是一個阻塞操作，它將來自上游運算符的塊合併為一個解決方案塊以傳遞給其下游運算符（逆）。`DFESplitChunks`

DFEMinus — 採用左右側輸入關係，並保留左側關係的值，這些關係在右側關係中不具對應值 (如指定聯結變數所定義)。如果跨兩個關係的變數沒有重疊，那麼這個運算子只會傳回左側輸入關係。

DFE NotExists — 取一個左右輸入關係，並保留左關係中沒有對應值 (如指定聯結變數所定義) 的右側關係中的值。如果跨兩個關係的變數沒有重疊，那麼這個運算子會傳回空的關係。

DFE OptionalJoin — 執行左外部聯結 (也稱為可選聯結)：左側至少有一個聯結夥伴在右側的解決方案會加入，而左側沒有聯結夥伴的解決方案會依原樣轉送。這是一項封鎖操作。

DFE PipelineJoin — 針對引數定義的元組模式加入輸入。`pattern`

DFE PipelineRange Count — 計算符合指定模式的解決方案數目，並傳回包含 count 值的單一一進位解決方案。

DFE PipelineScan — 掃描資料庫中的指定`pattern`引數，無論是否有欄上的指定篩選器。

DFEProject – 採取多個輸入資料行並只投影所需的資料行。

DFEReduce – 對指定的變數執行指定的彙總函數。

DFE RelationalJoin — 使用合併聯結，根據指定的模式索引鍵，連接上一個運算子的輸入。這是一項封鎖操作。

DFE RouteChunks — 從其單一傳入邊緣取得輸入區塊，並沿著多個傳出邊緣路由這些區塊。

DFE SelectRows — 此運算子選擇性地從左側輸入關係解決方案中取出列，以轉送至其下游運算子。根據運算子右側輸入關係中提供的資料列識別碼選取的資料列。

DFESerialize — 將查詢的最終結果序列化為 JSON 字串序列，將每個輸入解決方案映射到適當的變數名稱。對於節點和邊緣結果，這些結果會序列化為實體屬性和中繼資料的映射。

DFESort — 取得輸入關係，並根據提供的排序索引鍵產生排序關係。

DFE SplitBy Group — 將每個單一輸入區塊從一個傳入邊緣分割為較小的輸出區塊，對應於由來自另一個傳入邊緣的對應輸入區塊的資料列 ID 識別的資料列群組。

DFE SplitChunks — 將每個單一輸入區塊分割成較小的輸出區塊 (反轉為)。`DFEMergeChunks`

DFE StreamingHash IndexBuild — 的串流版本`DFEHashIndexBuild`。

DFE StreamingGroup ByHash 指數 — 串流版本的`DFEGroupByHashIndex`.

DFESubquery – 此運算子會出現在所有計畫的開頭處，並封裝 DFE 引擎上執行的計畫部分，即 OpenCypher 的整個計畫。

DFE SymmetricHash Jo in — 使用雜湊聯結，根據指定的模式索引鍵，連接前一個運算子的輸入。這是一項非封鎖操作。

DFESync — 此運算子是支援非封鎖計劃的同步處理運算子。該運算子從兩個傳入邊緣取得解決方案，並將這些解決方案轉送至適當的下游邊緣。為達同步處理之目的，沿著這些邊緣之一的輸入可以在內部緩衝。

DFETee – 這是一個分支運算子，其會將相同的一組解決方案傳送至多個運算子。

DFE TermResolution — 對其輸入執行本地化或全域化作業，分別產生本地化或全域化識別碼的欄。

— 將輸入資料欄中的值清單展開至輸出資料欄，作為個別元素。

DFEUnion — 接受兩個或多個輸入關係，並使用所需的輸出結構描述產生這些關係的聯集。

SolutionInjection— 出現在解釋輸出中所有其他項目之前，「單位輸出」欄中的值為 1。不過，它提供為無操作，而且實際上並沒有將任何解決方案注入至 DFE 引擎。

TermResolution— 出現在計劃結束時，並將物件從 Neptune 引擎轉換為 OpenCypher 物件。

openCypher **explain** 輸出中的資料行

Neptune 產生以做為 OpenCypher Explain 輸出的查詢計畫資訊包含每列都有一個運算子的資料表。資料表包含以下資料行：

ID – 此運算子在計畫中的數字 ID。

Out #1 (和 Out #2) – 來自此運算子的下游運算子的 ID。最多可有兩個下游運算子。

Name – 此運算子的名稱。

Arguments – 運算子的任何相關詳細資訊。這包括輸入結構描述、輸出結構描述、模式 (適用於 `PipelineScan` 和 `PipelineJoin`) 等項目。

Mode – 描述基本運算子行為的標籤。此資料行大多為空白 (`-`)。例外是 `TermResolution`，其中模式可以是 `id2value_opencypher`，表示從 ID 解析為 OpenCypher 值。

Units In – 以輸入形式傳遞至這個運算子的解決方案數目。沒有上游運算子的運算子 (例如 `DFEPipelineScan`、`SolutionInjections`，以及未插入靜態值的 `DFESubquery`) 將具有零值。

Units Out – 產生為此運算子之輸出的解決方案數目。`DFEDrain` 是一種特殊情況，其中要被排除的解決方案數目記錄在 `Units In` 中，並且 `Units Out` 始終為零。

Ratio – `Units Out` 與 `Units In` 的比率。

Time (ms) – 此運算子取用的 CPU 時間，以毫秒為單位。

openCypher Explain 輸出的基本範例

以下是 openCypher `explain` 輸出的基本範例。對於具有機場代碼 ATL 的節點，查詢是其航線資料集內的單一節點查詢，而該代碼會使用 `details` 模式以預設 ASCII 輸出格式調用 `explain`：

```
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      #
#########################################################################################
```

在頂層，SolutionInjection 出現在其他所有項目之前，有 1 個單位輸出。請注意，它實際上並沒有注入任何解決方案。您可以看到下一個運算子 DFESubquery 具有 0 個單位輸入。

在 SolutionInjection 之後，位於頂層的是 DFESubquery 和 TermResolution 運算子。DFESubquery 會封裝要推送至 DFE 引擎之查詢執行計畫的多個部分 (對於 OpenCypher 查詢，整個查詢計畫是由 DFE 執行)。查詢計畫中的所有運算子都會在 DFESubquery 所參考的 subQuery1

內形成巢狀。唯一的例外是 TermResolution，其會將內部 ID 具體化為完全序列化的 OpenCypher 物件。

向下推送至 DFE 引擎的所有運算子都具有以 DFE 字首開頭的名稱。如上所述，整個 OpenCypher 查詢計畫是由 DFE 執行，因此，除了最終的 TermResolution 運算子之外，所有運算子都會以 DFE 開頭。

在 subQuery1 內部，可有零個或多個 DFEChunkLocalSubQuery 或 DFELoopSubQuery 運算子，封裝在記憶體受限機制中執行的推送執行計畫的一部分。這裡的 DFEChunkLocalSubQuery 包含一個用作子查詢輸入的 SolutionInjection。若要在輸出中尋找該子查詢的資料表，請搜尋在 DFEChunkLocalSubQuery 或 DFELoopSubQuery 運算子的 Arguments 資料行中指定的 subQuery=*graph URI*。

在 subQuery1 中，ID 為 0 的 DFEPipelineScan 會掃描資料庫找出指定的 pattern。該模式會對所有標籤掃描屬性 code 儲存為變數 ?n_code2 的實體 (您可以透過將 airport 附加到 n:airport 根據特定標籤進行篩選)。inlineFilters 引數顯示等於 ATL 的 code 屬性的篩選。

接下來，DFEChunkLocalSubQuery 運算子加入包含 DFEPipelineJoin 的子查詢的中繼結果。這可確保 ?n 實際上是一個節點，因為先前的 DFEPipelineScan 會掃描任何具有 code 屬性的實體。

具有限制之關係查詢的 **explain** 輸出範例

此查詢會尋找類型為 route 的兩個匿名節點之間的關係，且最多可傳回 10 個。同樣，explain 模式是 details，且輸出格式是預設的 ASCII 格式。以下是 explain 輸出：

在這裡，DFEPipelineScan 會掃描從匿名節點 ?anon_node7 開始，並在另一個匿名節點 ?anon_node21 結束的邊緣，關係類型儲存為 ?p_type1。有一個 ?p_type1 為 el://route 的篩選條件 (其中 el 代表邊緣標籤)，它對應至查詢字串中的 [p:route]。

DFEDrain 收集限制為 10 的輸出解決方案，如其 Arguments 資料行所示。一旦達到限制或產生了所有解決方案，不管哪一個先發生，DFEDrain 都會終止。

```
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      #
#################################################################################################
```

值表達式函數的 **explain** 輸出範例

函數為：

```
MATCH (a) RETURN DISTINCT labels(a)
```

在下面的 explain 輸出中，DFEPipelineScan (ID 0) 會掃描所有節點標籤。這對應至 MATCH (a)。

DFEChunkLocalSubquery (ID 1) 彙總每個 ?a 的 ?a 的標籤。這對應至 labels(a)。您可以透過 DFEApply 和 DFEReduce 看到此情況。

BindRelation (ID 2) 用來將資料行泛型 ?__gen_labelsOfa2 重命名為 ?labels(a)。

DFEDistinctRelation (ID 4) 僅擷取不同的標籤（多個：機場節點會給出重複的標籤 (a)：["airport"])。這對應至 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      #
##################################################################################################
```

數學值表達式函數的 **explain** 輸出範例

在此範例中，RETURN abs(-10) 會執行簡單的評估，取得常數 -10 的絕對值。

DFEChunkLocalSubQuery (ID 1) 對靜態值 -10 執行解決方案注入，該值儲存在變數 ?100 中。

DFEApply (ID 2) 是對 ?100 變數中儲存的靜態值執行絕對值函數 abs() 的運算子。

以下是查詢和產生的 explain 輸出：

```
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      #
#########################################################################################################
```

可變長度路徑 (VLP) 查詢的 **explain** 輸出範例

這是處理可變長度路徑查詢的更複雜查詢計畫的範例。為了清楚起見，此範例僅顯示 explain 輸出的一部分。

在 subQuery1 中，注入 ...graph_1 子查詢的 DFEPipelineScan (ID 0) 和 DFEChunkLocalSubQuery (ID 1) 負責掃描具有 YPO 程式碼的節點。

在 subQuery1 中，注入 ...graph_2 子查詢的 DFEChunkLocalSubQuery (ID 2) 負責掃描具有 LAX 程式碼的節點。

在 subQuery1 中，DFEChunkLocalSubQuery (ID 3) 會注入 ...graph3 子查詢，該子查詢包含 DFELoopSubQuery (ID 17)，其又會插入 ...graph5 子查詢。此操作負責解析兩個節點之間的查詢字串中的 -[*2]-> 可變長度模式。

```
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      #
...
```

Neptune openCypher 中的交易

Amazon Neptune 中的 OpenCypher 實作會使用 Neptune 定義的交易語義。不過，Bolt 驅動程式提供的隔離層級對 Bolt 交易語義有一些特定的含意，如以下各節所述。

唯讀 Bolt 交易查詢

有各種方式可以處理唯讀查詢，其中具有不同的交易模型和隔離層級，如下所示：

隱含唯讀交易查詢

以下是唯讀隱含交易的範例：

```
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();
}
```

因為僅供讀取複本只接受唯讀查詢，所以針對僅供讀取複本的所有查詢都會以讀取隱含交易的形式執行，而不管工作階段組態中設定哪種存取模式 Neptune 會在 SNAPSHOT 隔離語義下將讀取隱含交易評估為唯讀查詢。

若失敗，預設會重試讀取隱含交易。

自動遞交唯讀交易查詢

以下是唯讀自動遞交交易的範例：

```
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();
}
```

如果在工作階段組態中將存取模式設定為 READ，Neptune 會在 SNAPSHOT 隔離語義下將自動遞交交易查詢評估為唯讀查詢。請注意，僅供讀取複本只接受唯讀查詢。

如果您沒有傳入工作階段設定，則預設會在變動查詢隔離的情況下處理自動遞交查詢，因此務必傳入明確將存取模式設定為 READ 的工作階段組態。

若失敗，不會重試唯讀自動遞交查詢。

明確唯讀交易查詢

以下為明確唯讀交易的範例：

```
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();
}
```

如果在工作階段組態中將存取模式設定為 READ，Neptune 會在 SNAPSHOT 隔離語義下將明確唯讀交易評估為唯讀查詢。請注意，僅供讀取複本只接受唯讀查詢。

如果您沒有傳入工作階段設定，則預設會在變動查詢隔離的情況下處理唯讀交易，因此務必傳入明確將存取模式設定為 READ 的工作階段組態。

若失敗，預設會重試唯讀明確查詢。

變動 Bolt 交易查詢

與唯讀查詢一樣，有各種方式可以處理變動查詢，其中具有不同的交易模型和隔離層級，如下所示：

隱含變動交易查詢

以下為隱含變動交易的範例：

```
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();
}
```

做為變動查詢一部分進行的讀取是在 READ COMMITTED 隔離下執行，這是對 Neptune 變動交易的一般保證。

無論您是否特別傳入工作階段組態，一律都會將交易視為寫入交易。

如需衝突，請參閱 使用鎖定等待逾時的衝突解決機制。

自動遞交變動交易查詢

變動自動遞交查詢會繼承與變動隱含交易相同的行為。

如果您沒有傳入工作階段組態，預設會將交易視為寫入交易。

若失敗，不會自動重試變動自動遞交查詢。

明確變動交易查詢

以下為明確變動交易的範例：

```
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();
}
```

明確變動查詢會繼承與隱含變動交易相同的行為。

如果您沒有傳入工作階段組態，預設會將交易視為寫入交易。

如需衝突，請參閱 使用鎖定等待逾時的衝突解決機制。

Neptune openCypher 限制

OpenCypher 的 Amazon Neptune 版本仍然不支援 Cypher 查詢語言參考第 9 版中指定的一切，如 符合开放密码规范 中所詳述。未來的版本預計將解決其中許多限制。

Neptune openCypher 例外狀況

在 Amazon Neptune 上使用 OpenCypher 時，可能會發生各種例外狀況。以下是您可能從 HTTPS 端點或 Bolt 驅動程式收到的常見例外狀況 (來自 Bolt 驅動程式的所有例外都會報告為伺服器狀態例外狀況)：

HTTP 代碼	錯誤訊息	是否可擷取？	修正方式	
400	(語法錯誤，直接從 OpenCypher 剖析器傳播)	否	更正查詢語法，然後再試一次。	
500	`Operation terminated (out of memory)`	是	重新設計查詢以新增其他篩選條件，來減少所需的記憶體	
500	操作已終止 (超過截止日期)	是	增加資料庫叢集參數群組中的查詢逾時，或重試請求。	
500	操作已終止 (遭使用者取消)	是	重試 請求。	
500	資料庫重設正在進行中。請在叢集可用之後重試查詢。	是	重設完成時再試一次。	
500	操作失敗，因為並行操作發生衝突 (請再試一次)。交易目前正在復原。	是	使用指數退避和重試策略再試一次。	

HTTP 代碼	錯誤訊息	是否可擷取？	修正方式	
400	*(####)* 操作/功能不受支援例外狀況	否	不支援指定的操作。	
400	已嘗試在唯讀複本上進行 openCypher 更新	否	將目標端點變更為寫入器端點。	
400	MalformedQueryException（Neptune 不顯示內部解析器狀態）	否	更正查詢語法，然後再試一次。	
400	無法刪除節點，因為它仍然具有關係。若要刪除此節點，您必須先刪除其關係。	否	請使用 `MATCH(n) DETACH DELETE(n)` ，而不是使用 `MATCH (n) DELETE n`	

HTTP 代碼	錯誤訊息	是否可擷取？	修正方式	
400	無效的操作：正在嘗試移除節點的最後一個標籤。一個節點必須至少具有一個標籤。	否	Neptune 要求所有節點至少有一個標籤，如果節點建立時沒有用標籤明確標示，則會指派預設標籤 `vertex`。變更查詢和/或應用程式邏輯，以免刪除最後一個標籤。節點的單例標籤可以透過設定新標籤並刪除舊標籤予以更新。	
500	已經違反的最大請求數，ConfiguredQueueCapacity= {} 表示 connID = {}	是	無論堆疊和通訊協定為何，目前都只能處理 8,192 個並行請求。	
500	已違反最大連線限制。	是	只允許每個執行個體 1000 個並行 Bolt 連線 (對於 HTTP 沒有限制)。	
400	預期 [其中一個：節點、關係或路徑]，卻得到常值	否	檢查您是否正在傳遞正確的參數、更正查詢語法，然後再試一次。	

HTTP 代碼	錯誤訊息	是否可擷取？	修正方式	
400	屬性值必須是簡單的常值。或者：預期 Set 屬性的映射，但找不到一個。	否	SET 子句只接受簡單常數，不接受複合類型。	
400	找不到傳遞的實體進行刪除	否	檢查您正在嘗試刪除的實體是否存在於資料庫中。	
400	使用者沒有資料庫的存取權。	否	檢查正在使用的 IAM 角色的政策。	
400	沒有權杖做為請求的一部分傳遞	否	在啟用 IAM 的叢集上，必須將正確簽署的權杖做為查詢請求的一部分傳遞。	
400	傳播錯誤訊息。	否	請透過要求識別碼聯絡 Sup AWS port 部門。	
500	操作已終止 (內部錯誤)	是	請透過要求識別碼聯絡 Sup AWS port 部門。	

使用 SPARQL 存取 Neptune 圖形

SPARQL 是一種資源描述架構 (RDF) 的查詢語言，其是專為網路設計的圖形資料格式。Amazon Neptune 與 SPARQL 1.1 相容。這表示您可以連線到 Neptune 資料庫執行個體，並使用 SPARQL 1.1 查詢語言規格所述的查詢語言來查詢圖形。

SPARQL 的查詢包含 SELECT 子句，用於指定要傳回的變數，和 WHERE 子句，用於指定要比對圖形中的哪些資料。如果您不熟悉 SPARQL 查詢，請參閱 SPARQL 1.1 查詢語言中的編寫簡易查詢。

> **Important**
>
> 若要載入資料，SPARQL UPDATE INSERT 可能非常適用於小型資料集，但如果您需要從檔案載入大量資料，請參閱使用 Amazon Neptune 大量載入器擷取資料。

如需有關 Neptune 之 SPARQL 實作細節的詳細資訊，請參閱 SPARQL 標準合規。

開始之前，您必須準備好以下事項：

- Neptune 資料庫執行個體。如需建立 Neptune 資料庫執行個體的相關資訊，請參閱 建立新的 Neptune 資料庫叢集。
- 與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體。

主題

使用 RDF4J 主控台連線到 Neptune 資料庫執行個體

RDF4J 主控台可讓您在 REPL (read-eval-print 迴圈) 環境中試驗資源描述架構 (RDF) 圖形和查詢。

您可從 RDF4J 主控台新增遠端圖形資料庫做為儲存庫和查詢。本小節將逐步引導您完成 RDF4J 主控台的設定，以遠端連線到 Neptune 資料庫執行個體。

使用 RDF4J 主控台連線到 Neptune

1. 從 RDF4J 網站的下載頁面下載 RDF4J SDK。
2. 解壓縮 RDF4J SDK zip 檔案。
3. 在終端機上，導覽至 RDF4J SDK 目錄，然後輸入下列命令來執行 RDF4J 主控台：

   ```
   bin/console.sh
   ```

   您應該會看到類似下列的輸出：

   ```
   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.
   >
   ```

   您現在進入 > 提示。這是 RDF4J 主控台的一般提示。您可使用此提示設定儲存庫和其他操作。儲存庫有專用可執行查詢的提示。
4. 在 > 提示下，輸入以下命令為您的 Neptune 資料庫執行個體建立 SPARQL 儲存庫：

   ```
   create sparql
   ```
5. RDF4J 主控台會提示您輸入連接到 SPARQL 端點所需變數的值。

   ```
   Please specify values for the following variables:
   ```

   指定下列值：

變數名稱	Value

SPARQL 查詢端點	`https://your-neptune-endpoint :port/sparql`
SPARQL 更新端點	`https://your-neptune-endpoint :port/sparql`
本機儲存庫 ID [端點@localhost]	`neptune`
儲存庫標題 [SPARQL 端點儲存庫 @localhost]	`Neptune DB instance`

如需尋找 Neptune 資料庫執行個體地址的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

如果操作成功，您會看到以下訊息：

```
Repository created
```

6. 在 > 提示下，輸入以下命令以連線到 Neptune 資料庫執行個體：

```
open neptune
```

如果操作成功，您會看到以下訊息：

```
Opened repository 'neptune'
```

您現在進入 `neptune>` 提示。在此提示下，您可以針對 Neptune 圖形執行查詢。

> **Note**
>
> 既然您已新增儲存庫，您下次執行 `bin/console.sh` 時便能立即執行 `open neptune` 命令以連線到 Neptune 資料庫執行個體。

7. 在 `neptune>` 提示下，輸入下列命令執行 SPARQL 查詢，使用 `?s ?p ?o` 查詢和限制 10，以傳回圖形中的最多 10 個三元組 (subject-predicate-object)。若要查詢其他內容，請將 `sparql` 命令之後的文字換成其他的 SPARQL 查詢。

```
sparql select ?s ?p ?o where {?s ?p ?o} limit 10
```

使用 RDF4J Workbench 連線到 Neptune 資料庫執行個體

本小節逐步引導您使用 RDF4J Workbench 和 RDF4J Server 連線到 Amazon Neptune 資料庫執行個體。RDF4J Server 是必要的，因為其做為 Neptune SPARQL HTTP REST 端點和 RDF4J Workbench 之間的 Proxy。

RDF4J Workbench 提供了簡單的介面以用來試驗圖形，包括可載入本機檔案。如需詳細資訊，請參閱 RDF4J 文件中的新增區段。

必要條件

開始之前，請執行以下動作：

- 安裝 Java 1.8 或更新版本。
- 安裝 RDF4J Server 和 RDF4J Workbench。如需詳細資訊，請參閱安裝 RDF4J Server 和 RDF4J Workbench。

使用 RDF4J Workbench 連線到 Neptune

1. 在 Web 瀏覽器中，導覽到部署 RDF4J Workbench Web 應用程式的 URL。例如，如果您使用 Apache Tomcat，URL：https://*ec2_hostname*:8080/rdf4j-workbench/。
2. 若系統要求您 Connect to RDF4J Server (連接到 RDF4J 伺服器)，請確認 RDF4J 伺服器已安裝並執行中，且伺服器 URL 正確無誤。接著繼續進行下一個步驟。
3. 在左側窗格中，選擇 New repository (新增儲存庫)。

   在 New repository (新增儲存庫) 中：

   - 在 Type (類型) 下拉式清單中，選擇 SPARQL endpoint proxy (SPARQL 端點 Proxy)。
   - 在 ID 中，輸入 neptune。
   - 針對標題，輸入 Neptune 資料庫執行個體。

   選擇下一步。
4. 在 New repository (新增儲存庫) 中：

- 在 SPARQL query endpoint URL (SPARQL 查詢端點 URL) 中，輸入 `https://your-neptune-endpoint:port/sparql`。
- 在 SPARQL update endpoint URL (SPARQL 更新端點 URL) 中，輸入 `https://your-neptune-endpoint:port/sparql`。

如需尋找 Neptune 資料庫執行個體地址的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

選擇建立。

5. neptune 儲存庫現在會顯示在儲存庫清單中。可能需要幾分鐘的時間才能使用新的儲存庫。
6. 在表格的 Id 欄中，選擇 neptune 連結。
7. 從左側窗格選擇 Query (查詢)。

> **Note**
>
> 如果 Explore (探索) 底下的功能表項目停用，您可能需要重新連線到 RDF4J 伺服器，並再次選擇 neptune 儲存庫。
> 若要這樣做，您可以使用右上角的 [change] ([變更]) 連結。

8. 在查詢欄位中，輸入以下 SPARQL 查詢，然後選擇 Execute (執行)。

```
select ?s ?p ?o where {?s ?p ?o} limit 10
```

先前範例使用 `?s ?p ?o` 查詢和限制 10，以傳回圖形中的最多 10 個三元組 (subject-predicate-object)。

使用 Java 連線至 Neptune 資料庫執行個體

本節逐步引導您執行完整的 Java 範例，以連線到 Amazon Neptune 資料庫執行個體並執行 SPARQL 查詢。

請從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

使用 Java 連線至 Neptune

1. 在 EC2 執行個體上安裝 Apache Maven。首先，輸入以下內容，以新增包含 Maven 套件的儲存庫：

   ```
   sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   ```

   輸入以下內容以設定套件的版本編號：

   ```
   sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
   ```

   接著即可使用 yum 安裝 Maven：

   ```
   sudo yum install -y apache-maven
   ```

2. 本範例僅經過 Java 8 的測試。輸入以下內容將 Java 8 安裝在 EC2 執行個體上：

   ```
   sudo yum install java-1.8.0-devel
   ```

3. 輸入以下內容將 Java 8 設定為 EC2 執行個體上預設的執行時間：

   ```
   sudo /usr/sbin/alternatives --config java
   ```

   出現提示時，輸入 Java 8 的數字。

4. 輸入以下內容將 Java 8 設定為 EC2 執行個體上預設的編譯器：

   ```
   sudo /usr/sbin/alternatives --config javac
   ```

   出現提示時，輸入 Java 8 的數字。

5. 在新目錄中，建立一個 pom.xml 檔案，然後在文字編輯器中開啟檔案。

6. 將以下內容複製到 pom.xml 檔案中並儲存它 (您通常可以將版本編號調整為最新的穩定版本)：

   ```
   <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**
>
> 如果您要修改現有的 Maven 專案，所需的相依性將在先前的程式碼中反白顯示。

7. 若要為範例來源碼 (`src/main/java/com/amazonaws/`) 建立子目錄，請在命令列輸入下列命令：

```
mkdir -p src/main/java/com/amazonaws/
```

8. 在 src/main/java/com/amazonaws/ 目錄中，建立一個名為 App.java 的檔案，然後在文字編輯器中開啟檔案。
9. 將以下內容複製到 App.java 檔案。將 *your-neptune-endpoint* 取代為 Neptune 資料庫執行個體的地址。

> **Note**
>
> 如需尋找 Neptune 資料庫執行個體主機名稱的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

```
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. 使用下列 Maven 命令編譯並執行範例：

```
mvn compile exec:java
```

先前範例使用 `?s ?p ?o` 查詢和限制 10，以傳回圖形中的最多 10 個三元組 (subject-predicate-object)。若要查詢其他項目，請將查詢換成其他 SPARQL 查詢。

範例中重複出現的結果將印出所傳回之各變數的值。`Value` 物件會轉換成 `String`，然後列印。如果您變更查詢的 `SELECT` 部分，您必須修改程式碼。

SPARQL HTTP API

SPARQL HTTP 請求在以下端點接受：`https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`

如需使用 SPARQL 連線到 Amazon Neptune 的詳細資訊，請參閱 使用 SPARQL 存取 Neptune 圖形。

如需 SPARQL 通訊協定和查詢語言的詳細資訊，請參閱 SPARQL 1.1 通訊協定和 SPARQL 1.1 查詢語言規格。

下列主題提供 SPARQL RDF 序列化格式，以及如何搭配 Neptune 使用 SPARQL HTTP API 的相關資訊。

內容

使用 HTTP REST 端點連線到 Neptune 資料庫執行個體

Amazon Neptune 會提供 HTTP 端點進行 SPARQL 查詢。REST 介面相容於 SPARQL 1.1 版。

> **Important**
>
> 版本：1.0.4.0 (2020 年 10 月 12 日) 已使與 Amazon Neptune 的所有連線都必須使用 TLS 1.2 和 HTTPS。再也不可能使用不安全的 HTTP，或搭配 1.2 之前的 TLS 版本使用 HTTPS，連線到 Neptune。

以下說明引導您使用 curl 命令、透過 HTTPS 連線和使用 HTTP 語法連線到 SPARQL 端點。請從與您的 Neptune 資料庫執行個體位於同一虛擬私有雲端 (VPC) 的 Amazon EC2 執行個體依照以下指示進行。

對 Neptune 資料庫執行個體進行 SPARQL 查詢時所用的 HTTP 端點為：`https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`。

> **Note**
>
> 如需尋找 Neptune 資料庫執行個體主機名稱的相關資訊，請參閱 連線至 Amazon Neptune 端點 一節。

使用 HTTP POST 執行 QUERY

以下範例使用 curl 透過 HTTP POST 提交 SPARQL **QUERY**。

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

先前範例使用 ?s ?p ?o 查詢和限制 10，以傳回圖形中的最多 10 個三元組 (subject-predicate-object)。若要查詢其他項目，請將查詢換成其他 SPARQL 查詢。

> **Note**
>
> SELECT 和 ASK 查詢的回應預設 MIME 類型為 application/sparql-results+json。
> CONSTRUCT 和 DESCRIBE 查詢回應的預設 MIME 類型為 application/n-quads。
> 如需 Neptune 用於序列化的媒體類型清單，請參閱 Neptune SPARQL 使用的 RDF 序列化格式。

使用 HTTP POST 執行 UPDATE

以下範例使用 curl 透過 HTTP POST 提交 SPARQL **UPDATE**。

```
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
```

上述範例插入以下三元組到 SPARQL 預設圖形：<https://test.com/s> <https://test.com/p> <https://test.com/o>

用於多部分 SPARQL 回應的選用 HTTP 結尾標頭

> **Note**
>
> 此功能從 Neptune 引擎 1.0.3.0 版開始可用。

通常會以多個部分或區塊傳回 SPARQL 查詢和更新的 HTTP 回應。很難診斷在查詢或更新開始傳送這些區塊之後發生的失敗，特別是因為第一個區塊到達時，其 HTTP 狀態碼為 200。

除非您明確請求結尾標頭，否則 Neptune 只會透過將錯誤訊息附加到訊息本文 (通常已損毀) 來報告此類失敗。

若要使偵測和診斷此類問題更容易進行，您可以在請求中包括傳輸編碼 (TE) 結尾標頭 (`te: trailers`) (例如，請參閱有關 TE 請求標頭的 MDN 頁面)。這樣做會導致 Neptune 在回應區塊的結尾標頭內包括兩個新的標頭欄位：

- `X-Neptune-Status` – 包含回應碼，後面接著簡短名稱。例如，若成功，結尾標頭將是：`X-Neptune-Status: 200 OK`。若失敗，回應代碼將是 Neptune 引擎錯誤代碼，例如 `X-Neptune-Status: 500 TimeLimitExceededException`。
- `X-Neptune-Detail` – 對於成功的請求而言是空的。若發生錯誤，它會包含 JSON 錯誤訊息。由於 HTTP 標頭值中只允許 ASCII 字元，因此 JSON 字串會進行 URL 編碼。錯誤訊息仍會附加至回應訊息本文。

SPARQL 在 Neptune 中使用的 RDF 媒體類型

資源描述架構 (RDF) 資料可以多種不同方法序列化，大部分都能為 SPARQL 取用或輸出：

Neptune SPARQL 使用的 RDF 序列化格式

- RDF/XML – RDF 的 XML 序列化，以 RDF 1.1 XML 語法定義。媒體類型：`application/rdf+xml`。一般副檔名：`.rdf`。
- N-Triples – 以行為基礎的純文字格式編碼 RDF 圖形，以 RDF 1.1 N-Triples 定義。媒體類型：`application/n-triples`、`text/turtle` 或 `text/plain`。一般副檔名：`.nt`。
- N-Quads – 以行為基礎的純文字格式編碼 RDF 圖形，以 RDF 1.1 N-Quads 定義。這是 N-Triples 的延伸。媒體類型：`application/n-quads`，若使用 7 位元 US-ASCII 編碼，則為 `text/x-nquads`。一般副檔名：`.nq`。
- Turtle – 以 RDF 1.1 Turtle 定義的 RDF 文字語法，使用精簡而自然的文字型式，搭配常見使用模式和資料類型的縮寫，完整編寫 RDF 圖形。Turtle 使用 N-Triples 及 SPARQL 三元組模式語法提供多層次的相容性。媒體類型：`text/turtle`一般副檔名：`.ttl`。
- TriG – 以 RDF 1.1 TriG 定義的 RDF 文字語法，使用精簡而自然的文字型式，搭配常見使用模式和資料類型的縮寫，完整編寫 RDF 圖形。TriG 是 Turtle 格式的延伸。媒體類型：`application/trig`。一般副檔名：`.trig`。
- N3 (Notation3) – 以 Notation3 (N3)：可閱讀的 RDF 語法定義的宣告和邏輯語言。N3 透過新增方程式 (為圖表本身的常值)、變數、邏輯暗示和功能述詞，擴充 RDF 資料模型，並提供 RDF/XML 的文字語法替代方案。媒體類型：`text/n3`。一般副檔名：`.n3`。
- JSON-LD – 以 JSON-LD 1.0 定義的資料序列化和簡訊格式。媒體類型：`application/ld+json`。一般副檔名：`.jsonld`。

- TriX – XML 中的 RDF 序列化，以 TriX：XML 中的 RDF 三元組定義。媒體類型：application/trix。一般副檔名：.trix。
- SPARQL JSON 結果 – 使用 SPARQL 1.1 查詢結果 JSON 格式的 RDF 序列化。媒體類型：application/sparql-results+json。一般副檔名：.srj。
- RDF4J 二進位格式 – 編碼 RDF 資料的二進位格式，以 RDF4J 二進位 RDF 格式記錄。媒體類型：application/x-binary-rdf。

Neptune SPARQL 使用的 SPARQL 結果序列化格式

- SPARQL XML 結果 – SPARQL 查詢語言提供的變數繫結 XML 格式和布林值結果格式，以 SPARQL 查詢結果 XML 格式 (第 2 版) 定義。媒體類型：application/sparql-results+xml。一般副檔名：.srx。
- SPARQL CSV 和 TSV 結果 – 使用逗號分隔值和定位字元分隔值來表達 SELECT 查詢的 SPARQL 查詢結果，以 SPARQL 1.1 查詢結果 CSV 和 TSV 格式定義。媒體類型：逗號分隔值為 text/csv，定位字元分隔值為 text/tab-separated-values。一般副檔名：逗號分隔值為 .csv，定位字元分隔值為 .tsv。
- 二進位結果表 – 編碼 SPARQL 查詢輸出的二進位格式。媒體類型：application/x-binary-rdf-results-table。
- SPARQL JSON 結果 – 使用 SPARQL 1.1 查詢結果 JSON 格式的 RDF 序列化。媒體類型：application/sparql-results+json。

Neptune 可用來匯入 RDF 資料的媒體類型

Neptune 大量載入器支援的媒體類型

- N-Triples
- N-Quads
- RDF/XML
- Turtle

SPARQL UPDATE LOAD 可以匯入的媒體類型

- N-Triples
- N-Quads
- RDF/XML

- Turtle
- TriG
- N3
- JSON-LD

Neptune 可用來匯出查詢結果的媒體類型

若要指定 SPARQL 查詢回應的輸出格式，請以查詢請求傳送 "Accept: *media-type*" 標頭。例如：

```
curl -H "Accept: application/nquads" ...
```

SPARQL SELECT 可從 Neptune 輸出的 RDF 媒體類型

- SPARQL JSON 結果 (這是預設值)
- SPARQL XML 結果
- 二進位結果表 (媒體類型：`application/x-binary-rdf-results-table`)
- 逗號分隔值 (CSV)
- 定位字元分隔值 (TSV)

SPARQL ASK 可從 Neptune 輸出的 RDF 媒體類型

- SPARQL JSON 結果 (這是預設值)
- SPARQL XML 結果
- 布林值 (媒體類型：`text/boolean`，表示 "true" 或 "false")

SPARQL CONSTRUCT 可從 Neptune 輸出的 RDF 媒體類型

- N-Quads (這是預設值)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3

- TriX
- TriG
- SPARQL JSON 結果
- RDF4J 二進位 RDF 格式

SPARQL DESCRIBE 可從 Neptune 輸出的 RDF 媒體類型

- N-Quads (這是預設值)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- SPARQL JSON 結果
- RDF4J 二進位 RDF 格式

使用 SPARQL UPDATE LOAD 將資料匯入至 Neptune

SPARQL UPDATE LOAD 命令的語法是在 SPARQL 1.1 更新建議中指定：

```
LOAD SILENT (URL of data to be loaded) INTO GRAPH (named graph into which to load the
 data)
```

- **`SILENT`** – (選用) 即使在處理期間發生錯誤，也會導致操作傳回成功。

  這在單一交易包含多個陳述式 (例如，`"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;"`) 時很有用，而且即使某些遠端資料無法處理，您也想要交易完成。
- *`###### URL`* – (必要) 指定一個遠端資料檔案，其中包含要載入至圖形的資料。

  遠端檔案必須具有下列其中一個副檔名：

  - NTriples 為 `.nt`。
  - NQuads 為 `.nq`。

- Trig 為 `.trig`。
- RDF/XML 為 `.rdf`。
- Turtle 為 `.ttl`。
- N3 為 `.n3`。
- JSON-LD 為 `.jsonld`。

- **`INTO GRAPH`***`##########`* – (選用) 指定應載入資料的圖形。

  Neptune 會將每個三元組與一個具名圖形建立關聯。您可以使用後援具名圖形 URI 來指定預設的具名圖形 (`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`)，如下所示：

  ```
  INTO GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

> **Note**
>
> 當您需要載入大量資料時，我們建議您使用 Neptune 大量載入器，而非 UPDATE LOAD。如需有關大量載入器的詳細資訊，請參閱 使用 Amazon Neptune 大量載入器擷取資料。

您可以使用 `SPARQL UPDATE LOAD` 直接從 Amazon S3 載入資料，或從取自於自我託管 Web 伺服器的檔案載入資料。要載入的資源必須和 Neptune 伺服器位於同一區域，而且 VPC 中必須允許資源的端點。如需建立 Amazon S3 端點的相關資訊，請參閱 建立 Amazon S3 VPC 端點。

所有 `SPARQL UPDATE LOAD` URI 都必須以 `https://` 開頭。這包括 Amazon S3 URL。

與 Neptune 大量載入器相反，呼叫 `SPARQL UPDATE LOAD` 是可完全交易的。

使用 SPARQL UPDATE LOAD 直接從 Amazon S3 將檔案載入至 Neptune

由於 Neptune 不允許您在使用 SPARQL UPDATLOAD 時將 IAM 角色傳遞給 Amazon S3，因此有問題的 Amazon S3 儲存貯體必須是公開的，或者您必須在 LOAD 查詢中使用預先簽署的 Amazon S3 URL。

若要為 Amazon S3 檔案產生預先簽署的 URL，您可以使用如下 AWS CLI 命令：

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to load)
```

然後，您可以在 LOAD 命令中使用產生的預先簽署 URL：

```
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)'
```

如需詳細資訊，請參閱對請求進行身分驗證：使用查詢參數。Boto3 文件展示了如何使用 Python 指令碼來產生一個預先簽署的 URL。

此外，必須正確設定要載入的檔案內容類型。

1. 當您使用 -metadata 參數將檔案上傳至 Amazon S3 時，請設定檔案的內容類型，如下所示：

   ```
   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. 確認媒體類型資訊真實存在。執行：

   ```
   curl -v bucket-name/folder-name
   ```

   此命令的輸出應該顯示您在上傳檔案時所設定的媒體類型資訊。

3. 然後，您可以使用 SPARQL UPDATE LOAD 命令將這些檔案匯入至 Neptune：

   ```
   curl https://your-neptune-endpoint:port/sparql \
     -d "update=LOAD <https://s3.amazonaws.com/bucket-name/my-rdf-input/test.rdf>"
   ```

上述步驟僅適用於公有 Amazon S3 儲存貯體，或者您在 LOAD 查詢中使用預先簽署 Amazon S3 URL 存取的儲存貯體。

您還可以設定一個 Web 代理服務器從私有 Amazon S3 儲存貯體，如下所示：

搭配 SPARQL UPDATE LOAD 使用 Web 伺服器將檔案載入至 Neptune

1. 在 VPC 內執行的電腦上安裝 Web 伺服器，此為要載入檔案的 VPC，且是 Neptune 託管所在。例如，使用 Amazon Linux，您可能會以下列方式安裝 Apache：

   ```
   sudo yum install httpd mod_ssl
   ```

```
sudo /usr/sbin/apachectl start
```

2. 定義您要載入之 RDF 檔案內容的 MIME 類型。SPARQL 使用 Web 伺服器傳送的 `Content-type` 標頭來決定內容的輸入格式，所以您必須定義 Web 伺服器的相關 MIME 類型。

   例如，假設您使用以下副檔名來識別檔案格式：

   - NTriples 為 `.nt`。
   - NQuads 為 `.nq`。
   - Trig 為 `.trig`。
   - RDF/XML 為 `.rdf`。
   - Turtle 為 `.ttl`。
   - N3 為 `.n3`。
   - JSON-LD 為 `.jsonld`。

   如果您使用 Apache 2 做為 Web 伺服器，您要編輯檔案 `/etc/mime.types` 並新增以下類型：

   ```
   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. 確認 MIME 類型映射有效。一旦啟動並執行 Web 伺服器，並以您選擇的格式託管 RDF 檔案，您就可以從本機主機向 Web 伺服器傳送請求，測試組態。

   例如，您可能會傳送類似以下的請求：

   ```
   curl -v http://localhost:80/test.rdf
   ```

   然後，在 `curl` 的詳細輸出中，您應該會看到如下一行：

   ```
   Content-Type: application/rdf+xml
   ```

   這會顯示已成功定義的內容類型映射。

4. 您現在可以使用 SPARQL UPDATE 命令載入資料：

```
curl https://your-neptune-endpoint:port/sparql \
    -d "update=LOAD <http://web_server_private_ip:80/test.rdf>"
```

> **Note**
>
> 使用 SPARQL UPDATE LOAD 可以在載入大型來源檔案時於 Web 伺服器上觸發逾時。Neptune 會在檔案資料串流輸入的同時進行處理，對大型檔案而言，所需時間可能會比伺服器上設定的逾時時間長。這樣反而可能會造成伺服器關閉連線，當 Neptune 在串流中遇到未預期的 EOF 時，可能會導致下列錯誤訊息：
>
> ```
> {
>   "detailedMessage":"Invalid syntax in the specified file",
>   "code":"InvalidParameterException"
> }
> ```
>
> 如果您收到此訊息，但不認為來源檔案包含無效的語法，請嘗試增加 Web 伺服器上的逾時時間設定。您也可以透過在伺服器上啟用偵錯日誌並尋找逾時時間來診斷問題。

使用 SPARQL UPDATE UNLOAD 從 Neptune 刪除資料

Neptune 也提供自訂 SPARQL 操作 (UNLOAD)，用於移除遠端來源中指定的資料。UNLOAD 可被視為 LOAD 操作的對應項。它的語法是：

> **Note**
>
> 此功能從 Neptune 引擎 1.0.4.1 版開始可用。

```
UNLOAD SILENT (URL of the remote data to be unloaded) FROM GRAPH (named graph from
 which to remove the data)
```

- **SILENT** – (選用) 即使在處理資料時發生錯誤，也會導致操作傳回成功。

  這在單一交易包含多個陳述式 (例如，"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;") 時很有用，而且即使某些遠端資料無法處理，您也想要交易完成。
- ######## *URL* – (必要) 指定一個遠端資料檔案，其中包含要從圖形卸載的資料。

遠端檔案必須具有下列其中一個副檔名 (這些是 UPDATE-LOAD 支援的相同格式)：

- NTriples 為 `.nt`。
- NQuads 為 `.nq`。
- Trig 為 `.trig`。
- RDF/XML 為 `.rdf`。
- Turtle 為 `.ttl`。
- N3 為 `.n3`。
- JSON-LD 為 `.jsonld`。

UNLOAD 操作將會從您的資料庫叢集中移除此檔案包含的所有資料。

任何 Amazon S3 身分驗證都必須包含在 URL 中，才能卸載資料。您可以預先簽署 Amazon S3 檔案，然後使用產生的 URL 安全地存取該檔案。例如：

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to unload)
```

然後：

```
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)'
```

如需詳細資訊，請參閱對請求進行身分驗證：使用查詢參數。

- **FROM GRAPH** *(############)* – (選用) 指定應從中卸載遠端資料的具名圖形。

  Neptune 會將每個三元組與一個具名圖形建立關聯。您可以使用後援具名圖形 URI 來指定預設的具名圖形 (`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`)，如下所示：

  ```
  FROM GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

UNLOAD 會以 LOAD 對應至 INSERT DATA { *(inline data)* } 的相同方式對應至 DELETE DATA { *(inline data)* }。如同 DELETE DATA，UNLOAD 無法對包含空白節點的資料運作。

例如，如果本機 Web 伺服器提供名為 data.nt 的檔案，其中包含以下兩個三元組：

```
<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> .
```

以下 UNLOAD 命令將從具名圖形 (<http://example.org/graph1>) 中刪除這兩個三元組：

```
UNLOAD <http://localhost:80/data.nt> FROM GRAPH <http://example.org/graph1>
```

這將與使用以下 DELETE DATA 命令具有相同的效果：

```
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> .
  }
}
```

UNLOAD 命令擲回的例外狀況

- **InvalidParameterException** – 資料中有空白節點。HTTP 狀態：400 錯誤的請求。

  訊息： Blank nodes are not allowed for UNLOAD

- **InvalidParameterException** – 資料中有不完整的語法。HTTP 狀態：400 錯誤的請求。

  訊息：Invalid syntax in the specified file.

- **UnloadUrlAccessDeniedException** – 存取遭拒。HTTP 狀態：400 錯誤的請求。

  訊息：Update failure: Endpoint *(Neptune endpoint)* reported access denied error. Please verify access.

- **`BadRequestException`** – 無法擷取遠端資料。HTTP 狀態：400 錯誤的請求。

  訊息：(取決於 HTTP 回應)。

SPARQL 查詢提示

您可以在 Amazon Neptune 中使用查詢提示來指定特定 SPARQL 查詢的最佳化和評估策略。

查詢提示使用在 SPARQL 查詢中嵌入的額外三重模式表示，包含以下部分：

```
scope hint value
```

- scope – 決定查詢提示要套用到查詢的哪些部分，例如查詢中的特定群組或完整查詢。
- hint – 識別要套用的提示類型。
- value – 確定正在考慮的系統層面行為。

這些查詢提示和範圍會公開成為 Amazon Neptune 命名空間 `http://aws.amazon.com/neptune/vocab/v01/QueryHints#` 中的預先定義用語。本節範例包含命名空間做為 `hint` 前綴，此前綴已定義並包含在查詢中：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

例如，以下說明如何在 `SELECT` 查詢中包含 `joinOrder` 提示：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... {
 hint:Query hint:joinOrder "Ordered" .
 ...
}
```

上述查詢會指示 Neptune 引擎依「給定」順序評估查詢中的聯結，並停用任何自動重新排序。

使用查詢提示時請考慮以下情況：

- 您可以將不同的查詢提示合併在單一查詢中。例如，您可以使用 `bottomUp` 查詢提示來將子查詢標註為由下而上的評估，並使用 `joinOrder` 查詢提示來修復子查詢中的聯結順序。
- 您可以在不同的非重疊範圍中，多次使用相同的查詢提示。

- 查詢提示是提示。雖然查詢引擎通常的目標是要考慮給定的查詢提示，也可以予以忽略。
- 查詢提示為語意保留。新增查詢提示不會變更查詢的輸出 (除了沒有提供排序保證時的可能結果順序 - 也就是沒有使用 ORDER BY 明確強制執行結果順序)。

以下章節提供 Neptune 中可用的查詢提示以及其用法的詳細資訊。

主題

Neptune 中的 SPARQL 查詢提示範圍

下表顯示 Amazon Neptune 中 SPARQL 查詢提示的可行範圍、相關聯的提示和說明。這些項目中的 `hint` 字首代表提示的 Neptune 命名空間：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

範圍	支援的提示	描述
`hint:Query`	joinOrder	查詢提示適用於整個查詢。
`hint:Query`	queryTimeout	逾時值適用於整個查詢。
`hint:Query`	rangeSafe	針對整個查詢停用類型提升。
`hint:Query`	queryId	查詢 ID 值適用於整個查詢。
`hint:Query`	useDFE	已啟用 (或停用) 將 DFE 用於整個查詢。

範圍	支援的提示	描述
`hint:Group`	joinOrder	查詢提示適用於指定群組中的最上層元素，但不適用於巢狀元素 (如子查詢) 或父元素。
`hint:SubQuery`	evaluationStrategy	提示指定及套用到巢狀的 SELECT 子查詢。子查詢是獨立評估的，不會考慮在子查詢之前計算的解決方案。

joinOrder SPARQL 查詢提示

當您提交 SPARQL 查詢時，Amazon Neptune 查詢引擎會調查查詢的結構。它會重新排序部分查詢，並嘗試將評估所需的工作量和查詢回應時間減到最低。

例如，一系列的連接三重模式通常不會依給定的順序評估。它使用啟發和統計資料 (例如個別模式的選擇性以及它們如何透過共用變數連接) 進行重新排序。此外，如果您的查詢包含更複雜的模式，例如子查詢、FILTER、或複雜的 OPTIONAL 或 MINUS 區塊，Neptune 查詢引擎會盡可能將它們重新排序，致力使評估順序更有效率。

用於更複雜的查詢時，Neptune 選擇評估查詢的順序不一定是最佳的順序。例如，Neptune 可能會錯過在查詢評估期間發生的執行個體資料特定特性 (如圖表中出現 Power 節點)。

如果您知道資料的確切特性，並希望手動指示查詢執行的順序，請使用 Neptune `joinOrder` 查詢提示來指定依給定順序評估查詢。

joinOrder SPARQL 提示語法

`joinOrder` 查詢提示是以三重模式指定，該模式包含在 SPARQL 查詢中。

為更清楚起見，以下語法使用已定義並包含在查詢中的 `hint` 字首，指定 Neptune 查詢提示命名空間：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
scope hint:joinOrder "Ordered" .
```

可用範圍

- `hint:Query`

- `hint:Group`

如需更多查詢提示範圍的詳細資訊，請參閱 Neptune 中的 SPARQL 查詢提示範圍。

`joinOrder` SPARQL 提示範例

本節示範使用和不使用 `joinOrder` 查詢提示編寫的查詢以及相關的最佳化。

此範例假設資料集包含下列項目：

- 一個人名為 `John`，且 `:likes` 1,000 個人，包括 `Jane`。
- 一個人名為 `Jane`，且 `:likes` 10 個人，包括 `John`。

無查詢提示

以下 SPARQL 查詢會從一組社交網路資料中擷取所有名為 `John` 和 `Jane` 且彼此喜歡 (like) 的雙人組：

```
PREFIX : <https://example.com/>
SELECT ?john ?jane {
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Neptune 查詢引擎可能會以不同於寫入的順序評估陳述式。例如，它可能選擇以下列順序評估：

1. 尋找名為 `John` 的所有人。
2. 尋找透過 `:likes` 邊緣連接到 `John` 的所有人。
3. 以人名 `Jane` 篩選這些人。
4. 以「透過 `:likes` 邊緣連接到 `John`」篩選這些人。

根據資料集，以此順序評估的結果是在第二個步驟會擷取 1,000 個實體。第三個步驟則將這些再縮減到單一節點 `Jane`。然後，最終步驟判斷 `Jane` 也 `:likes` `John` 節點。

查詢提示

從 Jane 節點開始是有利的，因為她只有 10 個傳出 :likes 邊緣。這可藉由避免在第二步驟擷取 1,000 個實體，減少查詢評估期間的工作量。

以下範例會使用 joinOrder 查詢提示，藉由停用查詢的所有自動聯結重新排序，來確保先處理 Jane 節點及其傳出邊緣：

```
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 .
}
```

適用的真實世界案例可以是在社交網路的應用，網路中的人被分類為具有許多連接的影響者或具有少量連接的普通使用者。在這類案例中，以先前範例中的查詢為例，可以確保先處理普通使用者 (Jane) 再處理影響者 (John)。

查詢提示和重新排序

這個範例可以再進一步處理。如果您知道單一節點的 :name 屬性是唯一的，可以透過重新排序和使用 joinOrder 查詢提示來加速查詢。此步驟可確保先擷取唯一的節點。

```
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 .
}
```

在這種情況下，您可以將查詢降低到每一個步驟只有單一動作，如下：

1. 尋找符合 :name Jane 的單一個人節點。
2. 尋找符合 :name John 的單一個人節點。
3. 確認第一個節點透過 :likes 邊緣連接到第二個節點。

4. 確認第二個節點透過 `:likes` 邊緣連接到第一個節點。

> **Important**
>
> 如果您選擇錯誤的順序，`joinOrder` 查詢提示可能導致效能大幅下降。例如，如果 `:name` 屬性不是唯一，前述範例的效率會很差。如果 100 個節點的名稱都是 `Jane`，且 1,000 個節點的名稱都是 `John`，則此查詢會造成對 `:likes` 邊緣檢查 1,000* 100 (共 100,000 個) 雙人組。

`evaluationStrategy` SPARQL 查詢提示

`evaluationStrategy` 查詢提示會告訴 Amazon Neptune 查詢引擎，應該將標註的查詢片段當做獨立單位由下而上評估。這表示之前評估步驟的解決方案不會用來計算查詢片段。查詢片段被當做獨立單位而評估，其生成的解決方案在計算後與查詢的其餘部分聯結。

使用 `evaluationStrategy` 查詢提示暗示區塊化 (非管道化的) 查詢計畫，這意味著使用查詢提示標註的片段解決方案，在主記憶體中會具體化和緩衝。使用此查詢提示可能會大幅增加評估查詢所需的主記憶體，尤其是在標註的查詢片段計算大量結果時。

`evaluationStrategy` SPARQL 提示語法

`evaluationStrategy` 查詢提示是以三重模式指定，該模式包含在 SPARQL 查詢中。

為更清楚起見，以下語法使用已定義並包含在查詢中的 `hint` 字首，指定 Neptune 查詢提示命名空間：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

可用範圍

- `hint:SubQuery`

> **Note**
>
> 只有巢狀子查詢支援此查詢提示。

如需更多查詢提示範圍的詳細資訊，請參閱 Neptune 中的 SPARQL 查詢提示範圍。

evaluationStrategy SPARQL 提示範例

本節示範使用和不使用 evaluationStrategy 查詢提示編寫的查詢以及相關的最佳化。

此範例假設資料集有下列特性：

- 包含 1,000 個有標記 :connectedTo 的邊緣。
- 每個 component 節點平均連接到 100 個其他 component 節點。
- 節點之間四躍點循環連接的數量通常是大約 100 個。

無查詢提示

以下 SPARQL 查詢會擷取透過四個躍點彼此循環連接的所有 component 節點：

```
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 .
}
```

Neptune 查詢引擎的做法是使用以下步驟評估此查詢：

- 擷取圖中的全部 connectedTo 邊緣，共 1,000 個。
- 擴大 100 倍 (從 component2 傳出的 connectedTo 邊緣數量)。

  中間結果：100,000 個節點。
- 擴大 100 倍 (從 component3 傳出的 connectedTo 邊緣數量)。

  中間結果：10,000,000 個節點。
- 掃描 10,000,000 個節點的循環結束。

這會得到串流查詢計畫，其主記憶體為不變數量。

查詢提示和子查詢

您可能想要犧牲主記憶體空間換取加速運算。使用 evaluationStrategy 查詢提示重寫查詢，您可以強制引擎計算兩個較小、具體化子集的聯結。

```
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 .
    }
  }
}
```

相較於反覆使用先前三重模式的結果做為即將發生模式的輸入時評估序列中的三重模式，evaluationStrategy 提示會使得兩個子查詢被獨立評估。兩種子查詢的中繼結果都是產生 100,000 個節點，然後聯結在一起形成最終輸出。

特別是，當您在較大的執行個體類型上執行 Neptune 時，將這兩個 100,000 子集臨時儲存在主記憶體中，會增加記憶體使用量，進而大幅加快評估速度。

queryTimeout SPARQL 查詢提示

queryTimeout 查詢提示指定的逾時比資料庫參數群組中設定的 neptune_query_timeout 值短。

如果查詢因為此提示而終止，則會擲出 TimeLimitExceededException，並帶有 Operation terminated (deadline exceeded) 訊息。

queryTimeout SPARQL 提示語法

```
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 .
 ...
}
```

逾時值以毫秒表示。

逾時值必須小於資料庫參數群組中設定的 neptune_query_timeout 值。否則，會擲出 MalformedQueryException 例外狀況，並帶有 Malformed query: Query hint 'queryTimeout' must be less than neptune_query_timeout DB Parameter Group 訊息。

queryTimeout 查詢提示應該在主要查詢的 WHERE 子句中指定，或在其中一個子查詢的 WHERE 子句中指定，如下例所示。

它必須在所有查詢/子查詢和 SPARQL Updates 區段 (例如 INSERT 和 DELETE) 上設定一次。否則，會擲出 MalformedQueryException 例外狀況，並帶有 Malformed query: Query hint 'queryTimeout' must be set only once 訊息。

可用範圍

queryTimeout 提示可同時套用至 SPARQL 查詢和更新。

- 在 SPARQL 查詢中，它可以出現在主要查詢或子查詢的 WHERE 子句中。
- 在 SPARQL 更新中，可以在 INSERT、DELETE 或 WHERE 子句中設定。如果有多個更新子句，則只能在其中一個設定。

如需更多查詢提示範圍的詳細資訊，請參閱 Neptune 中的 SPARQL 查詢提示範圍。

queryTimeout SPARQL 提示範例

下例在 UPDATE 查詢的主要 WHERE 子句中使用 hint:queryTimeout：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
INSERT {
    ?s ?p ?o
} WHERE {
    hint:Query hint:queryTimeout 100 .
    ?s ?p ?o .
}
```

在此，hint:queryTimeout 位在子查詢的 WHERE 子句中：

```
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 .
      }
   }
}
```

rangeSafe SPARQL 查詢提示

使用此查詢提示來關閉 SPARQL 查詢的類型提升。

當您提交其包含的 FILTER 超過數值或範圍的 SPARQL 查詢時，Neptune 查詢引擎通常必須在執行查詢時使用類型提升。這表示它必須檢查可以保留您正在篩選之值的每種類型的值。

例如，如果您要篩選等於 55 的值，則引擎必須尋找等於 55 的整數、等於 55L 的長整數、等於 55.0 的浮點數等等。每個類型提升都需要對儲存體進行額外查詢，這可能會導致明顯簡單的查詢需要非預期的長時間才能完成。

通常類型提升不是必要的，因為您事先知道只需要找到一個特定類型的值即可。若是這種情況，您可以使用 rangeSafe 查詢提示來關閉類型提升，以大幅加快查詢速度。

rangeSafe SPARQL 提示語法

rangeSafe 查詢提示會採取 true 值來關閉類型提升。它也接受 false 值 (預設值)。

範例。以下範例展示如何在篩選 o 大於 1 的整數值時關閉類型提升：

```
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>)
```

queryId SPARQL 查詢提示

使用此查詢提示，將您自己的 queryId 值指派給 SPARQL 查詢。

範例：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * WHERE {
  hint:Query hint:queryId "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
  {?s ?p ?o}}
```

您指派的值在 Neptune 資料庫的所有查詢中必須是唯一的。

`useDFE` SPARQL 查詢提示

使用此查詢提示來啟用使用 DFE 執行查詢。根據預設，Neptune 不會在未將此查詢提示設定為 `true` 的情況下使用 DFE，因為 neptune_dfe_query_engine 執行個體參數預設為 `viaQueryHint`。如果將該執行個體參數設定為 `enabled`，則除了 `useDFE` 查詢提示設定為 `false` 的查詢以外，所有查詢都會使用 DFE 引擎。

啟用將 DFE 用於查詢的範例：

```
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 .
}
```

與 DESCRIBE 搭配使用的 SPARQL 查詢提示

SPARQL `DESCRIBE` 查詢提供了一種請求資源描述的靈活機制。不過，SPARQL 規格並未定義 `DESCRIBE` 的精確語義。

從引擎 1.2.0.2 版開始，Neptune 支援適用於不同情況的數種不同 `DESCRIBE` 模式和演算法。

此範例資料集可協助說明不同的模式：

```
@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 .
```

以下範例假設使用 SPARQL 查詢請求資源 `:JaneDoe` 的描述，如下所示：

```
DESCRIBE <https://example.com/JaneDoe>
```

`describeMode` SPARQL 查詢提示

`hint:describeMode` SPARQL 查詢提示用來選取下列其中一種 Neptune 支援的 SPARQL DESCRIBE 模式：

`ForwardOneStep` DESCRIBE 模式

您可以使用如下的 `describeMode` 查詢提示調用 `ForwardOneStep` 模式：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "ForwardOneStep"
}
```

`ForwardOneStep` 模式只會傳回要描述之資源的屬性和正向連結。在範例情況下，這表示它傳回具有 `:JaneDoe` (要描述的資源) 的三元組做為主旨：

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b301990159 .
```

請注意，DESCRAPE 查詢可能會傳回具有空白節點的三元組 (例如 `_:b301990159`)，與輸入資料集相比，它們每次都有不同的 ID。

SymmetricOneStep DESCRIBE 模式

如果您未提供查詢提示，則 SymmetricOneStep 是預設 DESCRIBE 模式。您也可以使用如下的 `describeMode` 查詢提示明確地調用它：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SymmetricOneStep"
}
```

在 SymmetricOneStep 語義下，DESCRIBE 會傳回要描述之資源的屬性、正向連結和反向連結：

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b318767375 .

_:b318767631 rdf:subject :JaneDoe .

:RichardRoe :knows :JaneDoe .

:ref_s2 rdf:subject :JaneDoe .
```

簡潔界限描述 (**CBD**) DESCRIBE 模式

簡潔界限描述 (CBD) 模式是使用如下的 `describeMode` 查詢提示來調用的：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "CBD"
}
```

在 CBD 語義下，DESCRIBE 會傳回要描述之資源的簡潔界限描述 (如 W3C 定義)：

```
: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 .
```

RDF 資源的簡潔界限描述 (也就是 RDF 圖形中的節點) 是以該節點為中心且可單獨的最小子圖形。實際上，這表示如果您將此圖形視為一棵樹，將指定的節點作為根，則沒有空白節點 (bnode) 作為該樹的葉子。由於 bnode 無法從外部處理或在後續查詢中使用，因此只瀏覽圖形從目前節點中尋找下一個單一跳轉，這是不夠的。您還必須前進足夠遠才能找到可以在後續查詢中使用的項目 (也就是說，bnode 以外的項目)。

計算 CBD

鑑於來源 RDF 圖形中的特定節點 (起始節點或根節點)，該節點的 CBD 計算方式如下：

1. 子圖形中包含了來源圖形中的所有陳述式，其中陳述式的「主旨」是起始節點。
2. 遞迴地，對於子圖形中迄今為止具有空白節點「物件」的所有陳述式，子圖形中包含了來源圖形中的所有陳述式，其中陳述式的「主旨」是該空白節點，並且尚未包含在子圖形中。
3. 遞迴地，對於子圖形中迄今為止包含的所有陳述式，會針對來源圖形中這些陳述式的所有具體化，包括從每個具體化的 `rdf:Statement` 節點開始的 CBD。

這會產生一個子圖形，其中「物件」節點是 IRI 參考或常值，或是不做為圖形中任何陳述式之「主旨」的空白節點。請注意，無法使用單一 SPARQL SELECT 或 CONSTRUCT 查詢來計算 CBD。

對稱簡潔界限描述 (**SCBD**) DESCRIBE 模式

對稱簡潔界限描述 (SCBD) 模式是使用如下的 `describeMode` 查詢提示來調用的：

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SCBD"
```

```
}
```

在 SCBD 語義下，DESCRIBE 會傳回資源的對稱簡潔界限描述 (如 W3C 在使用 VoID 詞彙描述連結的資料集中所定義)：

```
: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 .
```

CBD 和 SCBD 優於 ForwardOneStep 和 SymmetricOneStep 模式的地方是空白節點始終會擴展以包含其表示法。這可能是一個重要的優勢，因為您無法使用 SPARQL 查詢空白節點。此外，CBD 和 SCBD 模式還會考慮具體化。

請注意，describeMode 查詢提示也可以是 WHERE 子句的一部分：

```
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>
}
```

describeIterationLimit SPARQL 查詢提示

hint:describeIterationLimit SPARQL 查詢提示會對迭代擴展次數上限提供選用限制，而這些擴展是針對迭代 DESCRIBE 演算法 (例如 CBD 和 SCBD) 所執行的。

DESCRIBE 限制是透過 AND 結合在一起。因此，如果同時指定了迭代限制和陳述式限制，則必須符合這兩個限制，然後才能切斷 DESCRIBE 查詢。

此值的預設值為 5。您可以將其設定為 ZERO (0)，對迭代擴展的次數指定無限制。

describeStatementLimit SPARQL 查詢提示

`hint:describeStatementLimit` SPARQL 查詢提示會對 SHARECT 查詢回應中可能存在的陳述式數目上限提供「選用」限制。它僅適用於迭代 DESCRIBE 演算法，例如 CBD 和 SCBD。

DESCRIBE 限制是透過 AND 結合在一起。因此，如果同時指定了迭代限制和陳述式限制，則必須符合這兩個限制，然後才能切斷 DESCRIBE 查詢。

此值的預設值為 5000。您可以將其設定為 ZERO (0)，對傳回的陳述式數目指定無限制。

關於預設圖形的 SPARQL DESCRIBE 行為

SPARQL `DESCRIBE` 查詢表單可讓您擷取資源的相關資訊，而無需知道資料結構，也不必撰寫查詢。如何組合這些資訊留給 SPARQL 實作決定。Neptune 提供了數個查詢提示，調用不同的模式和演算法，供 `DESCRIBE` 使用。

在 Neptune 的實作中，無論模式為何，`DESCRIBE` 只會使用 SPARQL 預設圖形中存在的資料。這與 SPARQL 處理資料集的方式一致 (請參閱 SPARQL 規格中的指定 RDF 資料集)。

在 Neptune 中，除非使用 `FROM` 和/或 `FROM NAMED` 子句指定特定的具名圖形，否則預設圖形會包含資料庫中所有具名圖形之聯集中的所有唯一三元組。Neptune 的所有 RDF 資料都儲存在具名圖形中。如果插入沒有具名圖形內容的三元組，Neptune 會將其儲存在指定的具名圖形 `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` 中。

當使用 `FROM` 子句指定一或多個具名圖形時，預設圖形是這些具名圖形中所有唯一三元組的聯集。如果沒有 `FROM` 子句，但有一個或多個 `FROM NAMED` 子句，則預設圖形是空的。

SPARQL **DESCRIBE** 範例

請考慮下列資料：

```
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" .
```

對於此查詢：

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM ex:g1
FROM NAMED ex:g2
WHERE {
  GRAPH ex:g2 { ?s ?p "b" . }
}
```

Neptune 會傳回：

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
```

在這裡，首先評估圖形模式 `GRAPH ex:g2 { ?s ?p "b" }`，這會導致 `?s` 的繫結，然後對預設圖形評估 `DESCRIBE` 部分，現在只是 `ex:g1`。

不過，對於此查詢：

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM NAMED ex:g1
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Neptune 不會傳回任何內容，因為當 `FROM NAMED` 子句存在，而沒有任何 `FROM` 子句時，預設圖形是空的。

在以下查詢中，`DESCRIBE` 會在沒有 `FROM` 或 `FROM NAMED` 子句存在的情況下使用：

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

在此情況下，預設圖形是由資料庫中所有具名圖形之聯集中的所有唯一三元組成 (正式稱為 RDF 合併)，因此 Neptune 將傳回：

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
ex:s ex:p3 "b" .
ex:s ex:p3 "d" .
```

SPARQL 查詢狀態 API

若要取得 SPARQL 查詢的狀態，請使用 HTTP GET 或 POST 向 `https://`*your-neptune-endpoint*`:`*port*`/sparql/status` 端點提出請求。

SPARQL 查詢狀態請求參數

queryId (選用)

執行中 SPARQL 查詢的 ID。只顯示指定查詢的狀態。

SPARQL 查詢狀態回應語法

```
{
    "acceptedQueryCount": integer,
    "runningQueryCount": integer,
    "queries": [
      {
        "queryId":"guid",
        "queryEvalStats":
          {
            "subqueries": integer,
            "elapsed": integer,
            "cancelled": boolean
          },
        "queryString": "string"
      }
    ]
}
```

SPARQL 查詢狀態回應值

接受 QueryCount

自上次重新啟動 Neptune引擎以來接受的查詢數目。

執行 QueryCount

目前執行中的 SPARQL 查詢數。

queries

目前 SPARQL 查詢的清單。

queryId

查詢的 GUID ID。Neptune 會自動將此 ID 值指派給每個查詢，或者您也可以指派自己的 ID (請參閱 將自訂 ID 注入至 Neptune Gremlin 或 SPARQL 查詢)。

查詢 EvalStats

此查詢的統計資訊。

subqueries

此查詢中的子查詢數。

elapsed

到目前為止查詢已執行的毫秒數。

cancelled

True 表示查詢已取消。

queryString

提交的查詢。

SPARQL 查詢狀態範例

下面是使用 `curl` 和 HTTP `GET` 的狀態命令範例。

```
curl https://your-neptune-endpoint:port/sparql/status
```

此輸出會顯示單一執行中查詢。

```
{
    "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}"
        }
    ]
}
```

SPARQL 查詢取消

若要取得 SPARQL 查詢的狀態，請使用 HTTP GET 或 POST 向 `https://`*your-neptune-endpoint*`:`*port*`/sparql/status` 端點提出請求。

SPARQL 查詢取消請求參數

cancelQuery

(必要) 告知狀態命令取消查詢。此參數不採用數值。

queryId

(必要) 要取消之執行中 SPARQL 查詢的 ID。

silent

(選用) 如果 `silent=true`，則執行中查詢會遭到取消，且 HTTP 回應代碼為 200。如果 `silent` 不存在或 `silent=false`，則會取消查詢並顯示 HTTP 500 狀態碼。

SPARQL 查詢取消範例

範例 1：描配 `silent=false` 進行取消

以下是使用狀態命令的範例，其使用 `curl` 搭配設定為 `false` 的 `silent` 參數來取消查詢：

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
```

```
  -d "silent=false"
```

除非查詢已啟動串流結果，否則取消的查詢隨後將傳回 HTTP 500 代碼，回應如下：

```
{
  "code": "CancelledByUserException",
  "requestId": "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47",
  "detailedMessage": "Operation terminated (cancelled by user)"
}
```

如果查詢已傳回 HTTP 200 代碼 (OK)，且在取消之前已啟動串流結果，則逾時例外狀況資訊會傳送至一般輸出串流。

範例 2：搭配 **`silent=true`** 進行取消

以下是與上述相同狀態命令的範例，但 `silent` 參數現在設定為 `true`：

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=true"
```

此命令會傳回與 `silent=false` 時相同的回應，但取消的查詢現在會傳回 HTTP 200 代碼，其回應如下：

```
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
  },
  "results" : {
    "bindings" : [ ]
  }
}
```

在 Amazon Neptune 中使用 SPARQL 1.1 圖形存放區 HTTP 通訊協定 (GSP)

在 SPARQL 1.1 圖形存放區 HTTP 通訊協定建議中，W3C 定義了用於管理 RDF 圖形的 HTTP 通訊協定。它會定義用於移除、建立和取代 RDF 圖形內容的操作，以及用於將 RDF 陳述式新增至現有內容的操作。

圖形存放區通訊協定 (GSP) 提供了一種便利的方式來操作您的整個圖形，而不必撰寫複雜的 SPARQL 查詢。

從 版本：1.0.5.0 (2021 年 7 月 27 日) 開始，Neptune 完全支援此通訊協定。

圖形存放區通訊協定 (GSP) 的端點為：

```
https://your-neptune-cluster:port/sparql/gsp/
```

若要使用 GSP 存取預設圖形，請使用：

```
https://your-neptune-cluster:port/sparql/gsp/?default
```

若要使用 GSP 存取具名圖形，請使用：

```
https://your-neptune-cluster:port/sparql/gsp/?graph=named-graph-URI
```

Neptune GSP 實作的特殊詳細資訊

Neptune 完全實作了定義 GSP 的 W3C 建議。不過，有一些規格未涵蓋的情況。

其中一種情況是，PUT 或 POST 請求在請求內文中指定一個或多個具名圖形，這些圖形與請求 URL 指定的圖形不同。僅在請求內文 RDF 格式支援具名圖形時，才可能發生這種情況，例如，使用 `Content-Type: application/n-quads` 或 `Content-Type: application/trig`。

在此情況下，Neptune 會新增或更新內文中存在的所有具名圖形，以及在 URL 中指定的具名圖形。

例如，假設從空白資料庫開始，您會傳送 PUT 請求，以將選票 upsert 至三個圖形。其中一個名為 `urn:votes`，包含來自所有選舉年的所有選票。另外兩個名為 `urn:votes:2005` 和 `urn:votes:2019`，包含來自特定選舉年的選票. 請求及其承載看起來像這樣：

```
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>
```

在執行請求之後，資料庫中的資料看起來像這樣：

```
<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>
```

另一個不明確的情況是，在請求 URL 本身中使用任一 PUT、POST、GET 或 DELETE 指定多個圖形 例如：

```
POST "http://your-Neptune-cluster:port/sparql/gsp/?
graph=urn:votes:2005&graph=urn:votes:2019"
```

或者：

```
GET "http://your-Neptune-cluster:port/sparql/gsp/?default&graph=urn:votes:2019"
```

在此情況下，Neptune 會傳回 HTTP 400，其中訊息指出只能在請求 URL 中指定一個圖形。

使用 SPARQL **explain** 分析 Neptune 查詢執行

Amazon Neptune 已新增名為 explain 的 SPARQL 功能。這個功能是自助式工具，用於了解 Neptune 引擎採取的執行方法。您透過將 `explain` 參數新增到提交 SPARQL 查詢的 HTTP 呼叫，來進行叫用。

此 `explain` 功能可提供查詢執行計劃邏輯結構的相關資訊。您可以使用此資訊來識別潛在的評估和執行瓶頸。然後，您可以使用查詢提示，以改善您的查詢執行計劃。

主題

- SPARQL 查詢引擎在 Neptune 的運作方式
- 如何使用 SPARQL explain 來分析 Neptune 查詢執行
- 在 explain 中調用 SPARQL 的範例
- Neptune SPARQL explain 運算子

- Neptune 中 SPARQL explain 的限制

SPARQL 查詢引擎在 Neptune 的運作方式

若要使用 SPARQL `explain` 功能提供的資訊，您需要了解一些 Amazon Neptune SPARQL 查詢引擎運作方式的詳細資訊。

此引擎會將每個 SPARQL 查詢轉換為運算子管道。從第一個運算子開始，名為繫結清單的中繼解決方案將套用至此運算子管道。您可以將繫結清單視為資料表，其中資料表標頭是查詢中所用的部分變數。因此，資料表中的每一列代表評估時間點之前的結果。

假設系統已為我們的資料定義兩個命名空間前綴：

```
@prefix ex:   <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

以下是此內容中簡單繫結清單的範例：

```
?person       | ?firstName
-------------------------------------------------------
ex:JaneDoe    | "Jane"
ex:JohnDoe    | "John"
ex:RichardRoe | "Richard"
```

對於三個人中的每一個人，清單會將 `?person` 變數繫結至該人員的識別符，且將 `?firstName` 變數繫結至該人員的名字。

一般情況下，變數可以維持在未繫結的狀態，例如，對於資料中不存在值的查詢中變數的 `OPTIONAL` 選項。

`PipelineJoin` 運算子是 `explain` 輸出中存在 Neptune 查詢引擎運算子的範例。它會將輸入做為上個運算子的傳入繫結集，並針對三重模式來進行聯結，也就是 `(?person, foaf:lastName, ?lastName)`。這個運算子會在此輸入串流使用 `?person` 變數的繫結，將這些繫結取代為三重模式，並從資料庫中查詢三元素。

當從上個資料表中在傳入繫結內容中執行時，`PipelineJoin` 會評估三個查詢，也就是下列項目：

```
(ex:JaneDoe,    foaf:lastName, ?lastName)
(ex:JohnDoe,    foaf:lastName, ?lastName)
(ex:RichardRoe, foaf:lastName, ?lastName)
```

這種方法稱為 as-bound 評估。系統會根據傳入解決方案重新聯結這個評估程序的解決方案，並在傳入解決方案中填補偵測到的 `?lastName`。假設您找到所有三個人員的姓氏，運算子會產生一個傳出繫結清單，這會像下面這樣：

```
?person       | ?firstName | ?lastName
---------------------------------------
ex:JaneDoe    | "Jane"     | "Doe"
ex:JohnDoe    | "John"     | "Doe"
ex:RichardRoe | "Richard"  | "Roe"
```

這個傳出繫結清單則做為輸入，供管道中下一個運算子使用。最後，管道中最後一個運算子的輸出會定義查詢結果。

運算子管道通常是線性的，這是因為每個運算子會為單一連線的運算子發出解決方案。不過，在某些情況下，他們可以擁有更為複雜的結構。例如，SPARQL 查詢中的 `UNION` 運算子會映射到 `Copy` 操作。這個操作會複製繫結並將副本轉送到兩個子計劃，一個用於 `UNION` 的左側而另一個用於右側。

如需運算子的詳細資訊，請參閱 Neptune SPARQL `explain` 運算子。

如何使用 SPARQL **`explain`** 來分析 Neptune 查詢執行

SPARQL `explain` 功能是 Amazon Neptune 中的自助式工具，可協助您了解 Neptune 引擎採取的執行方法。若要叫用 `explain`，您會以 `explain=`*`mode`* 的形式將參數傳遞到 HTTP 或 HTTPS 請求。

模式值可以是 `static` `dynamic` 或 `details` 的其中之一。

- 在靜態模式中，`explain` 只會列印查詢計劃的靜態結構。
- 在動態模式中，`explain` 還包括查詢計劃的動態層面。這些層面可能包含經由運算子流動的中繼繫結數和傳入繫結與傳出繫結的比率，以及運算子所耗費的時間。
- 在詳細資訊模式中，`explain` 會列印 `dynamic` 模式中顯示的資訊及其他詳細資訊，例如，實際 SPARQL 查詢字串，以及構成聯結運算子基礎之模式的預估範圍計數。

Neptune 支援使用 `explain`，搭配在 W3C SPARQL 1.1 通訊協定規格中所列的所有三個 SPARQL 查詢存取通訊協定，即：

1. HTTP GET
2. 使用 URL 編碼參數的 HTTP POST

3. 使用文字參數的 HTTP POST

如需 SPARQL 查詢引擎的詳細資訊，請參閱 SPARQL 查詢引擎在 Neptune 的運作方式。

如需叫用 SPARQL `explain` 產生之輸出種類的詳細資訊，請參閱 在 `explain` 中調用 SPARQL 的範例。

在 **`explain`** 中調用 SPARQL 的範例

本區段中的範例示範您透過調用 SPARQL `explain` 功能，在 Amazon Neptune 中分析查詢執行可以產生的各種輸出類型。

主題

- 了解 Explain 輸出
- 詳細資訊模式輸出的範例
- 靜態模式輸出的範例
- 不同參數編碼方式
- 除了文字/純文字之外的其他輸出類型
- DFE 啟用時的 SPARQL explain 輸出範例

了解 Explain 輸出

在這個範例中，Jane Doe 知道兩個人，也就是 John Doe 和 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" .
.
```

若要判斷 Jane Doe 知道的所有人的名字，您可以編寫以下查詢：

```
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"
```

此簡單查詢會傳回下列結果：

```
firstName
John
Richard
```

接著，透過新增 `-d "explain=dynamic"` 和使用預設的輸出類型 (而不是 `text/csv`)，來變更 `curl` 命令以叫用 `explain`：

```
 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"
```

查詢現在會在適合列印 ASCII 格式 (HTTP 內容類型 `text/plain`) (也就是預設的輸出類型) 中傳回輸出：

```
################################################################################################
# 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         #
#############################################################################################
```

對於在 Name 欄及其引數中的運算子詳細資訊，請參閱 Explain 運算子。

以下以逐列說明輸出：

1. 主要查詢的第一步一律是使用 SolutionInjection 運算子來注入解決方案。然後，系統會透過評估過程將解決方案擴展至最終結果。

   在這種情況下，它會注入所謂的通用解決方案 { }。存在 VALUES 子句或 BIND 時，此步驟也可能注入更為複雜的變數繫結來開始使用。

   Units Out 欄表示這個單一解決方案會自此運算子傳出。Out #1 欄會指定此運算子將結果送入其中的運算子。在這個範例中，所有運算子會連接到此資料表中的以下運算子。
2. 第二個步驟是 PipelineJoin。它收到做為上個運算子 (Units In := 1) 產生的輸入單一 (完全無限制) 解決方案。並將其聯結至 pattern 引數定義的元組模式。此會對應到模式的簡單查詢。在這種情況下，會將三重模式定義為下列各項：

   ```
   distinct( ex:JaneDoe, foaf:knows, ?person )
   ```

   joinType := join 引數表示一般聯結操作 (其他類型包括 optional 聯結、existence check 聯結等)。

   distinct := true 引數表示您從資料庫 (非重複項目) 擷取的剛好是不同的符合項目，且您將不同的符合項目繫結至變數 joinProjectionVars := ?person，來去重複。

   事實上，Units Out 欄值為 2 表示有兩個傳出的解決方案。特別是，反映兩個人員的 ?person 變數繫結，此資料說明 Jane Doe 知道：

   ```
   ?person
   -------------
   ```

```
ex:JohnDoe
ex:RichardRoe
```

3. 兩種解決方案從階段 2 流程做為輸入 (`Units In := 2`) 傳入至第二個 `PipelineJoin`。此運算子會將兩個舊解決方案與以下三重模式聯結：

   ```
   distinct(?person, foaf:firstName, ?firstName)
   ```

   已知運算子傳入解決方案會將 `?person` 變數繫結至 `ex:JohnDoe` 或 `ex:RichardRoe`。基於此，`PipelineJoin` 會擷取名字，John 和 Richard。兩個傳出解決方案 (Units Out := 2) 則如下所示：

   ```
   ?person       | ?firstName
   ---------------------------
   ex:JohnDoe    | John
   ex:RichardRoe | Richard
   ```

4. 下一個投影運算子使用階段 3 (`Units In := 2`) 兩個解決方案的輸入並投射到 `?firstName` 變數。這樣就不需在映射中繫結所有其他變數並在兩個繫結 (`Units Out := 2`) 上進行傳遞：

   ```
   ?firstName
   ----------
   John
   Richard
   ```

5. 為了改善效能，Neptune 會盡可能在其指派給 URI 和字串常值這類字詞的內部識別符上 (而不是在字串本身) 操作。最後一個運算子 `TermResolution` 會透過這些內部識別符執行映射，並傳回對應的字詞字串。

   在一般 (非說明) 查詢評估中，會將最後一個運算子運算的結果序列化為請求的序列化格式並串流到用戶端。

詳細資訊模式輸出的範例

> **Note**
>
> SPARQL Explain 詳細資料模式從 Neptune 引擎 1.0.2.1 版開始可用。

假設您執行的查詢與先前在詳細資訊模式 (而不是動態模式) 中執行的一樣：

```
 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"
```

如本範例所示，輸出與其他一些詳細資訊相同，例如，輸出頂端的查詢字串，以及 PipelineJoin 運算子的 patternEstimate 計數：

```
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         #
#########################################################################################
```

靜態模式輸出的範例

假設您執行的查詢與先前在靜態模式 (預設) (而不是詳細資訊模式) 中執行的一樣：

```
 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"
```

如此範例所示，輸出都是相同的，但它省略最後三個欄位：

```
#########################################################################################
# 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 #
```

```
################################################################################################################
```

不同參數編碼方式

以下範例查詢說明叫用 SPARQL `explain` 時兩種不同的參數編碼方法。

使用 URL 編碼 – 此範例使用參數的 URL 編碼，並指定「動態」輸出：

```
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"
```

直接指定參數 – 與上個查詢相同，差別在於它直接透過 POST 來傳遞參數：

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic"
```

除了文字/純文字之外的其他輸出類型

上述範例使用預設 `text/plain` 輸出類型。Neptune 還可以透過其他兩種 MIME 類型格式 (即 `explain` 和 `text/csv`) 格式化 SPARQL `text/html` 輸出。您透過設定 HTTP `Accept` 標頭 (也就是您在 `curl` 使用 `-H` 旗標可以做的事) 來進行叫用，如下所示：

```
 -H "Accept: output type"
```

以下是一些範例：

`text/csv` 輸出

此查詢透過指定 `-H "Accept: text/csv"` 來呼叫 CSV MIME 類型輸出：

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic" \
  -H "Accept: text/csv"
```

CSV 格式 (匯入到試算表或資料庫的實用格式)，透過分號 (`;`) 將每個 `explain` 列中的欄位分隔，如下所示：

```
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
```

`text/html` 輸出

如果您指定 `-H "Accept: text/html"`，則 `explain` 會產生 HTML 資料表：

```
<!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>
```

HTML 在瀏覽器中顯示如下：

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

DFE 啟用時的 SPARQL **`explain`** 輸出範例

以下是 Neptune DFE 替代查詢引擎啟用時的 SPARQL `explain` 輸出範例：

```
################################################################################################
# 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        #
##########################################################################################
```

Neptune SPARQL **explain** 運算子

以下各節描述目前 Amazon Neptune 中可用的 SPARQL `explain` 功能運算子和參數。

> **Important**
>
> SPARQL `explain` 功能仍在進行改善，這裡記載的運算子和參數在未來的版本中可能會加以變更。

主題

- Slice運算子
- SolutionInjection運算子
- Sort運算子
- VariableAlignment運算子

Aggregation運算子

該運算子會執行一個或多個彙總，以實作 count、max、min、sum 等 SPARQL 彙總運算子的語意。

Aggregation 附帶使用 groupBy 子句的選用分組，以及選用的 having 限制。

引數

- groupBy – (選用) 提供的 groupBy 子句會根據分組的傳入解決方案來指定運算式序列。
- aggregates – (必要) 指定彙總運算式的排序清單。
- having – (選用) 會按照 SPARQL 查詢中 having 子句的指示，將限制新增至群組中的篩選條件。

ConditionalRouting運算子

該運算子會根據指定的條件路由傳入解決方案。符合該條件的解決方案會路由至 Out #1 所參考的運算子 ID，而不符合條件的解決方案則會路由至 Out #2 參考的運算子。

引數

- condition – (必要) 路由條件。

Copy運算子

該運算子會依指定模式的規定來委派解決方案串流。

模式

- forward – 將解決方案轉送至 Out #1 識別的下游運算子。
- duplicate – 複製解決方案並將其分別轉送至 Out #1 和 Out #2 識別的兩個運算子。

Copy 並沒有任何引數。

`DFENode`運算子

這個運算子是 DFE 替代查詢引擎所執行之計畫的抽象。詳細的 DFE 計畫會在此運算子的引數中加以概述。引數目前已過載，以包含 DFE 計畫的詳細執行期統計資料。它包含按 DFE 查詢執行的各個步驟所花費的時間。

DFE 查詢計畫的邏輯最佳化抽象語法樹 (AST) 會與規劃時所考慮之運算子類型的相關資訊，以及要執行運算子的相關聯最佳和最差成本一起列印。AST 目前由以下類型的節點組成：

- `DFEJoinGroupNode` – 代表一個或多個 `DFEPatternNodes` 的聯結。
- `DFEPatternNode` – 使用從基礎資料庫中投影出來的相符元組封裝基礎模式。

子區段 `Statistics & Operator histogram` 包含有關 `DataflowOp` 計畫執行時間，以及每個運算子所使用之 CPU 時間明細的詳細資訊。這個子區段下面有一個資料表，其會列印 DFE 所執行之計畫的詳細執行期統計資料。

> **Note**
>
> 因為 DFE 是在實驗室模式下發行的一項實驗功能，所以其 `explain` 輸出的確切格式可能會有所變更。

`Distinct`運算子

該運算子會運算變數子集上不同的投影，藉此消除重複項目。因此，流入的解決方案數量會大於或等於流出的解決方案數量。

引數

- `vars` – (必要) 要套用 `Distinct` 投影的變數。

`Federation`運算子

將指定的查詢傳遞至指定的遠端 SPARQL 端點。

引數

- `endpoint` – (必要) SPARQL `SERVICE` 陳述式中的端點 URL。這可以是常數字串，或者如果查詢端點是根據相同查詢中的變數來決定，則它可以是變數名稱。

- query – (必要) 要傳送至遠端端點的重建查詢字串。即使用戶端未指定任何字首，引擎也會將預設字首新增至此查詢。
- silent – (必要) 布林值，指出 SILENT 關鍵字是否出現在此關鍵字之後。SILENT 會告知引擎，即使遠端 SERVICE 部分失敗，也不要讓整個查詢失敗。

Filter運算子

該運算子會篩選傳入的解決方案。唯有符合篩選條件的解決方案可轉送至上游運算子，其他所有解決方案則會遭到捨棄。

引數

- condition – (必要) 篩選條件。

HashIndexBuild運算子

該運算子會接受繫結清單並將其多工緩衝處理至雜湊索引，該雜湊索引的名稱是由 solutionSet 引數所定義。後續的運算子通常會對這個解決方案集執行聯結操作，再用該名稱指向解決方案集。

引數

- solutionSet – (必要) 雜湊索引解決方案集的名稱。
- sourceType – (必要) 所取得雜湊索引中存放繫結的來源類型：
  - pipeline – 將運算子管道中來自下游運算子的傳入解決方案多工緩衝處理至雜湊索引。
  - binding set – 將 sourceBindingSet 引數指定的固定繫結集多工緩衝處理至雜湊索引。
- sourceBindingSet – (選用) 如果 sourceType 引數值為 binding set，則此引數會指定將靜態繫結集多工緩衝處理至雜湊索引。

HashIndexJoin運算子

該運算子會對 solutionSet 引數辨識的雜湊索引解決方案集執行傳入解決方案聯結操作。

引數

- solutionSet – (必要) 要據以聯結的解決方案集名稱。這必須是在先前步驟中使用 HashIndexBuild 運算子建構的雜湊索引解決方案。
- joinType – (必要) 要執行的聯結類型：

  - join – 一般聯結，所有共用變數間需要有完全相符的項目。
  - optional – optional 聯結，其會使用 SPARQL OPTIONAL 運算子語義。
  - minus – minus 操作會使用 SPARQL MINUS 運算子語義來保留沒有任何聯結夥伴的映射。
  - existence check – 檢查是否有聯結夥伴，並將 existenceCheckResultVar 變數繫結至檢查結果。
- constraints – (選用) 進行聯結時所考量的其他聯結限制。不符合這些限制的聯結皆會予以捨棄。
- existenceCheckResultVar – (選用) 只能用在 joinType 等於 existence check 的聯結 (請參閱先前提及的 joinType 引數)。

MergeJoin運算子

該運算子會對 solutionSets 辨識的多個解決方案集執行合併聯結。

引數

- solutionSets – (必要) 要互相聯結的解決方案集。

NamedSubquery運算子

該運算子會針對 subQuery 引數識別的子查詢觸發評估作業，並將結果多工緩衝處理至 solutionSet 引數指定的解決方案集。運算子的傳入解決方案會轉送至該子查詢，接著轉送至下一個運算子。

引數

- subQuery – (必要) 要評估的子查詢名稱；輸出中會明確呈現該子查詢。
- solutionSet – (必要) 用來儲存子查詢結果的解決方案集名稱。

PipelineJoin運算子

該運算子會接收先前運算子的輸出做為輸入，並將其聯結至 pattern 引數定義的元組模式。

引數

- pattern- (必填) 模式，它採用的形式 subject-predicate-object，並可選擇-graph 元組的基礎連接。如果 distinct 為指定的模式值，則聯結僅會從 projectionVars 引數指定的投影變數中擷取不同解決方案，而不是所有相符的解決方案。

- inlineFilters – (選用) 要套用至模式中變數的一組篩選條件。系統會一併評估模式與這些篩選條件。
- joinType – (必要) 要執行的聯結類型：
  - join – 一般聯結，所有共用變數間需要有完全相符的項目。
  - optional – optional 聯結，其會使用 SPARQL OPTIONAL 運算子語義。
  - minus – minus 操作會使用 SPARQL MINUS 運算子語義來保留沒有任何聯結夥伴的映射。
  - existence check – 檢查是否有聯結夥伴，並將 existenceCheckResultVar 變數繫結至檢查結果。
- constraints – (選用) 進行聯結時所考量的其他聯結限制。不符合這些限制的聯結皆會予以捨棄。
- projectionVars – (選用) 投影變數。此引數需與 distinct := true 結合使用，以在指定的一組變數內強制擷取不同投影。
- cutoffLimit – (選用) 擷取的聯結夥伴數量截止限制。雖然沒有任何預設限制，但您可以在執行聯結操作時將此引數設為 1 以實作 FILTER (NOT) EXISTS 子句，其足以證明或駁斥聯結夥伴的存在。

PipelineCountJoin運算子

該運算子為 PipelineJoin 的變體；其不會執行聯結操作，而是計算相符的聯合夥伴數量，並將計數繫結至 countVar 引數指定的變數。

引數

- countVar – (必要) 應繫結計數結果 (即聯結夥伴數量) 的變數。
- pattern- (必填) 模式，它採用的形式 subject-predicate-object，並可選擇-graph 元組的基礎連接。如果 distinct 為指定的模式值，則聯結僅會從 projectionVars 引數指定的投影變數中擷取不同解決方案，而不是所有相符的解決方案。
- inlineFilters – (選用) 要套用至模式中變數的一組篩選條件。系統會一併評估模式與這些篩選條件。
- joinType – (必要) 要執行的聯結類型：
  - join – 一般聯結，所有共用變數間需要有完全相符的項目。
  - optional – optional 聯結，其會使用 SPARQL OPTIONAL 運算子語義。
  - minus – minus 操作會使用 SPARQL MINUS 運算子語義來保留沒有任何聯結夥伴的映射。
  - existence check – 檢查是否有聯結夥伴，並將 existenceCheckResultVar 變數繫結至檢查結果。

- constraints – (選用) 進行聯結時所考量的其他聯結限制。不符合這些限制的聯結皆會予以捨棄。
- projectionVars – (選用) 投影變數。此引數需與 distinct := true 結合使用，以在指定的一組變數內強制擷取不同投影。
- cutoffLimit – (選用) 擷取的聯結夥伴數量截止限制。雖然沒有任何預設限制，但您可以在執行聯結操作時將此引數設為 1 以實作 FILTER (NOT) EXISTS 子句，其足以證明或駁斥聯結夥伴的存在。

PipelinedHashIndexJoin運算子

這是一個 all-in-one 構建哈希索引和連接運算符。它會取得繫結清單、將它們多工緩衝處理至雜湊索引，然後針對雜湊索引聯結傳入的解決方案。

引數

- sourceType – (必要) 所取得雜湊索引中儲存繫結的來源類型：
  - pipeline – 導致 PipelinedHashIndexJoin 將運算子管道中來自下游運算子的傳入解決方案多工緩衝處理至雜湊索引。
  - binding set – 導致 PipelinedHashIndexJoin 將 sourceBindingSet 引數指定的固定繫結集多工緩衝處理至雜湊索引。
- sourceSubQuery – (選用) 如果 sourceType 引數值為 pipeline，此引數會指定進行評估並多工緩衝處理至雜湊索引的子查詢。
- sourceBindingSet – (選用) 如果 sourceType 引數值為 binding set，則此引數會指定將靜態繫結集多工緩衝處理至雜湊索引。
- joinType – (必要) 要執行的聯結類型：
  - join – 一般聯結，所有共用變數間需要有完全相符的項目。
  - optional – optional 聯結，其會使用 SPARQL OPTIONAL 運算子語義。
  - minus – minus 操作會使用 SPARQL MINUS 運算子語義來保留沒有任何聯結夥伴的映射。
  - existence check – 檢查是否有聯結夥伴，並將 existenceCheckResultVar 變數繫結至檢查結果。
- existenceCheckResultVar – (選用) 只會用於 joinType 等於 existence check 的聯結 (請參閱上述的 joinType 引數)。

Projection運算子

該運算子會在變數子集上進行投影。流入的解決方案數量會等同於流出的解決方案數量，但解決方案的圖形會因模式設定而異。

模式

- retain – 僅在解決方案中保留由 vars 引數指定的變數。
- drop – 捨棄由 vars 引數指定的所有變數。

引數

- vars – (必要) 要保留或捨棄的變數，取決於模式設定。

PropertyPath運算子

啟用遞迴屬性路徑，例如 + 或 *。Neptune 根據 iterationTemplate 引數指定的範本實作固定點迭代方法。系統會針對每個固定點反覆運算項目將已知的左側或右側變數繫結至範本，直到找不到任何新的解決方案為止。

引數

- iterationTemplate – (必要) 用來實作固定點迭代的子查詢範本名稱。
- leftTerm – (必要) 位於屬性路徑左側的字詞 (變數或常值)。
- rightTerm – (必要) 位於屬性路徑右側的字詞 (變數或常值)。
- lowerBound – (必要) 固定點迭代的下限 (適用於 * 查詢的 0，或適用於 + 查詢的 1)。

TermResolution運算子

該運算子會將內部字串識別符值轉換為各自對應的外部字串，或將外部字串轉換為內部字串識別符值，視模式而定。

模式

- value2id – 將常值和 URI 等字詞映射至對應的內部 ID 值 (內部值編碼)。
- id2value – 將內部 ID 值映射至對應的常值和 URI 等字詞 (內部值解碼)。

引數

- vars – (必要) 指定字串或內部字串 ID 需進行映射的變數。

Slice運算子

該運算子會使用 SPARQL LIMIT 和 OFFSET 子句的語意，在傳入解決方案串流上實作配量。

引數

- limit – (選用) 對要轉送之解決方案的限制。
- offset – (選用) 評估解決方案以進行轉送的偏移量。

SolutionInjection運算子

該運算子不會接收任何輸入，其會將靜態解決方案插入查詢計劃，並在 solutions 引數中加以記錄。

查詢計劃一律會從插入的靜態項目開始執行。如果可以結合各種來源的靜態繫結，進而從查詢本身衍生取得 (例如從 VALUES 或 BIND 子句) 要插入的靜態解決方案，則 SolutionInjection 運算子就會插入這些衍生的靜態解決方案。在最簡單的情況下，這會反映出外部 VALUES 子句所指示的繫結。

如果不能自查詢衍生取得任何靜態解決方案，SolutionInjection 就會插入空白的解決方案，也就是所謂的通用解決方案。在整個查詢評估過程中，系統會擴展並倍增這個解決方案。

引數

- solutions – (必要) 運算子插入的解決方案序列。

Sort運算子

該運算子會使用指定排序條件來排列解決方案集。

引數

- sortOrder – (必要) 變數的排序清單，各包含 ASC (遞增) 或 DESC (遞減) 識別符，其會按照順序來排列解決方案集。

VariableAlignment運算子

該運算子會逐一檢查所有解決方案，並針對其中的兩個變數執行校準作業。此處的變數指的是指定的 `sourceVar` 值和指定的 `targetVar` 值。

如果解決方案中的 `sourceVar` 和 `targetVar` 值相同，系統就會將該變數視為已經過校準並轉送解決方案，然後投射出多餘的 `sourceVar`。

若變數繫結至不同值，解決方案就會完全遭到排除。

引數

- `sourceVar` – (必要) 要與目標變數相比的來源變數。若在解決方案中成功完成校準，表示兩個變數擁有相同值，且系統會投射出來源變數。
- `targetVar` – (必要) 與來源變數相比的目標變數。即使校準成功，該值仍會保留不變。

Neptune 中 SPARQL **explain** 的限制

Neptune SPARQL `explain` 功能版本具有下列限制。

Neptune 目前僅在 SPARQL SELECT 查詢中支援 Explain

如需其他查詢形式評估程序的相關資訊，例如 ASK、CONSTRUCT、DESCRIBE 和 SPARQL UPDATE 查詢，您可以將這些查詢轉換為 SELECT 查詢。然後，改用 `explain` 來檢查對應的 SELECT 查詢。

例如，若要取得與 `ASK WHERE {...}` 查詢相關的 `explain` 資訊，使用 `explain` 執行對應的 `SELECT WHERE {...} LIMIT 1`。

同樣地，對於 `CONSTRUCT {...} WHERE {...}` 查詢，請捨棄 `CONSTRUCT {...}` 部分並使用 `explain`，在第二個 `WHERE {...}` 子句執行 SELECT 查詢。對第二個 WHERE 子句進行評估通常也揭露處理 CONSTRUCT 查詢的主要挑戰，因為從第二個 WHERE 傳到 CONSTRUCT 範本的解決方案通常只需要直接替換。

解釋運算子在未來版本可能變更

SPARQL `explain` 運算子及其參數可能在未來版本變更。

解釋輸出在未來版本可能變更

例如，欄標題可能會變更，且可能會將更多欄新增到資料表。

Neptune 中使用 **SERVICE** 延伸模組的 SPARQL 聯合查詢

Amazon Neptune 完整支援使用 SERVICE 關鍵字的 SPARQL 聯合查詢延伸模組。(如需詳細資訊，請參閱 SPARQL 1.1 聯合查詢。)

> **Note**
>
> 從 版本 1.0.1.0.200463.0 (2019 年 10 月 15 日) 開始就可以使用這項功能。

SERVICE 關鍵字會指示 SPARQL 查詢引擎對遠端 SPARQL 端點執行部分查詢，並撰寫最終查詢結果。只有 READ 操作是可行的。不支援 WRITE 和 DELETE 操作。Neptune 只能針對可在其虛擬私有雲端 (VPC) 內存取的 SPARQL 端點執行聯合查詢。不過，您也可以在 VPC 中使用反向代理，使外部資料來源可在 VPC 內存取。

> **Note**
>
> 當 SPARQL SERVICE 用來將一個查詢聯合到相同 VPC 中兩個以上的 Neptune 叢集時，安全群組必須設定為允許所有這些 Neptune 叢集彼此通話。

> **Important**
>
> SPARQL 1.1 聯合會在將查詢和參數傳遞至外部 SPARQL 端點時，代您提出服務請求。您有責任驗證外部 SPARQL 端點是否滿足您應用程式的資料處理和安全需求。

Neptune 聯合查詢的範例

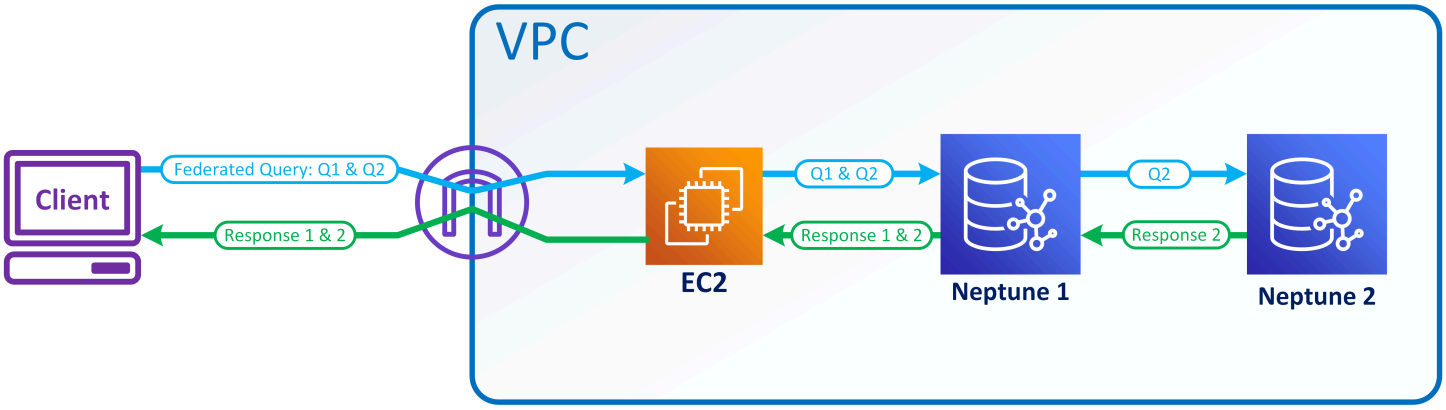
以下簡單範例說明 SPARQL 聯合查詢的運作方式。

假設客戶將以下查詢傳送至 Neptune-1，網址為 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 評估第一個查詢模式 (Q-1)，即 `?person rdf:type foaf:Person`，會使用結果來解析 Q-2 (`?person foaf:knows ?friend`) 中的 `?person`，並將產生的模式轉送至 Neptune-2，網址為 `http://neptune-2:8182/sparql`。
2. Neptune-2 評估 Q-2，並將結果傳回至 Neptune-1。
3. Neptune-1 聯結這兩種模式的解決方案，並將結果傳回給客戶。

下圖顯示此流程。



> **Note**
>
> 根據預設，最佳化工具會確定 `SERVICE` 指令在查詢執行中的哪個點執行。您可以使用 joinOrder 查詢提示覆寫此置放。

Neptune 中聯合查詢的存取控制

Neptune 使用 AWS Identity and Access Management (IAM) 進行身份驗證和授權。聯合查詢的存取控制可涉及多個 Neptune 資料庫執行個體。這些執行個體可能會有不同的存取控制需求。在某些情況下，這會限制您進行聯合查詢的能力。

考慮上一節中展示的簡單範例。Neptune-1 會透過呼叫其所用的同一憑證呼叫 Neptune-2。

- 如果 Neptune-1 需要 IAM 身分驗證和授權，但 Neptune-2 不需要，則您只需要 Neptune-1 的適當 IAM 許可，即可進行聯合查詢。
- 如果 Neptune-1 和 Neptune-2 都需要 IAM 身分驗證和授權，您需要連線兩個資料庫的 IAM 許可才能進行聯合查詢。兩個叢集也必須位於相同 AWS 帳戶並位於相同區域中。目前不支援跨區域和/或跨帳戶聯合查詢架構。

- 不過，在 Neptune-1 未啟用 IAM，但 Neptune-2 卻啟用它的情況下，您無法進行聯合查詢。原因是 Neptune-1 無法擷取您的 IAM 憑證，並將其傳遞至 Neptune-2，以授權查詢的第二個部分。

Neptune 的圖形視覺化工具

除了 Neptune 圖形筆記本內建的視覺化功能之外，您還可以使用 AWS 合作夥伴和協力廠商所建置的解決方案，將儲存在 Neptune 中的資料視覺化。

複雜的圖形視覺化可協助組織中的資料科學家、管理員和其他角色，以互動方式探索圖形資料，而不必知道如何撰寫複雜的查詢。

主題

- 開放程式碼圖形總管
- Tom Sawyer 軟體
- Cambridge Intelligence
- Graphistry
- metaphacts
- G.V( )
- 連結的

開放程式碼圖形總管

圖形總管是一種用於圖形資料的開放原始碼低程式碼視覺化探索工具，可在 Apache-2.0 授權下使用。它可讓您瀏覽圖形資料庫中的標記屬性圖 (LPG) 或資源描述架構 (RDF) 資料，而不必撰寫圖形查詢。圖形總管旨在協助資料科學家、商務分析師，以及組織中的其他角色，以互動方式探索圖形資料，而不必學習圖形查詢語言。

圖形總管提供了一個反應型 Web 應用程式，可以部署為容器來視覺化圖形資料。您可以連接到 Amazon Neptune 或其他圖形資料庫，這些資料庫提供阿帕奇 TinkerPop 格林或 SPARQL 1.1 端點。

- 您可以使用分面篩選條件快速查看資料的摘要，或透過將文字輸入至搜尋列來搜尋資料。
- 您也可以透過互動方式探索節點和邊緣連線。您可以檢視節點鄰近項目以查看物件彼此之間的關聯性，然後向下鑽研以視覺方式檢查邊緣和屬性。
- 您也可以自訂圖形配置、顏色、圖示，以及要針對節點和邊緣顯示的預設屬性。對於 RDF 圖形，您也可以自訂資源 URI 的命名空間。
- 對於涉及圖形資料的報告和簡報，您可以設定並儲存您以高解析度 PNG 格式所建立的檢視。您也可以將相關聯的資料下載至 CSV 或 JSON 檔案，以便進一步處理。

在 Neptune 圖形筆記本中使用圖形總管

與 Neptune 一起使用圖形總管的最簡單方法是在 Neptune 圖形筆記本中使用。

如果您使用 Neptune 工作台託管 Neptune 筆記本，圖形總管會自動與筆記本一起部署，並連接到 Neptune。

建立筆記本之後，移至 Neptune 主控台以啟動圖形總管：

1. 移至 Neptune。
2. 在筆記本下，選擇您的筆記本。
3. 在「動作」下選擇開啟圖形總管

如何在 AWS Fargate 上的 Amazon ECS 中執行圖形總管並連線到 Neptune

您也可以建立圖形資源管理器 Docker 映像，然後在本機電腦或託管服務 (例如 Amazon 彈性運算雲端 (Amazon EC2) 或 Amazon E lastic Container Service (Amazon ECS) 上執行，如圖形資源管理器專案中讀取我的入門章節所述。 GitHub

例如，本節提供在 Amazon ECS 中執行圖形資源管理器的 step-by-step 說明： AWS Fargate

1. 建立新的 IAM 角色，並將這些政策附加到該角色。

   - AmazonECS TaskExecution RolePolicy
   - CloudWatchLogsFull存取

   讓角色名稱方便立即使用。
2. 建立 Amazon ECS 叢集，其中基礎設施設定為 FARGATE 並具有下列聯網選項：

   - VPC：設定為 Neptune 資料庫所在的 VPC。
   - Subnets：設定為該 VPC 的公有子網路 (移除所有其他子網路)。
3. 建立新的 JSON 任務定義，如下所示：

```
{
  "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. 使用預設設定啟動新任務，但下列欄位除外：

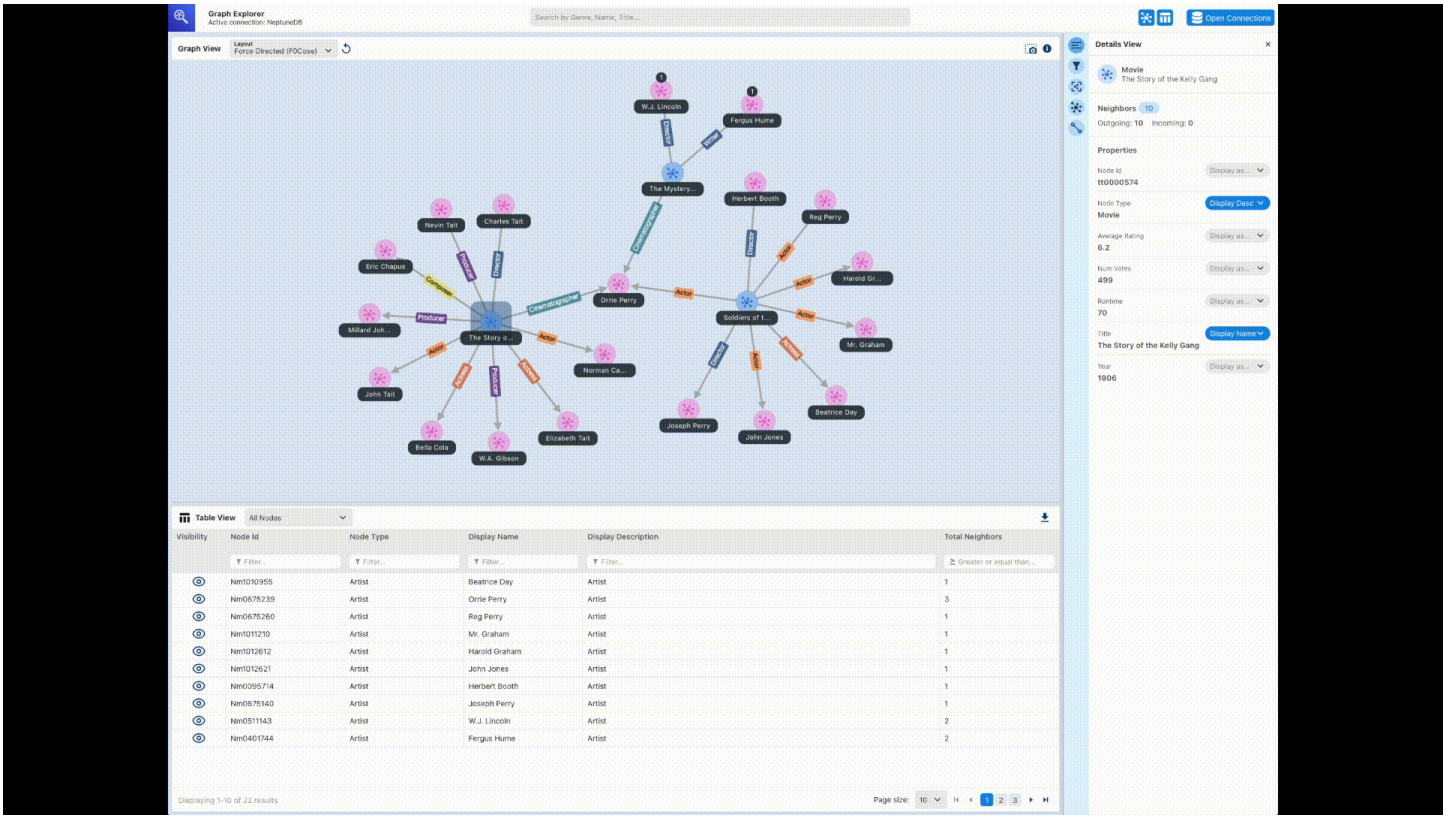
   - Environment (環境)
     - 運算選項 => 啟動類型
   - Deployment configuration (部署組態)
     - 應用程式類型 => 任務
     - 系列 => *### JSON ####)*
     - 修訂版 => *(##)*
   - 聯網
     - VPC => *(####### Neptune VPC)*
     - 子網路 => *(## VPC ###### - #########)*
     - 安全群組 => 建立新的安全群組
     - 安全群組名稱 => 圖形總管
     - 安全群組描述 => 用於存取圖形總管的安全群組
     - 安全群組的傳入規則 =>
       1. 80 Anywhere
       2. 443 Anywhere
5. 選取建立。
6. 任務開始後，複製執行中任務的公有 IP，然後導覽至：`https://(your public IP)/explorer`。
7. 接受使用已產生但無法辨識之憑證的風險，或將其新增至您的金鑰鏈。
8. 現在您可以將連線新增至 Neptune。為屬性圖 (LPG) 或 RDF 建立新連線，並設定下列欄位：

```
Using proxy server => true
Public or Proxy Endpoint => https://(your public IP address)
Graph connection URL => https://(your Neptune endpoint):8182
```

您現在應該已連線。

圖形總管示範

這個短片讓您了解如何使用圖形總管輕鬆地將圖形資料視覺化：



Tom Sawyer 軟體

Tom Sawyer Perspectives是一種低程式碼圖形和資料的視覺化和分析開發平台，適用於Amazon Neptune 中儲存的資料。整合式設計與預覽介面，以及廣泛的 API 程式庫，可讓您快速建立自訂的生產品質視覺化應用程式。透過設計 point-and-click 師介面和 30 種內建分析演算法，您可以設計和開發應用程式，以深入瞭解來自數十個來源聯合的資料。

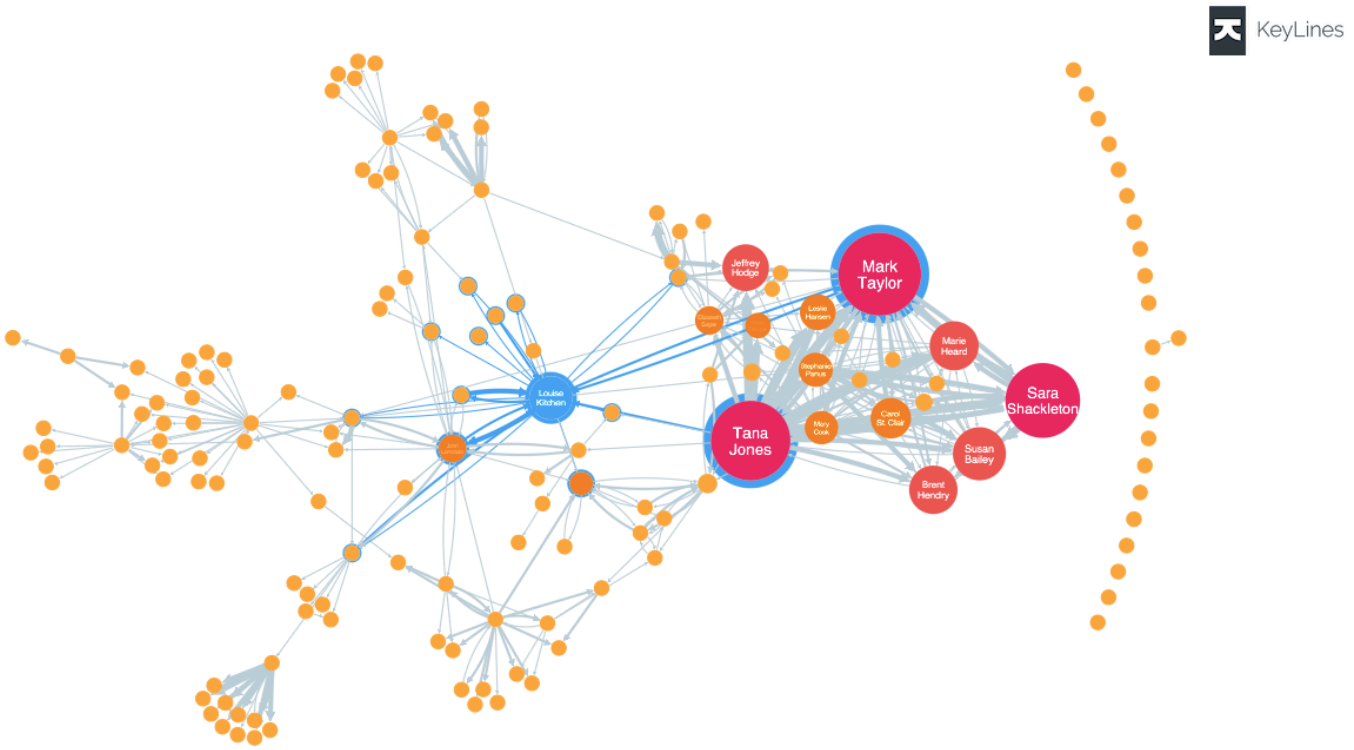
Tom Sawyer Graph Database Browser 可讓您輕鬆地視覺化和分析 Amazon Neptune 中的資料。您可以查看和了解資料中的連線，而無需廣泛了解查詢語言或結構描述。您只要載入所選節點的鄰近項目，並以任何您需要的指示建置視覺化，就能在沒有技術知識的情況下與資料互動，。您還可以利用五種獨特的圖形配置，以提供最大意義的方式顯示圖形，並且可以套用集中性、叢集和路徑尋找分析來顯示以前看不到的模式。若要查看 Graph Database Browser 與 Neptune 整合的範例，請參閱這篇部落格文章。若要開始免費試用 Graph Database Browser，請造訪 AWS Marketplace。

Cambridge Intelligence

Cambridge Intelligence 提供資料視覺化技術，用於探索和了解 Amazon Neptune 資料。圖形視覺化工具包（KeyLines適用於 JavaScript 開發人員和 ReGraphReact 開發人員）提供了一種簡單的方法來為 Web 應用程序構建高度互動和可定制的工具。這些工具組利用 WebGL 和 HTML5 Canvas 實現快速效能，它們支援高階圖形分析功能，並將靈活性和可擴展性與安全、強大的架構結合在一起。這些 SDK 會同時使用 Neptune Neptune 和 RDF 資料。

查看這些針對 Gremlin 資料、SPARQL 資料和 Neptune 架構的整合教學課程。

以下是範例 KeyLines 視覺效果：

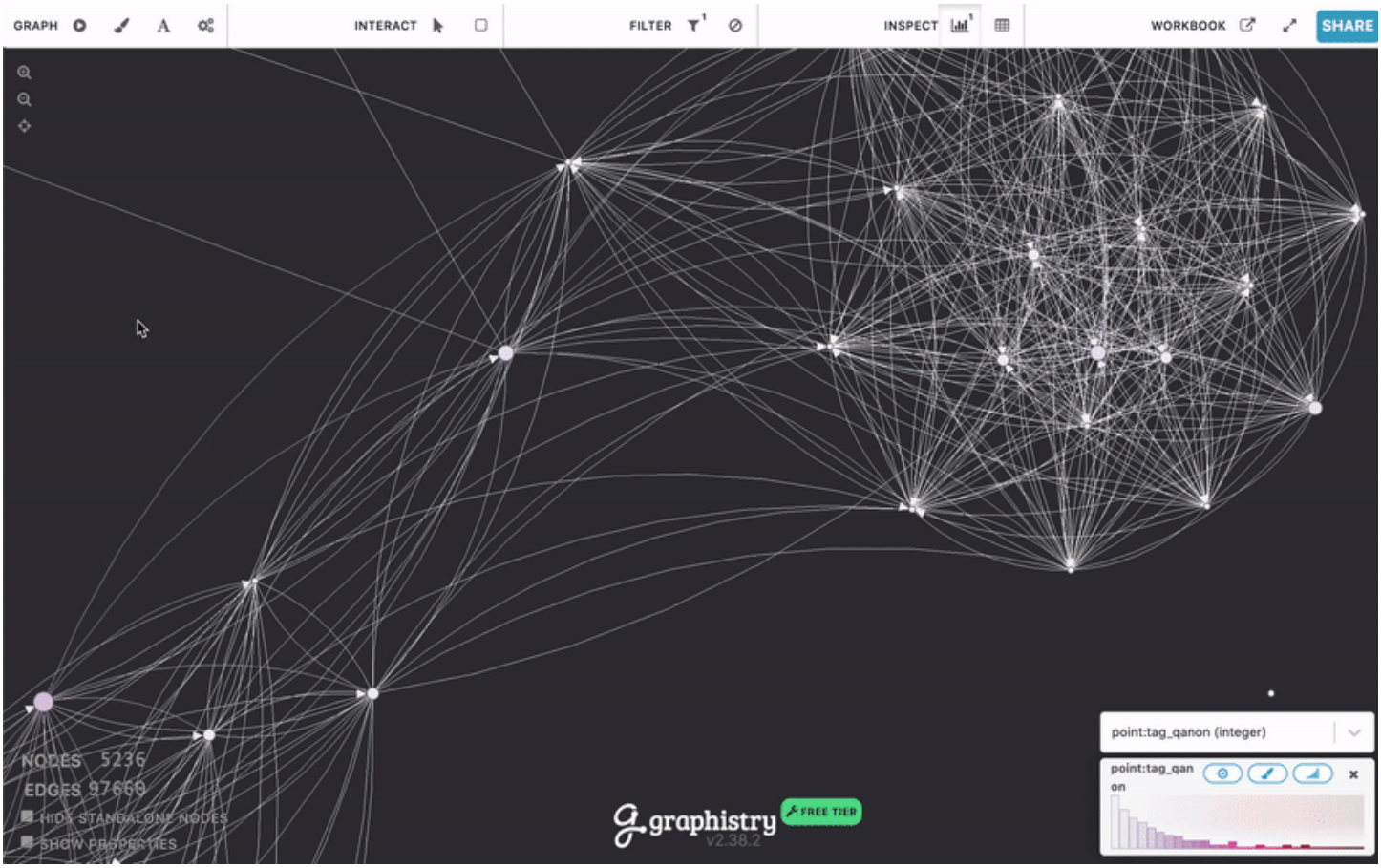


Graphistry

Graphstry 是一種視覺化圖形智慧平台，其會利用 GPU 加速以取得豐富的視覺體驗。團隊可以使用各種功能在 Graphstry 上進行協作，從無程式碼探索檔案和資料庫，到共用 Jupyter 筆記本和 Streamlit 儀表板，再到在您自己的應用程式中使用內嵌 API。

只要設定和啟動 graph-app-kit，並僅修改幾行程式碼，您就可以開始使用低編碼的完全互動式儀表板。如需使用 Graphistry 和 Neptune 建立第一個儀表板的演練，請參閱這篇部落格文章。您也可以嘗試 Neptune PyGraphistry示範。 PyGraphistry 是筆記本的 Python 視覺化圖形分析程式庫。查看此教程筆記本以獲取 Neptune PyGraphistry 演示。

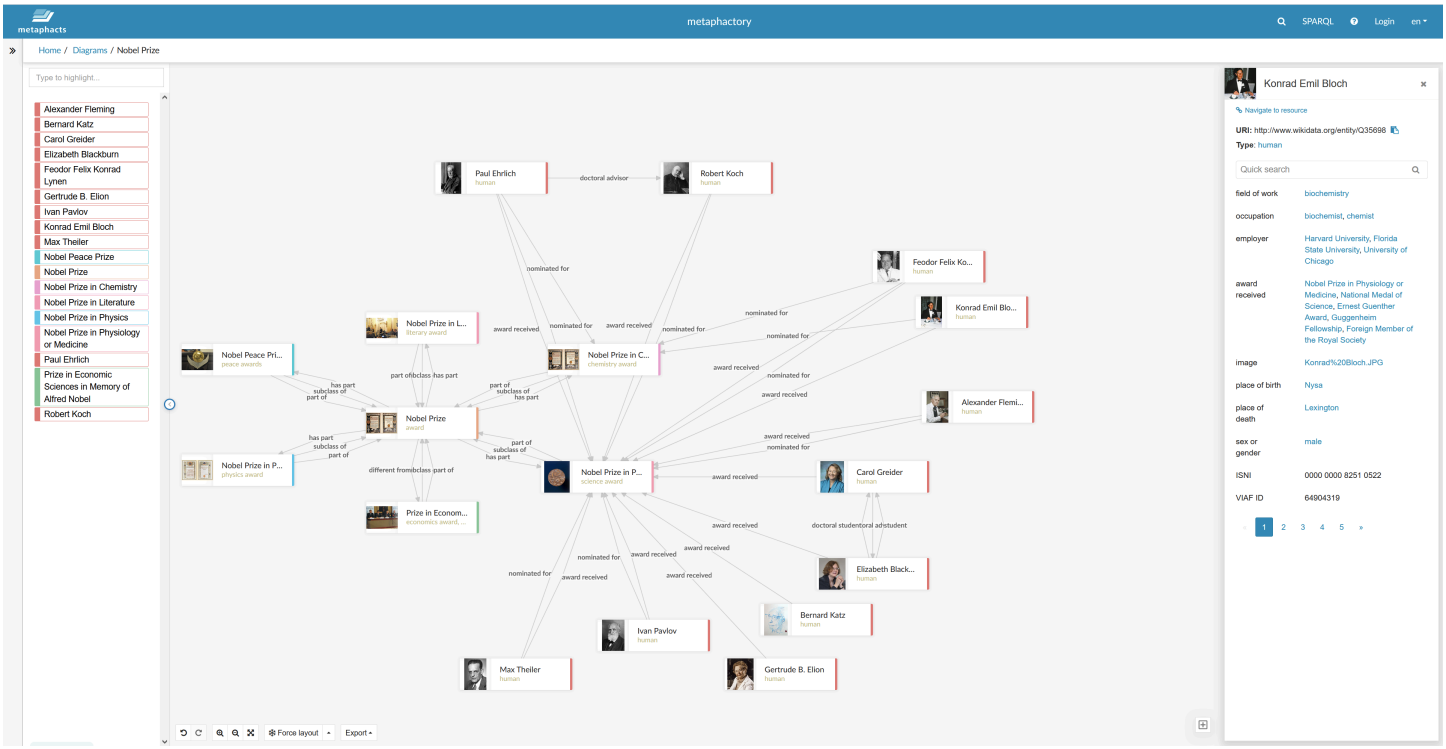
要開始使用，請訪問 AWS Marketplace 中的石墨工學。



metaphacts

metaphacts 提供了一個靈活的開放平台，用於描述和查詢圖形資料，以及視覺化知識圖譜並與其互動。使用 metaphactory，您可以使用 RDF 資料模型，建置互動式 Web 應用程式，例如 Neptune 中知識圖譜頂端的視覺化和儀表板。metaphactory 平台支援低程式碼開發體驗，其中包含用於資料載入的 UI、支援 OWL 和 SHACL 的視覺本體建模介面、SPARQL 查詢 UI 和查詢目錄，以及一組用於圖形探索、視覺化、搜尋和編寫的豐富 Web 元件。

以下是範例 metaphactory 視覺化：



此平台專為工程、製造、製藥、生命科學、金融、保險等領域而設計並在其中高效使用。若要查看範例解決方案架構，請參閱這篇部落格文章。

若要開始免費試用 metaphactory，請造訪 AWS Marketplace。

G.V( )

G.V () 是面向開發人員和數據分析師的強大的 Gremlin 集成開發環境（IDE）工具。使用它，您可以在 Neptune 中以交互方式查詢、視覺化和更新圖形資料。G.V () 提供內建的 Gemlin 語言自動完成功能，根據您的圖形資料模型，在您輸入查詢時提供建議和文件。

您也可以使用 Gremlin 查詢除錯功能，深入撰寫、偵錯、測試和分析圖形周遊程序。

使用 OpenAI 提供支援的自然語言處理功能，G.V () 可以從文字提示中產生精確的圖形資料結構描述的 Gremlin 查詢，以便透過自然語言查詢您的資料。

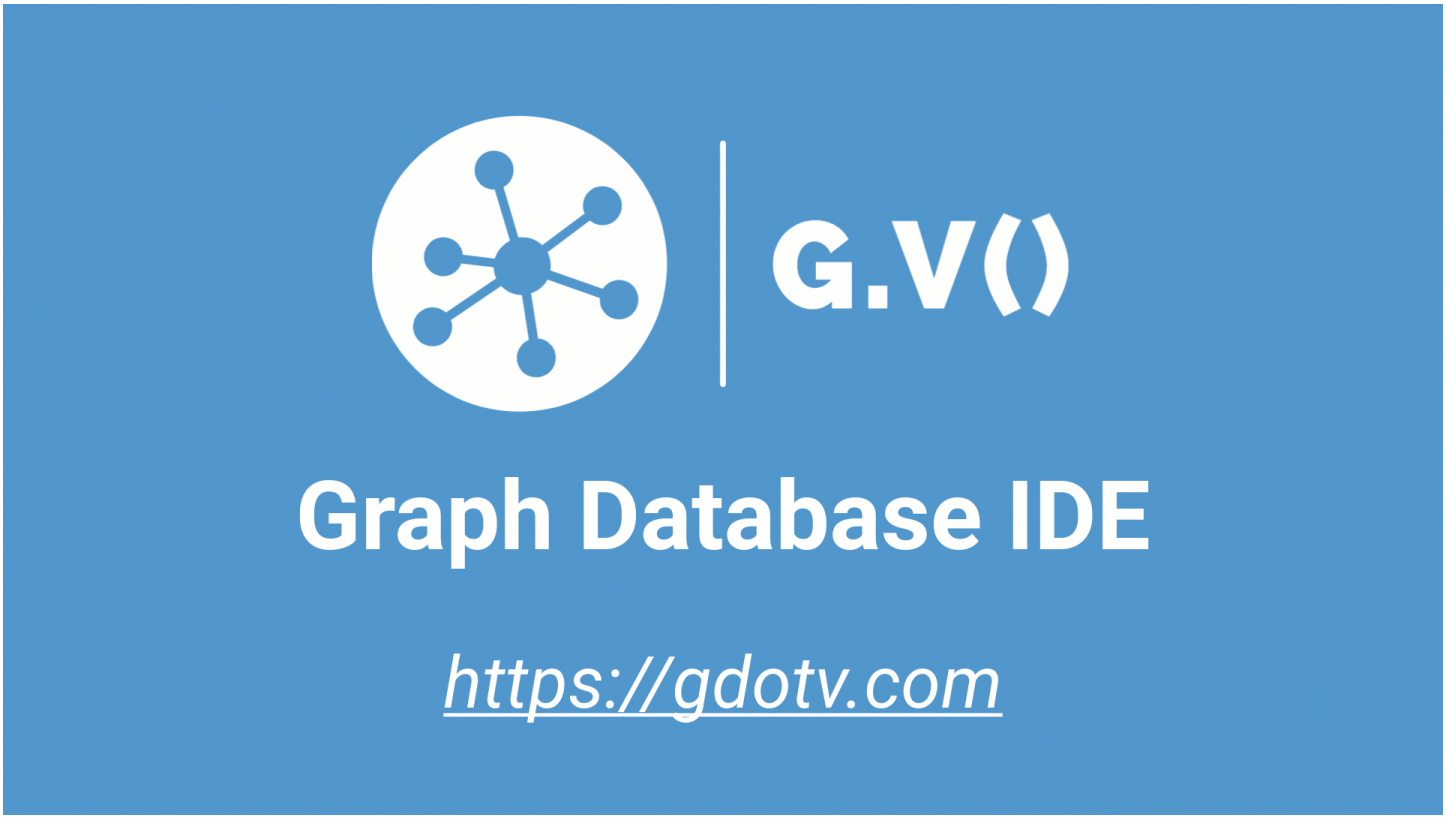
「圖表資料總管」可讓您瀏覽和修改圖形，以快速建構新的圖形結構並維護現有的圖形結構。

G.V () 為查詢結果提供多種視覺化格式，可協助您以互動方式解譯查詢輸出並瀏覽圖形。這些包括資料表、圖形，JSON 和 Gemlin 主控台輸出格式。

G.V () 與 Amazon Neptune 完全相容，並提供許多專門針對 Amazon Neptune 的其他功能，例如慢速查詢或稽核日誌深入解析，以及 IAM 身份驗證支援。若要進一步了解，請查看文件。

G.V () 不斷發展，每月都會收到新功能。要了解有關 G.V ( ) 的更多信息，請訪問 G.V ( ) 網站開始免費試用。

請參閱以下 G.V ( ) 的演示：



連結的

Linkurious 為技術和非技術使用者以及各種使用案例提供不同的圖形智慧解決方案。

Linkurious 企業資源管理器是專為團隊構建的 off-the-shelf 圖形可視化和分析軟件，可以跟上您的 day-to-day 活動的需求，並幫助數據驅動的專業人員做大事-簡單。完全可配置且易於使用，它可以輕鬆適應您的需求，並使新手或進階使用者能夠在 AWS Neptune 中快速視覺化資料，無論資料的大小或複雜性如何，都能直觀地探索您的資料集，並在團隊或企業層級無縫協作。

Linkurious 企業崗樓利用 Linkurious 企業資源管理器的力量，並增加了創新的檢測和案例管理功能，以提供由圖形技術提供支持的集成檢測和調查軟件。一方面，它使您能夠配置警報，利用 Neptune 數據庫和 Neptune Analytics 自動顯示複雜連接數據中的異常或模式。另一方面，它結合了案例管理和協作功能，以幫助團隊有效地管理調查工作流程。

Ogma 是一個商業 JavaScript 庫，可幫助您為應用程序開發功能強大的大規模交互式圖形可視化。它利用 WebGL 渲染和高性能佈局，使用戶能夠在幾秒鐘內顯示和與數千個節點和邊緣進行交互。它還提

供了各種功能來自定義您的應用程序並創建豐富的用戶體驗。最後，它配備了全面的文檔和工具，例如教程，數十個示例和交互式遊樂場。

要開始使用，請申請 30 天免費試用 Linkurious 企業或奧格瑪。

從 Neptune 資料庫叢集匯出資料

有數種好方法可從 Neptune 資料庫叢集匯出資料：

- 對於少量資料，只需使用一或多個查詢的結果即可。
- 對於 RDF 資料，圖形儲存通訊協定 (GSP) 可使匯出變得容易。例如：

```
curl --request GET \
  'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/graph'
```

- 還有一個功能強大且靈活的開放原始碼工具，用於匯出 Neptune 資料，即 `neptune-export`。下列各節描述此工具的功能及其使用方式。

主題

- 使用 neptune-export
- 使用 Neptune-Export 服務匯出 Neptune 資料
- 使用 neptune-export 命令列工具從 Neptune 匯出資料
- 由 Neptune-Export 和 neptune-export 匯出的檔案
- 用來控制 Neptune 匯出程序的參數
- 對 Neptune 匯出程序進行疑難排解

使用 neptune-export

您可以採取兩種不同的方式來使用開放原始碼 neptune-export 工具：

- 作為 Neptune-Export 服務。 使用 Neptune-Export 服務從 Neptune 匯出資料時，您可以透過 REST API 觸發和監控匯出工作。
- 作為 **neptune-export** Java 命令列公用程式。 若要使用此命令列工具匯出 Neptune 資料，您必須在可存取 Neptune 資料庫叢集的環境中執行它。

Neptune-Export 服務和 neptune-export 命令列工具兩者都會將資料發佈至 Amazon Simple Storage Service (Amazon S3)，並使用 Amazon S3 伺服器端加密 (SSE-S3) 進行加密。

> **Note**
>
> 最佳實務是在所有 Amazon S3 儲存貯體上啟用存取記錄，讓您稽核所有對這些儲存貯體的存取。

如果您嘗試從 Neptune 資料庫叢集匯出資料，但該叢集的資料在進行匯出時發生變更，則無法保證所匯出資料的一致性。也就是說，如果您的叢集在匯出工作正在進行時為寫入流量提供服務，則匯出的資料可能會有不一致的情況。無論您是從叢集中的主要執行個體匯出，還是從一或多個僅供讀取複本匯出，都是如此。

為了保證匯出的資料一致，最好從複製的資料庫叢集中匯出。這兩者都會為匯出工具提供了資料的靜態版本，並確保匯出工作不會減慢原始資料庫叢集中的查詢速度。

若要讓此操作更輕鬆，您可以指示您要在觸發匯出工作時複製來源資料庫叢集。如果這樣做，匯出程序會自動建立複製、並將其用於匯出，然後在匯出完成時將其刪除。

使用 Neptune-Export 服務匯出 Neptune 資料

您可以使用以下步驟，來使用 Neptune-Export 服務，將資料從 Neptune 資料庫叢集匯出至 Amazon S3：

安裝 Neptune-Export 服務

使用 AWS CloudFormation 範本建立堆疊：

安裝 Neptune-Export 服務

1. 在 AWS CloudFormation 主控台啟動 AWS CloudFormation 堆疊，方法是在下表中選擇其中一個啟動堆疊按鈕。

區域	檢視	在設計工具中檢視	啟動
美國東部 (維吉尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國東部 (俄亥俄)	檢視	在設計工具中檢視	Launch Stack
美國西部 (加利佛尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國西部 (奧勒岡)	檢視	在設計工具中檢視	Launch Stack
加拿大 (中部)	檢視	在設計工具中檢視	Launch Stack
南美洲 (聖保羅)	檢視	在設計工具中檢視	Launch Stack
歐洲 (斯德哥爾摩)	檢視	在設計工具中檢視	Launch Stack
歐洲 (愛爾蘭)	檢視	在設計工具中檢視	Launch Stack
歐洲 (倫敦)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
歐洲 (巴黎)	檢視	在設計工具中檢視	Launch Stack
歐洲 (法蘭克福)	檢視	在設計工具中檢視	Launch Stack
中東 (巴林)	檢視	在設計工具中檢視	Launch Stack
中東 (阿拉伯聯合大公國)	檢視	在設計工具中檢視	Launch Stack
以色列 (特拉維夫)	檢視	在設計工具中檢視	Launch Stack
非洲 (開普敦)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (香港)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (東京)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (首爾)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (新加坡)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (雪梨)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (孟買)	檢視	在設計工具中檢視	Launch Stack
中國 (北京)	檢視	在設計工具中檢視	Launch Stack
中國 (寧夏)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
AWS GovCloud (美國西部)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國東部)	檢視	在設計工具中檢視	Launch Stack

2. 在 Select Template (選取範本) 頁面上，請選擇 Next (下一步)。
3. 在指定詳細資訊頁面上，設定下列參數。

   - **VPC** – 設定 Neptune-Export 服務的最簡單方法是將其安裝在與 Neptune 資料庫相同的 Amazon VPC 中。如果要將其安裝在個別的 VPC 中，您可以使用 VPC 對等互連，在 Neptune 資料庫叢集的 VPC 與 Neptune-Export 服務 VPC 之間建立連線。
   - **Subnet1** – Neptune-Export 服務必須安裝在 VPC 中的子網路中，該子網路允許從子網路到網際網路的傳出 IPv4 HTTPS 流量。這樣，Neptune-Export 服務可以呼叫 AWS 批次 API，來建立和執行匯出工作。

     如果已使用 Neptune 文件中 建立資料庫叢集 頁面上的 CloudFormation 範本，建立了 Neptune 叢集，則您可以使用來自該堆疊的 `PrivateSubnet1` 和 `PrivateSubnet2` 輸出，來填入此參數和下一個參數。
   - **Subnet2** – VPC 中的第二個子網路，其允許從子網路到網際網路的傳出 IPv4 HTTPS 流量。
   - **EnableIAM** – 將其設為 `true`，以使用 AWS Identity and Access Management (IAM) 保護 Neptune-Endpoint API。建議您這樣做。

     如果您確實啟用 IAM 身分驗證，則必須以 `Sigv4` 簽署對端點的所有 HTTPS 請求。您可以使用 awscurl 等工具代表您簽署請求。
   - **VPCOnly** – 將其設為 `true`，使匯出端點成為唯一 VPC，以便您只能從安裝 Neptune-Export 服務的 VPC 內存取它。這會限制 Neptune-Export API 只能從該 VPC 內使用。

     建議您將 `VPCOnly` 設定為 `true`。
   - **NumOfFilesULimit** – 在 `ulimits` 容器屬性中，為 `nofile` 指定介於 10,000 與 1,000,000 之間的值。預設值為 10,000，除非您的圖形包含大量的唯一標籤，否則我們建議您保留預設值。
   - **PrivateDnsEnabled** (布林值) – 指示是否要將私有託管區域與指定的 VPC 建立關聯。預設值為 `true`。

在啟用此旗標的情況下建立 VPC 端點時，所有 API Gateway 流量都會透過 VPC 端點路由傳送，且公用 API Gateway 端點呼叫會變成停用狀態。如果將 `PrivateDnsEnabled` 設為 `false`，則會啟用公用 API Gateway 端點，但 Neptune 匯出服務無法透過私有 DNS 端點連線。然後，您可以使用 VPC 端點的公用 DNS 端點呼叫匯出服務，如這裡所述。

4. 選擇 Next (下一步)。
5. 在 Options (選項) 頁面上，選擇 Next (下一步)。
6. 在檢閱頁面上，選取第一個核取方塊以確認 AWS CloudFormation 將建立 IAM 資源。選取第二個核取方塊，確認新堆疊 `CAPABILITY_AUTO_EXPAND`。

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` 明確確認在建立堆疊時，無需事先檢閱，即可擴充巨集。使用者通常會在處理過的範本中建立變更集，以便在實際建立堆疊之前，檢閱巨集所做的變更。如需詳細資訊，請參閱 AWS CloudFormationCreateStack API。

然後選擇 Create (建立)。

啟用從 Neptune-Export 存取 Neptune

Neptune-Export 安裝完成後，更新您的 Neptune VPC 安全群組，以允許從 Neptune-Export 存取。建立 Neptune-Export AWS CloudFormation 堆疊後，輸出索引標籤會包含 `NeptuneExportSecurityGroup` ID。更新您的 Neptune VPC 安全群組，以允許從這個 Neptune-Export 安全群組存取。

啟用從 VPC 型 EC2 執行個體存取 Neptune-Export 端點

如果您讓 Neptune-Export 端點成為唯一 VPC，則只能從安裝 Neptune-Export 服務的 VPC 內存取它。若要在您可以從中進行 Neptune-Export API 呼叫的 VPC 中，允許來自 Amazon EC2 執行個體的連線，請將 AWS CloudFormation 堆疊建立的 `NeptuneExportSecurityGroup` 附加至該 Amazon EC2 執行個體。

使用 Neptune-Export API 執行 Neptune-Export 工作

AWS CloudFormation 堆疊的輸出索引標籤也包含 `NeptuneExportApiUri`。每當您將請求傳送到 Neptune-Export 端點時，請使用此 URI。

執行匯出工作

- 請確定匯出執行所在的使用者或角色已獲授與 `execute-api:Invoke` 許可。
- 如果在安裝 Neptune-Export 時，於 AWS CloudFormation 堆疊中將 `EnableIAM` 參數設為 `true`，則您需要以 `Sigv4` 簽署對 Neptune-Export API 的所有請求。我們建議您使用 awscurl 向 API 提出請求。這裡的所有範例都假設 IAM 身分驗證已啟用。
- 如果在安裝 Neptune-Export 時，於 AWS CloudFormation 堆疊中將 `VPCOnly` 參數設為 `true`，則必須從 VPC 內呼叫 Neptune-Export API，通常從位於 VPC 的 Amazon EC2 執行個體進行此呼叫。

若要開始匯出資料，請使用 `command` 和 `outputS3Path` 請求參數以及 `endpoint` 匯出參數，將請求傳送至 `NeptuneExportApiUri` 端點。

以下是請求範例，其會從 Neptune 匯出屬性圖資料，並將其發佈至 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)" }
      }'
```

同樣地，以下是將 RDF 資料從 Neptune 匯出至 Amazon S3 的請求範例：

```
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)" }
      }'
```

如果您省略了 `command` 請求參數，根據預設，Neptune-Export 會嘗試從 Neptune 匯出屬性圖資料。

如果上一個命令執行成功，輸出將如下所示：

```
{
  "jobName": "neptune-export-abc12345-1589808577790",
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f"
}
```

監控您剛開始的匯出工作

若要監控執行中工作，請將其 jobID 附加到您的 NeptuneExportApiUri，如下所示：

```
curl \
  (your NeptuneExportApiUri)(the job ID)
```

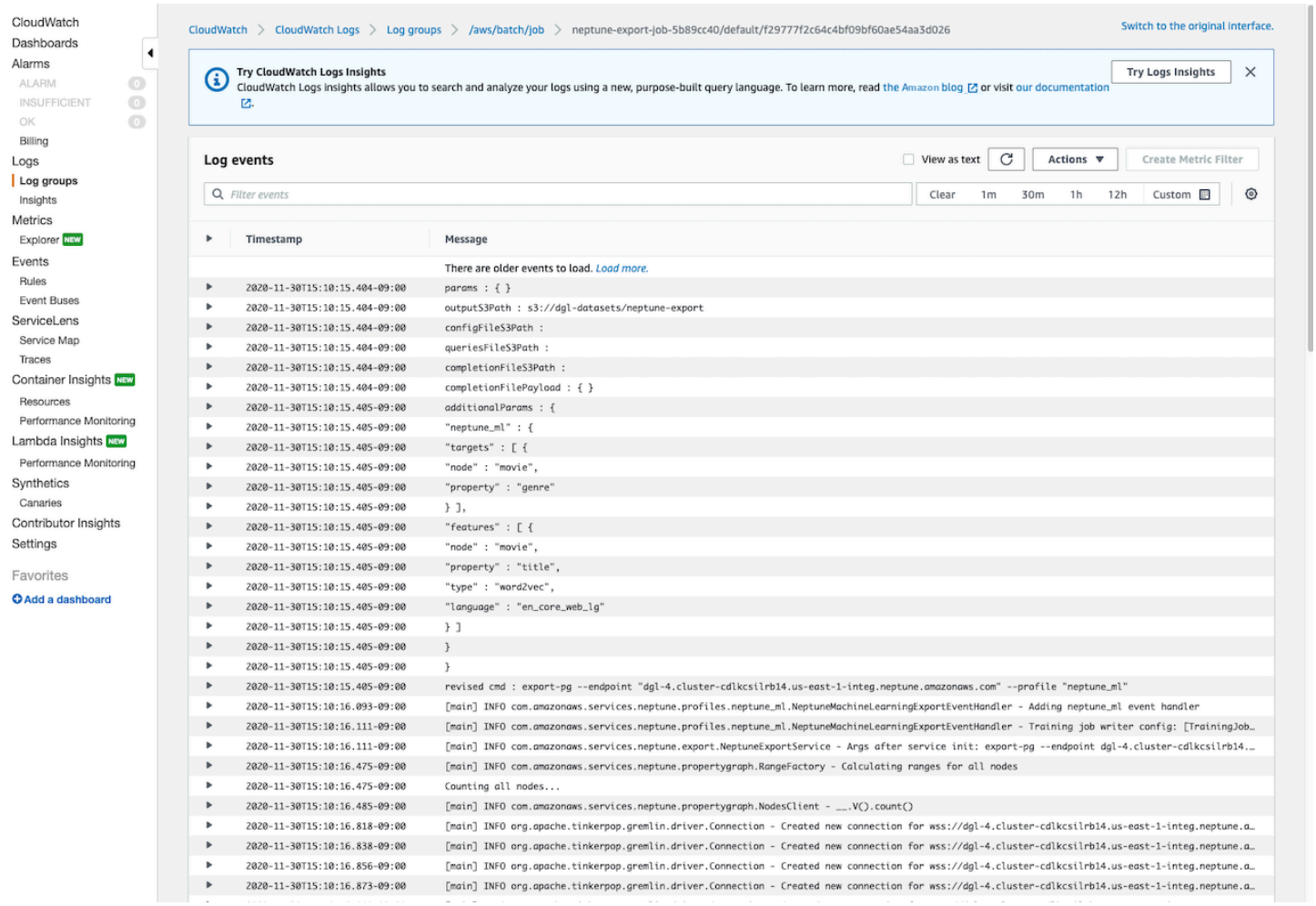
如果服務尚未啟動匯出工作，回應將如下所示：

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "pending"
}
```

當您在匯出工作開始之後重複此命令時，回應會如下所示：

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "running",
  "logs": "https://us-east-1.console.aws.amazon.com/cloudwatch/home?..."
}
```

如果您使用狀態呼叫所提供的 URI 開啟 CloudWatch Logs 中的日誌，則接著可以詳細監控匯出的進度：



取消執行中的匯出工作

使用 AWS Management Console 取消執行中的匯出工作

1. 於 https://console.aws.amazon.com/batch/ 開啟 AWS Batch 主控台。
2. 選擇 Jobs (任務)。
3. 根據 `jobID` 找出您要取消的執行中工作。
4. 選取取消工作。

若要使用 Neptune 匯出 API 取消執行中的匯出工作：

將 `HTTP DELETE` 請求傳送到附加 `jobID` 的 `NeptuneExportApiUri`，如下所示：

```
curl -X DELETE \
  (your NeptuneExportApiUri) (the job ID)
```

使用 **neptune-export** 命令列工具從 Neptune 匯出資料

您可以使用以下步驟，來使用 `neptune-export` 命令列公用程式，將資料從 Neptune 資料庫叢集匯出至 Amazon S3：

使用 **neptune-export** 命令列公用程式的先決條件

開始之前

- 具有第 8 版 JDK – 您需要安裝 Java SE 開發套件 (JDK) 第 8 版。
- 下載 neptune-export 公用程式 – 下載並安裝 neptune-export.jar 檔案。
- 確定 **neptune-export** 可以存取 Neptune VPC – 從其可存取 VPC (Neptune 資料庫叢集所在) 的位置執行 neptune-export。

  例如，您可以在 Neptune VPC 內的 Amazon EC2 執行個體上執行它，或在與 Neptune VPC 對等互連的個別 VPC 中執行，或在個別的堡壘主機上執行。
- 確定 VPC 安全群組授與 **neptune-export** 的存取權 – 檢查附加至 Neptune VPC 的 VPC 安全群組是否允許從與 `neptune-export` 環境相關聯的 IP 地址或安全群組存取您的資料庫叢集。
- 設定必要的 IAM 許可 – 如果您的資料庫已啟用 AWS Identity and Access Management (IAM) 資料庫身分驗證，請確保 `neptune-export` 執行所在的角色與允許連線 Neptune 的 IAM 政策相關聯。如需 Neptune 政策的相關資訊，請參閱 使用 IAM 政策。

  如果您想要在查詢請求中使用 `clusterId` 匯出參數，`neptune-export` 執行所在的角色需要下列 IAM 許可：

  - `rds:DescribeDBClusters`
  - `rds:DescribeDBInstances`
  - `rds:ListTagsForResource`

  如果想要從複製的叢集匯出，`neptune-export` 執行所在的角色需要下列 IAM 許可：

  - `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`

  若要將匯出的資料發佈到 Amazon S3，`neptune-export` 執行所在的角色需要 Amazon S3 位置的下列 IAM 許可：

  - `s3:PutObject`
  - `s3:PutObjectTagging`
  - `s3:GetObject`

- 設定 **`SERVICE_REGION`** 環境變數 – 設定 `SERVICE_REGION` 環境變數以識別資料庫叢集所在的區域 (如需區域識別符清單，請參閱連線至 Neptune)。

執行 `neptune-export` 公用程式以啟動匯出操作

使用下列命令從命令行執行 neptune-export 並啟動匯出操作：

```
java -jar neptune-export.jar nesvc \
  --root-path (path to a local directory) \
  --json (the JSON file that defines the export)
```

此命令有兩個參數：

開始匯出時 neptune-export 的參數

- **`--root-path`** – 本機目錄的路徑，其中匯出檔案在發佈到 Amazon S3 之前會寫入該目錄。
- **`--json`** – 定義匯出的 JSON 物件。

使用 neptune-export 命令列公用程式的範例命令

若要直接從來源資料庫叢集匯出屬性圖資料：

```
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)"
            }
          }'
```

若要直接從來源資料庫叢集匯出 RDF 資料：

```
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)"
            }
          }'
```

如果您省略了 command 請求參數，根據預設，neptune-export 公用程式會嘗試從 Neptune 匯出屬性圖資料。

若要從您複製的資料庫叢集匯出：

```
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
            }
          }'
```

若要使用 IAM 身分驗證從資料庫叢集匯出：

```
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
            }
          }'
```

由 Neptune-Export 和 **neptune-export** 匯出的檔案

匯出完成時，匯出檔案會發佈至您所指定的 Amazon S3 位置。發佈至 Amazon S3 的檔案都會使用 Amazon S3 伺服器端加密 (SSE-S3) 進行加密。發佈至 Amazon S3 的資料夾和檔案會有所不同，取決於您要匯出屬性圖還是 RDF 資料。如果開啟檔案發佈所在的 Amazon S3 位置，您會看到下列內容：

Amazon S3 中所匯出檔案的位置

- **nodes/** – 此資料夾包含逗號分隔值 (CSV) 或 JSON 格式的節點資料檔案。

  在 Neptune，節點可有一個或多個標籤。具有不同個別標籤 (或多個標籤的不同組合) 的節點會寫入至不同的檔案，這表示沒有個別檔案包含具有不同標籤組合之節點的資料。如果節點具有多個標籤，這些標籤會在指派給檔案之前依字母順序排序。
- **edges/** – 此資料夾包含逗號分隔值 (CSV) 或 JSON 格式的邊緣資料檔案。

  與節點檔案一樣，邊緣資料會根據其標籤組合寫入至不同的檔案。基於模型訓練的目的，會根據邊緣標籤的組合，加上邊緣起始和結束節點的標籤組合，將邊緣資料指派給不同的檔案。
- **statements/** – 此資料夾包含 Turtle、N-Quads、N-Triples 或 JSON 格式的 RDF 資料檔案。
- **config.json** – 此檔案包含匯出程序所推斷之圖形的結構描述。
- **lastEventId.json** – 此檔案包含資料庫 Neptune 串流上最後一個事件的 `commitNum` 和 `opNum`。只有當您將 `includeLastEventId` 匯出參數設為 `true`，且您要從中匯出資料的資料庫已啟用 Neptune 串流時，匯出程序才會包含此檔案。

用來控制 Neptune 匯出程序的參數

無論您使用的是 Neptune-Export 服務還是 `neptune-export` 命令行公用程式，您用來控制匯出的參數大部分都是相同的。它們包含傳遞至 Neptune-Export 端點或傳遞至命令行上 `neptune-export` 的 JSON 物件。

傳入匯出程序的物件最多可有五個頂層欄位：

```
-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) }
   }'
```

內容

- 使用 rdfExportScope 和 sparql 匯出特定邊緣
- 用namedGraph來匯出單一具名圖形

command 參數

command 頂層參數決定要匯出屬性圖資料還是 RDF 資料。如果您省略 command 參數，匯出程序預設為匯出屬性圖資料。

- **export-pg** – 匯出屬性圖資料。
- **export-rdf** – 匯出 RDF 資料。

outputS3Path 參數

outputS3Path 頂層參數為必要參數，且必須包含可將已匯出檔案發佈至其中之 Amazon S3 位置的 URI：

```
  "outputS3Path" : "s3://(your Amazon S3 bucket)/(path to output folder)"
```

此值必須以 s3:// 開始，其後跟著有效的儲存貯體名稱，以及選擇性地跟著儲存貯體內的資料夾路徑。

jobSize 參數

jobSize 頂層參數只會與 Neptune-Export 服務搭配使用，而不會與 neptune-export 命令列公用程式搭配使用，而且是選用的。它可讓您描述正在啟動的匯出工作大小，這有助於判斷專用於工作的運算資源數量及其最大並行層級。

```
  "jobsize" : "(one of four size descriptors)"
```

四個有效的大小描述項如下：

- small – 並行上限：8。適用於高達 10 GB 的儲存磁碟區。
- medium – 並行上限：32。適用於高達 100 GB 的儲存磁碟區。
- large – 並行上限：64。適用於超過 100 GB 但小於 1 TB 的儲存磁碟區。
- xlarge – 並行上限：96。適用於超過 1 TB 的儲存磁碟區。

根據預設，在 Neptune-Export 服務上啟動的匯出會以 `small` 工作的形式執行。

匯出的效能不僅取決於 `jobSize` 設定，還取決於您要從中匯出的資料庫執行個體數目、每個執行個體的大小，以及工作的有效並行層級。

對於屬性圖匯出，您可以使用 cloneClusterReplica伯爵 參數設定資料庫執行個體的數目，也可以使用 concurrency 參數設定工作的有效並行層級。

`params` 物件

`params` 頂層參數是 JSON 物件，其中包含您用來控制匯出程序本身的參數，如 匯出 `params` 頂層 JSON 物件中的參數欄位 中所述。`params` 物件中的有些欄位是屬性圖匯出特有的，有些欄位則是 RDF 特有的。

`additionalParams` 物件

`additionalParams` 頂層參數是 JSON 物件，您可以使用其中包含的參數，控制在資料匯出之後套用至該資料的動作。目前，`additionalParams` 僅用於匯出 Neptune ML 的訓練資料。

匯出 **params** 頂層 JSON 物件中的參數欄位

Neptune 匯出 params JSON 物件可讓您控制匯出，包括所匯出資料的類型和格式。

所匯出參數 **params** 物件中可能欄位的清單

下面列出了可以出現在 params 物件中的所有可能頂層欄位。只有這些欄位的子集才會出現在任何一個物件中。

所有匯出類型通用的欄位清單

- cloneCluster
- cloneClusterInstanceType
- cloneClusterReplicaCount
- clusterId
- endpoint
- endpoints
- profile
- useIamAuth
- includeLastEventId

屬性圖匯出的欄位清單

- concurrency
- edgeLabels
- filter
- filterConfigFile
- gremlinFilter
- gremlinNodeFilter
- gremlinEdgeFilter
- format
- nodeLabels
- scope

RDF 匯出的欄位清單

- format
- rdfExportScope
- sparql
- namedGraph

所有匯出類型通用的欄位

params 中的 cloneCluster 欄位

(選用)。預設：false。

如果 cloneCluster 參數設為 true，則匯出程序會使用資料庫叢集的快速複製：

```
  "cloneCluster" : true
```

依預設，匯出程序會從您使用 endpoint、endpoints 或 clusterId 參數指定的資料庫叢集中匯出資料。不過，如果您的資料庫叢集在匯出進行時使用中，且資料正在變更，則匯出程序無法保證要匯出之資料的一致性。

若要確保匯出的資料一致，請改用 cloneCluster 參數，從資料庫叢集的靜態複製進行匯出。

複製的資料庫叢集是在與來源資料庫叢集相同的 VPC 中建立的，並繼承來源的安全群組、子網路群組和 IAM 資料庫身分驗證設定。匯出完成時，Neptune 會刪除複製的資料庫叢集。

根據預設，複製的資料庫叢集由與來源資料庫叢集中主要執行個體具有相同執行個體類型的單一執行個體組成。您可以變更用於所複製資料庫叢集的執行個體類型，方法為使用 cloneClusterInstanceType 指定不同的執行個體類型。

> **Note**
>
> 如果您沒有使用 cloneCluster 選項，而且要直接從主資料庫叢集匯出，則可能需要增加要從中匯出資料之執行個體上的逾時。對於大型資料集，逾時應設定為數小時。

params 中的 cloneClusterInstanceType 欄位

(選用)。

如果 cloneCluster 參數存在且設為 true，您可以使用 cloneClusterInstanceType 參數，指定用於所複製資料庫叢集的執行個體類型：

根據預設，複製的資料庫叢集由與來源資料庫叢集中主要執行個體具有相同執行個體類型的單一執行個體組成。

```
  "cloneClusterInstanceType" : "(for example, r5.12xlarge)"
```

params 中的 cloneClusterReplicaCount 欄位

(選用)。

如果 cloneCluster 參數存在且設為 true，則您可以使用 cloneClusterReplicaCount 參數，指定所複製資料庫叢集中建立的僅供讀取複本數目：

```
  "cloneClusterReplicaCount" : (for example, 3)
```

根據預設，複製的資料庫叢集由單一主要執行個體組成。cloneClusterReplicaCount 參數可讓您指定應該建立多少個僅供讀取複本執行個體。

params 中的 clusterId 欄位

(選用)。

clusterId 參數會指定要使用之資料庫叢集的 ID：

```
  "clusterId" : "(the ID of your DB cluster)"
```

如果您使用 clusterId 參數，則匯出程序會使用該資料庫叢集中的所有可用執行個體來擷取資料。

> **Note**
>
> endpoint、endpoints 和 clusterId 參數互斥。僅使用其中一個。

params 中的 endpoint 欄位

(選用)。

使用 endpoint 來指定資料庫叢集中 Neptune 執行個體的端點，匯出程序可以查詢該資料庫叢集來擷取資料 (請參閱 端點連線)。這僅是 DNS 名稱，不包括通訊協定或連接埠：

```
"endpoint" : "(a DNS endpoint of your DB cluster)"
```

使用叢集或執行個體端點，但不使用主讀取器端點。

> **Note**
>
> endpoint、endpoints 和 clusterId 參數互斥。僅使用其中一個。

params 中的 endpoints 欄位

(選用)。

使用 endpoints 來指定資料庫叢集中端點的 JSON 陣列，匯出程序可以查詢該資料庫叢集來擷取資料 (請參閱 端點連線)。這些僅是 DNS 名稱，不包括通訊協定或連接埠：

```
"endpoints": [
  "(one endpoint in your DB cluster)",
  "(another endpoint in your DB cluster)",
  "(a third endpoint in your DB cluster)"
  ]
```

如果您的叢集中有多個執行個體 (主要和一個或多個僅供讀取複本)，則您可以使用 endpoints 參數，將查詢分發到這些端點的清單，以改善匯出效能。

> **Note**
>
> endpoint、endpoints 和 clusterId 參數互斥。僅使用其中一個。

params 中的 profile 欄位

(必須匯出 Neptune ML 的訓練資料，除非 *neptune_ml* 欄位存在於 *additionalParams* 欄位)。

profile 參數為特定工作負載提供幾組預先設定的參數。目前，匯出程序僅支援 neptune_ml 設定檔

如果您要匯出 Neptune ML 的訓練資料，請將下列參數新增至 params 物件：

```
"profile" : "neptune_ml"
```

params 中的 useIamAuth 欄位

(選用)。預設：false。

如果要從中匯出資料的資料庫已啟用 IAM 身分驗證，則您必須將 useIamAuth 參數設為 true：

```
"useIamAuth" : true
```

params 中的 includeLastEventId 欄位

如果您將 includeLastEventId 設為 true，且您要從中匯出資料的資料庫已啟用 Neptune 串流，則匯出程序會將 lastEventId.json 檔案寫入至您指定的匯出位置。此檔案包含串流中最後一個事件的 commitNum 和 opNum。

```
"includeLastEventId" : true
```

由匯出程序建立的複製資料庫會繼承其父項的串流設定。如果父項已啟用串流，複製也會同樣啟用串流。複製上的串流內容將會反映建立複製時的父項內容 (包括相同的事件 ID)。

屬性圖匯出的欄位

params 中的 concurrency 欄位

(選用)。預設：4。

concurrency 參數會指定匯出程序應該使用的平行查詢數目：

```
"concurrency" : (for example, 24)
```

一個好的指導方針是將並行層級設定為所有執行個體上 vCPU 數目的兩倍，而您要從這些執行個體中匯出資料。例如，一個 r5.xlarge 執行個體具有 4 個 vCPU。如果您要從 3 個 r5.xlarge 執行個體組成的叢集匯出，則可以將並行層級設為 24 (= 3 x 2 x 4)。

如果您使用的是 Neptune-Export 服務，則並行層級會受到 JobSize 設定的限制。例如，小型工作支援並行層級 8。如果您嘗試使用 concurrency 參數，為小型工作指定並行層級 24，則有效層級保持在 8。

如果您從複製的叢集匯出，匯出程序會根據複製的執行個體大小和工作大小來計算適當的並行層次。

params 中的 edgeLabels 欄位

(選用)。

使用 edgeLabels 僅匯出這些具有所指定標籤的邊緣：

```
  "edgeLabels" : ["(a label)", "(another label"]
```

JSON 陣列中的每個標籤都須是單一的簡單標籤。

scope 參數的優先順序高於 edgeLabels 參數，因此如果 scope 值不包括邊緣，則 edgeLabels 參數沒有任何作用。

params 中的 filter 欄位

(選用)。

使用 filter 指定僅應該匯出具有特定標籤的節點和/或邊緣，並篩選針對每個節點或邊緣匯出的屬性。

filter 物件的一般結構 (無論是內嵌還是篩選器組態檔案) 如下所示：

```
  "filter" : {
    "nodes": [ (array of node label and properties objects) ],
    "edges": [ (array of edge definition an properties objects) ]
  }
```

- **nodes** – 包含節點和節點屬性的 JSON 陣列，其形式如下：

  ```
    "nodes : [
      {
        "label": "(node label)",
        "properties": [ "(a property name)", "(another property name)", ( ... ) ]
      }
    ]
  ```

  - label – 節點的一或多個屬性圖標籤。

    採用單一值，或者，如果節點具有多個標籤，則採用值陣列。

  - properties – 包含您要匯出之節點屬性的名稱陣列。
- **edges** – 包含邊緣定義的 JSON 陣列，其形式如下：

  ```
    "edges" : [
      {
  ```

```
        "label": "(edge label)",
        "properties": [ "(a property name)", "(another property name)", ( ... ) ]
      }
    ]
```

- `label` – 邊緣的屬性圖標籤。採用單一值。
- `properties` – 包含您要匯出之邊緣屬性的名稱陣列。

params 中的 **filterConfigFile** 欄位

(選用)。

使用 `filterConfigFile` 以 `filter` 參數採用的相同形式指定包含篩選器組態的 JSON 檔案：

```
 "filterConfigFile" : "s3://(your Amazon S3 bucket)/neptune-export/(the name of the
JSON file)"
```

如需 `filterConfigFile` 檔案的格式，請參閱 篩選條件。

params 中用於屬性圖資料的 **format** 欄位

(選用)。預設值：`csv` (逗號分隔值)

`format` 參數會指定所匯出屬性圖資料的輸出格式：

```
 "format" : (one of: csv, csvNoHeaders, json, neptuneStreamsJson)
```

- **csv** – 逗號分隔值 (CSV) 格式化的輸出，具有根據 Gremlin 載入資料格式進行格式化的資料行標頭。
- **csvNoHeaders** – 沒有資料列標頭的 CSV 格式化資料。
- **json** – JSON 格式化資料。
- **neptuneStreamsJson** – 使用 GREMLIN_JSON 變更序列化格式的 JSON 格式化資料。

params 中的 **gremlinFilter** 欄位

(選用)。

`gremlinFilter` 參數可讓您提供 Gremlin 程式碼片段，例如用來同時篩選節點和邊緣的 `has()` 步驟：

```
"gremlinFilter" : (a Gremlin snippet)
```

欄位名稱和字串值應以逸出雙引號括住。對於日期和時間，您可以使用 datetime 方法。

下列範例只會匯出這些其日期建立屬性值大於 2021-10-10 的節點和邊緣：

```
"gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
```

params 中的 gremlinNodeFilter 欄位

(選用)。

gremlinNodeFilter 參數可讓您提供 Gremlin 程式碼片段，例如用來篩選節點的 has() 步驟：

```
"gremlinNodeFilter" : (a Gremlin snippet)
```

欄位名稱和字串值應以逸出雙引號括住。對於日期和時間，您可以使用 datetime 方法。

下列範例只會匯出這些其 deleted 布林屬性值為 true 的節點：

```
"gremlinNodeFilter" : "has(\"deleted\", true)"
```

params 中的 gremlinEdgeFilter 欄位

(選用)。

gremlinEdgeFilter 參數可讓您提供 Gremlin 程式碼片段，例如用來篩選邊緣的 has() 步驟：

```
"gremlinEdgeFilter" : (a Gremlin snippet)
```

欄位名稱和字串值應以逸出雙引號括住。對於日期和時間，您可以使用 datetime 方法。

下列範例只會匯出這些其 strength 數值屬性值為 5 的邊緣：

```
"gremlinEdgeFilter" : "has(\"strength\", 5)"
```

params 中的 nodeLabels 欄位

(選用)。

使用 nodeLabels 僅匯出這些具有所指定標籤的節點：

```
  "nodeLabels" : ["(a label)", "(another label"]
```

JSON 陣列中的每個標籤都須是單一的簡單標籤。

scope 參數的優先順序高於 nodeLabels 參數，因此如果 scope 值不包括節點，則 nodeLabels 參數沒有任何作用。

params 中的 scope 欄位

(選用)。預設：all。

scope 參數會指定僅匯出節點、僅匯出邊緣，還是同時匯出節點和邊緣：

```
  "scope" : (one of: nodes, edges, or all)
```

- nodes – 僅匯出節點及其屬性。
- edges – 僅匯出邊緣及其屬性。
- all – 同時匯出節點和邊緣及其性質 (預設值)。

RDF 匯出的欄位

params 中用於 RDF 資料的 format 欄位

(選用)。預設：turtle

format 參數會指定所匯出 RDF 資料的輸出格式：

```
  "format" : (one of: turtle, nquads, ntriples, neptuneStreamsJson)
```

- **turtle** – Turtle 格式化輸出。
- **nquads** – 沒有資料列標頭的 N-Quads 格式化資料。
- **ntriples** – N-Triples 格式化資料。
- **neptuneStreamsJson** – 使用 SPARQL NQUADS 變更序列化格式的 JSON 格式化資料。

params 中的 rdfExportScope 欄位

(選用)。預設：graph。

rdfExportScope 參數會指定 RDF 匯出的範圍：

```
  "rdfExportScope" : (one of: graph, edges, or query)
```

- graph – 匯出所有 RDF 資料。
- edges – 僅匯出這些代表邊緣的三元組。
- query – 匯出使用 sparql 欄位提供之 SPARQL 查詢所擷取的資料。

params 中的 sparql 欄位

(選用)。

sparql 參數可讓您指定 SPARQL 查詢以擷取要匯出的資料：

```
  "sparql" : (a SPARQL query)
```

如果使用 sparql 欄位提供查詢，您也必須將 rdfExportScope 欄位設為 query。

params 中的 namedGraph 欄位

(選用)。

此namedGraph參數可讓您指定 IRI，將匯出限制為單一具名圖形：

```
  "namedGraph" : (Named graph IRI)
```

namedGraph參數只能與設定為的rdfExportScope欄位一起使用graph。

篩選匯出內容的範例

以下範例說明篩選所匯出資料的方式。

篩選屬性圖資料的匯出

使用 **scope** 僅匯出邊緣的範例

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "scope": "edges"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

使用 **nodeLabels** 和 **edgeLabels** 僅匯出具有特定標籤之節點和邊緣的範例

下列範例中的 nodeLabels 參數指定只應匯出具有 Person 標籤或 Post 標籤的節點。edgeLabels 參數指定僅應匯出具有 likes 標籤的邊緣：

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "nodeLabels": ["Person", "Post"],
    "edgeLabels": ["likes"]
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

使用 **filter** 僅匯出所指定節點、邊緣和屬性的範例

此範例中的 filter 物件會匯出 country 節點及其 type、code 和 desc 屬性，也會匯出 route 邊緣及其 dist 屬性。

```
{
  "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"
}
```

使用 **gremlinFilter** 的範例

此範例會使用 gremlinFilter 僅匯出在 2021-10-10 之後建立的節點和邊緣 (也就是說，具有其值大於 2021-10-10 的 created 屬性)：

```
{
  "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"
}
```

使用 **gremlinNodeFilter** 的範例

此範例會使用 gremlinNodeFilter 僅匯出已刪除的節點 (布林 deleted 屬性值為 true 的節點)：

```
{
  "command": "export-pg",
```

```
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinNodeFilter" : "has(\"deleted\", true)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

使用 **gremlinEdgeFilter** 的範例

此範例會使用 gremlinEdgeFilter 僅匯出其 strength 數值屬性值為 5 的邊緣：

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinEdgeFilter" : "has(\"strength\", 5)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

合併 **filter**、**gremlinNodeFilter**、**nodeLabelsedgeLabels** 和 **scope**

此範例中的 filter 物件會匯出：

- country 節點及其 type、code 及 desc 屬性
- airport 節點及其 code、icao 及 runways 屬性
- route 邊緣及其 dist 屬性

gremlinNodeFilter 參數會篩選節點，以便僅匯出其 code 屬性值以 A 開始的節點。

nodeLabels 和 edgeLabels 參數會進一步限制輸出，以便僅匯出 airport 節點和 route 邊緣。

最後，scope 參數會從匯出中消除邊緣，這樣只會在輸出中留下指定的 airport 節點。

```
{
  "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"
}
```

篩選 RDF 資料的匯出

使用 **rdfExportScope** 和 **sparql** 匯出特定邊緣

此範例會匯出其述詞為 <http://kelvinlawrence.net/air-routes/objectProperty/route> 且其物件不是常值的三元組：

```
{
  "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"
}
```

用**namedGraph**來匯出單一具名圖形

此範例會匯出屬於已命名圖形 < http://aws.amazon.com/neptune/vocab/v01/ DefaultNamedGraph > 的三元組：

```
{
  "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"
}
```

對 Neptune 匯出程序進行疑難排解

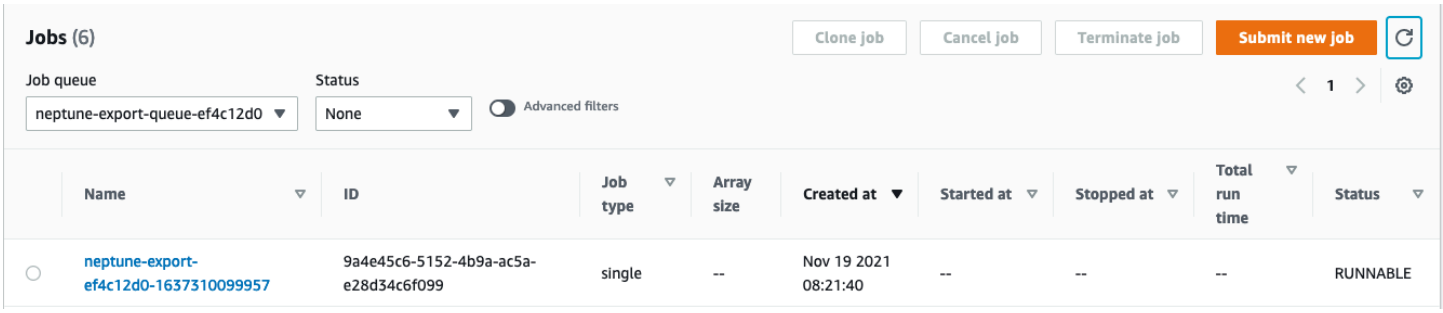
Amazon Neptune 匯出程序會使用 AWS Batch 佈建匯出 Neptune 資料所需的運算和儲存資源。匯出執行時，您可以使用 `logs` 欄位中的連結，存取匯出工作的 CloudWatch 日誌。

不過，執行匯出之 AWS Batch 工作的 CloudWatch 日誌只有在 AWS Batch 工作正在執行時才可用。如果 Neptune 匯出報告匯出處於待定狀態，則不會有日誌連結，讓您可透過其存取 CloudWatch 日誌。如果匯出工作保持 `pending` 狀態超過數分鐘，則佈建基礎 AWS Batch 資源可能發生問題。

當匯出工作離開待定狀態時，您可以檢查其狀態，如下所示：

檢查 AWS Batch 工作的狀態

1. 於 https://console.aws.amazon.com/batch/ 開啟 AWS Batch 主控台。
2. 選取 neptune-export 工作佇列。
3. 尋找其名稱與 Neptune 匯出在您開始匯出時所傳回之 `jobName` 相符的工作。

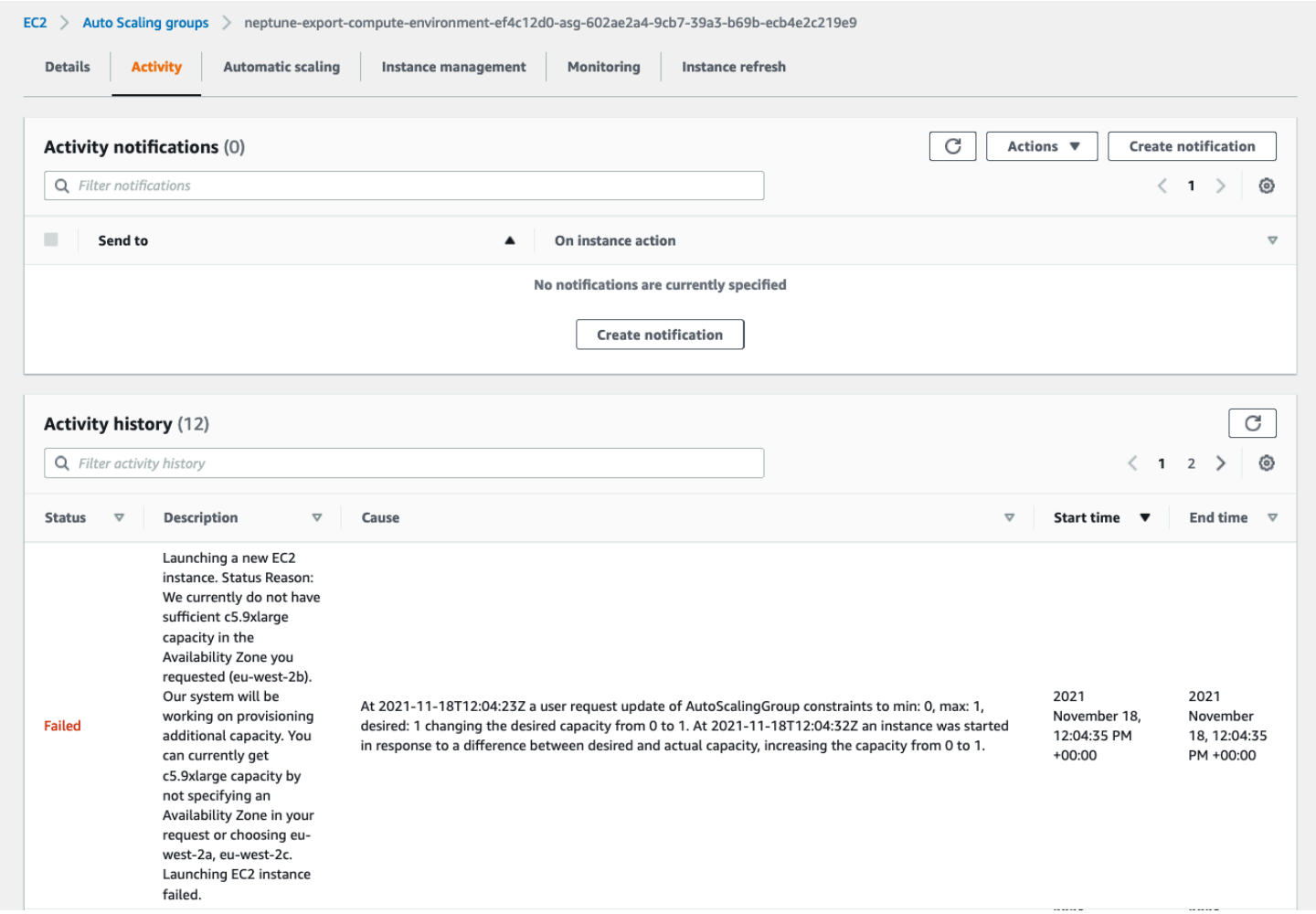


如果工作仍卡在 `RUNNABLE` 狀態，原因可能是網路或安全問題阻止容器執行個體加入基礎 Amazon Elastic Container Service (Amazon ECS) 叢集。請參閱本支援文章中有關驗證運算環境的網路和安全設定一節。

您可以檢查的另一件事就是找出自動擴展方面的問題：

檢查 AWS Batch 運算環境的 Amazon EC2 Auto Scaling 群組

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 為 neptune-export 運算環境選取 Auto Scaling 群組。
3. 開啟活動索引標籤，並檢查活動歷史記錄是否有未成功的事件。



Neptune 匯出常見錯誤

`org.eclipse.rdf4j.query.QueryEvaluationException: Tag mismatch!`

如果 `export-rdf` 工作經常由於 `Tag mismatch! QueryEvaluationException` 失敗，則對於 Neptune 匯出所使用的大型、長時間執行的查詢，Neptune 執行個體大小過小。

您可以縱向擴展到較大的 Neptune 執行個體，或將工作設定為從大型複製的叢集匯出，來避免發生此錯誤，如下所示：

```
'{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "cloneCluster": True,
    "cloneClusterInstanceType" : "r5.24xlarge"
```

```
  }
}'
```

管理 Amazon Neptune 資料庫

本節說明如何使用 AWS Management Console 和 AWS CLI 來管理和維護您的 Neptune 資料庫叢集。

Neptune 會在複寫拓撲中連線的資料庫伺服器叢集上運作。因此，管理 Neptune 通常會涉及將變更部署到多部伺服器，並確保所有 Neptune 複本都和主伺服器保持同步。

由於 Neptune 會隨著您的資料增加而透明地擴展基礎儲存體，因此管理 Neptune 只需要進行相對較少的磁碟儲存體管理。同樣地，因為 Neptune 會自動執行連續備份，因此 Neptune 叢集不需要為執行備份進行大量的規劃或停機時間。

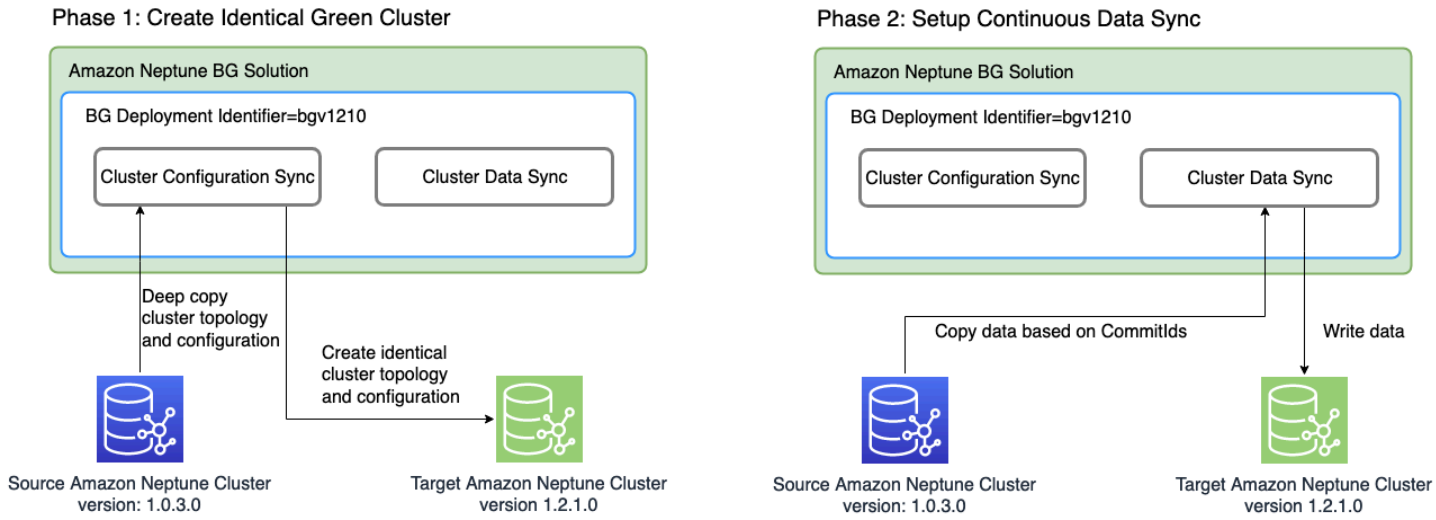
主題

使用 Neptune 藍/綠解決方案執行藍綠更新

Amazon Neptune 引擎升級可能需要應用程式停機，因為在安裝和驗證更新時無法使用資料庫。無論它們是手動還是自動啟動，都是如此。

Neptune 提供藍/綠部署解決方案，您可以使用 AWS CloudFormation 堆疊執行此解決方案，從而大幅減少此類停機時間。它會建立與您的藍色生產環境同步的綠色預備環境。然後，您可以更新該預備環境，以執行次要或主要引擎版本升級、圖形資料模型變更，或作業系統更新，並測試結果。最後，您可以快速將其切換為您的生產環境，停機時間很短。

Neptune 藍/綠解決方案經過兩個階段，如下圖所示：



第 1 階段會建立與生產叢集相同的綠色資料庫叢集

此解決方案會建立一個資料庫叢集，其具有唯一的藍/綠部署識別符，並具有與生產叢集相同的叢集拓撲。也就是說，它具有相同的資料庫執行個體數量和大小、相同的參數群組，以及與生產 (藍色) 資料庫叢集相同的組態，不同之處在於它已升級至您指定的目標引擎版本，但此版本必須高於您目前的 (藍色) 引擎版本。您可以指定目標的次要和主要引擎版本。如有必要，解決方案會執行任何所需的中繼升級，以達到指定的目標引擎版本。這個新叢集會成為綠色預備環境。

第 2 階段會設定連續資料同步

在綠色環境完全準備好之後，解決方案會使用 Neptune 串流，在來源 (藍色) 叢集與目標 (綠色) 叢集之間設定連續複寫。當它們之間的複寫差異達到零時，預備環境就可以進行測試。此時，您必須暫停寫入至藍色叢集，以避免任何進一步的複寫延遲。

您的目標引擎版本可能具有會影響應用程式的新功能或相依性。檢查引擎版本下的目標引擎版本頁面和介入引擎版本頁面，以查看自目前引擎版本以來發生了哪些變更。在將您的應用程式提升至生產環境之前，最好先在綠色叢集上執行整合測試或手動驗證該應用程式。

在測試並限定了綠色叢集中的變更之後，只要將應用程式中的資料庫端點從藍色叢集切換為綠色叢集即可。

轉換後，Neptune 藍/綠解決方案不會刪除舊的藍色生產環境。如果需要，您仍然可以存取它，以進行其他驗證和測試。標準帳單費用會確實套用至其執行個體，直到您刪除它們為止。藍色/綠色解決方案還會使用其他 AWS 服務，其費用按正常價格計費。清除一節中涵蓋了在完成解決方案後將其刪除的詳細資訊。

執行 Neptune 藍/綠堆疊的必要條件

啟動 Neptune 藍/綠堆疊之前：

- 務必在生產 (藍色) 叢集上啟用 Neptune 串流。
- 藍色叢集中的所有執行個體都必須處於可用狀態。您可以在 Neptune 主控台中，或使用 describe-db-instances API 來檢查執行個體狀態。
- 所有執行個體也必須與資料庫叢集參數群組同步。
- Neptune 藍/綠解決方案需要 VPC 中藍色叢集所在的 DynamoDB VPC 端點。請參閱使用 Amazon VPC 端點來存取 DynamoDB。
- 選擇在藍色生產資料庫叢集上的寫入工作負載盡可能輕時執行解決方案。例如，避免在大量載入發生時，或是在由於任何其他原因而可能有大量寫入操作時執行解決方案。

使用 AWS CloudFormation 範本執行 Neptune 藍/綠解決方案

您可以使用 AWS CloudFormation 部署 Neptune 藍/綠解決方案。CloudFormation 範本會在與您的藍色來源 Neptune 資料庫相同的 VPC 中建立 Amazon EC2 執行個體、在該處安裝解決方案，然後執行它。您可以在 CloudWatch 日誌中監控其進度，如監控進度中所述。

您可以使用這些連結來檢閱解決方案範本，或選取啟動堆疊按鈕，在 AWS CloudFormation 主控台中啟動它：

檢視	在設計工具中檢視	Launch Stack

在主控台中，從視窗右上角的下拉式清單中選擇要執行解決方案的 AWS 區域。

設定堆疊參數，如下所示：

- **DeploymentID** – 每個 Neptune 藍/綠部署獨有的識別符。

  其會用作綠色資料庫叢集識別符，以及用作字首以命名在部署期間建立的新資源。
- **NeptuneSourceClusterId** – 您要升級之藍色資料庫叢集的識別符。
- **NeptuneTargetClusterVersion:** – 您要將藍色資料庫叢集升級到的 Neptune 引擎版本。

  此版本必須高於目前藍色資料庫叢集的引擎版本。
- **DeploymentMode** – 指示這是新部署還是繼續先前部署的嘗試。當您使用與先前部署相同的 `DeploymentID` 時，請將 `DeploymentMode` 設定為 `resume`。

  有效值為：`new` (預設值) 和 `resume`。
- **GraphQueryType** – 資料庫的圖形資料類型。

  有效值為：`propertygraph` (預設值) 和 `rdf`。
- **SubnetId** – 來自藍色資料庫叢集所在之相同 VPC 的子網路 ID。(請參閱從同一 VPC 中的 Amazon EC2 執行個體連線至 Neptune 資料庫叢集)。

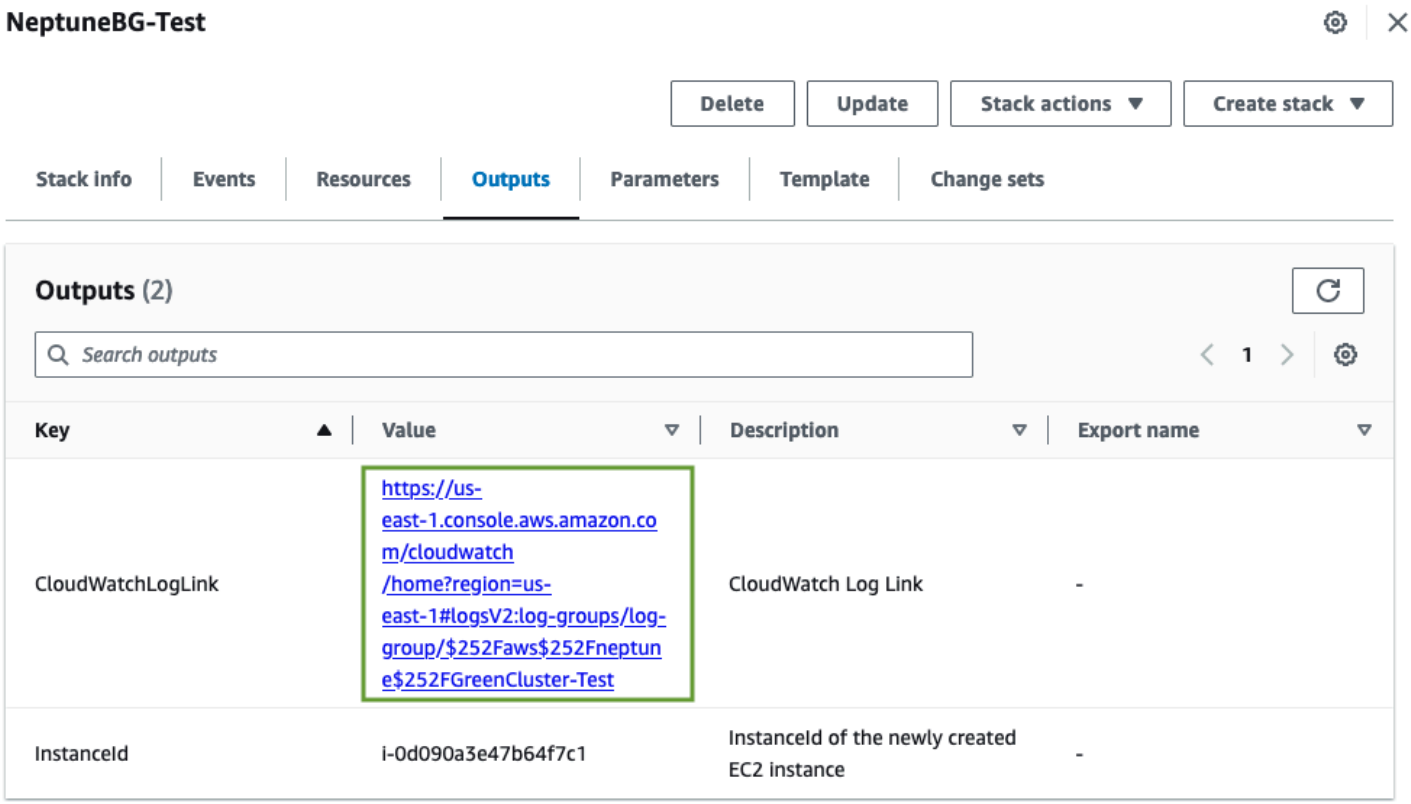
  如果您想要透過 EC2 Connect 使用 SSH 連線至執行個體，請提供公有子網路的 ID。
- **InstanceSecurityGroup** – Amazon EC2 執行個體的安全群組。

  安全群組必須有權存取藍色資料庫叢集，而且您必須能夠使用 SSH 連線至執行個體。請參閱 使用 VPC 主控台建立安全群組。

等到堆疊完成。一旦完成，解決方案就會啟動。然後，您可以使用 CloudWatch 日誌監控部署程序，如下一節所述。

監控 Neptune 藍/綠部署的進度

您可以前往 CloudWatch 主控台並查看 `/aws/neptune/`*`(Neptune Blue/Green deployment ID)`* CloudWatch 日誌群組中的日誌，以監控 Neptune 藍/綠解決方案的進度。您可以在解決方案的 AWS CloudFormation 堆疊輸出中找到 CloudWatch 日誌的連結：



如果您提供了公有子網路做為堆疊參數，也可以使用 SSH 連線至作為堆疊一部分而建立的 Amazon EC2 執行個體，並參考 `/var/log/cloud-init-output.log` 中的日誌。

日誌會顯示 Neptune 藍/綠解決方案所採取的動作，如此螢幕擷取畫面所示：

```
============================================================

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':
```

日誌訊息會顯示藍色叢集與綠色叢集之間的同步狀態：

```
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'}}
```

同步程序會檢查複寫延遲，方法是計算藍色叢集 `eventID` 上的最新串流與 DynamoDB 檢查點表格中存在的複寫檢查點之間的差異，這個表格是由 Neptune 對 Neptune 複寫堆疊所建立。使用這些訊息，您可以監控目前的複寫差異。

從生產藍色叢集切換至更新的綠色叢集

在將綠色叢集提升至生產之前，請確定藍色叢集與綠色叢集之間的遞交差異為零，然後停用藍色叢集的所有寫入流量。將資料庫端點切換至綠色叢集時，繼續寫入至藍色叢集，可能會因為將部分資料寫入這兩個叢集而導致資料損毀。您可能還不需要停用讀取流量。

如果您已在來源 (藍色) 叢集上啟用 IAM 身分驗證，請務必更新應用程式中使用的任何 IAM 政策，以指向綠色叢集 (如需這類政策的範例，請參閱此不受限制的存取政策)。

在停用寫入流量之後，請等待複寫完成，然後在綠色叢集上啟用寫入流量 (但不在藍色叢集上)。也會將讀取流量從藍色叢集切換到綠色叢集。

在 Neptune 藍/綠解決方案完成之後清除

將預備 (綠色) 叢集提升至生產之後，請清除 Neptune 藍/綠解決方案所建立的資源：

- 刪除為了執行解決方案而建立的 Amazon EC2 執行個體。
- 對於保持綠色叢集與藍色叢集同步的 Neptune 串流型複寫，刪除 AWS CloudFormation 範本。主要的一個具有您之前提供的堆疊名稱，另一個由後面跟著「-replication」的部署 ID 組成：亦即，*(DeploymentID)*-replication。

刪除 AWS CloudFormation 範本並不會刪除叢集本身。一旦確認了綠色叢集如預期般運作，您就可以選擇性地建立快照，然後再手動刪除藍色叢集。

Neptune 藍/綠解決方案最佳實務

- 在將綠色叢集切換至生產之前，值得徹底驗證其是否運作正常。檢查資料的一致性和資料庫的組態。有些新的引擎版本可能也需要用戶端升級。在升級之前檢查引擎版本備註。在生產中開始藍/綠升級之前，值得在開發、測試和生產前環境中測試這一切。
- 最好在維護時段期間執行從藍色伺服器到綠色伺服器的切換。
- 為了確保在升級和同步之後一切正常運作，值得在刪除原始叢集之前將其保留一段時間。如果引發不可預見的問題，它可能會很有用。
- 執行 Neptune 藍/綠解決方案時，請避免大量寫入操作 (例如大量載入)，因為這些操作可能會造成複寫延遲，帶來顯著的停機時間。理想情況下，關閉對藍色叢集的寫入與針對綠色叢集開啟它們之間的時間只是幾分鐘。

對 Neptune 藍/綠解決方案進行疑難排解

Neptune 藍/綠解決方引發的錯誤

- **`Cluster with id = (blue_green_deployment_id) already exists`** – 有一個現有的叢集，其識別符為 *`blue_green_deployment_id`*。

  如果叢集是在先前的 Neptune 藍/綠執行中建立，請提供新的部署 ID 或將部署模式設定為 `resume`。
- **`Streams should be enabled on the source Cluster for Blue Green Deployment`** – 在藍色 (來源) 叢集上啟用 Neptune 串流。
- **`No Bulkload should be in progress on source cluster: (cluster_id)`** – 如果 Neptune 藍/綠解決方案識別了大量載入正在進行，則其會終止。

  這是為了確保同步程序能夠跟上在進行的寫入。在啟動 Neptune 藍/綠解決方案之前，請避免或取消任何正在進行的大量載入。

- **`Blue Green deployment requires instances to be in sync with db cluster parameter group`** – 叢集參數群組的任何變更都應在整個資料庫叢集中保持同步。請參閱 Amazon Neptune 參數群組。
- **`Invalid target engine version for Blue Green Deployment`** – 目標引擎版本必須在 Amazon Neptune 的引擎版本 中列示為作用中，且必須高於來源 (藍色) 叢集的目前引擎版本。

建立具有 Neptune 許可的 IAM 使用者

若要存取 Neptune 主控台以建立和管理 Neptune 資料庫叢集，您需要建立具有所有必要許可的 IAM 使用者。

第一步是建立 Neptune 的服務連結角色政策：

建立 Amazon Neptune 的服務連結角色政策

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 在政策頁面上，選取建立政策。
4. 在建立政策頁面上，選取 JSON 索引標籤，然後複製下列服務連結角色政策：

```
{
  "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. 選取下一步：標籤，然後在新增標籤頁面上選取下一步：檢閱。
6. 在檢閱政策頁面上，將新原則命名為「NeptuneServiceLinked」。

如需服務連結角色的詳細資訊，請參閱使用 Neptune 的服務連結角色。

建立具有所有必要許可的新 IAM 使用者

接下來，建立新的 IAM 使用者，其中附加了將授予您所需許可的適當受管政策，以及您已建立的服務連結角色政策 (此處命名為 NeptuneServiceLinked)：

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側導覽窗格中，選擇使用者，然後在使用者頁面上，選擇新增使用者。
3. 在新增使用者頁面上，輸入新 IAM 使用者的名稱、選擇存取金鑰 – 程式化存取做為 AWS 憑證類型，然後選擇下一步：許可。
4. 在設定許可頁面的篩選器政策方塊中，輸入「Neptune」。現在，從列出的政策中選取以下政策：

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked (假設這是您之前建立的服務連結角色政策的命名)。
5. 接下來，在篩選器政策方塊中輸入「VPC」取代「Neptune」。從列出的政策中選取 AmazonVPCFullAccess。
6. 選取下一步：標籤，然後在新增標籤頁面中，選取下一步：檢閱。
7. 在檢閱頁面中，檢查下列所有政策現在是否已附加至新的使用者：

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked
   - AmazonVPCFullAccess

   選取建立使用者。
8. 最後，下載並儲存新使用者的存取金鑰 ID 和私密存取金鑰。

若要與其他服務 (例如 Amazon Simple Storage Service (Amazon S3)) 互通，您需要新增更多許可和信任關係。

Amazon Neptune 參數群組

您可以使用參數群組中的參數，在 Amazon Neptune 中管理資料庫組態。參數群組扮演引擎組態值的容器，以套用至一或多個資料庫執行個體。

參數群組有兩種類型，即資料庫叢集參數群組和資料庫參數群組。

- 「資料庫參數群組」會在執行個體層級套用，並且一般都會和 Neptune 圖形引擎的設定建立關聯，例如 `neptune_query_timeout` 參數。
- 「資料庫叢集參數群組」會套用到叢集中的每個執行個體，並且一般擁有較廣的設定。每個 Neptune 叢集都會與一個資料庫叢集參數群組相關聯。該叢集內的每個資料庫執行個體會繼承包含在資料庫叢集參數群組中的引擎組態值。

您在資料庫叢集參數群組中修改的任何組態值，都會覆寫資料庫參數群組中的預設值。若您在資料庫參數群組中編輯對應的值，則那些值會覆寫資料庫叢集參數群組中的設定。

如果您建立資料庫執行個體而未指定自訂資料庫參數群組，將會使用預設的資料庫參數群組。您無法修改預設資料庫參數群組的參數設定。相反，若要變更預設參數設定，您必須建立新的資料庫參數群組。並非所有資料庫引擎參數都可以在您建立的資料庫參數群組中變更。

參數群組是在與不同 Neptune 引擎版本相容的系列中建立的。預設參數群組系列為 `neptune1`，與 `1.2.0.0` 之前的所有引擎版本相容。從 版本：1.2.0.0 (2022 年 7 月 21 日) 開始，必須改用 `neptune1.2` 參數群組系列。這表示當您升級至 `1.2.0.0` 或更高版本時，必須先在 `neptune1.2` 系列中重新建立您的所有自訂參數群組，以便在升級時可以附加它們。

有些 Neptune 參數是靜態的，而其他參數是動態的。差異如下所示：

靜態參數

- 靜態參數是指僅在資料庫執行個體重新啟動後才會生效的參數。換言之，當您變更靜態參數並儲存執行個體資料庫參數群組時，您必須手動重新啟動資料庫執行個體，參數變更才會生效。目前，所有 Neptune 執行個體層級參數 (位於資料庫參數群組中，而非資料庫叢集參數群組中) 都是靜態的。
- 當您變更叢集層級靜態參數並儲存資料庫叢集參數群組時，參數變更會在您手動重新啟動叢集中的每個資料庫執行個體之後生效。

動態參數

- 動態參數是指在參數群組中更新參數之後幾乎立即生效的參數。換言之，在更新動態參數之後，不需要重新啟動資料庫執行個體，參數變更就會生效。

- 對於要跨所有資料庫執行個體套用的動態叢集參數變更，預期會有些微的延遲。
- 更新的動態參數值不會套用至目前執行中的請求，而只會套用至變更發生後提交的請求。
- 變更動態叢集層級參數時，根據預設，參數變更會立即套用至資料庫叢集，無需任何重新啟動。若要將參數變更延遲到叢集中的資料庫執行個體重新啟動之後，您可以使用 AWS CLI，將 `ApplyMethod` 設定為 `pending-reboot`，進行參數變更。

目前，除了下列新叢集參數以外，所有參數都是靜態的：

- `neptune_enable_slow_query_log` (叢集層級)
- `neptune_slow_query_log_threshold` (叢集層級)

以下是使用資料庫參數群組中的參數時，您應該知道的一些重要提示：

- 未正確設定資料庫參數群組中的參數，可能產生各種意外影響，包括降低效能和系統不穩定。修改資料庫參數時請務必謹慎，在修改資料庫參數群組之前，請備份您的資料。在建參數群組設定變更套用到生產資料庫執行個體之前，請先在測試資料庫執行個體上嘗試這些變更。
- 變更與資料庫執行個體關聯的資料庫參數群組時，您必須手動重新啟動執行個體，資料庫執行個體才會使用新的資料庫參數群組。

> **Note**
>
> 在 版本：1.2.0.0 (2022 年 7 月 21 日) 之前，每當主要 (寫入器) 執行個體重新啟動時，資料庫叢集中的所有僅供讀取複本執行個體都會自動重新啟動
> 從 版本：1.2.0.0 (2022 年 7 月 21 日) 開始，重新啟動主執行個體並不會導致任何複本執行個體重新啟動。這表示如果您要變更叢集層級參數，必須個別重新啟動每個執行個體，才能接收參數變更。

編輯資料庫叢集參數群組或資料庫參數群組

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Parameter groups (參數群組)。
3. 選擇您要編輯的資料庫參數群組的 Name (名稱) 連結。

(選用) 選擇 Create parameter group (建立參數群組) 來建立新的叢集參數群組，然後建立新的群組。然後，選擇新參數群組的 Name (名稱)。

> **Important**
>
> 若因為無法修改預設資料庫叢集參數群組，因此您只有預設的資料庫叢集參數群組，則此步驟是「必要」的。

4. 搜尋參數，然後按一下「名稱」欄旁的「值」欄位。
5. 輸入允許的值，然後選擇值欄位旁邊的核取方塊。
6. 選擇儲存變更。
7. 如果您要變更資料庫叢集參數，請重新啟動 Neptune 叢集中的每個資料庫執行個體，或如果要變更資料庫執行個體參數，請重新啟動一或多個特定執行個體。

建立資料庫叢集參數群組或資料庫參數群組

您可以輕鬆地使用 Neptune 主控台建立新的參數群組。

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在左側導覽窗格中，選擇 Parameter groups (參數群組)。
3. 選擇 Create DB parameter group (建立資料庫參數群組)。

   Create DB parameter group (建立資料庫參數群組) 頁面隨即出現。
4. 在參數群組系列清單中，選擇 neptune1，或者，如果您的目標是引擎 1.2.0.0 版或更高版本，請選擇 neptune1.2。
5. 在 Type (類型) 清單中選擇 DB Parameter Group (資料庫參數群組) 或 DB Cluster Parameter Group (資料庫叢集參數群組)。
6. 在 Group name (群組名稱) 方塊中輸入新資料庫參數群組的名稱。
7. 在 Description (描述) 方塊中輸入新資料庫參數群組的描述。
8. 選擇建立。

您也可以使用 AWS CLI 建立新的參數群組：

```
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)
```

Amazon Neptune 參數

您可以使用參數群組中的參數，在 Amazon Neptune 中管理資料庫組態。下列參數可用於設定 Neptune 資料庫：

叢集層級參數

- 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

執行個體層級參數

- neptune_dfe_query_engine
- neptune_query_timeout
- neptune_result_cache

已棄用的參數

- neptune_enforce_ssl

neptune_enable_audit_log (叢集層級參數)

此參數可切換 Neptune 的稽核記錄。

允許的值為 0 (已停用) 和 1 (已啟用)。預設值為 0。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

您可以將稽核日誌發佈到 Amazon CloudWatch，如 使用 CLI 將 Neptune 稽核記錄發佈至 CloudWatch 記錄 中所述。

neptune_enable_slow_query_log (叢集層級參數)

使用此參數可啟用或停用 Neptune 的慢速查詢記錄功能。

這是一個動態參數，表示變更其值不需要也不會導致資料庫叢集重新啟動。

允許的值為：

- **info** – 啟用慢速查詢記錄，並記錄可能造成效能降低的所選屬性。
- **debug** – 啟用慢速查詢記錄，並記錄查詢執行的所有可用屬性。
- **disable** – 停用慢速查詢記錄。

預設值為 disable。

您可以將慢速查詢日誌發佈到 Amazon CloudWatch，如 使用 CLI 將 Neptune 慢速查詢記錄檔發佈至記錄檔 CloudWatch 中所述。

neptune_slow_query_log_threshold (叢集層級參數)

此參數指定執行時間閾值 (以毫秒為單位)，而執行時間超過此閾值的查詢會被視為緩慢查詢。如果啟用了慢速查詢記錄，則執行時間超過此閾值的查詢將與其某些屬性一起記錄。

預設值為 5000 毫秒 (5 秒)。

這是一個動態參數，表示變更其值不需要也不會導致資料庫叢集重新啟動。

neptune_lab_mode (叢集層級參數)

設定後，此參數會啟用 Neptune 的特定實驗性功能。如需目前可用的實驗性功能，請參閱Neptune 實驗室模式。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

若要啟用或停用實驗性功能，請在此參數中包含 *(feature name)*=enabled 或 *(feature name)*=disabled。您可以啟用或停用多個功能，方法為使用逗號區隔它們，如下所示：

(feature #1 name)=enabled,*(feature #2 name)*=enabled

實驗室模式功能通常預設為停用。DFEQueryEngine 功能除外，因為從 Neptune 引擎 1.0.5.0 版 開始，預設會啟用此功能，與查詢提示 (DFEQueryEngine=viaQueryHint) 搭配使用。從 Neptune 引擎 1.1.1.0 版開始，DFE 引擎不再處於實驗室模式，而且現在可以使用執行個體資料庫參數群組中的 neptune_dfe_query_engine 執行個體參數來控制此引擎。

neptune_query_timeout (叢集層級參數)

指定圖形查詢的特定逾時持續時間 (以毫秒為單位)。

允許的值範圍從 10 到 2,147,483,647 (2^{31} -1)。預設值為 120,000 (2 分鐘)。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

> **Note**
>
> 如果您將查詢逾時值設得太高，特別是在無伺服器執行個體上，可能會產生非預期的成本。若沒有合理的逾時設定，您可能會不小心發出其執行時間可能會比預期長得多的查詢，進而產生您從未預期的成本。這在無伺服器執行個體上尤是如此，因為該執行個體在執行查詢時可能會縱向擴展為大型且昂貴的執行個體類型。
> 您可以使用查詢逾時值，避免此類非預期的費用，而這些逾時值符合大部分的查詢，且只會導致執行時間異常長的查詢逾時。

neptune_streams (叢集層級參數)

啟用或停用 Neptune 串流。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

允許值為 0 (停用，此為預設值) 和 1 (啟用)。

neptune_streams_expiry_days (叢集層級參數)

指定伺服器在經過多少天後就會刪除串流記錄。

允許的值從 1 到 90 (含)。預設值為 7。

此參數是在引擎 1.2.0.0 版中引進的。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

neptune_lookup_cache (叢集層級參數)

在 R5d 執行個體上，停用或重新啟用 Neptune 查閱快取。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

允許的值為 enabled 和 disabled。預設值為 disabled，但無論何時在資料庫叢集中建立 R5d 執行個體，都會自動將 neptune_lookup_cache 參數設定為 enabled，並在該執行個體上建立查閱快取。

neptune_autoscaling_config (叢集層級參數)

為 Neptune 自動擴展建立和管理的僅供讀取複本執行個體設定組態參數。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

使用您設定為 neptune_autoscaling_config 參數值的 JSON 字串，您可以指定：

- Neptune 自動擴展用於其建立的所有新僅供讀取複本執行個體的執行個體類型。
- 指派給這些僅供讀取複本的維護時段。
- 要與所有新的僅供讀取複本相關聯的標籤。

JSON 字串具有如下結構：

```
"{
  \"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\"
}"
```

請注意，字串中的引號必須全都使用反斜線字元 (\) 來逸出。

三個組態設定中若有任何一個未在 neptune_autoscaling_config 參數中指定，其會從資料庫叢集的主要寫入器執行個體的組態中複製。

neptune_ml_iam_role (叢集層級參數)

指定 Neptune ML 中使用的 IAM 角色 ARN。此值可以是任何有效的 IAM 角色 ARN。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

您可以在圖形上指定機器學習的預設 IAM 角色 ARN。

neptune_ml_endpoint (叢集層級參數)

指定用於 Neptune ML 的端點。此值可以是任何有效的 SageMaker 端點名稱。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

您可以在圖形上指定機器學習的預設 SageMaker 端點。

neptune_dfe_query_engine (執行個體層級參數)

從 Neptune 引擎 1.1.1.0 版開始，此資料庫執行個體參數用來控制 DFE 查詢引擎的使用方式。允許的值如下：

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

- **enabled** – 盡可能使用 DFE 引擎，但在 `useDFE` 查詢提示存在且設定為 `false` 的情況下除外。
- **viaQueryHint** (預設值) – 使 DFE 引擎僅用於明確包含 `useDFE` 查詢提示 (設定為 `true`) 的查詢。

如果尚未明確設定此參數，則會在啟動執行個體時使用預設值 `viaQueryHint`。

> **Note**
>
> 無論如何設定此參數，所有 OpenCypher 查詢都是由 DFE 引擎執行。

在 1.1.1.0 版之前，這是實驗室模式參數，而不是資料庫執行個體參數。

neptune_query_timeout (執行個體層級參數)

此資料庫執行個體參數會針對一個執行個體指定圖形查詢的逾時持續時間 (以毫秒為單位)。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

允許的值範圍從 10 到 2,147,483,647 (2^{31} -1)。預設值為 120,000 (2 分鐘)。

> **Note**
>
> 如果您將查詢逾時值設得太高，特別是在無伺服器執行個體上，可能會產生非預期的成本。若沒有合理的逾時設定，您可能會不小心發出其執行時間可能會比預期長得多的查詢，進而產生您從未預期的成本。這在無伺服器執行個體上尤是如此，因為該執行個體在執行查詢時可能會縱向擴展為大型且昂貴的執行個體類型。
> 您可以使用查詢逾時值，避免此類非預期的費用，而這些逾時值符合大部分的查詢，且只會導致執行時間異常長的查詢逾時。

neptune_result_cache (執行個體層級參數)

neptune_result_cache – 此資料庫執行個體參數會啟用或停用 快取查詢結果。

此參數是靜態的，表示對其所做的變更不會在任何執行個體上生效，直到重新啟動了該執行個體。

允許值為 0 (停用，此為預設值) 和 1 (啟用)。

neptune_enforce_ssl (DEPRECATED 叢集層級參數)

(已棄用) 曾經有允許 HTTP 連線至 Neptune 的區域，而且此參數用來在其設定為 1 時強制所有連線都使用 HTTPS。不過，這個參數不再相關，因為 Neptune 現在只在所有區域中接受 HTTPS 連線。

使用 AWS Management Console 啟動 Neptune 資料庫叢集

啟動新的 Neptune DB 叢集的最簡單方法是，使用可為您建立所有必要資源的 AWS CloudFormation 範本，如 建立資料庫叢集 中所述。

如果您願意，也可以使用 Neptune 主控台，手動啟動新的資料庫叢集，如這裡所述。

在可以存取 Neptune 主控台以建立 Neptune 叢集之前，請建立 IAM 使用者，其具有執行此操作的必要許可，如 建立具有 Neptune 許可的 IAM 使用者 所述。

然後，以該 IAM 使用者身分登入 AWS Management Console，並遵循下列步驟來建立新的資料庫叢集：

使用主控台啟動 Neptune 資料庫叢集

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 導覽至資料庫頁面，然後選擇建立資料庫，這會開啟建立資料庫頁面。
3. 在引擎選項下，引擎類型為 neptune，而且您可以選擇特定的引擎版本或接受預設值。
4. 在設定下，輸入新資料庫叢集的名稱，或接受此處提供的預設名稱。此名稱用於執行個體的端點地址，並且必須滿足以下限制條件：

   - 必須包含 1 到 63 個英數字元或連字號。
   - 第一個字元必須是字母。
   - 不能以一個連字號結尾或是連續包含兩個連字號。
   - 其對特定 AWS 區域內 AWS 帳戶的所有資料庫執行個體均須是唯一的。
5. 在範本下，選擇生產或開發和測試。
6. 在資料庫執行個體大小下，選擇執行個體大小。這會決定新資料庫叢集之主要寫入執行個體的處理和記憶體容量。

   如果您選取了生產範本，則只能從列出的可用記憶體最佳化類別中進行選擇，但是如果您選取了開發和測試，則也可以從更經濟的爆量類別中進行選擇 (如需爆量類別的討論，請參閱 T3 高載執行個體)。

   > **Note**
   >
   > 從 Neptune 引擎 1.1.0.0 版開始，Neptune 不再支援 R4 執行個體類型。

7. 在可用性與持久性下，您可以選擇是否啟用多可用區部署。生產範本預設會啟用多可用區部署，而開發和測試範本則不會啟用。如果啟用了多可用區部署，Neptune 會尋找您在不同可用區域 (AZ) 中建立的僅供讀取複本執行個體，以提高可用性。
8. 在連線下，從可用選項之間選取將託管新資料庫叢集的虛擬私有雲端 (VPC)。如果您想要 Neptune 為您建立 VPC，則可以在這裡選擇建立新的 VPC。您必須在這個相同的 VPC 中建立 Amazon EC2 執行個體，才能存取 Neptune 執行個體 (如需詳細資訊，請參閱 每個 Amazon Neptune 資料庫叢集都位於 Amazon VPC 中)。請注意，在建立了資料庫叢集之後，您無法變更 VPC。

   如有需要，您可以在其他連線能力組態下進一步設定叢集的連線能力：

   a. 在子網路群組下，您可以選擇要用於新資料庫叢集的 Neptune 資料庫子網路群組。如果您的 VPC 尚未有任何子網路群組，Neptune 會為您建立資料庫子網路群組 (請參閱 每個 Amazon Neptune 資料庫叢集都位於 Amazon VPC 中)。
   b. 在 VPC 安全群組下，選擇一個或多個現有的 VPC 安全群組，以保護對新資料庫叢集的網路存取，或者，如果想要 Neptune 為您建立安全群組，請選擇建立新的，然後為新的 VPC 安全群組提供名稱 (請參閱 使用 VPC 主控台建立安全群組)。
   c. 在資料庫連接埠 下，輸入資料庫將用於應用程式連線的 TCP/IP 連接埠。Neptune 會使用連接埠號碼 8182 做為預設值。
9. 如果您想要 Neptune 在 Neptune 工作台中為您建立 Jupyter 筆記本，請在筆記本組態下選擇建立筆記本 (請參閱 使用 Neptune 圖形筆記本快速開始 和 使用 Neptune 工作台託管 Neptune 筆記本)。然後，您可以選擇應該如何設定新的筆記本：

   a. 在筆記本執行個體類型下，從筆記本可用的執行個體類別之間進行選擇。
   b. 在筆記本名稱下，輸入筆記本的名稱。
   c. 如果想要的話，您也可以在描述 - 選用下輸入筆記本的描述。
   d. 在 IAM 角色名稱下，選擇讓 Neptune 為筆記本建立 IAM 角色，然後輸入新角色的名稱，或選擇從可用角色之間選取現有的 IAM 角色。
   e. 最後，選擇您的筆記本是直接連線至網際網路，還是透過 Amazon SageMaker 或透過 VPC 搭配 NAT 閘道連線至網際網路。如需詳細資訊，請參閱將筆記本執行個體連線至 VPC 中的資源。
10. 在標籤下，您最多可以將 50 個標籤與新資料庫叢集建立關聯。
11. 在其他組態下，有更多您可以為新資料庫叢集進行的設定 (在許多情況下，您可以略過它們並立即接受預設值)：

選項	您可以做什麼
DB instance identifier (資料庫執行個體識別符) :	您可以為叢集的寫入器執行個體提供名稱。如果未提供，則會使用以叢集名稱為基礎的預設識別符。如果提供，則指定所有資料庫執行個體中的唯一名稱，該執行個體的擁有者是您在目前區域中的 AWS 帳戶。資料庫執行個體識別符會區分大小寫，但全部以小寫形式儲存。
DB cluster parameter group (資料庫叢集參數群組)	選取資料庫叢集參數群組，以定義叢集中所有資料庫執行個體的預設組態。除非您另行選擇，否則 Neptune 會使用預設資料庫叢集參數群組。如需參數群組的詳細資訊，請參閱Amazon Neptune 參數群組。
DB parameter group (資料庫參數群組)	選取資料庫參數群組，以定義叢集中主要資料庫執行個體的組態。除非您另行選擇，否則 Neptune 會使用預設參數群組。如需參數群組的詳細資訊，請參閱參數群組。
IAM DB authentication (IAM 資料庫身分驗證)	如果您核取啟用 IAM 資料庫身分驗證，則會使用 AWS Identity and Access Management (IAM) 驗證對資料庫的所有存取。 **Important** 這需要您使用 AWS Signature 第 4 版簽署來簽署所有請求。如需更多詳細資訊，請參閱 Amazon Neptune AWS Identity and Access Management （IAM）概述。
容錯移轉優先順序	選擇 No preference 或容錯移轉的優先順序方案。如果您選擇一個方案，其中出現爭用，則會選擇與主要執行個體相同大小的複本。

選項	您可以做什麼
Backup retention period (備份保留期間)	選擇從 1 到 35 天的時間長度，Neptune 應會在此時間長度內保留此資料庫執行個體的自動備份。您只能將時間點還原 (PITR) 執行至備份保留期內的時間。
Copy tags to snapshots (將標籤複製到快照)	(預設為啟用) 此選項會將與資料庫叢集相關聯的所有標籤複製到其任何快照。
Enable encryption (啟用加密)	(預設為啟用) 此選項會使資料庫叢集中的資料進行靜態加密。 如果啟用，請選擇主金鑰，其用來保護用來加密此資料庫磁碟區的金鑰。您可以選取預設 `aws/rds` 金鑰，或從您帳戶中的主金鑰進行選擇，或從不同的帳戶輸入金鑰的 ARN。您可以在 IAM 主控台的加密金鑰索引標籤上建立新的主加密金鑰。如需更多詳細資訊，請參閱 靜態加密 Neptune 資源。
稽核日誌	如果您想要將資料庫叢集中的稽核日誌發佈到 CloudWatch Logs，請核取此選項。
Enable auto minor version upgrade (啟用自動次要版本升級)	(預設為啟用) 此選項會導致您的資料庫叢集在發行後自動升級至新的次要引擎版本。自動升級會在資料庫的維護期間進行。請參閱使用 `AutoMinorVersionUpgrade` 。
Maintenance window (維護時段)	您可以選取特定期間，您想要在此期間對資料庫叢集進行待定修改，例如對資料庫執行個體類別進行變更或自動修補引擎。任何此類維護操作都會在選取的期間內開始並完成。如果您未選取期間，Neptune 會任意指定維護期間。
啟用刪除保護	(預設為啟用) 刪除保護會防止您的資料庫叢集遭到刪除。您必須明確停用它，才能刪除資料庫叢集。

12. 選擇建立資料庫以啟動新的 Neptune 資料庫叢集及其主要執行個體。

    在 Amazon Neptune 主控台上，新的資料庫叢集會出現在資料庫清單中。資料庫叢集的狀態為 Creating (建立中)，直到建立完成且可供使用為止。狀態變更為 Available (可用) 時，您便能連接到資料庫叢集的主執行個體。根據資料庫執行個體類別和分配的存放區，新執行個體可能要幾分鐘後才能使用。

    若要檢視新建立的叢集，請在 Neptune 主控台中選擇資料庫檢視。

    > **Note**
    >
    > 如果您使用 AWS Management Console 刪除資料庫叢集內的所有 Neptune 資料庫執行個體，則主控台會自動刪除資料庫叢集本身。如果您使用的是 AWS CLI 或 SDK，則在刪除最後一個執行個體之後，您必須手動刪除資料庫叢集。

    請記下叢集端點的值。您需要此值來連線到您的 Neptune 資料庫叢集。

停止和啟動 Amazon Neptune 資料庫叢集

停止和啟動 Amazon Neptune 叢集可協助您管理開發和測試環境的成本。您可以暫時停用叢集中的所有資料庫執行個體，而非每次使用叢集時，設定和卸除所有資料庫執行個體。

主題

停止和啟動 Neptune 資料庫叢集的概觀

在不需要 Neptune 叢集的期間，您可以立即停止該叢集中的所有執行個體。一旦您需要叢集，即可隨時重新啟動它。啟動和停用可簡化用於下列操作之叢集的設定和卸除程序：開發、測試或不需要連續可用性的類似活動。無論叢集中有多少執行個體，您都可以透過單一動作在AWS Management Console中完成此操作。

資料庫叢集停用時，只需支付您指定的保留時段內叢集儲存、手動快照和自動備份儲存的費用。您無須支付任何資料庫執行個體小時數的費用。

Neptune 會在七天後自動啟動您的資料庫叢集，以確保它不會落後於任何必要的維護更新。

若要將輕度載入之 Neptune 叢集的費用降至最低，您可以停止該叢集，而非刪除其所有僅供讀取複本。對於具有一個或兩個以上執行個體的叢集，經常刪除和重建資料庫執行個體實際上僅會使用 AWS CLI 或 Neptune API，刪除也可能很難以正確的順序執行。例如，您必須在刪除主要執行個體之前刪除所有僅供讀取複本，以避免啟動容錯移轉機制。

如果您需要保持資料庫叢集執行中但想要減少容量，請勿使用啟動和停用。如果您的叢集太昂貴或不是非常忙碌，您可刪除一個或多個資料庫執行個體，或將您的所有資料庫執行個體變更為使用小型執行個體類別，但無法停止個別資料庫執行個體。

停止 Neptune 資料庫叢集

如果您會有一段時間不使用它，您可以停止執行中的 Neptune 資料庫叢集，然後在需要時重新啟動它。叢集停用時，需支付您指定的保留時段內叢集儲存、手動快照和自動備份儲存的費用，但無須支付資料庫執行個體小時數。

停止作業會先停止所有叢集的僅供讀取複本執行個體，然後再停止主要執行個體，以避免啟動容錯移轉機制。

使用 AWS Management Console 停止資料庫叢集

使用 AWS Management Console 停止 Neptune 叢集

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)，然後選擇一個叢集。您可以從這個頁面執行停用操作，或導覽至欲停用資料庫叢集的詳細資訊頁面。
3. 針對 Actions (動作)，選擇 Stop (停止)。

使用 AWS CLI 停止資料庫叢集

若要使用 AWS CLI 停止資料庫執行個體，請呼叫 stop-db-cluster 命令，並使用 `--db-cluster-identifier` 參數來識別要停止的資料庫叢集。

Example

```
aws neptune stop-db-cluster --db-cluster-identifier mydbcluster
```

使用 Neptune 管理 API 停止資料庫叢集

若要使用 Neptune 管理 API 來停止資料庫執行個體，請呼叫 StopDBCluster API 並使用 `DBClusterIdentifier` 參數，來識別您要停止的資料庫叢集。

停止資料庫叢集時可能發生的情況

- 您可以從快照還原它 (請參閱從資料庫叢集快照還原)。
- 您無法修改資料庫叢集或其任何資料庫執行個體的組態。
- 您無法從叢集新增或移除資料庫執行個體。
- 如果叢集仍然有任何關聯的資料庫執行個體，則無法刪除叢集。
- 一般而言，您必須重新啟動已停止的資料庫叢集，才能執行大部分的管理動作。
- 您已停止的叢集一重新啟動，Neptune 就會將任何排程的維護套用至這個叢集。請記住，Neptune 會在七天後自動啟動已停止的叢集，使其不會落後其維護狀態太多。
- Neptune 不會對已停止的資料庫叢集執行任何自動備份，因為在叢集停止時，基礎資料無法變更。
- Neptune 不會在資料庫叢集停止時延長備份保留期間。

啟動已停止的 Neptune 資料庫叢集

您只能啟動處於停止狀態的 Neptune 資料庫叢集。當您啟動叢集時，所有其資料庫執行個體會再次變為可用。叢集會保留其組態設定，例如端點、參數群組及 VPC 安全群組。

使用 AWS Management Console 啟動停止的資料庫叢集

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)，然後選擇一個叢集。您可以從這個頁面執行啟動操作，或導覽至資料庫叢集的詳細資訊頁面並從此處開始。
3. 在 Actions (動作) 中，選擇 Start (啟動)。

使用 AWS CLI 啟動停止的資料庫叢集

若要使用 AWS CLI 啟動停止的資料庫叢集，請使用 `--db-cluster-identifier` 參數呼叫 start-db-cluster 命令，以指定您要啟動的已停止資料庫叢集。提供您在建立資料庫叢集時選擇的叢集名稱，或使用您選擇的資料庫執行個體名稱，並在名稱結尾附加 `-cluster`。

Example

```
aws neptune start-db-cluster --db-cluster-identifier mydbcluster
```

使用 Neptune 管理 API 啟動停止的資料庫叢集

若要使用 Neptune 管理 API 啟動停止的 Neptune 資料庫叢集，請使用 `DBCluster` 參數呼叫 StartDBCluster API，以指定您要啟動的已停止資料庫叢集。提供您在建立資料庫叢集時選擇的叢集名稱，或使用您選擇的資料庫執行個體名稱，並在名稱結尾附加 `-cluster`。

使用快速重設 API 清空 Amazon Neptune 資料庫叢集

Neptune 快速重設 REST API 可讓您快速輕鬆地重設 Neptune 圖形，同時刪除其所有資料。

您可以使用 %db_reset 行魔法，在 Neptune 筆記本內執行此操作。

> **Note**
>
> 此功能從 Neptune 引擎 1.0.4.0 版開始可用。

- 在大部分情況下，快速重設操作會在幾分鐘內完成。持續時間可能會略有不同，取決於啟動操作時叢集上的負載。
- 快速重設操作不會產生額外的 I/O。
- 快速重設後，儲存磁碟區大小不會縮小。相反地，會在插入新資料時重複使用儲存體。這表示快速重設操作前後所建立之快照的磁碟區大小將相同。使用快速重設操作前後建立的快照所還原叢集的磁碟區大小也會相同
- 做為重設操作的一部分，資料庫叢集中的所有執行個體都會重新啟動。

> **Note**
>
> 在極少數情況下，這些伺服器重新啟動也可能導致叢集容錯移轉。

> **Important**
>
> 使用快速重設可能會中斷 Neptune 資料庫叢集與其他服務的整合。例如：
>
> - 快速重設會刪除資料庫中的所有串流資料，並完全重設串流。這表示若沒有新的組態，您的串流消費者可能不再運作。
> - 快速重設會移除有關 Neptune ML 正在使用之 SageMaker 資源的所有中繼資料，包括工作和端點。它們繼續存在於 SageMaker 中，而且您可以繼續使用現有的 SageMaker 端點進行 Neptune ML 推論查詢，但是 Neptune ML 管理 API 不再使用它們。
> - 也會透過快速重設來整合 (例如與 ElasticSearch 的全文檢索搜索整合)，並且必須手動重新建立它們，然後才能再次使用。

使用 API 刪除 Neptune 資料庫叢集中的所有資料

1. 首先，您會產生一個記號，然後可以將其用來執行資料庫重設。此步驟旨在協助防止任何人意外重設資料庫。

   您可以透過將 HTTP POST 請求傳送至資料庫叢集的寫入器執行個體上的 `/system` 端點，來指定 `initiateDatabaseReset` 動作以執行此操作。

   使用 JSON 內容類型的 `curl` 命令是：

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{ "action" : "initiateDatabaseReset" }'
   ```

   或者，使用 `x-www-form-urlencoded` 內容類型：

   ```
   curl -X POST \
     -H 'Content-Type: application/x-www-form-urlencoded' \
         https://your_writer_instance_endpoint:8182/system \
     -d 'action=initiateDatabaseReset '
   ```

   `initiateDatabaseReset` 請求會在其 JSON 回應中傳回重設記號，如下所示：

   ```
   {
     "status" : "200 OK",
     "payload" : {
       "token" : "new_token_guid"
     }
   }
   ```

   記號在發出後，有效期為一小時 (60 分鐘)。

   如果您將請求傳送到讀取器執行個體或狀態端點，Neptune 將擲回 `ReadOnlyViolationException`。

   如果您傳送多個 `initiateDatabaseReset` 請求，則只有最新產生的記號對第二個步驟有效，而您實際上在這個步驟中執行重設。

如果伺服器在您的 initiateDatabaseReset 請求之後立即重新啟動，則產生的記號將變成無效，並且您需要傳送新請求以取得新記號。

2. 接下來，您會將 performDatabaseReset 請求與您從 initiateDatabaseReset 取回的記號一起傳送至資料庫叢集的寫入器執行個體上的 /system 端點。這會從資料庫叢集中刪除所有資料。

   使用 JSON 內容類型的 curl 命令是：

```
curl -X POST \
  -H 'Content-Type: application/json' \
      https://your_writer_instance_endpoint:8182/system \
  -d '{
        "action" : "performDatabaseReset",
        "token" : "token_guid"
      }'
```

   或者，使用 x-www-form-urlencoded 內容類型：

```
curl -X POST \
  -H 'Content-Type: application/x-www-form-urlencoded' \
      https://your_writer_instance_endpoint:8182/system \
  -d 'action=performDatabaseReset&token=token_guid'
```

   請求會傳回 JSON 回應。如果接受請求，則回應是：

```
{
  "status" : "200 OK"
}
```

   如果您傳送的記號與發出的記號不符，則回應如下所示：

```
{
  "code" : "InvalidParameterException",
  "requestId":"token_guid",
  "detailedMessage" : "System command parameter 'token' : 'token_guid' does not
 match database reset token"
}
```

如果接受請求且開始重設，則伺服器會重新啟動並刪除資料。重設時，您無法將任何其他請求傳送至資料庫叢集。

使用快速重設 API 搭配 IAM-Auth

如果您在資料庫叢集上啟用了 IAM 身分驗證，則可以使用 awscurl，傳送使用 IAM-Auth 驗證的快速重設命令：

使用 awscurl 搭配 IAM-Auth 傳送快速重設請求

1. 正確設定 AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY 環境變數 (如果您使用的是臨時憑證，也會設定 AWS_SECURITY_TOKEN)。
2. initiateDatabaseReset 請求看起來像這樣：

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "initiateDatabaseReset" }'
```

3. performDatabaseReset 請求看起來像這樣：

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "performDatabaseReset" }'
```

使用 Neptune 工作台 **%db_reset** 行魔法重設資料庫叢集

Neptune 工作台支援 %db_reset 行魔法，可讓您在 Neptune 筆記本中執行快速資料庫重設。

如果在沒有任何參數的情況下調用魔法，您會看到一個畫面，詢問您是否要刪除叢集中的所有資料，並顯示一個核取方塊，要求您確認在刪除叢集資料後將無法再使用該叢集資料。此時，您可以選擇繼續刪除資料，或取消操作。

更危險的選項是使用 --yes 或 -y 選項來調用 %db_reset，這會導致在沒有進一步提示的情況下執行刪除。

您還可以透過兩個步驟執行重設，就像 REST API 一樣：

```
%db_reset --generate-token
```

回應為：

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

則執行：

```
%db_reset --token new_token_guid
```

回應為：

```
{
  "status" : "200 OK"
}
```

快速重設操作的常見錯誤代碼

Neptune 錯誤代碼	HTTP 狀態	訊息	範例
InvalidParameterException	400	系統命令參數 '*action*' 具有不支援的值 '*XXX*'	參數無效
InvalidParameterException	400	為以下動作提供了太多的值：*action*	一個快速重設請求，其中具有多個使用標頭 'Content-type:application/x-www-form-urlencoded' 傳送的動作
InvalidParameterException	400	重複欄位 'action'	一個快速重設請求，其中具有多個使用標頭 'Content-Type: application/json' 傳送的動作

Neptune 錯誤代碼	HTTP 狀態	訊息	範例
`MethodNotAllowedException`	400	錯誤的路由：*/bad_endpoint*	傳送至不正確端點的請求
`MissingParameterException`	400	缺少必要參數：[action]	快速重設請求未包含必要的 'action' 參數
`ReadOnlyViolationException`	400	不允許在僅供讀取複本執行個體上進行寫入	快速重設請求已傳送至讀取器或狀態端點
`AccessDeniedException`	403	缺少身分驗證字符	快速重設請求會在沒有正確簽章的情況下傳送至已啟用 IAM-Auth 的資料庫端點
`ServerShutdownException`	500	資料庫重設正在進行中。請在叢集可用之後重試查詢。	當快速重設開始時，現有和傳入的 Gremlin/Sparql 查詢會失敗。

將 Neptune 讀取器執行個體新增至資料庫叢集

在 Neptune 資料庫叢集中，有一個主要資料庫執行個體，以及最多 15 個 Neptune 讀取器執行個體 主要資料庫執行個體支援讀寫操作，並對叢集磁碟區執行所有資料修改。Neptune 讀取器執行個體會連線到與主要資料庫執行個體相同的儲存磁碟區，並僅支援讀取操作。

使用讀取器執行個體 ，從主要資料庫執行個體中卸載讀取工作負載。

建議您將資料庫叢集中的主要執行個體和 Neptune 讀取器分佈在數個可用區域上，以改善資料庫叢集的可用性。

下節描述如何在資料庫叢集中建立讀取器執行個體。

使用主控台建立 Neptune 讀取器執行個體

為 Neptune 資料庫叢集建立主要執行個體之後，您可以使用 Neptune 主控台，新增其他 Neptune 讀取器執行個體。

使用 AWS Management Console 建立 Neptune 讀取器執行個體

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選取您要在其中建立讀取器執行個體的資料庫叢集。
4. 選擇動作，然後選擇新增讀取器。
5. 在建立複本資料庫執行個體頁面上，指定 Neptune 複本的選項。下表顯示 Neptune 僅供讀取複本的設定。

對於此選項...	執行此作業
DB instance class (資料庫執行個體類別)	選擇資料庫執行個體類別，定義 Neptune 複本的處理和記憶體需求。如需 Neptune 在不同區域中提供之資料庫執行個體類別的目前清單，請參閱 Neptune 定價頁面。
Availability zone (可用區域)	指定可用區域。選擇與主要資料庫執行個體不同的區域。清單只包含資料庫叢集之資料庫子網路群組所對應的那些可用區域。
加密	啟用或停用加密。
Read replica source (僅供讀取複本來源)	選擇要為其建立 Neptune 複本之主要執行個體的識別符。
DB instance identifier (資料庫執行個體識別符) :	輸入執行個體的名稱，該名稱在您選取的區域中針對您的帳戶必須是唯一的。您可以選擇新增某些情報至名稱 (例如包括您選取的可用區域)，例如 `neptune-us-east-1c` 。
Database port (資料庫連線埠)	資料庫接受連線的連接埠號碼。

對於此選項...	執行此作業
DB parameter group (資料庫參數群組)	此執行個體的參數群組。
Log exports (日誌匯出)	選擇您要發佈的日誌 (若有的話)。
自動次要版本升級	當次要 Neptune 資料庫引擎版本升級變成可用時，如果您想要讓 Neptune 複本可以自動接收這些升級，請選擇是。 Auto Minor Version Upgrade (自動次要版本升級) 選項僅適用於次要升級。該選項不適用於引擎維護修補程式，這些修補程式一律會自動套用，以保持系統穩定。

6. 選擇建立僅供讀取複本以建立 Neptune 複本執行個體。

若要從資料庫叢集中移除 Neptune 讀取器執行個體，請遵循刪除 Amazon Neptune 中的資料庫執行個體中的指示。

使用主控台修改 Neptune 資料庫叢集

使用 AWS Management Console 修改資料庫執行個體時，您可以透過選取 Apply Immediately (立即套用)，來選擇立即套用變更。若您選擇立即套用變更，則系統會立即套用待定修改佇列中的新變更及任何變更內容。

若您未選擇立即套用變更，則系統會將變更內容放入待定修改佇列。在下一次維護時段期間，系統便會套用佇列中的任何待定變更。

> **Important**
>
> 如果有任何擱置中的修改需要停機才能執行，選擇立即套用變更可能會導致有問題的資料庫執行個體未預期的停機。資料庫叢集中的其他資料庫執行個體沒有任何停機時間。

> **Note**
>
> 當您在 Neptune 中修改資料庫叢集時，立即套用設定僅會影響對資料庫叢集識別符、IAM 資料庫身分驗證 的變更。所有其他的修改都會立即套用，無論 Apply Immediately (立即套用) 設定的值為何。

使用主控台修改資料庫叢集

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Clusters (叢集)，然後選擇您要修改的資料庫叢集。
3. 選擇 Actions (動作)，然後選擇 Modify cluster (修改叢集)。Modify DB cluster (修改資料庫叢集) 頁面隨即出現。
4. 變更您要的任何設定。

   > **Note**
   >
   > 在主控台上，某些執行個體層級變更僅會套用到目前的資料庫執行個體，而其他的變更則會套用到整個資料庫叢集。若要在主控台上變更於執行個體層級修改整個資料庫叢集的設定，請遵循修改資料庫叢集中的資料庫執行個體中的說明。

5. 當所有變更都如您所願時，請選擇 Continue (繼續) 並查看摘要。

6. 若要立即套用變更，請選取 Apply immediately (立即套用)。
7. 在確認頁面上，檢閱您的變更。如果都正確，請選擇 Modify cluster (修改叢集) 以儲存您的變更。

   若要編輯您的變更，請選擇 Back (返回)，或是若要取消您的變更，請選擇 Cancel (取消)。

修改資料庫叢集中的資料庫執行個體

使用主控台修改資料庫叢集中的資料庫執行個體

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中選擇 Instances (執行個體)，然後選擇您要修改的資料庫執行個體。
3. 選擇 Instance actions (執行個體動作)，然後選擇 Modify (修改)。Modify DB Instance (修改資料庫執行個體) 頁面隨即出現。
4. 變更您要的任何設定。

   > **Note**
   >
   > 有些設定會套用到整個資料庫叢集，且必須在叢集層級進行變更。若要變更這些設定，請遵循使用主控台修改 Neptune 資料庫叢集中的說明。
   > 在 AWS Management Console中，某些執行個體層級變更僅會套用到目前的資料庫執行個體，而其他的變更則會套用到整個資料庫叢集。

5. 當所有變更都如您所願時，請選擇 Continue (繼續) 並查看摘要。
6. 若要立即套用變更，請選取 Apply immediately (立即套用)。
7. 在確認頁面上，檢閱您的變更。如果都正確，請選擇 Modify DB Instance (修改資料庫執行個體) 以儲存您的變更。

   若要編輯您的變更，請選擇 Back (返回)，或是若要取消您的變更，請選擇 Cancel (取消)。

Amazon Neptune 中的效能和擴展

三種不同層級的 Neptune 資料庫叢集和執行個體擴展：

- 儲存體擴展
- 執行個體擴展；
- 讀取擴展

Neptune 中的儲存體擴展

Neptune 儲存體會隨著叢集磁碟區中的資料而自動擴展。隨著您的資料成長，在所有支援的區域中，您的叢集磁碟區儲存體最多可成長至 128 TiB，但中國和 GovCloud 除外，因其限制為 64 TiB。

每小時會檢查一次叢集磁碟區的大小，以決定儲存成本。

Neptune 資料庫耗用的儲存體以每月每 GB 增量計費，而耗用的 I/O 則以每百萬請求數增量計費。您只需支付 Neptune 資料庫使用的儲存體和 I/O，且不需要預先佈建。

如需定價資訊，請參閱 Neptune 產品頁面。

Neptune 中的執行個體擴展

您可以修改資料庫叢集中每個資料庫執行個體的資料庫執行個體類別，視需要擴展 Neptune 資料庫叢集。Neptune 支援數種最佳化的資料庫執行個體。

Neptune 中的讀取擴展

您可以在資料庫叢集中建立最多 15 個 Neptune 複本，以實現 Neptune 資料庫叢集的讀取擴展。在主要執行個體寫入更新之後，每個 Neptune 複本會以最短的複本延遲從叢集磁碟區傳回相同的資料 (通常遠低於 100 毫秒)。當讀取流量增加時，您可以建立額外的 Neptune 複本，並直接連線這些複本，以分散資料庫叢集的讀取負載。Neptune 複本的資料庫執行個體類別不必與主要執行個體相同。

如需將 Neptune 複本新增到資料庫叢集的相關資訊，請參閱 新增讀取器執行個體。

自動擴展 Amazon Neptune 資料庫叢集中的複本數目

您可以使用 Neptune 自動擴展，自動調整資料庫叢集中的 Neptune 複本數目，以符合您的連線能力和工作負載需求。自動調整規模可讓 Neptune 資料庫叢集處理工作負載的增加，然後，當工作負載降低時，自動擴展會移除不必要的複本，因此對於未使用的容量您無需支付費用。

您只能使用自動擴展，搭配已有一個主要寫入器執行個體和至少一個僅供讀取複本執行個體的叢集 (請參閱 Amazon Neptune 資料庫叢集和執行個體)。此外，叢集中的所有僅供讀取複本執行個體都必須處於可用狀態。如果任何僅供讀取複本處於可用以外的狀態，Neptune 自動擴展不會執行任何動作，直到叢集中的每個僅供讀取複本都可用為止。

如果您需要建立新的叢集，請參閱 建立資料庫叢集。

使用 AWS CLI，您可以定義擴展政策並將其套用至資料庫叢集。您也可以使用編 AWS CLI 輯或刪除自動調度資源政策。此政策會指定下列自動擴展參數：

- 要在叢集中具有的複本數目下限和上限。
- 複本-附加擴展活動之`ScaleOutCooldown`間的`ScaleInCooldown`間隔，以及複本刪除擴展活動之間的間隔。
- 向上或向下擴展的度量和指標觸發值。 CloudWatch

Neptune 自動擴展動作的頻率會以下列幾種方式加以控制：

- 最初，若要自動擴展以新增或刪除讀取器，必須違反 `CPUUtilization` 高警示至少 3 分鐘，或者必須違反低警示至少 15 分鐘。
- 在第一次新增或刪除之後，後續 Neptune 自動擴展動作的頻率會受到自動擴展政策中 `ScaleOutCooldown` 和 `ScaleInCooldown` 設定的限制。

如果您使用的指 CloudWatch 標達到您在政策中指定的高臨界值，並且自上次自動調度資 auto-scaling 作以來的`ScaleOutCooldown`間隔已經過去，並且您的資料庫叢集還沒有您設定的最大複本，Neptune auto auto-scaling 會使用與資料庫叢集的主要執行個體相同的執行個體類型建立新複本。

同樣地，如果指標達到您指定的低閾值、如果自上次自動擴展動作後 `ScaleInCooldown` 間隔已過了，以及如果資料庫叢集具有的複本數目超過您指定的複本數目下限，則 Neptune 自動擴展會刪除其中一個複本。

> **Note**
>
> Neptune 自動擴展只會移除它建立的複本。它不會移除預先存在的複本。

使用 neptune_autoscaling_config 資料庫叢集參數，您也可以指定 Neptune 自動擴展建立的新僅供讀取複本的執行個體類型、這些僅供讀取複本的維護時段，以及要與每個新僅供讀取複本相關聯的標籤。您可以在 JSON 字串中提供這些組態設定，做為 neptune_autoscaling_config 參數的值，如下所示：

```
"{
  \"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\"
}"
```

請注意，JSON 字串中的引號必須全都使用反斜線字元 (\) 來逸出。像往常一樣，字串中的所有空格都是選用的。

三個組態設定中若有任何一個未在 neptune_autoscaling_config 參數中指定，其會從資料庫叢集的主要寫入器執行個體的組態中複製。

當自動擴展新增僅供讀取複本執行個體時，它會在資料庫執行個體 ID 前面加上 autoscaled-reader (例如，autoscaled-reader-7r7t7z3lbd-20210828)。它也會將一個標籤新增至其使用金鑰 autoscaled-reader 和 值 TRUE 建立的每個僅供讀取複本。您可以在 AWS Management Console中資料庫執行個體詳細資訊頁面的標籤索引標籤上看到此標籤。

```
 "key" : "autoscaled-reader",  "value" : "TRUE"
```

透過自動擴展建立的所有僅供讀取複本執行個體的提升層是最低的優先順序，根據預設，其為 15。這表示在容錯移轉期間，優先順序較高的任何複本 (例如手動建立的複本) 將會優先提升。請參閱Neptune 資料庫叢集的容錯能力。

Neptune auto-scaling 是使用「應用程式自動調整」來實作，搭配使用 Neptune CPUUtilization CloudWatch 度量做為預先定義量度的目標追蹤調整規

在 Neptune 無伺服器資料庫叢集中使用自動擴展

當需求超過執行個體的容量時，Neptune Serverless 的回應速度會比 Neptune 自動擴展快得多，且其會縱向擴展執行個體而不是新增另一個執行個體。自動擴展旨在匹配工作負載相對穩定的增加或減少，而無伺服器則擅長處理需求的快速激增和抖動。

了解它們的優勢，您可以結合自動擴展和無伺服器，來建立彈性的基礎架構，其會有效地處理工作負載的變更，並在滿足需求，同時將成本降至最低。

若要允許自動擴展與無伺服器一起有效運作，請務必設定無伺服器叢集的 maxNCU 設定，使其足夠高到因應需求中的爆增和短暫變更。否則，暫時性變更不會觸發無伺服器擴展，這可能會導致自動擴展啟動了許多不必要的額外執行個體。如果將 maxNCU 設定足夠高，無伺服器擴展可以更快速且成本更低地處理這些變更。

如何為 Amazon Neptune 啟用自動擴展

只能使用 AWS CLI為 Neptune 資料庫叢集啟用自動擴展。您無法使用 AWS Management Console啟用自動擴展。

此外，以下 Amazon 區域不支援自動擴展：

- 非洲 (開普敦)：af-south-1
- 中東 (阿拉伯聯合大公國)：me-central-1
- AWS GovCloud (美國東部): us-gov-east-1
- AWS GovCloud (美國西部): us-gov-west-1

為 Neptune 資料庫叢集啟用自動擴展涉及三個步驟：

1. 向 Application Auto Scaling 註冊您的資料庫叢集

為 Neptune 資料庫叢集啟用自動擴展的第一個步驟是，使用 AWS CLI 或其中一個 Application Auto Scaling SDK，向 Application Auto Scaling 註冊叢集。叢集必須已有一個主要執行個體和至少一個僅供讀取複本執行個體：

例如，若要註冊叢集以自動調整規模，從一到八個額外的複本，您可以使用 AWS CLI register-scalable-target命令，如下所示：

```
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
```

這等同於使用 RegisterScalableTarget Application Auto Scaling API 操作。

該 AWS CLI `register-scalable-target` 命令會使用下列參數：

- **`service-namespace`** – 設為 `neptune`。

  此參數等同於 Application Auto Scaling API 中的 `ServiceNamespace` 參數。
- **`resource-id`** – 將其設定為 Neptune 資料庫叢集的資源識別符。資源類型是 `cluster`，其後跟著一個冒號 (':')，然後是資料庫叢集的名稱。

  此參數等同於 Application Auto Scaling API 中的 `ResourceID` 參數。
- **`scalable-dimension`** – 在這種情況下，可擴展維度是資料庫叢集中複本執行個體的數量，因此您會將此參數設定為 `neptune:cluster:ReadReplicaCount`。

  此參數等同於 Application Auto Scaling API 中的 `ScalableDimension` 參數。
- **`min-capacity`** – Application Auto Scaling 要管理的讀取器資料庫複本執行個體數量下限。此值應該設定在 0 到 15 的範圍內，而且必須等於或小於針對 `max-capacity` 中 Neptune 複本數量上限指定的值。資料庫叢集中至少必須有一個讀取器，自動擴展才能運作。

  此參數等同於 Application Auto Scaling API 中的 `MinCapacity` 參數。
- **`max-capacity`** – 資料庫叢集中讀取器資料庫複本執行個體的數量上限，包括 Application Auto Scaling 管理的預先存在執行個體和新執行個體。此值必須設定在 0 到 15 的範圍內，而且必須等於或大於針對 `min-capacity` 中 Neptune 複本數量下限指定的值。

  此`max-capacity` AWS CLI 參數相當於 Application Auto Scaling API 中的`MaxCapacity`參數。

當您註冊資料庫叢集時，Application Auto Scaling 會建立 `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster` 服務連結角色。如需詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的 Application Auto Scaling 的服務連結角色。

2. 定義要與資料庫叢集搭配使用的自動擴展政策

目標追蹤擴展政策會定義為 JSON 文字物件，此物件也可以儲存在文字檔案中。對於 Neptune，此原則目前只能使用 Neptune CPUUtilization CloudWatch 度量做為名為的預先定義度量`NeptuneReaderAverageCPUUtilization`。

以下是 Neptune 的範例目標追蹤擴展組態政策。

```
{
  "PredefinedMetricSpecification": { "PredefinedMetricType":
 "NeptuneReaderAverageCPUUtilization" },
  "TargetValue": 60.0,
  "ScaleOutCooldown" : 600,
  "ScaleInCooldown" : 600
}
```

這裡的 **TargetValue** 元素包含 CPU 使用率的百分比，超過此百分比，自動擴展會「橫向擴展」(也就是新增更多複本)，低於此百分比，自動擴展會「縮減」(也就是刪除複本)。在此情況下，觸發擴展的目標百分比為 60.0%。

ScaleInCooldown 元素指定在縮減活動完成後到另一個縮減活動開始前的時間長度 (以秒為單位)。預設為 300 秒。此處的值 600 指定在完成一個複本刪除與啟動另一個複本之間至少必須經過十分鐘。

ScaleOutCooldown 元素指定在橫向擴展活動完成後到另一個橫向擴展活動開始前的時間長度 (以秒為單位)。預設為 300 秒。此處的值 600 指定在完成一個複本新增與啟動另一個複本之間至少必須經過十分鐘。

DisableScaleIn 元素是一個布林值，如果存在並設定為 `true`，則會完全停用縮減，這表示自動擴展可能會新增複本，但永遠不會刪除任何複本。根據預設，會啟用縮減，且 `DisableScaleIn` 為 `false`。

向 Application Auto Scaling 註冊您的 Neptune 資料庫叢集，並以文字檔案定義 JSON 擴展政策之後，接著將擴展政策套用到已註冊的資料庫叢集。您可以使用 AWS CLI [put-scaling-policy](#)命令來執行此操作，參數如下：

```
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)
```

在套用了自動擴展政策時，就會在資料庫叢集上啟用自動擴展。

您也可以使用命 AWS CLI `put-scaling-policy`令來更新現有的 auto-scaling 政策。

另請參閱 PutScalingAp plication Auto Scaling API 參考中的政策。

從 Neptune 資料庫叢集移除自動擴展

若要從 Neptune 資料庫叢集移除 auto-scaling 資源，請使用 AWS CLI刪除擴展原則和取消註冊-可擴充目標命令。

維護 Amazon Neptune 資料庫叢集

Neptune 會定期針對其使用的所有資源執行維護，包括：

- 視需要更換基礎硬體設施。這會在背景中作業，您無需採取任何動作，而且通常不會影響您的運作。
- 更新基礎作業系統。資料庫叢集中執行個體的作業系統升級是為了提升效能和安全性，因此通常須盡速完成這些執行個體。通常，更新大約需要 10 分鐘。作業系統更新不會變更資料庫執行個體的資料庫引擎版本或資料庫執行個體類別。

  一般而言，最好先更新資料庫叢集中的讀取器執行個體，然後再更新寫入器執行個體。同時更新讀取器和寫入器可能會在發生容錯移轉時導致停機。請注意，在作業系統更新之前，資料庫執行個體不會自動備份，因此請務必在套用作業系統更新之前進行手動備份。
- 更新 Neptune 資料庫引擎。Neptune 會定期發布各種引擎更新，以導入新特色和改進功能,並修復錯誤。

引擎版本編號

引擎版本 1.3.0.0 之前的版本編號

在 2019 年 11 月之前，Neptune 一次僅支援一個引擎版本，而且所有引擎版本號碼皆採用以下格式：`1.0.1.0.200<xxx>`，其中 `xxx` 是修補程式號碼。新引擎版本均以舊版修補程式的形式發布。

自 2019 年 11 月起，Neptune 開始支援多重版本，讓客戶更能妥善管控其升級途徑。因此，引擎版本編號已變更。

從 2019 年 11 月起，直到引擎版本 1.3.0.0 為止，引擎版本編號有 5 個部分。以版本編號 `1.0.2.0.R2` 為例：

- 第一部分永遠是 1。
- 第二部分 (`0` 在 `1.0.2.0.R2` 中) 是資料庫主要版本號碼。
- 第三部分和第四部分 (`2.0` 在 `1.0.2.0.R2` 中) 都是次要版本號碼。
- 第五部分 (`R2` 在 `1.0.2.0.R2` 中) 是修補程式編號。

大多數更新都是修補程式更新，修補程式和次要版本更新之間的區別不是那麼清楚。

從引擎版本 1.3.0.0 開始的版本編號

從引擎版本 1.3.0.0 開始，Neptune 改變了引擎更新的編號和管理方式。

引擎版本編號現在有四個部分，每個部分對應一種發行版本類型，如下所示：

product-version.*major-version*.*minor-version*.*patch-version*

先前作為修補程式發行之排序的非中斷變更，現在會以次要版本的形式發行，您可以使用 AutoMinorVersionUpgrade 執行個體設定來管理。

這表示如果您希望每次發行新的次要版本時，都可以透過訂閱 RDS-EVENT-0156 事件來接收通知 (請參閱訂閱 Neptune 事件通知)。

修補程式版本目前保留給緊急目標修正，並使用版本號碼的最後部分的號碼 (*.*.*.1、*.*.*.2 等) 進行編號。

Amazon Neptune 中不同類型的引擎版本

與引擎版本號碼的四個部分相對應的四種引擎版本類型如下：

- 產品版本 – 只有在產品功能或介面發生徹底的根本變更時，才會變更。目前的 Neptune 產品版本為 1。
- 主要版本 – 主要版本引入重要的新功能和突破性的變更，一般都具有至少兩年的使用期限。
- 次要版本 – 次要版本可以包含新特色、改進功能和錯誤修復，但不包含重大變更。您可以選擇是否在下一個維護時段中自動套用新功能，也可以選擇在版本發行時收到通知。
- 修補程式版本 – 只在要解決緊急錯誤修正或重大安全性更新時發行修補程式版本。這些版本很少包含重大變更，而且會在發行後的下一個維護時段自動套用這些變更。

Amazon Neptune 主要版本更新

主要版本更新通常會導入一或多個重要的新功能，而且通常包含重大變更。其通常具有大約兩年的支援期限。Neptune 主要版本會列在引擎版本中，以及發行的日期及其預估的生命週期結束時間。

在您使用的主要版本到達其生命週期結束之前，主要版本更新是完全選用的。如果選擇升級到新的主要版本，您必須使用 AWS CLI 或 Neptune 主控台自行安裝新版本，如 主要版本升級 中所述。

但是，如果您使用的主要版本生命週期已經結束，系統會通知您必須升級到更新的更新的主要版本。然後，如果您未在通知後的寬限期內升級，系統會自動排程在下一個維護時段升級至最新的主要版本。如需更多資訊，請參閱引擎版本生命週期。

Amazon Neptune 次要版本更新

大多數 Neptune 引擎更新都是次要版本更新。更新次數相當頻率，並且不包含重大變更。

如果您在資料庫叢集的寫入器 (主要) 執行個體中將 AutoMinorVersionUpgrade 欄位設定為 true，次要版本更新會在發行後的下一個維護時段，自動將次要版本更新套用至資料庫叢集中的所有執行個體。

如果您在資料庫叢集的寫入器執行個體中將 AutoMinorVersionUpgrade 欄位設定為 false，則只有在明確安裝時才會套用這些更新。

> **Note**
>
> 次要版本更新是自給自足的 (而不是依賴相同主要版本的先前次要版本更新) 和累積 (包含先前次要版本更新中導入的所有功能和修正程式)。這意味著您可以安裝任何指定的次要版本更新，無論您是否已安裝以前的更新。

您可以透過訂閱 RDS-EVENT-0156 事件輕鬆追蹤次要版本的發行 (請參閱訂閱 Neptune 事件通知)。然後，每次發行新的次要版本時，您都會收到通知。

而且，無論您是否訂閱通知，都可以隨時查看有哪些更新擱置中。

Amazon Neptune 修補程式更新

如果發生安全問題或其他影響執行個體可靠性的嚴重缺陷，Neptune 會部署強制性修補程式。在下一個維護時段期間，會將修補程式套用至資料庫叢集中的所有執行個體，而不需要您的介入。

只有在未部署修補程式的風險大於與部署相關的任何風險和停機時間時，才會部署修補程式版本。不常發行修補程式 (通常幾個月才發行一次)，且幾乎不需要太長的維護時段。

規劃 Amazon Neptune 主要引擎版本生命週期

Neptune 引擎版本幾乎總是在行事曆季度結束時達到其生命週期結束。僅在發生重要安全性或可用性問題時，才會發生例外狀況。

當引擎版本到達其生命週期結束時，您需要將 Neptune 資料庫升級至更新的版本。

一般而言，Neptune 引擎版本繼續可供使用，如下所示：

- 次要引擎版本：次要引擎版本在發行後至少仍能使用 6 個月。
- 主要引擎版本：主要引擎版本在發行後至少仍能使用 12 個月。

在引擎版本到達生命週期結束之前至少 3 個月，AWS 會將自動電子郵件通知傳送至與 AWS 帳戶關聯的電子郵件地址，並將相同的訊息張貼至您的 AWS 運作狀態儀表板。這會讓您有時間規劃和準備升級。

當引擎版本到達其生命週期結束時，您將再也無法使用該版本建立新的叢集或執行個體，自動擴展也無法使用該版本建立執行個體。

實際到達生命週期結束的引擎版本將在維護時段期間自動升級。在引擎版本生命週期結束前 3 個月傳送給您的訊息將包含此自動更新所涉及內容的詳細資料，包括您將自動升級至哪個版本、對資料庫叢集的影響，以及我們建議的動作。

Important

您負責將資料庫引擎版本保持在最新狀態。AWS 會促使所有客戶將其資料庫升級至最新的引擎版本，以便得益於最新的安全性、隱私權和可用性防禦措施。如果在超過棄用日期的不受支援引擎或軟體 (「舊版引擎」) 上操作資料庫，您可能會面臨更大的安全性、隱私權和操作風險 (包括停機事件)。

在任何引擎上操作您的資料庫都會受制於控管您使用 AWS 服務的協議。舊版引擎未正式推出。如果 AWS 確定舊版引擎對服務、AWS、其關係企業或任何第三方構成安全性或責任風險，或有傷害風險，則 AWS 不再對舊版引擎提供支援，且 AWS 可能隨時限制對任何舊版引擎的存取或使用。您決定繼續在舊版引擎中執行您的內容，可能會導致您的內容無法使用、損毀或無法復原。在舊版引擎上執行的資料庫受服務水準協議 (SLA) 例外狀況約束。

在舊版引擎上執行的資料庫和相關軟體包含錯誤、瑕疵和/或有害元件。因此，儘管協議或服務條款有任何相反的規定，AWS 仍按「原狀」提供舊版引擎。

管理 Neptune 資料庫叢集的引擎更新

Note

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 或 30 秒的停機時間，之後就可以繼續使用資料庫叢集。在極少數情況下，執行個體需要執行多可用區容錯移轉，才能完成維護更新作業。

對於可能需要較長時間才能套用的主要版本升級，您可以使用藍綠部署策略將停機時間降至最低。

決定您目前使用的引擎版本

您可以使用AWS CLI get-engine-status 命令來檢查您的資料庫叢集目前所使用的引擎版本：

```
aws neptunedata get-engine-status
```

JSON 輸出包括如下所示之 "dbEngineVersion" 欄位：

```
  "dbEngineVersion": "1.3.0.0",
```

查看有哪些更新正在擱置中且可用

您可以使用 Neptune 主控台，檢查資料庫叢集的待處理更新。選取左欄中的資料庫，然後在資料庫窗格中選取您的資料庫叢集。待處理更新會列在維護欄中。如果依序選取動作和維護，您可選取三個選項：

- 立即升級。
- 在下一個時段升級。
- 延遲升級。

您可以使用 AWS CLI 列出待處理的引擎更新，如下所述：

```
aws neptune describe-pending-maintenance-actions \
  --resource-identifier (ARN of your DB cluster)
  --region (your region) \
  --engine neptune
```

您也可以使用 AWS CLI 列出可用的引擎版本，如下所述：

```
aws neptune describe-db-engine-versions \
  --region (your region) \
  --engine neptune
```

可用的引擎版本清單僅包含具有高於現有版本編號的發行版本，並且已定義其升級路徑者。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。次要升級會導入影響程式碼 (即使沒有重大更新) 的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方法是使用 Neptune 藍綠部署解決方案。這樣您就可以在新版本上執行應用程式和查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

Neptune 維護時段

每週維護時段是 30 分鐘時間，在此期間會套用排程的引擎更新和其他系統變更。大多數維護事件都能在 30 分鐘的維護時段內完成，但較大型的維護事件有時可能需要更長的時間才能完成。

每個資料庫叢集都有每週 30 分鐘的維護時段。若您在建立資料庫叢集時沒有指定偏好的時間點，則 Neptune 會隨機選取一週內的某一天，然後從 8 小時的時段中隨機分配 30 分鐘的時段，該時段因地區而異。

例如，這裡有幾個 AWS 區域中使用的維護時段的 8 小時的區塊：

區域	時間區塊
美國西部 (奧勒岡) 區域	上午 6 時至下午 2 時 (UTC)
美國西部 (加利佛尼亞北部) 區域	上午 6 時至下午 2 時 (UTC)
美國東部 (俄亥俄) 區域	上午 3 時至上午 11 時 (UTC)
歐洲 (愛爾蘭) 區域	下午 10 時至次日上午 6 時 (UTC)

維護時段決定了擱置作業的開始處理時間，以及大部分維護作業在時段內完成，但較大規模的維護作業可能會在時段結束後繼續進行。

移動資料庫叢集維護時段

理想情況下，您的維護時段應該會在叢集最低用量時下降。如果這不是您目前的時段，您則可以將其移動到更好的時間點，如下所示：

若要變更您的資料庫叢集維護時段

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇資料庫。
3. 選擇資料庫叢集，以變更其維護時段。
4. 選擇 Modify (修改)。
5. 選擇修改叢集頁面底部的顯示更多。
6. 在偏好的維護時段區段中，根據您的喜好設定維護時段的日期、時間和持續時間。
7. 選擇 Next (下一步)。

   在確認頁面上，檢閱您的變更。
8. 選取 Apply immediately (立即套用)，就能立刻將變更套用至維護時段。
9. 選擇提交來套用您的變更。

   若要編輯您的變更，請選擇返回，或是若要取消您的變更，請選擇取消。

使用 `AutoMinorVersionUpgrade` 控制自動次要版本更新

> **Important**
>
> `AutoMinorVersionUpgrade` 僅適用於引擎版本 1.3.0.0 以上的次要版本升級。

如果您在資料庫叢集的寫入器 (主要) 執行個體中將 `AutoMinorVersionUpgrade` 欄位設定為 `true`，次要版本更新會在發行後的下一個維護時段中，自動將次要版本更新套用至資料庫叢集的所有執行個體。

如果您在資料庫叢集的寫入器執行個體中將 `AutoMinorVersionUpgrade` 欄位設定為 `false`，則只有在明確安裝時才會套用這些更新。

> **Note**
>
> 無論 AutoMinorVersionUpgrade 參數的設定方式為何，修補程式版本 (*.*.*.1、*.*.*.2 等) 一律會在下一個維護時段自動安裝。

您可以使用 AWS Management Console 設定 AutoMinorVersionUpgrade，如下所示：

使用 Neptune 主控台設定 **AutoMinorVersionUpgrade**

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇您要為其設定 AutoMinorVersionUpgrade 的資料庫叢集的主要 (寫入器) 執行個體。
4. 選擇 Modify (修改)。
5. 選擇修改叢集頁面底部的顯示更多。
6. 在展開的頁面底部，選擇開啟次要版本自動升級或關閉次要版本自動升級。
7. 選擇 Next (下一步)。

   在確認頁面上，檢閱您的變更。
8. 若要將變更套用至自動升級次要版本，請選取立即套用。
9. 選擇提交來套用您的變更。

   若要編輯您的變更，請選擇返回，或是若要取消您的變更，請選擇取消。

您也可以使用 AWS CLI 設定 AutoMinorVersionUpgrade 欄位。例如，若要將其設定為 true，您可以使用如下所示之命令：

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --auto-minor-version-upgrade \
  --apply-immediately
```

同樣的，若要將其設定為 false，請使用如下所示之命令：

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
```

```
  --no-auto-minor-version-upgrade \
  --apply-immediately
```

手動安裝更新至您的 Neptune 引擎

安裝主要版本引擎升級

主要引擎版本一律必須手動安裝。為了將停機時間減到最少，並提供足夠的時間進行測試和驗證，安裝新的主要版本的最佳方法通常是使用 Neptune 藍綠部署解決方案。

在某些情況下，您也可以使用用來建立資料庫叢集的 AWS CloudFormation 範本來安裝主要版本升級 (請參閱 使用 AWS CloudFormation 範本更新 Neptune 資料庫叢集的引擎版本)。

如果想要立即安裝主要版本更新，您可以使用如下所示的 CLI 命令：

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (identifier for your neptune cluster) \
  --engine neptune \
  --engine-version (the new engine version) \
  --apply-immediately
```

請務必指定要升級為哪個引擎版本。若未指定，引擎可能會升級為非最新的版本，或是非您預期的版本。

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。

如果您的叢集使用自訂叢集參數群組，請務必使用此參數來指定：

```
  --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣的，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必使用此參數來指定：

```
  ---db-instance-parameter-group-name (name of the custom instance parameter group)
```

使用 AWS Management Console 安裝次要版本引擎升級

使用 Neptune 主控台執行次要版本升級

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇資料庫，然後選取您要修改的資料庫叢集。

3. 選擇 Modify (修改)。
4. 在執行個體規格下方，選擇您要升級至哪個新版本。
5. 選擇 Next (下一步)。
6. 若要立即套用變更，請選擇立即套用。
7. 選擇提交以更新您的資料庫叢集。

使用 AWS CLI 安裝次要版本引擎升級

只要使用類似下列的命令，您不必等到下一個維護時段就能執行次要版本升級：

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your-neptune-cluster) \
  --engine-version (new-engine-version) \
  --apply-immediately
```

如果您使用 AWS CLI 手動升級，請務必包含您要升級的引擎版本。若未指定，引擎可能會升級為並非最新的版本，或是並非您預期的版本。

從 1.2.0.0 之前的版本升級至引擎 1.2.0.0 版或更新版本

引擎版本 1.2.0.0 會導入幾項重要變更，這些變更可能會使從舊版升級變得比以往更加複雜：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

使用 AWS CloudFormation 範本更新 Neptune 資料庫叢集的引擎版本

您可以重複使用用來建立 Neptune 資料庫叢集的 Neptune AWS CloudFormation 範本，以更新其引擎版本。

Neptune 引擎版本升級可以是次要或主要升級。使用 AWS CloudFormation 範本可協助進行主要版本升級，這些升級通常包含重大變更。由於主要版本升級可能包含不能與現有應用程式回溯相容的資料庫變更，因此您在升級時可能還需要對應用程式進行變更。一律先測試再升級，而且我們強烈建議您一律在升級前建立資料庫叢集的手動快照。

請注意，您必須為每個主要版本進行個別的引擎升級。您無法跳過主要版本並直接升級到以下主要版本。

在 2023 年 5 月 17 日之前，如果您使用 Neptune AWS CloudFormation 堆疊升級引擎版本，它只會建立新的空白資料庫叢集來取代您目前的資料庫叢集。不過，從 2023 年 5 月 17 日起，Neptune AWS CloudFormation 堆疊現在支援就地引擎升級，以保留您現有的資料。

對於主要版本升級，您的範本應在 `DBCluster` 中設定下列屬性：

- `DBClusterParameterGroup` (自訂或預設)
- `DBInstanceParameterGroupName`
- `EngineVersion`

同樣地，對於附加到 DBCluster 的 DBInstances，您應該設定：

- `DBParameterGroup` (自訂/預設)

確保您的所有參數群組都定義在範本中，無論它們是預設的還是自訂的。

若是自訂參數群組，請確定現有自訂參數群組的系列與新引擎版本相容。1.2.0.0 之前的引擎版本已使用參數群組系列 `neptune1`，而 1.2.0.0 之後的引擎版本則需要參數群組系列 `neptune1.2`。如需更多資訊，請參閱Amazon Neptune 參數群組。

對於主要引擎版本升級，請在 `DBCluster` `DBInstanceParameterGroupName` 欄位中指定具有適當系列的參數群組。

預設參數群組應升級至與新引擎版本相容的參數群組。

請注意，Neptune 會在引擎升級後自動重新啟動資料庫執行個體。

主題

範例：次要引擎從 1.2.0.1 升級至 1.2.0.2

尋找您要升級的資料庫叢集，以及您用來建立它的範本。例如：

```
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
```

將 EngineVersion 屬性從 1.2.0.1 更新為 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
```

現在使用 AWS CloudFormation 來執行修訂的範本。

範例：主要版本從 1.1.1.0 升級至 1.2.0.2，其中具有預設參數群組

尋找您要升級的 DBCluster，以及您用來建立它的範本。例如：

```
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
```

- 將預設 DBClusterParameterGroup 更新為新引擎版本使用的參數群組系列中的參數群組 (此處的 default.neptune1.2)。
- 對於附加至 DBCluster 的每個 DBInstance，將預設 DBParameterGroup 更新為新引擎版本使用的系列中的參數群組 (此處的 default.neptune1.2)。
- 將 DBInstanceParameterGroupName 屬性設定為該系列中的預設參數群組 (此處的 default.neptune1.2)。
- 將 EngineVersion 屬性從 1.1.0.0 更新為 1.2.0.2。

範本應如下所示：

```
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:
```

現在使用 AWS CloudFormation 來執行修訂的範本。

範例：主要版本從 1.1.1.0 升級至 1.2.0.2，其中具有自訂參數群組

尋找您要升級的 DBCluster，以及您用來建立它的範本。例如：

```
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
```

- 將自訂DBClusterParameterGroup系列更新為此處新引擎版本所使用的自訂系列default.neptune1.2)。
- 對於每個DBInstance附加到的DBCluster，請將自訂DBParameterGroup系列更新為新引擎版本所使用的自訂系列 (此處default.neptune1.2)。
- 將 DBInstanceParameterGroupName 屬性設定為該系列中的參數群組 (此處的 default.neptune1.2)。
- 將 EngineVersion 屬性從 1.1.0.0 更新為 1.2.0.2。

範本應如下所示：

```
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
```

現在使用 AWS CloudFormation 來執行修訂的範本。

範例：主要版本從 1.1.1.0 升級至 1.2.0.2，其中混有預設和自訂參數群組

尋找您要升級的 DBCluster，以及您用來建立它的範本。例如：

```
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
```

- 對於自訂叢集參數群組，請將 DBClusterParameterGroup 系列更新為對應至新引擎版本的系列，即 neptune1.2。
- 對於預設叢集參數群組，請將 DBClusterParameterGroup 更新為對應至新引擎版本的預設值，即 default.neptune1.2。
- 對於附加至 DBCluster 的每個 DBInstance，請將預設 DBParameterGroup 更新為新引擎版本使用的系列中的參數群組 (此處的 default.neptune1.2)，並將自訂參數群組更新為使用新引擎版本所支援系列的參數群組 (此處的 neptune1.2)。
- 將 DBInstanceParameterGroupName 屬性設定為新引擎版本支援的系列中的參數群組。

範本應如下所示：

```
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
```

現在使用 AWS CloudFormation 來執行修訂的範本。

Neptune 中的資料庫複製

使用資料庫複製，您可以快速且經濟實惠地在 Amazon Neptune 中建立所有資料庫的複本。複製資料庫在第一次建立時只需最少的額外空間。資料庫複製使用寫入時複製通訊協定。無論在來源資料庫或複製資料庫上，在資料變更時會複製資料。您可以從相同的資料庫叢集建立多個複製。您也可以從其他複製建立額外的複製。如需複製時寫入通訊協定如何在 Neptune 儲存體內容中運作的詳細資訊，請參閱寫入時複製通訊協定。

您可以在各種使用案例中使用資料庫複製，尤其是您不想要對生產環境產生影響的使用案例，如下所示：

- 實驗並評估變更的影響，例如結構描述變更或參數群組變更。
- 執行工作負載密集的操作，例如匯出資料或執行分析查詢。
- 在非生產環境中建立生產資料庫叢集的副本，以進行開發或測試。

使用 AWS Management Console 建立資料庫叢集複本

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Instances (執行個體)。選擇您要建立其複製之資料庫叢集的主要執行個體。
3. 選擇 Instance actions (執行個體動作)，然後選擇 Create clone (建立複製)。
4. 在 Create Clone (建立複製) 頁面上，輸入複製資料庫叢集主要執行個體的名稱，做為 DB instance identifier (資料庫執行個體識別符)。

   如果想要的話，請對複製資料庫叢集設定任何其他設定。如需不同資料庫叢集設定的詳細資訊，請參閱 使用主控台啟動。
5. 選擇 Create Clone (建立複製) 來啟動複製資料庫叢集。

使用 AWS CLI 建立資料庫叢集複本

- 呼叫 Neptune restore-db-cluster-to-point-in-time AWS CLI 命令並提供下列值：
  - `--source-db-cluster-identifier` – 要建立其複製的來源資料庫叢集的名稱。
  - `--db-cluster-identifier` – 複製資料庫叢集的名稱。
  - `--restore-type copy-on-write` – `copy-on-write` 值指出應該建立複製資料庫叢集。
  - `--use-latest-restorable-time` – 這會指定應使用最新可還原的備份時間。

> **Note**
>
> restore-db-cluster-to-point-in-time AWS CLI 命令只會複製資料庫叢集，而不會還原該資料庫叢集的資料庫執行個體。

下列 Linux/UNIX 範例會從 `source-db-cluster-id` 資料庫叢集建立複本，並將複本命名為 `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
```

如果 \ 行尾逸出字元取代為 Windows ^ 對等字元，則相同的範例適用於 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
```

限制

Neptune 中的資料庫複製具有下列限制：

- 您無法跨 AWS 區域建立複製資料庫。複製資料庫必須建立在與來源資料庫相同的區域中。
- 複製的資料庫永遠都會使用 Neptune 引擎版本的最新修補程式，而此引擎版本是由從其複製的資料庫所使用。即使來源資料庫尚未升級至該修補程式版本，也是如此。但是，引擎版本本身不會改變。
- 目前，您最多可在 Neptune DB 叢集的每個複本進行 15 次複製，包括以其他複製為基礎的複製。在達到該限制之後，您必須製作另一個資料庫副本，而不是複製它。不過，如果您建立新副本，最多也可以有 15 個複製。
- 目前不支援跨帳戶資料庫複製。

- 您可以提供不同的 virtual private cloud (VPC) 進行複製。不過，那些 VPC 中的子網路必須映射至相同的可用區域集。

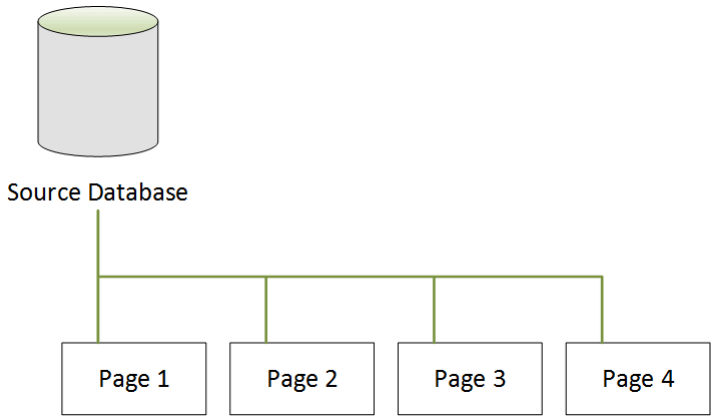
用於資料庫複製的寫入時複製通訊協定

下列範例說明寫入時複製協定的運作方式。

- 複製之前的 Neptune 資料庫
- 複製之後的 Neptune 資料庫
- 對來源資料庫進行變更時
- 對複製資料庫進行變更時

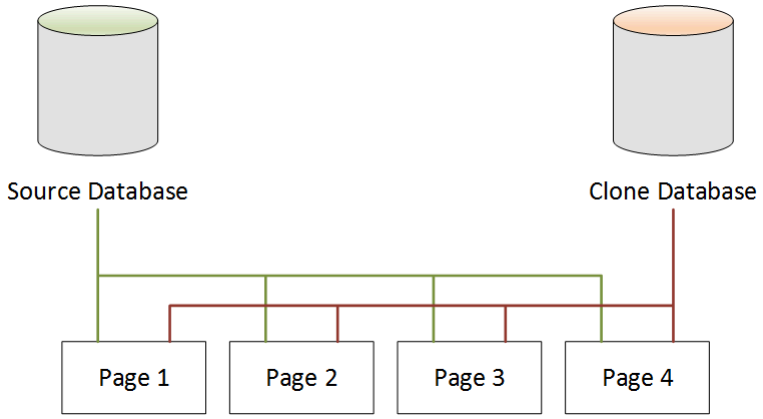
複製之前的 Neptune 資料庫

來源資料庫中的資料存放於頁面。在下圖中，來源資料庫有四個頁面。



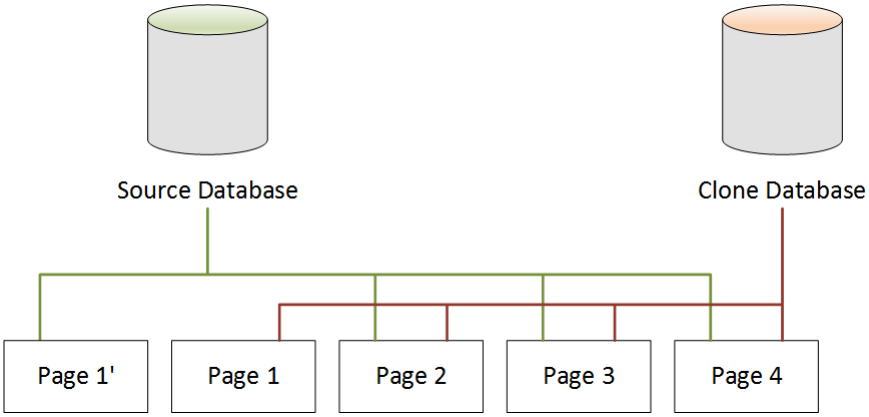
複製之後的 Neptune 資料庫

如下圖所示，在資料庫複製之後，來源資料庫中沒有任何變更。來源資料庫與複製資料庫都指向相同的四個頁面。不會實際複製任何頁面，所以不需要額外的儲存空間。



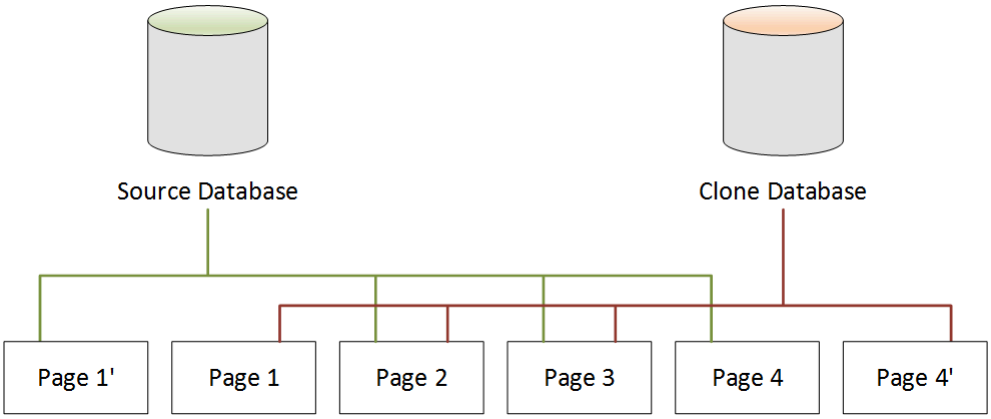
對來源資料庫進行變更時

在下列範例中，來源資料庫會對 Page 1 中的資料進行變更。系統不會寫入至原始 Page 1，而是使用額外儲存空間建立新的頁面，稱為 Page 1'。來源資料庫現在指向新的 Page 1'，同時也指向 Page 2、Page 3 和 Page 4。複製資料庫繼續指向 Page 1 到 Page 4。



對複製資料庫進行變更時

在下圖中，複製資料庫也做了變更，這次在 Page 4 中。不是寫入至原始 Page 4，而是使用額外儲存空間建立新的頁面，稱為 Page 4'。來源資料庫繼續指向 Page 1'，同時也指向 Page 2 到 Page 4，但複製資料庫現在會指向 Page 1 到 Page 3，同時也指向 Page 4'。



如第二個案例所示，在資料庫複製之後，在建立複製時不需要額外的儲存空間。不過，當來源資料庫與複製資料庫中發生變更時，只會建立已變更的頁面，如第三個及第四個案例所示。隨著來源資料庫與複製資料庫在一段時間後發生更多變更，您需要不斷增加更多的儲存空間來擷取並存放變更。

刪除來源資料庫

刪除來源資料庫不會影響與其相關聯的複製資料庫。複製資料庫會繼續指向來源資料庫先前所擁有的頁面。

管理 Amazon Neptune 執行個體

以下章節提供有關執行個體層級操作的資訊。

主題

Neptune T3 高載執行個體類別

除了固定效能執行個體類別 (例如 R5 和 R6) 外，Amazon Neptune 還可讓您選擇使用爆量效能 T3 執行個體。在開發圖形應用程式時，您想要資料庫快速且積極回應，但是您不需要一直使用它。Neptune 的 db.t3.medium 執行個體類別就是您在這種情況下應該使用的執行個體類別，成本遠低於最便宜的固定效能執行個體類別。

高載執行個體會在 CPU 效能的基準層級執行，直到工作負載需要更多，然後只要工作負載需要，就會在工作負載需要時高過基準。如果平均 CPU 使用率在 24 小時期間內未超過基準，則每小時價格會涵蓋高載。對於大多數開發和測試情況，這會轉換為低成本的良好性能。

如果您從 T3 執行個體類別開始，稍後可以在準備投入生產、使用 AWS Management Console、AWS CLI 或其中一個 AWS SDK 時，輕鬆切換到固定效能執行個體類別。

T3 高載由 CPU 點數管理

一個 CPU 點數代表一個虛擬 CPU 核心 (vCPU) 一分鐘的完整使用率。這也可以轉換成 vCPU 兩分鐘的 50% 使用率，或兩分鐘內的 25% 使用率，依此類推。

T3 執行個體會在閒置時產生 CPU 積分，並在作用中時使用積分，兩者均以毫秒為單位測量而得。db.t3.medium 執行個體類別有兩個 vCPU，每個 vCPU 閒置時，每小時獲得 12 個 CPU 信用點數。這表示每個 vCPU 的 20% 使用率會產生零 CPU 點數餘額。獲得的 12 個 CPU 點數支出 vCPU 的 20% 使用率 (因為 60 分鐘的 20% 也是 12)。因此，這 20% 的使用率就是 baseline (基準) 使用率，不會產生正或負的 CPU 點數餘額。

閒置時間 (CPU 使用率低於可用總數的 20%) 會導致 CPU 積分儲存在積分餘額儲存貯體中，db.t3.medium 執行個體類別的上限為 576 (24 小時內可產生的 CPU 積分上限，即 2 x 12 x 24)。超過這個限制，則會捨棄 CPU 積分。

必要時，即使在 CPU 積分餘額已低於零，CPU 使用率可以根據工作負載的需要，到爆量增加至 100%。如果執行個體連續 24 小時維持負餘額，則該期間內產生的每 -60 個 CPU 積分會收取 0.05 美元的額外費用。不過，對於大多數的開發和測試工作負載而言，高載通常是由高載前後的閒置時間所涵蓋。

> **Note**
>
> Neptune 的 T3 執行個體類別設定如 Amazon EC2 無限制模式。

使用 AWS Management Console 建立 T3 高載執行個體

在 AWS Management Console 中，您可以建立主要資料庫叢集執行個體，或使用執行個體類別 db.t3.medium 的讀取複本執行個體，或者您可以修改現有執行個體，以使用 db.t3.medium 執行個體類別。

例如，若要在 Neptune 主控台中建立新的資料庫叢集主要執行個體：

- 選擇 Create Database (建立資料庫)。
- 選擇等於或晚於 1.0.2.2 的資料庫引擎版本。
- 在 Purpose (目的) 下，選擇 Development and Testing (開發和測試)。
- 做為 DB instance class (資料庫執行個體類別)，接受預設值：db.t3.medium — 2 vCPU, 4 GiB RAM。

使用 AWS CLI 建立 T3 高載執行個體

你也可以使用 AWS CLI 來做同樣的事情：

```
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
```

修改 Neptune 資料庫執行個體 (並立即套用)

您可以將大多數的變更立即套用到 Amazon Neptune 資料庫執行個體，或是將它們延遲到下一次的維護時段。而諸如參數群組變更等部分修改作業，則要求您手動重新啟動資料庫執行個體，才能使變更生效。

> **Important**
>
> 如果 Neptune 必須重新啟動您的資料庫執行個體，才能套用變更，則修改會導致中斷。因此，請先檢視修改作業對資料庫與應用程式所造成的影響，再修改資料庫執行個體設定。

常用設定與停機時間影響

下表包含您可以變更的設定、套用變更的時間，以及變更是否會造成資料庫執行個體停機的詳細資訊。

資料庫執行個體設定	停機時間備註	
DB instance class (資料庫執行個體類別)	在此變更期間會發生中斷，無論是立即套用此變更，還是在下一個維護時段套用它都一樣。	
DB instance identifier (資料庫執行個體識別符) :	資料庫執行個體會重新啟動，並在此變更期間發生中斷，無論是立即套用此變更，還是在下一個維護時段套用它都一樣。	
Subnet group (子網路群組)	資料庫執行個體會重新啟動，並在此變更期間發生中斷，無論是立即套用此變更，還是在下一個維護時段套用它都一樣。	
安全群組	無論何時指定變更應發生，都會盡快以非同步方式套用變更，且不會造成中斷結果。	–

資料庫執行個體設定	停機時間備註
憑證授權單位	根據預設，當您指派新的憑證授權機構時，資料庫執行個體會重新啟動。
Database Port (資料庫連接埠)	變更永遠會立即發生，進而導致資料庫執行個體重新啟動，並發生中斷。
DB parameter group (資料庫參數群組)	變更此設定不會導致中斷。參數群組名稱本身會立即變更，但在您重新啓動實例而不需容錯移轉的情況下，才會套用實際參數變更。在此情況下，資料庫執行個體不會自動重新啟動，且參數變更也不會在下一個維護時段期間套用。但是，如果您修改新關聯的資料庫參數群組中的動態參數，則會立即套用這些變更，而不需重新開機。 如需更多詳細資訊，請參閱 在 Amazon Neptune 中重新啟動資料庫執行個體。
DB cluster parameter group (資料庫叢集參數群組)	資料庫參數群組名稱會立即變更。

資料庫執行個體設定	停機時間備註	
Backup retention period (備份保留期間)	如果您指定變更應立即發生，則此變更不會立即發生。否則，如果您將設定從非零值變更為另一個非零值，則系統會盡快以非同步的方式來套用變更。在下一次維護時段期間，任何其他變更也會發生。從零變更為非零值，或是從非零值變更為零時，將會發生中斷。	
稽核日誌	如果您想要透過 CloudWatch Logs 使用稽核記錄，請選取稽核日誌。您也必須將資料庫叢集參數群組中的 `neptune_enable_audit_log` 參數設定為 `enable` (1)，才能啟用稽核記錄。	
Auto minor version upgrade (自動次要版本升級)	如果您想要讓 Neptune 資料庫叢集可以自動接收可用的次要引擎版本升級，請選取啟用自動次要版本升級。 自動次要版本升級選項僅適用於 Amazon Neptune 資料庫叢集的次要引擎版本升級，不適用於為維護系統穩定性而套用的一般修補程式。	

重新命名 Neptune 資料庫執行個體

您可以使用 AWS Management Console 重新命名 Amazon Neptune 資料庫執行個體。重新命名資料庫執行個體可以發揮深遠的效果。以下是您在重新命名資料庫執行個體之前應該要先知道的事項清單。

- 重新命名資料庫執行個體時，資料庫執行個體的端點會隨之變更，因為 URL 包含您指派給資料庫執行個體的名稱。您應該將流量從舊的 URL 重新導向到新的 URL。
- 重新命名資料庫執行個體時，資料庫執行個體所用的舊 DNS 名稱會立即刪除，但其會保留在快取內幾分鐘。重新命名的資料庫執行個體的新 DNS 名稱會在大約 10 分鐘後生效。重新命名的資料庫執行個體必須等到新名稱生效後才可使用。
- 重新命名執行個體後，您將無法使用現有的資料庫執行個體名稱。
- 與資料庫執行個體相關聯的所有僅供讀取複本，在重新命名後仍會保持與該執行個體的關聯。例如，假設您有一個做為生產資料庫的資料庫執行個體，且該執行個體有多個關聯的僅供讀取複本。若您重新命名資料庫執行個體，接著在生產環境中將它替換成資料庫快照，則重新命名後的資料庫執行個體仍會擁有與其關聯的僅供讀取複本。
- 如果您重複使用資料庫執行個體名稱，與資料庫執行個體名稱關聯的指標和事件也會保留。例如，如果您提升僅供讀取複本，並將其重新命名為與先前主要執行個體相同的名稱，則與主要執行個體相關聯的事件和指標隨後會與重新命名後的執行個體相關聯。
- 資料庫執行個體標籤隨資料庫執行個體保留，無論是否重新命名。
- 重新命名的資料庫執行個體將保留其資料庫快照。

使用 Neptune 主控台重新命名資料庫執行個體

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇想要重新命名的資料庫執行個體旁邊的選項按鈕。
4. 在 Instance actions (執行個體動作) 功能表中，選擇 Modify (修改)。
5. 在 DB instance identifier (資料庫執行個體識別符) 文字方塊中，輸入新名稱。選擇 Apply immediately (立即套用)，然後選擇 Continue (繼續)。
6. 選擇 Modify DB instance (修改資料庫執行個體) 以完成變更。

在 Amazon Neptune 中重新啟動資料庫執行個體

在某些情況下，若您修改 Amazon Neptune 資料庫執行個體、變更與執行個體相關聯的資料庫參數群組，或是變更執行個體所使用參數群組中的靜態資料庫參數，您必須重新啟動執行個體，才能套用變更。

重新啟動資料庫執行個體，將重新啟動資料庫引擎服務。重新啟動也適用於關聯資料庫參數群組有任何變更正在等待處理的資料庫執行個體。重新啟動資料庫執行個體會暫時中斷執行個體，在此期間，資料庫執行個體狀態設定為重新啟動中。如果 Neptune 執行個體是針對多可用區域設定的，重新啟動可能會透過容錯移轉進行。重新啟動完成後將建立 Neptune 事件。

如果您的資料庫執行個體為異地同步備份部署，您可在選擇 Reboot (重新啟動) 選項時強制從某一可用區域容錯移轉到另一個區域。當您強制容錯移轉資料庫執行個體時，Neptune 會自動切換到位於另一個可用區域中的待命複本。它接著會更新資料庫執行個體的 DNS 記錄，指向待命資料庫執行個體。因此，您必須清除和重新建立資料庫執行個體任何現有的連線。

當您想要進行資料庫執行個體的故障模擬測試，或在發生故障後恢復操作到原始可用區域時，Reboot with failover (使用容錯移轉重新啟動) 是比較有利的。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的高可用性 (多可用區域)。重新啟動資料庫叢集時，它會容錯移轉到待命複本。重新啟動 Neptune 複本不會啟動容錯移轉。

重新啟動所需的時間為損毀復原程序的函數。若要改善重新啟動時間，建議在重新啟動程序期間盡可能減少資料庫活動，以降低傳輸中交易的轉返活動。

在主控台上，如果資料庫執行個體不是處於可用狀態，重新啟動選項可能停用。這可能由多種原因導致，例如，備份進行中、客戶正請求修改，或維護時段的動作。

> **Note**
>
> 在 版本：1.2.0.0 (2022 年 7 月 21 日) 之前，每當主要 (寫入器) 執行個體重新啟動時，資料庫叢集中的所有僅供讀取複本都會自動重新啟動。
> 從 版本：1.2.0.0 (2022 年 7 月 21 日) 開始，重新啟動主要執行個體並不會導致任何複本重新啟動。這表示如果您要變更叢集參數，必須個別重新啟動每個執行個體，才能接收參數變更 (請參閱 參數群組)。

使用 Neptune 主控台重新啟動資料庫執行個體

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。

2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇想要重新啟動的資料庫執行個體。
4. 選擇 Instance actions (執行個體動作)，然後選擇 Reboot (重新啟動)。
5. 若要強制從某一可用區域容錯移轉到另一可用區域，請由重新啟動資料庫執行個體對話方塊中選取重新啟動並容錯移轉？。
6. 選擇 Reboot (重新啟動)。若要取消重新啟動，請改為選擇 Cancel (取消)。

在 Amazon Neptune 中刪除資料庫執行個體

只要執行個體已啟動，且執行個體上的刪除保護停用，您隨時都能刪除任何狀態的 Amazon Neptune 資料庫執行個體。

如果已啟用刪除保護，您就無法刪除資料庫執行個體。

您只能刪除已停用刪除保護的資料庫執行個體。無論您使用主控台、AWS CLI 還是 API 來刪除資料庫執行個體時，Neptune 都會強制執行刪除保護。

當您使用AWS Management Console建立生產資料庫執行個體時，預設會啟用刪除保護。

如果您使用 AWS CLI 或 API 命令來建立資料庫執行個體，則預設會停用刪除保護。

如果要刪除未啟用刪除保護的資料庫執行個體，請先修改執行個體來將其 `DeletionProtection` 欄位設定成 `false`。

啟用或停用刪除保護並不會導致停機。

刪除資料庫執行個體之前為其建立最終快照

若要刪除資料庫執行個體，則必須指定執行個體的名稱，並決定您是否要為該執行個體擷取最終資料庫快照。如果要刪除的資料庫執行個體狀態為 Creating (正在建立)，便無法擷取最終資料庫快照。如果資料庫執行個體處於故障狀態，且狀態顯示為 failed (失敗)、incompatible-restore (不相容還原) 或 incompatible-network (不相容網路)，則僅有將 `SkipFinalSnapshot` 參數設定為 `true` 時，才能刪除該執行個體。

如果使用 AWS Management Console 刪除資料庫叢集內的所有 Neptune 資料庫執行個體，則會自動刪除整個資料庫叢集。若您使用的是 AWS CLI 或軟體開發套件，則刪除最後一個執行個體之後，您必須手動刪除資料庫叢集。

> **Important**
>
> 如果您刪除整個資料庫叢集，其所有自動備份都會同時刪除，且無法將其復原。這表示，除非您選擇手動建立最終資料庫快照，否則稍後無法將資料庫執行個體還原至其最終狀態。刪除叢集時，並不會刪除執行個體的手動快照。

如果您要刪除的資料庫執行個體具備僅供讀取複本，建議您提升僅供讀取複本或將其刪除。

無論有沒有包含最終資料庫快照，您都可以參考下方範例來刪除資料庫執行個體。

刪除不含最終快照的資料庫執行個體

如果您想要快速刪除資料庫執行個體，可以略過最終資料庫快照的建立步驟。刪除資料庫執行個體時，系統會一併刪除所有自動備份，且無法復原。系統並不會刪除手動快照。

使用 Neptune 主控台刪除不含最終資料庫快照的資料庫執行個體

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 在 Instances (執行個體) 清單中，選擇待刪除資料庫執行個體旁邊的選項按鈕。
4. 選擇 Instance actions (執行個體動作)，然後選擇 Delete (刪除)。
5. 在 Create final snapshot? (是否建立最終快照？) 方塊中，選擇 No (否)。
6. 選擇 Delete (刪除)。

刪除包含最終快照的資料庫執行個體

如果您希望稍後可以還原刪除的資料庫執行個體，則可以建立最終資料庫快照。系統會一併刪除所有自動備份，且無法復原。系統並不會刪除手動快照。

使用 Neptune 主控台刪除包含最終資料庫快照的資料庫執行個體

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 在 Instances (執行個體) 清單中，選擇待刪除資料庫執行個體旁邊的選項按鈕。
4. 選擇 Instance actions (執行個體動作)，然後選擇 Delete (刪除)。
5. 在 Create final snapshot? (是否建立最終快照？) 方塊中，選擇 Yes (是)。
6. 在 Final snapshot name (最終快照名稱) 方塊中，輸入最終資料庫快照的名稱。
7. 選擇 Delete (刪除)。

您可以檢查執行個體的運作狀態、判斷其執行個體類型、了解您目前安裝的引擎發行版本，並使用執行個體狀態 API 來取得有關執行個體的其他資訊。

Amazon Neptune Serverless

Amazon Neptune Serverless 是一種隨需自動擴展組態，其建構目的旨在視需要擴展您的資料庫叢集，以滿足處理需求的大幅增加，然後在需求減少時再次縮減規模。它有助於將監控工作負載和調整 Neptune 資料庫容量的程序自動化。因為容量會根據應用程式需求自動調整，所以您只需為應用程式實際需要的資源支付費用。

Neptune Serverless 的使用案例

Neptune Serverless 支援多種類型的工作負載。它適合於高要求、高度變化的工作負載，而且如果您的資料庫使用量通常在短時間內很大，然後是長時間的輕度活動或完全沒有活動，則它可能非常有幫助。Neptune Serverless 對於以下使用案例特別有用：

- 變數工作負載 – 在 CPU 活動中突然且無法預測地增加的工作負載。透過 Neptune Serverless，您的圖形資料庫會自動擴展容量以滿足工作負載的需求，並在活動激增結束時縮小規模。您不再需要針對峰值或平均容量進行佈建。您可以指定容量上限來處理尖峰工作負載，而且除非需要，否則不會使用該容量。

  Neptune Serverless 提供的擴展精細程度可協助您讓容量緊密地滿足工作負載的需求。Neptune Serverless 可以根據需要的內容以精細的增量新增或移除容量。只需要多一些容量時，它可以只增加一半的 Neptune 容量單位 (NCU)。
- 多租用戶應用程式 – 利用 Neptune Serverless，您可以為需要執行的每個應用程式建立個別的資料庫叢集，而不必個別管理這些租用戶叢集。每個租用戶叢集可能有不同的忙碌和閒置期間，取決於多個因素，但 Neptune Serverless 可以有效地擴展它們，而無需您的介入。
- 新的應用程式 – 當部署新的應用程式時，您通常不確定它將需要多少資料庫容量。使用 Neptune Serverless，您可以設定可自動擴展的資料庫叢集，以滿足新應用程式在開發時的容量需求。
- 容量規劃 – 假設您通常透過修改叢集中所有資料庫執行個體的資料庫執行個體類別，藉以調整資料庫容量，或驗證工作負載的最佳資料庫容量。使用 Neptune Serverless，您可以避免此管理負荷。相反地，您可以將現有的資料庫執行個體從佈建修改為無伺服器或從無伺服器修改為佈建，而不必建立新的資料庫叢集或執行個體。
- 開發與測試 – Neptune Serverless 也非常適合於開發和測試環境。使用 Neptune Serverless，您可以建立具有足夠大容量的資料庫執行個體，以測試要求最嚴苛的應用程式，並且在測試之間系統可能閒置的所有其他時間使用較低的最小容量。

Neptune Serverless 僅會擴展運算容量。您的儲存磁碟區保持不變，且不受無伺服器擴展的影響。

> **Note**
>
> 您也可以使用 Neptune 自動擴展搭配 Neptune Serverless，來處理不同種類的工作負載變化。

Amazon Neptune Serverless 限制條件

- 並非所有區域都可用 – Neptune Serverless 僅在下列區域可用：
  - 美國東部 (維吉尼亞北部)：`us-east-1`
  - 美國東部 (俄亥俄)：`us-east-2`
  - 美國西部 (加利佛尼亞北部)：`us-west-1`
  - 美國西部 (奧勒岡)：`us-west-2`
  - 加拿大 (中部)：`ca-central-1`
  - 歐洲 (斯德哥爾摩)：`eu-north-1`
  - 歐洲 (愛爾蘭)：`eu-west-1`
  - 歐洲 (倫敦)：`eu-west-2`
  - 歐洲 (法蘭克福)：`eu-central-1`
  - 亞太區域 (東京)：`ap-northeast-1`
  - 亞太區域 (新加坡)：`ap-southeast-1`
  - 亞太區域 (雪梨)：`ap-southeast-2`
- 無法在早期引擎版本中使用 – Neptune Serverless 僅能在引擎 1.2.0.1 版或更新版本中使用。
- 與 Neptune 查詢快取不相容 – 查閱快取不會使用無伺服器資料庫執行個體。
- 無伺服器執行個體中的記憶體上限為 256 GB – 將 `MaxCapacity` 設定為 128 NCU (支援的最高設定) 可讓 Neptune Serverless 執行個體擴展至 256 GB 的記憶體，這相當於 `R6g.8XL` 佈建執行個體類型的記憶體數量。

Neptune Serverless 資料庫叢集中的容量擴展

設定 Neptune Serverless 資料庫叢集類似於設定一般佈建的叢集，但有其他組態用於擴展的最小和最大單位，而執行個體類型設定為 `db.serverless`。擴展組態是在 Neptune 容量單位 (NCU) 中定義，每個單位都包含 2 GiB (gibibyte) 記憶體 (RAM)，以及相關聯的虛擬處理器容量 (vCPU) 和網路。它會設定為 `ServerlessV2ScalingConfiguration` 物件的一部分，以 JSON 表示，如下所示：

```
"ServerlessV2ScalingConfiguration": {
  "MinCapacity": (minimum NCUs, a floating-point number such as   1.0),
  "MaxCapacity": (maximum NCUs, a floating-point number such as 128.0)
}
```

在任何時候，每個 Neptune 寫入器或讀取器執行個體都有一個由浮點數測量的容量，代表該執行個體目前正在使用的 NCU 數目。您可以使用執行個體層級的 CloudWatch ServerlessDatabase容量指標，瞭解指定資料庫執行個體目前正在使用多少 NCU，以及 NCUUtilise 指標，找出執行個體使用的最大容量百分比。這兩個指標也可在資料庫叢集層級使用，以顯示整體資料庫叢集的平均資源使用率。

當您建立 Neptune Serverless 資料庫叢集時，可以為所有無伺服器執行個體設定 Neptune 容量單位 (NCU) 的數目下限和上限。

您指定的最小 NCU 值會設定資料庫叢集中無伺服器執行個體可縮小的最小大小，同樣地，最大 NCU 值會建立無伺服器執行個體可以增長的最大大小。您可以設定的最大 NCU 值最高為 128.0 NCU，而最小值最低為 1.0 NCU。

Neptune 會持續追蹤每個 Neptune Serverless 執行個體上的負載，方法是監控 CPU、記憶體和網路等資源的使用率。負載是由應用程式的資料庫操作、伺服器的背景處理，以及其他管理任務所產生。

當無伺服器執行個體上的負載達到目前容量的上限時，或是當 Neptune 偵測到任何其他效能問題時，執行個體就會自動縱向擴展。當執行個體上的負載下降時，容量會縮減至設定的最小容量單位，CPU 容量會比記憶體先釋放。這種架構允許以受控的逐步下降方式釋放資源，並有效地處理需求波動。

您可以使讀取器執行個體與寫入器執行個體一起擴展，或透過設定其提升層來獨立擴展。提升層 0 和 1 中的讀取器執行個體會與寫入器同時擴展，這會使它們的大小調整到適當的容量，以在發生容錯移轉時快速從寫入器處接管工作負載。提升層 2 到 15 中的讀取器與寫入器執行個體各自獨立擴展，而且讀取器彼此也獨立擴展。

如果您已將 Neptune 資料庫叢集建立為多可用區域叢集以確保高可用性，Neptune Serverless 會隨著資料庫負載而縱向擴展和縮減所有可用區域中的執行個體。您可以將次要可用區域中讀取器執行個體的提升層設定為 0 或 1，以便其隨著主要可用區域中寫入器執行個體的容量進行縱向擴展和縮減規模，隨時準備好接管目前的工作負載。

> **Note**
>
> Neptune 資料庫叢集的儲存體由您分散在三個可用區域中的所有資料的六個副本組成，不論您是否已將叢集建立為多可用區域叢集。儲存體複寫是由儲存子系統處理，不受 Neptune Serverless 影響。

為 Neptune Serverless 資料庫叢集選擇最小容量值

您可以針對最小容量設定的最小值為 1.0 NCU。

確定不要將最小值設定為低於應用程式有效操作所需的值。將其設定太低可能會在特定記憶體密集型工作負載中造成更高的逾時率。

將最小值設定為盡可能低可以節省成本，因為當需求低時，您的叢集將使用最少的資源。不過，如果您的工作負載往往會大幅波動 (從非常低到非常高)，您可能想要設定較高的最小值，因為較高的最小值可讓 Neptune Serverless 執行個體更快地縱向擴展。

原因是 Neptune 會根據目前的容量選擇擴展增量。如果目前容量很低，Neptune 一開始就會緩慢地縱向擴展。如果最小值較高，Neptune 會從較大的擴展增量開始，因此可以更快地縱向擴展，以因應工作負載大幅突然增加的情況。

為 Neptune Serverless 資料庫叢集選擇最大容量值

您可以針對最大容量設定的最大值是 128.0 NCU，而您可以針對為最大容量設定的最小值為 2.5 NCU。無論您設定的最大容量值為何，它都必須至少與您設定的最小容量值相同。

一般規則是設定足夠高的最大值，以處理應用程式可能會遇到的尖峰負載。將其設定太低可能會在特定記憶體密集型工作負載中造成更高的逾時率。

將最大值設定為盡可能高，其優點是即使是最意外的工作負載，您的應用程式也許都能夠處理。缺點是您會失去一些預測和控制資源成本的能力。非預期的需求激增最終可能是成本遠遠超出您預算所預期的。

精心設定最大值的好處在於，它可讓您滿足尖峰需求，同時也限制 Neptune 運算成本。

> **Note**
>
> 變更 Neptune Serverless 資料庫叢集的容量範圍會造成某些組態參數的預設值發生變更。Neptune 可以立即套用其中某些新的預設值，但某些動態參數變更僅在重新啟動之後才會生效。pending-reboot 狀態表示需要重新啟動才能套用某些參數的變更。

使用現有的組態來估計無伺服器需求

如果您通常會修改所佈建資料庫執行個體的資料庫執行個體類別，以滿足異常高或低的工作負載，則可以使用該體驗粗略估計相等的 Neptune Serverless 容量範圍。

估計最佳的最小容量設定

您可以套用您對現有 Neptune 資料庫叢集了解的內容，以估計最適合的無伺服器最小容量設定。

例如，如果您佈建的工作負載對於 T3 或 T4g 等小型資料庫執行個體類別的記憶體要求過高，請選擇提供與 R5 或 R6g 資料庫執行個體類別相當之記憶體的最小 ACU 設定。

或者，假設您在叢集具有低工作負載時，使用 db.r6g.xlarge 資料庫執行個體類別。該資料庫執行個體類別具有 32 GiB 的記憶體，因此您可以將最小 NCU 設定指定為 16，來建立可以縮減規模至與該容量大約相同的無伺服器執行個體 (每個 NCU 對應於大約 2 GiB 的記憶體)。如果您的 db.r6g.xlarge 執行個體有時並未充分利用，則您也許能夠指定較低的值。

如果您的資料庫執行個體可在記憶體或緩衝區快取中保留給定數量的資料時，您的應用程式運作效率最高，請考慮指定大到足以為此提供足夠記憶體的最小 NCU 設定。否則，當無伺服器執行個體縮減規模時，資料可能會從緩衝區快取中移出，而且一段時間後在執行個體再次縱向擴展時，必須將資料讀回緩衝區快取。如果要將資料帶回緩衝區快取的 I/O 量很大，則選擇較高的最小 ACU 值可能值得一試。

如果您發現無伺服器執行個體大部分時間都以特定容量執行，則將最小容量設定為略低於該容量，這會有很好的效果。當目前容量並未明顯低於所需容量時，Neptune Serverless 可最有效地估計縱向擴展的數量和速度。

在混合組態 (具有佈建的寫入器和 Neptune Serverless 讀取器) 中，讀取器不會與寫入器一起擴展。因為它們獨立擴展，所以針對其設定較低的最小容量可能會導致過長的複寫延遲。當有高度寫入密集型工作負載時，它們可能沒有足夠的容量來跟上寫入器所做的變更。在此情況下，請設定與寫入器容量相當的最小容量。尤其，如果您在位於提升層 2 至 15 的讀取器中觀察到複本延遲，請增加叢集的最小容量設定。

估計最佳的最大容量設定

您也可以套用您對現有 Neptune 資料庫叢集了解的內容，以估計最適合的無伺服器最大容量設定。

例如，假設您在叢集具有高工作負載時，使用 db.r6g.4xlarge 資料庫執行個體類別。該資料庫執行個體類別具有 128 GiB 的記憶體，因此您可以將最大 NCU 設定指定為 64，以設定相等的 Neptune Serverless 執行個體 (每個 NCU 對應至大約 2 GiB 的記憶體)。如果您的 db.r6g.4xlarge 執行個體無法一直處理工作負載，則您可指定一個較高的值，讓資料庫執行個體進一步縱向擴展。

如果工作負載出現非預期的激增情況很少見，則設定足夠高的最大容量設定，即使在這些激增期間也能維持應用程式效能，這可能有意義。另一方面，您可能想要設定較低的最大容量，可在異常激增期間減少輸送量，但這可讓 Neptune 毫無問題地處理預期的工作負載，並限制成本。

Neptune Serverless 資料庫叢集和執行個體的其他組態

除了針對 Neptune Serverless 資料庫叢集設定最小和最大容量之外，還有其他幾個需要考慮的組態選項。

在資料庫叢集中結合無伺服器與佈建的執行個體

資料庫叢集不一定只是無伺服器 - 您可以建立無伺服器執行個體與佈建執行個體的組合 (混合組態)。

例如，假設您需要的寫入容量超過無伺服器執行個體中所能提供的。在此情況下，您可以使用常大的佈建寫入器設定叢集，並仍對讀取器使用無伺服器執行個體。

或者，假設叢集上的寫入工作負載有變化，但讀取工作負載穩定。在此情況下，您可以使用無伺服器寫入器和一個或多個佈建讀取器來設定叢集。

如需如何建立混合組態資料庫叢集的相關資訊，請參閱 使用 Amazon Neptune Serverless。

設定 Neptune Serverless 執行個體的提升層

對於包含多個無伺服器執行個體或佈建和無伺服器執行個體混合的叢集，請注意每個無伺服器執行個體的提升層。比起佈建的資料庫執行個體，此設定控制無伺服器執行個體更多的行為。

在中 AWS Management Console，您可以使用 [建立資料庫]、[修改執行個體] 和 [新增讀取器] 頁面的 [其他組態] 下的 [容錯移轉優先順序] 指定 您可以在資料庫頁面的選用優先順序方案欄中看到現有執行個體的這個屬性。您也可以在資料庫叢集或執行個體的詳細資訊頁面上查看此屬性。

對於佈建的執行個體，選擇第 0–15 層僅決定 Neptune 在容錯移轉操作期間選擇將讀取器執行個體提升為寫入器的順序。對於 Neptune Serverless 讀取器執行個體，層數也會決定執行個體縱向擴展以符合寫入器執行個體的容量，還是僅根據其自己的工作負載獨立擴展。

第 0 層或第 1 層的 Neptune Serverless 讀取器執行個體保持至少與寫入器執行個體一樣高的最小容量，以便它們準備好在容錯移轉時從寫入器接管。如果寫入器是佈建的執行個體，Neptune 會估計相等的無伺服器容量，並使用該估計值做為無伺服器讀取器執行個體的最小容量。

第 2–15 層中的 Neptune Serverless 讀取器執行個體對其最小容量沒有相同的限制條件，並與寫入器各自獨立擴展。當它們處於閒置狀態時，可以縮減規模到叢集容量範圍中指定的最小 NCU 值。不過，如果讀取工作負載快速激增，這可能會導致問題。

保持讀取器容量與寫入器容量一致

要記住的一件重要事項是，您要確保您的讀取器執行個體可以跟上您的寫入器執行個體，以防止過長的複寫延遲。在兩種情況下，無伺服器讀取器執行個體不會自動與寫入器執行個體同步縮減，這是特別需要關注的問題：

- 當佈建您的寫入器，且您的讀取器是無伺服器時。
- 當您的寫入器是無伺服器，且您的無伺服器讀取器位於提升層 2-15 時。

在這兩種情況下，請將最小無伺服器容量設定為符合預期的寫入器容量，以確保讀取器操作不會逾時且可能導致重新啟動。在佈建的寫入器執行個體的情況下，請設定最小容量以符合佈建執行個體的容量。在無伺服器寫入器的情況下，最佳設定可能較難預測。

由於執行個體容量範圍是在叢集層級設定，因此所有無伺服器執行個體都是由相同的最小和最大容量設定控制。第 0 層和第 1 層中的讀取器執行個體會與寫入器執行個體同步縮減，但提升層 2-15 中的執行個體彼此獨立擴展，且與寫入器執行個體也是各自獨立擴展，取決於它們的工作負載。如果您將最小容量設得太低，第 2 層到第 15 層中的閒置執行個體可能縮減規模後太低，以致無法足夠快速縱向擴展來處理寫入器活動中突然激增的情況。

避免將逾時值設得太高

如果您在無伺服器執行個體上將查詢逾時值設得太高，可能會產生非預期的成本。

若沒有合理的逾時設定，您可能會不小心發出一個查詢，需要功能強大且昂貴的執行個體類型，且會持續執行很長時間，從而產生您從未預期的成本。您可以使用查詢逾時值來避免此情況，而這些逾時值滿足大部分的查詢，且只會導致執行時間異常長的查詢逾時。

對於使用參數設定的一般查詢逾時值，以及使用查詢提示設定的每個查詢逾時值，都是如此。

最佳化您的 Neptune Serverless 組態

如果 Neptune Serverless 資料庫叢集未針對其正在執行的工作負載進行調校，您可能會注意到它並未以最佳方式執行。您可以調整最小和/或最大容量設定，以便其可在沒有遇到記憶體問題的情況下進行擴展。

- 增加叢集的最小容量設定。這可以更正下列情況：閒置執行個體縮減至其中記憶體少於應用程式和已啟用功能所需的容量。
- 增加叢集的最大容量設定。這可以更正下列情況：忙碌的資料庫無法縱向擴展至具有足夠記憶體的容量，以處理工作負載和任何已啟用的記憶體密集型功能。

- 變更有問題執行個體上的工作負載。例如，您可以將讀取器執行個體新增至叢集，以將讀取負載分散至更多的執行個體中。
- 調校應用程式的查詢，使其使用更少的資源。
- 嘗試使用佈建的執行個體，其大於 Neptune Serverless 內可用的最大 NCU，以查看它是否更符合工作負載的記憶體和 CPU 需求。

使用 Amazon Neptune Serverless

您可以將新的 Neptune 資料庫叢集建立為無伺服器資料庫叢集，或在某些情況下，您可以轉換現有的資料庫叢集以使用無伺服器。您也可以在無伺服器資料庫叢集中的資料庫執行個體與無伺服器執行個體之間進行來回轉換。您只能在支援 Neptune 無伺服器的其中一個 AWS 區域 位置使用，但有一些其他限制 (請參閱Amazon Neptune Serverless 限制條件)。

您也可以使用 Neptune AWS CloudFormation 堆疊來建立 Neptune Serverless 資料庫叢集。

建立一個使用無伺服器的新資料庫叢集

若要建立一個使用無伺服器的 Neptune 資料庫叢集，您可以透過您建立一個佈建叢集所用的同一方式使用 AWS Management Console來執行此操作。不同之處在於，您需要在資料庫執行個體大小下將資料庫執行個體類別設定為無伺服器。當您這麼做時，需要針對叢集設定無伺服器容量範圍。

您也可以使用像這樣的命令來創建無服務器數據庫集群（在 Windows 上，用 '^' 替換 '\'）：AWS CLI

```
aws neptune create-db-cluster \
  --region (an 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"
```

您也可以指定 `serverless-v2-scaling-configuration` 參數，如下所示：

```
  --serverless-v2-scaling-configuration '{"MinCapacity":1.0, "MaxCapacity":128.0}'
```

然後，您可以執行 `ServerlessV2ScalingConfiguration` 屬性的 `describe-db-clusters` 命令，其應會傳回您指定的容量範圍設定：

```
"ServerlessV2ScalingConfiguration": {
```

```
    "MinCapacity": (the specified minimum number of NCUs),
    "MaxCapacity": (the specified maximum number of NCUs)
}
```

將現有的資料庫叢集或執行個體轉換為無伺服器

如果您具有正在使用引擎 1.2.0.1 版或更新版本的 Neptune 資料庫叢集，則可以將其轉換為無伺服器。此過程確實會產生一些停機時間。

第一步是將容量範圍新增至現有叢集。您可以使用 AWS Management Console，或者使用這樣的 AWS CLI 命令來執行此操作（在 Windows 上，將 '\' 替換為 '^'）：

```
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)
```

下一步是建立新的無伺服器資料庫執行個體，以取代叢集中現有的主要執行個體 (寫入器)。同樣地，您可以使用 AWS Management Console 或執行此操作以及所有後續步驟 AWS CLI。在任一情況下，請將資料庫執行個體類別指定為無伺服器。該 AWS CLI 命令看起來像這樣（在 Windows 上，用 '^' 替換 '\'）：

```
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
```

當新的寫入器執行個體變成可用時，請執行容錯移轉，使其成為叢集的寫入器執行個體：

```
aws neptune failover-db-cluster \
  --db-cluster-identifier (ID of the DB cluster) \
  --target-db-instance-identifier (instance ID of the new serverless instance)
```

接下來，刪除舊的寫入器執行個體：

```
aws neptune delete-db-instance \
  --db-instance-identifier (instance ID of the old writer instance) \
```

```
  --skip-final-snapshot
```

最後，執行相同的操作來建立新的無伺服器執行個體，以取代您想要轉換為無伺服器執行個體的每個現有的佈建讀取器執行個體，並刪除現有的佈建執行個體 (讀取器執行個體不需要容錯移轉)。

修改現有無伺服器資料庫叢集的容量範圍

您可以使用類似這樣的 AWS CLI 變更 Neptune Serverless 資料庫叢集的容量範圍 (在 Windows 上，將「\」取代為「^」)：

```
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
```

變更容量範圍會造成某些組態參數的預設值產生變更。Neptune 可以立即套用其中某些新的預設值，但某些動態參數變更僅在重新啟動之後才會生效。狀態 `pending-reboot` 表示需要重新啟動才能套用某些參數的變更。

將無伺服器資料庫執行個體變更為佈建

若要將 Neptune Serverless 執行個體轉換為佈建的執行個體，您只需將其執行個體類別變更為其中一個佈建的執行個體類別即可。請參閱修改 Neptune 資料庫執行個體 (並立即套用)。

使用 Amazon 監控無伺服器容量 CloudWatch

您可以用 CloudWatch 來監視資料庫叢集中 Neptune 無伺服器執行個體的容量和使用率。有兩個 CloudWatch 指標可讓您在叢集層級和執行個體層級追蹤目前的無伺服器容量：

- **`ServerlessDatabaseCapacity`** – 作為執行個體層級指標，`ServerlessDatabaseCapacity` 會報告目前的執行個體容量，以 NCU 為單位。作為叢集層級指標，其會報告叢集中所有資料庫執行個體的所有 `ServerlessDatabaseCapacity` 值的平均值。
- **`NCUUtilization`** – 此指標報告可能使用的容量百分比。其計算方式為將目前的 `ServerlessDatabaseCapacity` (在執行個體層級或叢集層級) 除以資料庫叢集的最大容量設定。

  如果此指標在叢集層級接近 100%，表示叢集已盡可能高地擴展，請考慮增加最大容量設定。

  如果對於讀取器執行個體，它接近 100%，而寫入器執行個體未接近最大容量，請考慮新增更多的讀取器執行個體來分散讀取工作負載。

請注意，CPUUtilization 和 FreeableMemory 指標對無伺服器執行個體的意義與對佈建執行個體的意義略有不同。在無伺服器內容中，CPUUtilization 是百分比，其計算方法為目前使用的 CPU 數量除以最大容量可用的 CPU 數量。同樣地，FreeableMemory 會報告執行個體達到最大容量時將可用的可用記憶體數量。

下列範例顯示如何使用 Linux AWS CLI 上的擷取指定資料庫執行個體的最小、最大和平均容量值 (在一小時內每 10 分鐘測量一次)。Linux date 命令指定相對於目前日期和時間的開始和結束時間。--query 參數中的 sort_by 函數根據 Timestamp 欄位依時間順序對結果進行排序。

```
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
```

使用 Neptune 串流即時擷取圖形變更

Neptune 串流會以全受管方式依圖形發生變更的順序，記錄每一筆變更。一旦啟用串流，Neptune 就會負責可用性、備份、安全性和過期。

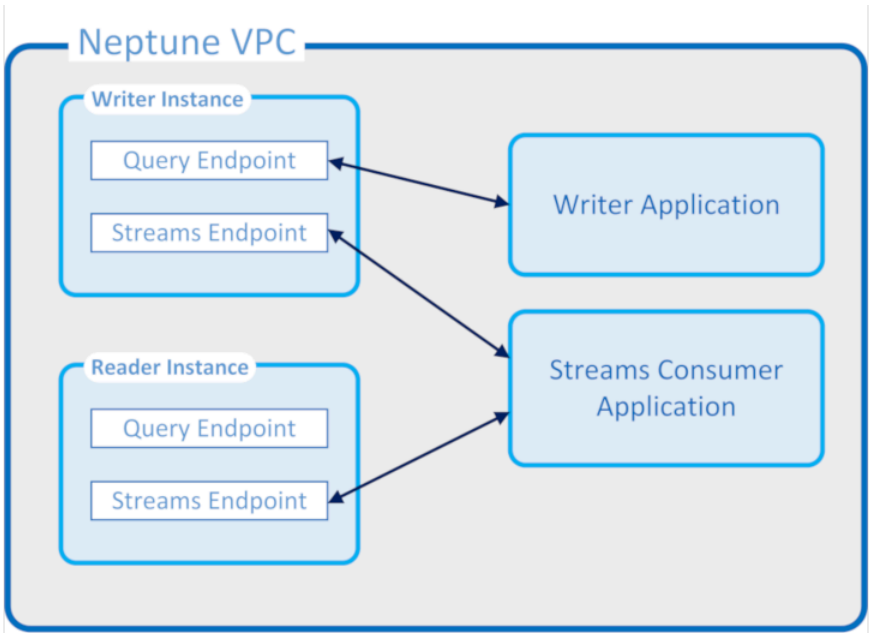
> **Note**
>
> 此功能從 版本 1.0.1.0.200463.0 (2019 年 10 月 15 日) 開始可在實驗室模式下使用，並且從 Neptune 引擎版本 1.0.2.2.R2 開始可用於生產用途。

以下是您可能想要在圖形發生變更時擷取其變更的一些使用案例：

- 您可能想要應用程式在進行特定變更時自動通知人員。
- 您可能想要在另一個資料存放區中維護圖形資料的最新版本，例如 Amazon OpenSearch 服務 ElastiCache、Amazon 或亞馬遜簡單儲存服務 (Amazon S3)。

Neptune 會針對變更日誌串流使用與圖形資料相同的原生儲存體。它會同步寫入變更日誌項目與進行這些變更的交易。您可以使用 HTTP REST API，從日誌串流擷取這些變更記錄。(如需相關資訊，請參閱呼叫串流 API。)

下圖顯示如何從 Neptune 串流擷取變更日誌資料。



Neptune 串流保證

- 交易一完成時，就可以立即從寫入器和讀取器二者讀取交易所做的變更 (除了讀取器中的任何正常複寫延遲外)。
- 變更記錄會嚴格地依其發生的順序顯示 (這包括交易內所做的變更)。
- 變更串流不包含重複項目。每個變更只會記錄一次。
- 變更串流已完成。未遺失或省略任何變更。
- 變更串流包含判斷資料庫本身在任何時間點是否狀態完整所需的所有資訊，前提是起始狀態已知。
- 串流可以隨時開啟或關閉。

Neptune 串流操作屬性

- 變更日誌串流是全受管的。
- 變更日誌資料會以同步方式寫入，做為進行變更之相同交易的一部分。
- 啟用 Neptune 串流時，您會產生與變更日誌資料相關聯的 I/O 和儲存費用。
- 根據預設，變更記錄會在建立後一週自動清除。從引擎 1.2.0.0 版開始，此保留期可以使用 neptune_streams_expiry_days 資料庫叢集參數變更為 1 與 90 之間的任意天數。
- 串流上的讀取效能會隨執行個體擴展。
- 您可以使用僅供讀取複本實現高可用性和讀取輸送量。您可以同時建立和使用的串流讀取器數量沒有限制。
- 變更日誌資料可跨多個可用區域進行複寫，使其具有高耐用性。
- 日誌資料與您的圖形資料本身一樣安全。它可以在靜態和傳輸中加密。可以使用 IAM、Amazon VPC 和 AWS Key Management Service (AWS KMS) 來控制存取權限。就像圖表資料一樣，它可以進行備份，並在稍後使用 point-in-time 還原 (PITR) 進行還原。
- 隨著每個交易同步寫入串流資料，會導致整體寫入效能略微降低。
- 串流資料不會碎片化，因為 Neptune 的設計是單一碎片化。
- 日誌串流 `GetRecords` API 使用的資源與所有其他 Neptune 圖形操作相同。這表示用戶端需要在串流請求和其他資料庫請求之間進行負載平衡。
- 串流停用時，所有日誌資料都會立即變成無法存取。這表示您必須先讀取您感興趣的所有日誌資料，然後才能停用記錄。
- 目前沒有與 AWS Lambda. 日誌串流不會產生可觸發 Lambda 函數的事件。

主題

使用 Neptune 串流

使用 Neptune 串流功能，您可以產生一系列完整的變更日誌項目，記錄對圖形資料所做的每項變更。如需此功能的概觀，請參閱 使用 Neptune 串流即時擷取圖形變更。

主題

啟用 Neptune 串流

您可以隨時設定 `neptune_streams` 資料庫叢集參數，以啟用或停用 Neptune 串流。將參數設為 `1` 可啟用串流，設為 `0` 則可停用串流。

> **Note**
>
> 在變更 `neptune_streams` 資料庫叢集參數之後，您必須重新啟動叢集中的所有資料庫執行個體，變更才會生效。

您可以設定 `neptune_streams_expiry_days` 資料庫叢集參數，以控制串流記錄在刪除之前保留在伺服器上的天數 (從 1 到 90)。預設值為 7。

Neptune Streams 最初是以實驗性功能形式引入，您可以使用資料庫叢集 `neptune_lab_mode` 參數在實驗室模式下啟用或停用該功能 (請參閱 Neptune 實驗室模式)。使用實驗室模式啟用串流現在已遭取代，且未來將停用。

停用 Neptune 串流

您可以隨時關閉執行中的 Neptune 串流。

若要關閉串流，請更新資料庫叢集參數群組，將 neptune_streams 參數的值設為 0。

> **Important**
> 一旦串流關閉，您再也無法存取變更日誌資料。在關閉串流之前，請務必閱讀您感興趣的內容。

呼叫 Neptune 串流 REST API

您可以使用 REST API 存取 Neptune 串流，將 HTTP GET 請求傳送至下列其中一個本機端點：

- 若為 SPARQL 圖形資料庫：`https://Neptune-DNS:8182/sparql/stream`。
- 對於 Gremlin 或 openCypher 圖形資料庫：`https://Neptune-DNS:8182/propertygraph/stream` 或 `https://Neptune-DNS:8182/pg/stream`。

> **Note**
> 從引擎 1.1.0.0 版開始，Gramlin 串流端點 (`https://Neptune-DNS:8182/gremlin/stream`) 正被棄用，以及其相關聯的輸出格式 (GREMLIN_JSON) 也一樣。基於回溯相容性仍支援它，但可能會在未來的版本中將其移除。

只允許 HTTP GET 操作。

Neptune 支援回應的 gzip 壓縮，前提是 HTTP 請求包含 Accept-Encoding 標頭，其中將 gzip 指定為接受的壓縮格式 (也就是 "Accept-Encoding: gzip")。

參數

- limit – 長整數 (選用)。範圍：1–100,000。預設值：10

  指定要傳回的記錄數上限。回應也有無法修改的 10 MB 大小限制，而且會優先於 limit 參數中指定的記錄數量。如果達到 10 MB 限制，回應會包含違反閾值的記錄。

- iteratorType – 字串 (選用)。

  此參數可採用下列其中一個值：

  - AT_SEQUENCE_NUMBER (預設值) – 指出讀取應該從 commitNum 和 opNum 參數共同指定的事件序號開始。
  - AFTER_SEQUENCE_NUMBER – 指出讀取應該在 commitNum 和 opNum 參數共同指定的事件序號之後立即開始。
  - TRIM_HORIZON – 指出讀取應該從系統中的最後一個未修整記錄開始，這是變更日誌串流中最舊的未過期 (尚未刪除) 記錄。當您沒有特定的開始事件序號時，此模式在應用程式啟動期間很有用。
  - LATEST – 指出讀取應該從系統中的最新記錄開始，這是變更日誌串流中最新的未過期 (尚未刪除) 記錄。當需要從串流的目前頂端讀取記錄，以免處理較舊的記錄 (例如在災難復原或零停機時間升級期間) 時，這會很有用。請注意，在此模式下，最多只會傳回一筆記錄。
- commitNum – 長整數，當 iteratorType 為 AT_SEQUENCE_NUMBER 或 AFTER_SEQUENCE_NUMBER 時為必要。

  要從 change-log 串流讀取之開始記錄的遞交編號。

  當 iteratorType 為 TRIM_HORIZON 或 LATEST 時會忽略此參數。
- opNum – 長整數，選用 (預設為 1)。

  所指定遞交內的操作序號，即在變更日誌串流資料中開始讀取之處。

變更 SPARQL 圖形資料的操作通常只會為每個操作產生單一變更記錄。不過，變更 Gremlin 圖形資料的操作可能會為每個操作產生多個變更記錄，如以下範例所示：

- INSERT – Gremlin 頂點可以有多個標籤，而且 Gremlin 元素可以有多個屬性。插入元素時，會為每個標籤和屬性產生個別的變更記錄。
- UPDATE – 變更 Gremlin 元素屬性時，會產生兩個變更記錄：第一個用於移除先前的值，第二個用於插入新值。
- DELETE – 系統會針對刪除的每個元素屬性產生個別的變更記錄。例如，刪除具有屬性的 Gremlin 邊緣時，每個屬性都會產生一個變更記錄，之後會產生一個用於刪除邊緣標籤的變更記錄。

  刪除 Gremlin 頂點時，會先刪除所有傳入和傳出邊緣屬性，接著依序刪除邊緣標籤、頂點屬性，最後再刪除頂點標籤。其中每個刪除都會產生一筆變更記錄。

Neptune 串流 API 回應格式

Neptune 串流 REST API 請求的回應具有下列欄位：

- `lastEventId` – 串流回應中上次變更的序列識別符。事件 ID 由兩個欄位組成：`commitNum` 識別已變更圖形的交易，以及 opNum 識別該交易內的特定操作。如以下範例所示。

```
  "eventId": {
    "commitNum": 12,
    "opNum": 1
  }
```

- `lastTrxTimestamp` – 已請求遞交交易的時間，以毫秒為單位，從 Unix epoch 開始。
- `format` – 要傳回之變更記錄的序列化格式。Gremlin 或 openCypher 變更記錄的可能值是 `PG_JSON`，而 SPARQL 變更記錄的可能值是 `NQUADS`。
- `records` – 回應中包含之序列化變更日誌串流記錄的陣列。`records` 陣列中的每筆記錄都包含下列欄位：
  - `commitTimestamp` – 已請求遞交交易的時間，以毫秒為單位，從 Unix epoch 開始。
  - `eventId` – 串流變更記錄的序列識別符。
  - `data`— 序列化的小鬼、SPARQL 或變更記錄。 OpenCypher 下節 (Neptune 串流中的序列化格式) 會詳細描述每筆記錄的序列化格式。
  - `op` – 已建立變更的操作。
  - `isLastOp` – 僅當此操作是其交易中的最後一個操作時才存在。存在時，其會設定為 `true`。有助於確保取用整個交易。
- `totalRecords` – 回應中的記錄總數。

例如，下列回應會針對包含多項操作的交易傳回 Gremlin 變更資料：

```
{
  "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
}
```

下列回應會傳回交易中最後一個操作的 SPARQL 變更資料 (交易編號 97 中 `EventId(97, 1)` 所識別的操作)。

```
{
  "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
}
```

Neptune 串流 API 例外狀況

下表描述 Neptune 串流例外狀況。

錯誤程式碼	HTTP 代碼	OK to Retry? (確定要重試嗎？)	訊息
`InvalidParameterException`	400	否	提供了無效的或 out-of-range 值作為輸入參數。
`ExpiredStreamException`	400	否	所有請求的記錄都超過允許的最大存留期且已過期。
`ThrottlingException`	500	是	請求速度超出傳輸量上限。
`StreamRecordsNotFoundException`	404	否	找不到請求的資源。可能無法正確指定串流。
`MemoryLimitExceededException`	500	是	由於缺乏記憶體，請求處理未成功，但可以在伺服器較不忙碌時重試。

Neptune 串流中的序列化格式

Amazon Neptune 會使用兩種不同格式，將圖形變更資料序列化至日誌串流，取決於是使用 Gremlin 還是 SPARQL 建立圖形。

這兩種格式共用通用的記錄序列化格式 (如 Neptune 串流 API 回應格式 中所述)，其中包含下列欄位：

- `commitTimestamp` – 已請求遞交交易的時間，以毫秒為單位，從 Unix epoch 開始。

- `eventId` – 串流變更記錄的序列識別符。
- `data`— 序列化的小鬼、SPARQL 或變更記錄。 OpenCypher 以下幾節會詳細描述每筆記錄的序列化格式。
- `op` – 已建立變更的操作。

主題

- PG_JSON 變更序列化格式
- SPARQL NQUADS 變更序列化格式

PG_JSON 變更序列化格式

> **Note**
>
> 從引擎 1.1.0.0 版開始，Gramlin 串流端點 (`https://Neptune-DNS:8182/gremlin/stream`) 輸出的 Gremlin 串流輸出格式 (`GREMLIN_JSON`) 正被棄用。它被 PG_JSON 取代，目前與 `GREMLIN_JSON` 相同。

Gremlin 或 openCypher 變更記錄 (包含在日誌串流回應的 `data` 欄位中) 具有下列欄位：

- `id` – 字串 (必要)。

  Gremlin 或 openCypher 元素的 ID。

- `type` – 字串 (必要)。

  此 Gremlin 或 openCypher 元素的類型。必須是下列其中之一：

  - `vl` - Gremlin 的頂點標籤；openCypher 的節點標籤。
  - `vp` - Gremlin 的頂點屬性；openCypher 的節點屬性。
  - `e` - Gremlin 的邊緣和邊緣標籤；OpenCypher 的關係和關係類型。
  - `ep` - Gremlin 的邊緣屬性；openCypher 的關係屬性。

- `key` – 字串 (必要)。

  屬性名稱 對於元素標籤，這是「標籤」。

- `value` – `value` 物件 (必要)。

這是 JSON 物件，其中包含值本身的 `value` 欄位，以及該值之 JSON 資料類型的 `datatype` 欄位。

```
  "value": {
    "value": "the new value",
    "dataType": "the JSON datatype of the new value"
  }
```

- `from` – 字串 (選用)。

  如果這是邊緣 (type="e")，則為對應「來源」頂點或來源節點的 ID。

- `to` – 字串 (選用)。

  如果這是邊緣 (type="e")，則為對應「目標」頂點或目標節點的 ID。

Gremlin 範例

- 以下是 Gremlin 頂點標籤的範例。

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the vertex label",
    "dataType": "String"
  }
}
```

- 以下是 Gremlin 頂點屬性的範例。

```
{
  "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"
  }
}
```

- 以下是 Gremlin 邊緣的範例。

```
{
  "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"
}
```

openCypher 範例

- 以下是 openCypher 節點標籤的範例。

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the node label",
    "dataType": "String"
  }
}
```

- 以下是 openCypher 節點屬性的範例。

```
{
  "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"
  }
}
```

- 以下是 openCypher 關係的範例。

```
{
  "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"
}
```

SPARQL NQUADS 變更序列化格式

Neptune 會使用 W3C RDF 1.1 N-Quads 規格中定義的資源描述架構 (RDF) N-QUADS 語言，將變更記錄到圖形中的 SPARQL 四元組。

變更記錄中的 `data` 欄位只包含 `stmt` 欄位，其中保留 N-QUADS 陳述式，表示已變更的 quad，如下列範例所示。

```
  "stmt" : "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
```

Neptune 串流範例

以下範例說明如何在 Amazon Neptune 中存取變更日誌串流資料。

主題

- AT_SEQUENCE_NUMBER 變更日誌
- AFTER_SEQUENCE_NUMBER 變更日誌
- TRIM_HORIZON 變更日誌
- LATEST 變更日誌
- 壓縮變更日誌

AT_SEQUENCE_NUMBER 變更日誌

以下範例顯示 Gremlin 或 openCypher `AT_SEQUENCE_NUMBER` 變更日誌。

```
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
}
```

這一個顯示 AT_SEQUENCE_NUMBER 變更日誌的 SPARQL 範例。

```
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
}
```

AFTER_SEQUENCE_NUMBER 變更日誌

以下範例顯示 Gremlin 或 openCypher AFTER_SEQUENCE_NUMBER 變更日誌。

```
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
}
```

TRIM_HORIZON 變更日誌

以下範例顯示 Gremlin 或 openCypher TRIM_HORIZON 變更日誌。

```
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
}
```

LATEST 變更日誌

以下範例顯示 Gremlin 或 openCypher LATEST 變更日誌。請注意，API 參數 `limit`、`commitNum` 和 `opNum` 是完全選用的。

```
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
}
```

壓縮變更日誌

以下範例顯示 Gremlin 或 openCypher 壓縮變更日誌。

```
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
}
```

用 AWS CloudFormation 於使用串流使用者應用程式設定 Neptune 到海王星的複製

您可以使用 AWS CloudFormation 範本來設定 Neptune 串流取用者應用程式，以支援 Neptune 到 Neptune 的複寫。

主題

- 選擇您所在地區的 AWS CloudFormation 範本
- 新增有關您要建立之 Neptune 串流消費者堆疊的詳細資訊
- 執行 AWS CloudFormation 範本
- 使用最新的 Lambda 成品更新串流輪詢器

選擇您所在地區的 AWS CloudFormation 範本

若要在 AWS CloudFormation 主控台上啟動適當的 AWS CloudFormation 堆疊，請根據您要使用的 AWS 區域選擇下表中的其中一個 Launch Stack 按鈕。

區域	檢視	在設計工具中檢視	啟動
美國東部 (維吉尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國東部 (俄亥俄)	檢視	在設計工具中檢視	Launch Stack
美國西部 (加利佛尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國西部 (奧勒岡)	檢視	在設計工具中檢視	Launch Stack
加拿大 (中部)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
南美洲 (聖保羅)	檢視	在設計工具中檢視	Launch Stack
歐洲 (斯德哥爾摩)	檢視	在設計工具中檢視	Launch Stack
歐洲 (愛爾蘭)	檢視	在設計工具中檢視	Launch Stack
歐洲 (倫敦)	檢視	在設計工具中檢視	Launch Stack
Europe (Paris)	檢視	在設計工具中檢視	Launch Stack
歐洲 (法蘭克福)	檢視	在設計工具中檢視	Launch Stack
Middle East (Bahrain)	檢視	在設計工具中檢視	Launch Stack
中東 (阿拉伯聯合大公國)	檢視	在設計工具中檢視	Launch Stack
以色列 (特拉維夫)	檢視	在設計工具中檢視	Launch Stack
非洲 (開普敦)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (東京)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (香港)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (首爾)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (新加坡)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (悉尼)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
亞太區域 (孟買)	檢視	在設計工具中檢視	Launch Stack
中國 (北京)	檢視	在設計工具中檢視	Launch Stack
中國 (寧夏)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國西部)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國東部)	檢視	在設計工具中檢視	Launch Stack

在 Create Stack (建立堆疊) 頁面中，選擇 Next (下一步)。

新增有關您要建立之 Neptune 串流消費者堆疊的詳細資訊

Specify Stack Details (指定堆疊詳細資訊) 頁面提供的屬性和參數，可用於控制應用程式設定。

堆疊名稱 — 您要建立的新 AWS CloudFormation 堆疊名稱。您通常可以使用預設值 `NeptuneStreamPoller`。

Parameters (參數) 下提供下列項目：

串流消費者執行所在 VPC 的網路組態

- **`VPC`** – 提供輪詢 Lambda 函數將在其中執行的 VPC 名稱。
- **`SubnetIDs`** – 要建立網路界面的子網路。新增與 Neptune 叢集對應的子網路。
- **`SecurityGroupIds`** – 提供安全群組的 ID，這些群組會授予來源 Neptune 資料庫叢集的寫入傳入存取權。
- **`RouteTableIds`** – 這是在 Neptune VPC 中建立 Amazon DynamoDB 端點的必要項目，如果您還沒有此項目，請建立一個。您必須提供與子網路相關聯的路由表 ID 逗號分隔清單。
- **`CreateDDBVPCEndPoint`** – 預設為 `true` 的布林值，指出是否需要建立 Dynamo 資料庫 VPC 端點。如果您已在 VPC 中建立 DynamoDB 端點，只需要將其變更為 `false` 即可。

- **CreateMonitoringEndPoint** – 預設為 `true` 的布林值，指出是否需要建立一個監控 VPC 端點。如果您已在 VPC 中建立監控端點，只需要將其變更為 `false` 即可。

串流輪詢器

- **ApplicationName** – 您通常可將此設定保留為預設值 (`NeptuneStream`)。如果使用不同的名稱，其必須是唯一的。
- **LambdaMemorySize** – 用來設定 Lambda 輪詢器函數可用的記憶體大小。預設值是 2,048 MB。
- **LambdaRuntime** – 從 Neptune 串流擷取項目之 Lambda 函數中使用的語言。您可以將此項設為 `python3.9` 或 `java8`。
- **LambdaS3Bucket** – 包含 Lambda 程式碼成品的 Amazon S3 儲存貯體。除非您要使用從不同 Amazon S3 儲存貯體載入的自訂 Lambda 輪詢函數，否則請保留空白。
- **LambdaS3Key** – 對應至 Lambda 程式碼成品的 Amazon S3 金鑰。除非您要使用自訂的 Lambda 輪詢函數，否則請保留空白。
- **LambdaLoggingLevel** – 一般而言，會將此設定保留為預設值，即 `INFO`。
- **ManagedPolicies** – 列出用於執行 Lambda 函數的受管政策。一般而言，除非您要使用自訂的 Lambda 輪詢函數，否則請保留空白。
- **StreamRecordsHandler** – 一般而言，除非您要為 Neptune 串流中的記錄使用自訂處理器，否則請保留空白。
- **StreamRecordsBatchSize** – 要從串流擷取的記錄數目上限。您可以使用此參數調整效能。預設值 (`5000`) 是很好的開始。允許的上限為 10,000。數字愈大，從串流讀取記錄所需的網路呼叫就愈少，但處理記錄所需的記憶體就愈多。此參數的較低值會產生較低的輸送量。
- **MaxPollingWaitTime** – 兩次輪詢之間的最長等待時間 (以秒為單位)。決定調用 Lambda 輪詢器輪詢 Neptune 串流的頻率。將此值設為 0 可連續輪詢。最高值為 3,600 秒 (1 小時)。預設值 (60 秒) 是很好的開始，視圖表資料變更的速度而定。
- **MaxPollingInterval** – 最長連續輪詢期間 (以秒為單位)。使用此項來設定 Lambda 輪詢函數的逾時。此值應在 5 秒與 900 秒之間的範圍內。預設值 (600 秒) 是很好的開始。
- **StepFunctionFallbackPeriod**— 等待輪詢器的 step-function-fallback-period 單位數量，之後通過 Amazon E CloudWatch vents 調用步驟函數以從故障中恢復。預設值值 (5 分鐘) 是很好的開始。
- **StepFunctionFallbackPeriodUnit** – 用來測量前一個 `StepFunctionFallbackPeriodUnit` (`minutes`、`hours` 或 `days`) 的時間單位。通常預設值 (`minutes`) 就足夠了。

Neptune 串流

- **NeptuneStreamEndpoint** – (必要) Neptune 來源串流的端點。這會採取以下兩種形式之一：
  - **https://*your DB cluster*:*port*/propertygraph/stream** (或其別名 https://*your DB cluster*:*port*/pg/stream)。
  - **https://*your DB cluster*:*port*/sparql/stream**.
- **Neptune Query Engine** – 選擇 Gremlin、openCypher 或 SPARQL。
- **IAMAuthEnabledOnSourceStream** – 如果您的 Neptune 資料庫叢集使用的是 IAM 身分驗證，請將此參數設為 `true` 。
- **StreamDBClusterResourceId** – 如果您的 Neptune 資料庫叢集使用的是 IAM 身分驗證，請將此參數設為叢集資源 ID。資源 ID 和叢集 ID 不同。其改採的格式為：`cluster-` 加上 28 個英數字元。它可以在 Neptune 主控台的叢集詳細資訊下找到。

目標 Neptune 資料庫叢集

- **TargetNeptuneClusterEndpoint** – 目標備份叢集的叢集端點 (僅主機名稱)。

  請注意，如果您指定 `TargetNeptuneClusterEndpoint`，您也無法指定 `TargetSPARQLUpdateEndpoint`。
- **TargetNeptuneClusterPort** – 目標叢集的連接埠號碼。

  請注意，如果指定 `TargetSPARQLUpdateEndpoint`，則會忽略 `TargetNeptuneClusterPort` 的設定。
- **IAMAuthEnabledOnTargetCluster** – 如果要在目標叢集上啟用 IAM 身分驗證，則設定為 true。
- **TargetAWSRegion**— 目標備份叢集的 AWS 區域，例如`us-east-1`)。只有當目標備份叢集的 AWS 區域與 Neptune 來源叢集的區域不同時，您才必須提供此參數，如跨區域複寫的情況一樣。如果來源和目標區域相同，則此為選用參數。

  請注意，如果該`TargetAWSRegion`值不是 Neptune 支援的有效 AWS 區域，則程序會失敗。
- **TargetNeptuneDBClusterResourceId** – 選用：只有在目標資料庫叢集上啟用 IAM 身分驗證時，才需要此項。設定為目標叢集的資源 ID。
- **SPARQLTripleOnlyMode** – 決定是否啟用僅限三重模式的布林旗標。在僅限三重模式中，沒有具名圖形複寫。預設值為 `false`。
- **TargetSPARQLUpdateEndpoint** – SPARQL 更新之目標端點的 URL，例如 `https://abc.com/xyz`。此端點可以是任何支援四元組或三元組的 SPARQL 存放區。

請注意，如果指定 TargetSPARQLUpdateEndpoint，您也無法指定 TargetNeptuneClusterEndpoint，且會忽略 TargetNeptuneClusterPort 的設定。

- **BlockSparqlReplicationOnBlankNode** — 布林旗標，如果設定為true，則會停止 SPARQL (RDF) 資料的複寫。 BlankNode 預設值為 false。

警示

- **Required to create Cloud watch Alarm**— true 如果要為新堆疊建立 CloudWatch 警示，請將此設定為。
- **SNS Topic ARN for Cloudwatch Alarm Notifications**— 應在其中傳送 CloudWatch 警示通知的 SNS 主題 ARN (僅在啟用警示時才需要)。
- **Email for Alarm Notifications** – 應接收警示通知的電子郵件地址 (僅在警示啟用時才需要)。

對於警示通知的目的地，您可以僅新增 SNS、僅新增電子郵件，或同時新增 SNS 和電子郵件。

執行 AWS CloudFormation 範本

現在，您可以完成佈建 Neptune 串流消費者應用程式執行個體的程序，如下所示：

1. 在中 AWS CloudFormation的 [指定堆疊詳細資訊] 頁面上，選擇 [下一步]。
2. 在選項頁面上，選擇下一步。
3. 在檢閱頁面上，選取第一個核取方塊以確認 AWS CloudFormation 將建立 IAM 資源。選取第二個核取方塊，確認新堆疊 CAPABILITY_AUTO_EXPAND。

   > **Note**
   >
   > CAPABILITY_AUTO_EXPAND 明確確認在建立堆疊時，無需事先檢閱，即可擴充巨集。使用者通常會在處理過的範本中建立變更集，以便在實際建立堆疊之前，檢閱巨集所做的變更。如需詳細資訊，請參閱 AWS CloudFormation CreateStackAPI 參考資料中的AWS CloudFormation API。

   然後選擇 Create (建立)。

使用最新的 Lambda 成品更新串流輪詢器

您可以使用最新的 Lambda 程式碼成品更新串流輪詢器，如下所示：

1. 在中 AWS Management Console，導覽至主父系 AWS CloudFormation 堆疊 AWS CloudFormation 並選取。
2. 為堆疊選取更新選項。
3. 選取取代目前範本。
4. 對於範本來源，請選擇 Amazon S3 URL，然後輸入以下 S3 URL：

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_neptune.json
```

5. 選取下一步而不變更任何 AWS CloudFormation 參數。
6. 請選擇 Update Stack (建立堆疊)。

堆疊現在將使用最新的成品來更新 Lambda 成品。

使用 Neptune 串流跨區域複寫進行災難復原

Neptune 提供兩種實作跨區域容錯移轉功能的方式：

- 跨區域快照複製和還原
- 使用 Neptune 串流在兩個不同區域的兩個叢集之間複寫資料。

跨區域快照複製和還原對於在不同區域中復原 Neptune 叢集具有最低操作負荷。不過，由於快照是 Neptune 叢集的完整備份，因此在區域之間複製快照可能需要大量的資料傳輸時間。因此，跨區域快照複製和還原可以用於只需要數小時復原點目標 (RPO) 與數小時復原時間點目標 (RTO)的案例。

復原點目標 (RPO) 是由備份之間的時間來測量。它會定義在進行上次備份的時間與復原資料庫的時間之間可能會丟失多少資料。

復原時間點目標 (RTO) 是以執行復原操作所需的時間來測量。這是資料庫叢集在發生失敗之後容錯移轉至復原的資料庫所需的時間。

Neptune 串流提供一種方式，讓備份 Neptune 叢集始終與主要生產叢集保持同步。如果發生失敗，您的資料庫會容錯移轉至備份叢集。由於資料會持續複製到備份叢集，從而可隨時立即做為容錯移轉目標使用，因此會將 RPO 和 RTO 縮短為數分鐘。

以這種方式使用 Neptune 串流的缺點是維護複寫元件所需的操作負荷，以及始終有第二個 Neptune 資料庫叢集在線上的成本，可能相當大。

設定 Neptune 到 Neptune 複寫

您的主要生產資料庫叢集位於給定來源區域的 VPC 中。基於災難復原的目的，有三個您需要在不同的復原區域中複寫或模擬的主要項目：

- 儲存在叢集中的資料。
- 主要叢集的組態。這將包括它是否使用 IAM 身分驗證、它是否加密、其資料庫叢集參數、其執行個體參數、執行個體大小等等。
- 它使用的網路拓撲，包括目標 VPC、其安全群組等等。

您可以使用如下的 Neptune 管理 API 來收集該資訊：

- DescribeDBClusters
- DescribeDBInstances
- DescribeDBClusterParameters
- DescribeDBParameters
- DescribeVpcs

搭配您收集的資訊，您可以使用下列程序，在不同的區域中設定備份叢集，在發生失敗時，生產叢集可容錯移轉至其中。

1：啟用 Neptune 串流

您可以使用 ModifyDBClusterParameterGroup，將 `neptune_streams` 參數設定為 1。然後，重新啟動資料庫叢集中的所有執行個體，以便變更生效。

在啟用了 Neptune 串流之後，最好在來源資料庫叢集上執行至少一個新增或更新操作。這會將資料點填入變更串流，這些資料點可在稍後重新同步生產叢集與備份叢集時參考。

2：在您要設定備份叢集的區域中建立新的 VPC

在與主要叢集不同的區域中建立新的 Neptune 資料庫叢集之前，您需要在目標區域中建立新的 VPC 來託管叢集。主要叢集與備份叢集之間的連線是透過 VPC 對等互連建立的，這會使用不同 VPC 中私有子網路的流量。不過，若要在兩個 VPC 之間建立 VPC 對等互連，它們不得具有重疊的 CIDR 區

塊或 IP 地址空間。您不能只在這兩個區域中使用預設 VPC，因為預設 VPC 的 CIDR 區塊一律相同 (172.31.0.0/16)。

您可以在目標區域中使用現有 VPC，只要它符合下列條件即可：

- 它沒有與您主要叢集所在 VPC 的 CIDR 區塊重疊的 CIDR 區塊。
- 它尚未與另一個 VPC 對等互連，該 VPC 與主要叢集所在的 VPC 具有相同的 CIDR 區塊。

如果目標區域中沒有合適的 VPC 可用，請使用 Amazon EC2 CreateVpc API 建立一個 VPC。

3：建立主要叢集的快照，並將其還原至目標備份區域

現在，您可以在目標備份區域中的適當 VPC 中建立新的 Neptune 叢集，該區域是生產叢集的副本：

在備份區域中製作生產叢集的副本

1. 在目標備份區域中，重新建立生產資料庫叢集所使用的參數和參數群組。您可以使用 CreateDBClusterParameterGroup、CreateDBParameterGroup、ModifyDBClusterParamete 和 ModifyDBParameterGroup 來執行此操作。

   請注意，CopyDBClusterParameterGroup 和 CopyDBParameterGroup API 目前不支援跨區域複製。
2. 使用 CreateDBClusterSnapshot 在生產區域的 VPC 中建立生產叢集的快照。
3. 使用 CopyDBClusterSnapshot 將快照複製到目標備份區域中的 VPC。
4. 使用 RestoreDBClusterFromSnapshot，以使用複製的快照在目標備份區域的 VPC 中建立新的資料庫叢集。使用您已從主要生產叢集複製的組態設定和參數。
5. 新的 Neptune 叢集現已存在，但不包含任何執行個體。使用 CreateDBInstance 建立新的主要/寫入器執行個體，其與您的生產叢集的寫入器執行個體具有相同的執行個體類型。此時不需要建立額外的僅供讀取複本，除非您的備份執行個體在容錯移轉之前將用來服務目標區域中的讀取 I/O。

4：在主要叢集的 VPC 與新備份叢集的 VPC 之間建立 VPC 對等互連

透過設定 VPC 對等互連，您可讓主要叢集的 VPC 與備份叢集的 VPC 進行通訊，就像它們是單一私有網路一樣。若要執行此作業，請採用下列步驟：

1. 從生產叢集的 VPC 中， 呼叫 CreateVpcPeeringConnection API 以建立對等互連連線。
2. 從目標備份叢集的 VPC 中， 呼叫 AcceptVpcPeeringConnection API 以接受對等互連連線。

3. 從生產叢集的 VPC 中，使用 CreateRoute API 將路由新增至 VPC 的路由表，其會將所有流量重新導向至目標 VPC 的 CIDR 區塊，以便它可以使用 VPC 對等互連字首清單。
4. 同樣地，從目標備份叢集的 VPC 中，使用 CreateRoute API 將路由新增至 VPC 路由表，其會將流量路由至主要叢集的 VPC。

5：設定 Neptune 串流複寫基礎設施

既然兩個叢集都已部署，並且已在兩個區域之間建立網路通訊，請使用 Neptune to Neptune AWS CloudFormation 範本，透過支援資料複寫的額外基礎架構部署 Neptune 串流消費者 Lambda 函數。在主要生產叢集的 VPC 中執行此操作。

您需要為此 AWS CloudFormation 堆棧提供的參數是：

- **NeptuneStreamEndpoint** – 主要叢集的串流端點，採用 URL 格式。例如：`https://(cluster name):8182/pg/stream`。
- **QueryEngine** – 此必須為 `gremlin`、`sparql` 或 `openCypher`。
- **RouteTableIds** – 可讓您同時為 DynamoDB VPC 端點和監控 VPC 端點新增路由。

  如果兩個額外的參數 (即 `CreateMonitoringEndpoint` 和 `CreateDynamoDBEndpoint`) 尚未存在於主要叢集的 VPC 上，也必須將這兩個參數設定為 true。如果它們已經存在，請確保它們設置為 false，否則 AWS CloudFormation 創建將失敗。
- **SecurityGroupIds** – 指定 Lambda 取用者用來與主要叢集的 Neptune 串流端點通訊的安全群組。

  在目標備份叢集中，附加一個安全性群組，允許源自此安全群組的流量。
- **SubnetIds** – 主要叢集 VPC 中的子網路 ID 清單，Lambda 取用者可以使用這些子網路 ID 與主要叢集進行通訊。
- **TargetNeptuneClusterEndpoint** – 目標備份叢集的叢集端點 (僅主機名稱)。
- **TargetAWSRegion**— 目標備份叢集的 AWS 區域，例如`us-east-1`)。只有當目標備份叢集的 AWS 區域與 Neptune 來源叢集的區域不同時，您才必須提供此參數，如跨區域複寫的情況一樣。如果來源和目標區域相同，則此為選用參數。

  請注意，如果該`TargetAWSRegion`值不是 Neptune 支援的有效 AWS 區域，則程序會失敗。
- **VPC** – 主要叢集 VPC 的 ID。

所有其他參數都可以保留其預設值。

AWS CloudFormation 範本部署完成後，Neptune 會開始將主要叢集的任何變更複製到備份叢集。您可以在 Lambda 取用者函數產生的 CloudWatch 記錄中監視此複寫。

其他考量

- 如果您需要在主叢集和備份叢集之間使用 IAM 身份驗證，也可以在調用 AWS CloudFormation 範本時進行設定。
- 如果主要叢集已啟用靜態加密，請考慮在將快照複製到目標，並與目標區域中的新 KMS 金鑰建立關聯時，如何管理相關聯的 KMS 金鑰。
- 最佳實務是在應用程式中使用的 Neptune 端點前面使用 DNS CNAME。然後，如果您需要手動容錯移轉至目標備份叢集，則可以變更這些 CNAME 以指向目標叢集和/或執行個體端點。

使用 Amazon OpenSearch 服務在 Amazon Neptune 全文搜索

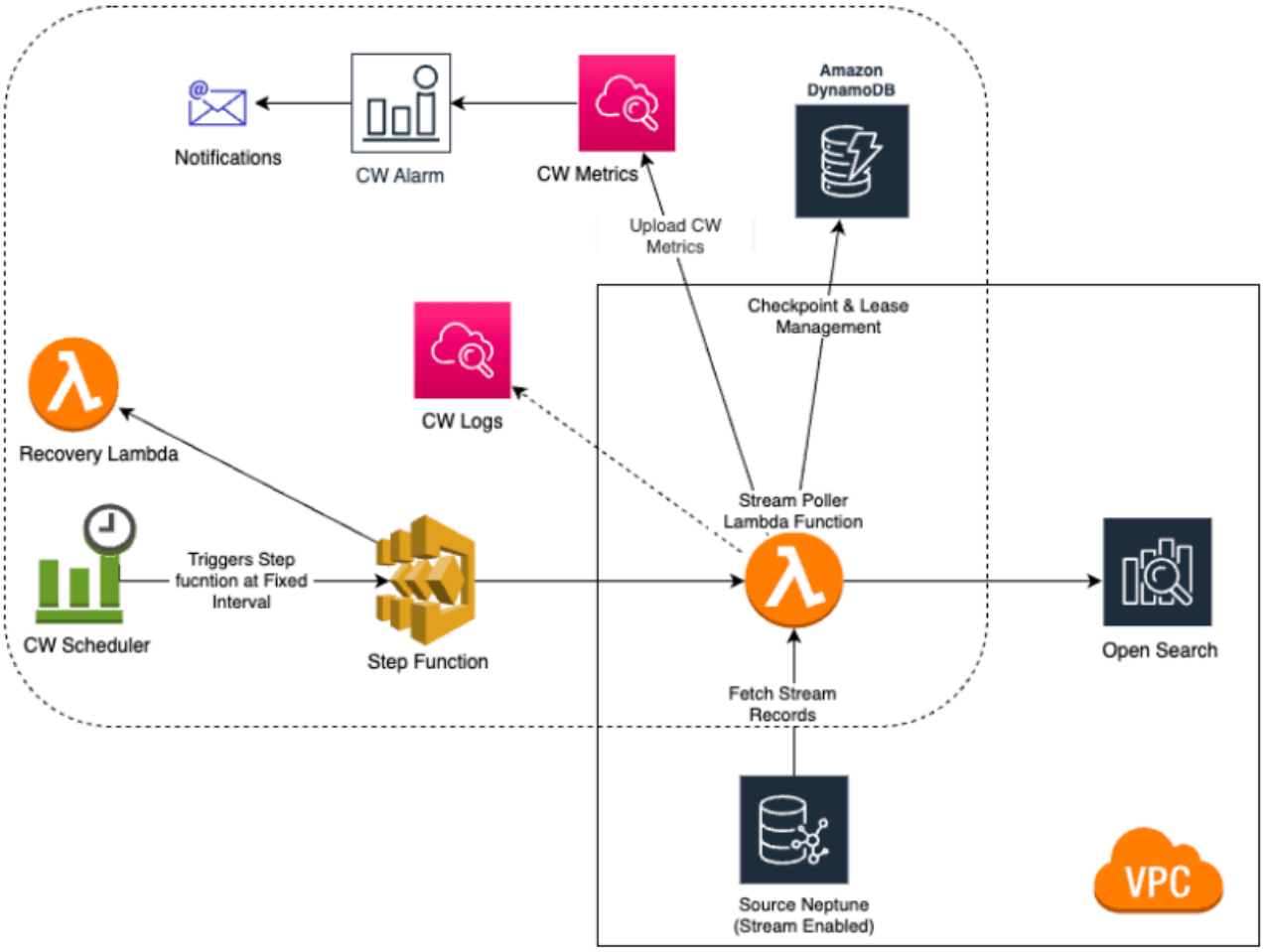
Neptune 與 Amazon OpenSearch 服務（服OpenSearch 務）集成，以支持 Girmlin 和 SPARQL 查詢中的全文搜索。此功能從 Neptune 引擎 1.0.2.1 版 開始可用，但我們建議與引擎版本 `1.0.4.2` 或更高版本搭配使用，以利用最新的修正。

從引擎版本 1.3.0.0 開始，Amazon Neptune 支援使用 Amazon 無伺 OpenSearch 服器服務在 Gremlin 和 SPARQL 查詢中進行全文檢索搜尋。

> **Note**
>
> 與 Amazon OpenSearch 服務集成時，Neptune 需要使用彈性搜索 7.1 版或更高版本，並與 OpenSearch 2.3、2.5 及更高版本一起使用。Neptune 也可與OpenSearch 無伺服器搭配使用。

您可以將 Neptune 與現有的 OpenSearch 服務叢集搭配使用，該叢集已根據OpenSearch 資料的 Neptune 資料模型. 或者，您可以使用 AWS CloudFormation 堆疊建立與 Neptune 連結的 OpenSearch 服務網域。

> **Important**
>
> 此處描述的 Neptune 到 OpenSearch 複寫程序不會複寫空白節點。這是需要注意的重要限制。此外，如果您在 OpenSearch 叢集上啟用精細的存取控制，則還需要在 Neptune 資料庫中啟用 IAM 身份驗證。



主題

- 亞馬遜海王星-到-複製 OpenSearch
- 複寫至 OpenSearch Serverless
- 從已啟用精細存取控制 (FGAC) 的 OpenSearch 叢集進行查詢
- 在 Neptune 全文檢索搜尋查詢中使用 Apache Lucene 查詢語法
- OpenSearch 資料的 Neptune 資料模型
- Neptune 全文檢索搜尋參數
- Amazon Neptune 中的非字串 OpenSearch 索引編製。
- Amazon Neptune 中的全文檢索搜尋查詢執行
- Neptune 中使用全文檢索搜尋的範例 SPARQL 查詢
- 在 Gremlin 查詢中使用 Neptune 全文檢索搜尋
- 對 Neptune 全文檢索搜尋進行疑難排解

亞馬遜海王星-到-複製 OpenSearch

Amazon Neptune 支持使用 Amazon 服務（ OpenSearch 服務）在 Gremlin 和 SPARQL 查詢中進行全文搜索。OpenSearch 您可以使用 AWS CloudFormation 堆疊將 OpenSearch 服務網域連結至 Neptune。 AWS CloudFormation 範本會建立串流消費者應用程式執行個體，以提供 Neptune-目標複寫。OpenSearch

在開始之前，您需要一個已啟用串流的現有 Neptune 資料庫叢集作為來源，以及一個 OpenSearch 服務網域做為複寫目標。

如果您已經擁有可由 Lambda 在 Neptune 資料庫叢集所在的 VPC 中存取的現有目標 OpenSearch 服務網域，則範本可以使用該網域。否則，您需要建立一個新的網域。

> **Note**
>
> 您建立的 OpenSearch 叢集和 Lambda 函數必須位於與 Neptune 資料庫叢集相同的 VPC 中，且叢集必須設定為 VPC 模式 (而非網際網路模式)。 OpenSearch

建議您使用新建立的 Neptune 執行個體搭配 OpenSearch 服務使用。如果您使用已包含資料的現有執行個體，則應在進行查詢之前執行 OpenSearch 服務資料同步，否則可能有資料不一致。此 GitHub 專案提供如何執行同步處理的範例：將 Neptune 匯出至 OpenSearch (https://github.com/awslabs/amazon-neptune-tools /樹/主export-neptune-to-elasticsearch/)。

> **Important**
>
> 與 Amazon OpenSearch 服務集成時，Neptune 需要使用彈性搜索 7.1 版或更高版本，並與 OpenSearch 2.3、2.5 和 future 兼容的 OpenSearch 版本一起使用。

> **Note**
>
> 從引擎版本 1.3.0.0 開始，Amazon Neptune 支援使用 Amazon 無伺 OpenSearch 服器服務在 Gremlin 和 SPARQL 查詢中進行全文檢索搜尋。

主題

- 使用 AWS CloudFormation 範本啟動海王星-目標複製 OpenSearch

- 在現有的 Neptune 資料庫上啟用全文檢索搜尋
- 更新串流輪詢器
- 停用和重新啟用串流輪詢器程序

使用 AWS CloudFormation 範本啟動海王星-目標複製 OpenSearch

啟動特定於您所在地區的 AWS CloudFormation 堆疊

以下每個 AWS CloudFormation 範本都會在特 AWS 定區域中建立串流消費者應用程式執行個體。若要使用 AWS CloudFormation 主控台啟動對應的堆疊，請根據您要使用的 AWS 區域選擇下表中的其中一個 Launch Stack 按鈕。

區域	檢視	在設計工具中檢視	啟動
美國東部 (維吉尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國東部 (俄亥俄)	檢視	在設計工具中檢視	Launch Stack
美國西部 (加利佛尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國西部 (奧勒岡)	檢視	在設計工具中檢視	Launch Stack
加拿大 (中部)	檢視	在設計工具中檢視	Launch Stack
南美洲 (聖保羅)	檢視	在設計工具中檢視	Launch Stack
歐洲 (斯德哥爾摩)	檢視	在設計工具中檢視	Launch Stack
歐洲 (愛爾蘭)	檢視	在設計工具中檢視	Launch Stack
歐洲 (倫敦)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
Europe (Paris)	檢視	在設計工具中檢視	Launch Stack
歐洲 (法蘭克福)	檢視	在設計工具中檢視	Launch Stack
Middle East (Bahrain)	檢視	在設計工具中檢視	Launch Stack
中東 (阿拉伯聯合大公國)	檢視	在設計工具中檢視	Launch Stack
以色列 (特拉維夫)	檢視	在設計工具中檢視	Launch Stack
非洲 (開普敦)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (香港)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (東京)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (首爾)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (新加坡)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (孟買)	檢視	在設計工具中檢視	Launch Stack
中國 (北京)	檢視	在設計工具中檢視	Launch Stack
中國 (寧夏)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國西部)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
AWS GovCloud (美國東部)	檢視	在設計工具中檢視	

在 Create Stack (建立堆疊) 頁面中，選擇 Next (下一步)。

新增關於您正在建立的新 OpenSearch堆疊的詳細資料

指定堆疊詳細資訊頁面提供您可以用來控制全文檢索搜尋設定的屬性和參數。

堆疊名稱 — 您要建立的新 AWS CloudFormation 堆疊名稱。您通常可以使用預設值 `NeptuneStreamPoller`。

Parameters (參數) 下提供下列項目：

串流消費者執行之 VPC 的網路組態

- **`VPC`** - 提供輪詢 Lambda 函數將在其中執行的 VPC 名稱。
- **`List of Subnet IDs`** - 要建立網路界面的子網路。新增與 Neptune 叢集對應的子網路。
- **`List of Security Group Ids`** - 提供安全群組的 ID，這些群組會授予來源 Neptune 資料庫叢集的寫入傳入存取權。
- **`List of Route Table Ids`** - 這是在 Neptune VPC 中建立 Amazon DynamoDB 端點的必要項目，如果您還沒有此項目，請建立一個。您必須提供與子網路相關聯的路由表 ID 逗號分隔清單。
- **`Require to create Dynamo DB VPC Endpoint`** - 預設為 `true` 的布林值。如果您已在 VPC 中建立 DynamoDB 端點，只需要將其變更為 `false` 即可。
- **`Require to create Monitoring VPC Endpoint`** - 預設為 `true` 的布林值。如果您已在 VPC 中建立監控端點，只需要將其變更為 `false` 即可。

串流輪詢器

- **`Application Name`** - 您通常可將此設定保留為預設值 (`NeptuneStream`)。如果使用不同的名稱，其必須是唯一的。
- **`Memory size for Lambda Poller`** - 用來設定 Lambda 輪詢器函數可用的記憶體大小。預設值是 2,048 MB。
- **`Lambda Runtime`** - 從 Neptune 串流擷取項目之 Lambda 函數中使用的語言。您可以將此項設為 `python3.9` 或 `java8`。

- **S3 Bucket having Lambda code artifacts** - 除非您要使用從其他 S3 儲存貯體載入的自訂 Lambda 輪詢函數，否則請保留空白。
- **S3 Key corresponding to Lambda Code artifacts** - 除非您要使用自訂的 Lambda 輪詢函數，否則請保留空白。
- **StartingCheckpoint** – 串流輪詢器的起始檢查點。預設值為 `0:0`，這表示從 Neptune 串流開頭處開始。
- **StreamPollerInitialState** - 輪詢器的初始狀態。預設值為 ENABLED，這表示串流複寫會在整個堆疊建立完成後立即開始。
- **Logging level for Lambda** - 一般而言，會將此設定保留為預設值 INFO。
- **Managed Policies for Lambda Execution** - 一般而言，除非您要使用自訂的 Lambda 輪詢函數，否則請保留空白。
- **Stream Records Handler** - 一般而言，除非您要為 Neptune 串流中的記錄使用自訂處理器，否則請保留空白。
- **Maximum records Fetched from Stream** - 您可以使用此參數調整效能。預設值 (100) 是很好的開始。允許的上限為 10,000。數字愈大，從串流讀取記錄所需的網路呼叫就愈少，但處理記錄所需的記憶體就愈多。
- **Max wait time between two Polls (in Seconds)** - 決定調用 Lambda 輪詢器輪詢 Neptune 串流的頻率。將此值設為 0 可連續輪詢。最高值為 3,600 秒 (1 小時)。預設值 (60 秒) 是很好的開始，視圖表資料變更的速度而定。
- **Maximum Continuous polling period (in Seconds)** - 用來設定 Lambda 輪詢函數的逾時。應介於 5 秒到 900 秒之間。預設值 (600 秒) 是很好的開始。
- **Step Function Fallback Period**— 等待輪詢器的 step-function-fallback-period 單位數量，之後通過 Amazon E CloudWatch vents 調用 step 函數以從故障中恢復。預設值值 (5 分鐘) 是很好的開始。
- **Step Function Fallback Period Unit** - 用來測量前述 `Step Function Fallback Period` 的時間單位 (分鐘、小時、天)。通常預設值 (分鐘) 即足夠。
- **Data replication scope**— 決定是要同時複製節點和邊緣，還是只複製節點 OpenSearch (這僅適用於 Gemlin 引擎資料)。預設值 (All) 通常是很好的開始。
- **Ignore OpenSearch missing document error**— 用來決定中是否 OpenSearch 可忽略遺失文件錯誤的旗標。遺失文件錯誤很少發生，但如果未忽略，則需要手動介入。預設值 (`True`) 通常是很好的開始。
- **Enable Non-String Indexing** - 要啟用或停用為沒有字串內容之欄位編製索引的旗標。如果將此旗標設定為`true`，則非字串欄位會在中編製索引 OpenSearch，或者 if `false` 僅對字串欄位進行索引。預設值為 `true`。

- **Properties to exclude from being inserted into OpenSearch**— 要從索引中 OpenSearch 排除的屬性或述詞索引鍵的逗號分隔清單。如果此 CFN 參數值保留空白，則所有屬性索引鍵都會編製索引。
- **Datatypes to exclude from being inserted into OpenSearch**— 要從索引中排除的屬性或述詞資料類型的逗號分隔清單。 OpenSearch 如果此 CFN 參數值保留空白，則會對所有可安全地轉換為 OpenSearch 資料類型的屬性值進行索引。

Neptune 串流

- **Endpoint of source Neptune Stream** - (必要) 這會採用以下兩種格式之一：
  - **https://*your DB cluster*:*port*/propertygraph/stream** (或其別名 `https://`*your DB cluster*`:`*port*`/pg/stream`)。
  - `https://`*your DB cluster*`:`*port*`/sparql/stream`
- **Neptune Query Engine** - 選擇 Gremlin 或 SPARQL。
- **Is IAM Auth Enabled?** - 如果您的 Neptune 資料庫叢集使用的是 IAM 身分驗證，請將此參數設為 `true`。
- **Neptune Cluster Resource Id** - 如果您的 Neptune 資料庫叢集使用的是 IAM 身分驗證，請將此參數設為叢集資源 ID。資源 ID 和叢集 ID 不同。其改採的格式為：`cluster-` 加上 28 個英數字元。它可以在 Neptune 主控台的叢集詳細資訊下找到。

目標 OpenSearch 叢集

- **Endpoint for OpenSearch service**— (必要) 在 VPC 中提供 OpenSearch 服務的端點。
- **Number of Shards for OpenSearch Index** - 預設值 (5) 通常是很好的開始。
- **Number of Replicas for OpenSearch Index** - 預設值 (1) 通常是很好的開始。
- **Geo Location Fields for Mapping** - 如您使用的是地理位置欄位，請在這裡列出屬性索引鍵。

警示

- **Require to create Cloud watch Alarm**— `true` 如果要為新堆疊建立 CloudWatch 警示，請將此設定為。
- **SNS Topic ARN for Cloudwatch Alarm Notifications**— 應在其中傳送 CloudWatch 警示通知的 SNS 主題 ARN (僅在啟用警示時才需要)。

- **`Email for Alarm Notifications`** - 應接收警示通知的電子郵件地址 (僅在警示啟用時才需要)。

對於警示通知的目的地，您可以僅新增 SNS、僅新增電子郵件，或同時新增 SNS 和電子郵件。

執行 AWS CloudFormation 範本

現在，您可以完成佈建 Neptune 串流消費者應用程式執行個體的程序，如下所示：

1. 在中 AWS CloudFormation的 [指定堆疊詳細資訊] 頁面上，選擇 [下一步]。
2. 在選項頁面上，選擇下一步。
3. 在檢閱頁面上，選取第一個核取方塊以確認 AWS CloudFormation 將建立 IAM 資源。選取第二個核取方塊，確認新堆疊 `CAPABILITY_AUTO_EXPAND`。

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` 明確確認在建立堆疊時，無需事先檢閱，即可擴充巨集。使用者通常會在處理過的範本中建立變更集，以便在實際建立堆疊之前，檢閱巨集所做的變更。如需詳細資訊，請參閱 AWS CloudFormation CreateStackAPI 參考資料中的AWS CloudFormation API 作業。

   然後選擇 Create (建立)。

在現有的 Neptune 資料庫上啟用全文檢索搜尋

如果您可以暫停寫入工作負載

在現有的 Neptune 資料庫上啟用全文檢索搜尋的最佳方式通常如下，前提是您可以暫停寫入工作負載。它需要建立一個複製、使用叢集參數啟用串流，並重新啟動所有執行個體。建立複製是一項相對較快的操作，因此所需的停機時間會受到限制。

必要的步驟如下：

1. 停止資料庫上的所有寫入工作負載。
2. 在資料庫上啟用串流 (請參閱啟用 Neptune 串流)。
3. 建立資料庫的複製 (請參閱 Neptune 中的資料庫複製)。

4. 繼續寫入工作負載。
5. 使用 github 上的export-neptune-to-elasticsearch工具，從複製的資料庫到 OpenSearch網域執行一次性同步。
6. 使用適用於您區域的AWS CloudFormation 範本，以持續更新的方式從原始資料庫開始同步 (範本中不需要變更任何組態)。
7. 刪除複製的資料庫和為export-neptune-to-elasticsearch工具建立的 AWS CloudFormation 堆疊。

如果您無法暫停寫入工作負載

如果您負擔不起在資料庫上暫停寫入工作負載，以下是所需停機時間甚至比上述建議方法還要少的方法，但需要謹慎完成：

1. 在資料庫上啟用串流 (請參閱啟用 Neptune 串流)。
2. 建立資料庫的複製 (請參閱 Neptune 中的資料庫複製)。
3. 針對串流 API 端點執行此類命令，以取得所複製資料庫上串流的最新 `eventID` (如需詳細資訊，請參閱呼叫 Neptune 串流 REST API)：

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   記下回應中 `lastEventId` 物件中 `commitNum` 和 `opNum` 欄位中的值。
4. 使用 github 上的export-neptune-to-elasticsearch工具，從複製的資料庫到 OpenSearch網域執行一次性同步。
5. 使用適用於您區域的AWS CloudFormation 範本，以持續更新的方式從原始資料庫開始同步。

   在建立堆疊時進行下列變更：在堆疊詳細資訊頁面的參數區段中，使用您在上面記錄的 `commitNum` 和 `opNum` 值，將 `StartingCheckpoint` 欄位的值設為 *commitNum*:*opnum*。
6. 刪除複製的資料庫和為export-neptune-to-elasticsearch工具建立的 AWS CloudFormation 堆疊。

更新串流輪詢器

使用最新的 Lambda 成品更新串流輪詢器

您可以使用最新的 Lambda 程式碼成品更新串流輪詢器，如下所示：

1. 在中 AWS Management Console，導覽至主父系 AWS CloudFormation 堆疊 AWS CloudFormation 並選取。
2. 為堆疊選取更新選項。
3. 選取取代目前範本。
4. 對於範本來源，請選擇 Amazon S3 URL，然後輸入以下 S3 URL：

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_elastic_search.json
```

5. 選取下一步而不變更任何 AWS CloudFormation 參數。
6. 請選擇 Update Stack (建立堆疊)。

堆疊現在將使用最新的成品來更新 Lambda 成品。

擴充串流輪詢器以支援自訂欄位

目前的串流輪詢器可以輕鬆地擴充為撰寫用於處理自訂欄位的自訂程式碼，以下部落格文章中會有這方面的詳細說明：使用 Neptune 串流擷取圖形變更。

> **Note**
>
> 在中新增自訂欄位時 OpenSearch，請務必將新欄位新增為述詞的內部物件 (請參閱Neptune 全文檢索搜尋資料模型)。

停用和重新啟用串流輪詢器程序

> **Warning**
>
> 停用串流輪詢器程序時要小心！如果程序的暫停時間超過串流到期時段，則可能會發生資料遺失。預設時段為 7 天，但是從引擎 1.2.0.0 版開始，您可以將自訂串流到期時段設為最多 90 天。

停用 (暫停) 串流輪詢器程序

1. 登錄到 AWS Management Console 並打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/.

2. 在導覽窗格中，選取規則。
3. 選取規則，其名稱包含您在用來設定串流輪詢器的 AWS CloudFormation 範本中作為「應用程式名稱」提供的名稱。
4. 選擇停用。
5. 開啟 Step Functions 主控台，網址為 https://console.aws.amazon.com/states/。
6. 選取與串流輪詢器程序對應的執行中步驟函數。同樣地，該步驟函數的名稱會包含您在用來設定串流輪詢器的 AWS CloudFormation 範本中，做為「應用程式名稱」提供的名稱。您可以按函數執行狀態進行篩選，以僅查看執行中函數。
7. 選擇停止。

重新啟用串流輪詢器程序

1. 登錄到 AWS Management Console 並打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/.
2. 在導覽窗格中，選取規則。
3. 選取規則，其名稱包含您在用來設定串流輪詢器的 AWS CloudFormation 範本中作為「應用程式名稱」提供的名稱。
4. 選擇停用。以所指定排程間隔為基礎的事件規則現在會觸發步驟函數的新執行。

複寫至 OpenSearch Serverless

從引擎版本 1.3.0.0 開始，Amazon Neptune 支援使用 Amazon OpenSearch Service Serverless 在 Gremlin 和 SPARQL 查詢中進行全文檢索搜尋。

如果要複寫至 OpenSearch Serverless，請將 Lambda 串流輪詢程式執行角色新增至 OpenSearch Serverless 集合的資料存取政策。Lambda 串流輪詢程式執行角色的 ARN 具有下列格式：

```
arn:aws:iam::(account ID):role/stack-name-NeptuneOSReplication-
NeptuneStreamPollerExecu-(uuid)
```

如需詳細資訊，請參閱 Amazon OpenSearch Serverless 的資料存取控制。

如果您已在 OpenSearch 叢集上啟用精細存取控制，則也需要在 Neptune 資料庫中啟用 IAM 身分驗證。

用於連線至 Neptune 資料庫的 IAM 實體 (使用者或角色) 應該同時具有 Neptune 和 OpenSearch Serverless 集合的許可。這表示您的使用者或角色必須具有如下附加的 OpenSearch Serverless 政策：

```
{
  "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)"
    }
  ]
}
```

如需更多資訊，請參閱Amazon Neptune 的自訂 IAM 資料存取政策陳述式。

從已啟用精細存取控制 (FGAC) 的 OpenSearch 叢集進行查詢

如果您已在 OpenSearch 叢集上啟用精細存取控制，則也需要在 Neptune 資料庫中啟用 IAM 身分驗證。

用於連線至 Neptune 資料庫的 IAM 實體 (使用者或角色) 應該同時具有 Neptune 和 OpenSearch 叢集的許可。這表示您的使用者或角色必須具有如下附加的 OpenSearch Service 政策：

```
{
  "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/*"
    }
  ]
}
```

如需詳細資訊，請參閱「Amazon Neptune 的自訂 IAM 資料存取政策陳述式」。

在 Neptune 全文檢索搜尋查詢中使用 Apache Lucene 查詢語法

OpenSearch 支援將 Apache Lucene 語法用於 query_string 查詢。這對在查詢中傳遞多個篩選條件特別有用。

Neptune 會使用巢狀結構，儲存 OpenSearch 文件中的屬性 (請參閱 Neptune 全文檢索搜尋資料模型)。使用 Lucene 語法時，您需要使用此巢狀模型中屬性的完整路徑。

以下是 Gremlin 範例：

```
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")
```

以下是 SPARQL 範例：

```
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 .
}
```

OpenSearch 資料的 Neptune 資料模型

Amazon Neptune 會使用統一的 JSON 文件結構，在 OpenSearch Service 中儲存 SPARQL 和 Gremlin 資料。OpenSearch 中的每份文件都會對應至一個實體，並儲存該實體的所有相關資訊。對 Gremlin 而言，頂點和邊緣皆視為實體，所以對應的 OpenSearch 文件會包含頂點、標籤和屬性的資訊。對 SPARQL 而言，主體可視為實體，所以對應的 OpenSearch 文件在一份文件中包含所有述詞物件對的資訊。

> **Note**
>
> Neptune 至 OpenSearch 的複寫實作只會儲存字串資料。但可加以修改以儲存其他資料類型。

統一的 JSON 文件結構如下所示。

```
{
  "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` - 代表文件的實體唯一 ID。
  - 對於 SPARQL 而言，這是主體 URI。
  - 對於 Gremlin 而言，這是 `Vertex_ID` 或 `Edge_ID`。
- `entity_type` - 表示頂點或邊緣的一或多個標籤，或為主體的零或多個 `rdf:type` 述詞值。
- `document_type` - 用來指定目前文件代表頂點、邊緣，還是 rdf 資源。
- `predicates` - 若為 Gremlin，儲存頂點或邊緣的屬性和值。若為 SPARQL，則儲存述詞物件對。

  屬性名稱會在 OpenSearch 中採用格式 `properties.name.value`。若要查詢該屬性，則必須採用該形式加以命名。
- `value` - Gremlin 的屬性值或 SPARQL 的物件值。
- `graph` - SPARQL 的具名圖形。
- `language` - SPARQL 中 `rdf:langString` 常值的語言標籤。

範例 SPARQL OpenSearch 文件

資料

```
@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
```

文件

```
{
  "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"
}
```

範例 Gremlin OpenSearch 文件

資料

```
# 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
```

文件

```
{
  "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"
      }
    ]
  }
}
```

Neptune 全文檢索搜尋參數

Amazon Neptune 會使用以下參數，在 Gremlin 和 SPARQL 中指定全文 OpenSearch 查詢：

- **`queryType`** – (必要) OpenSearch 查詢的類型。(如需查詢類型的清單，請參閱 OpenSearch 文件)。Neptune 支援下列 OpenSearch 查詢類型：
  - simple_query_string – 使用有所限制但可容錯之 Lucene 語法的剖析器，根據提供的查詢字串傳回文件。這是預設的查詢類型。

    此查詢使用簡單的語法，將提供的查詢字串剖析並分割成以特殊運算子為基礎的字詞。然後，查詢會先個別分析每個字詞，再傳回相符的文件。

    雖然其語法比 `query_string` 查詢限制更多，但查 `simple_query_string` 詢不會傳回無效語法的錯誤。而是會忽略查詢字串所有無效的部分。
  - match – `match` 查詢是執行全文檢索搜尋的標準查詢，包括模糊比對的選項。

- prefix – 傳回在所提供欄位中包含特定字首的文件。
- fuzzy – 傳回其中包含的字詞類似搜尋字詞的文件，如 Levenshtein 編輯距離所測量。

  編輯距離是指將一個字詞轉換成另一個字詞，所需要變更的單字元數量。這些變更可能包括：

  - 變更一個字元 (box 變成 fox)。
  - 移除一個字元(black 變成 lack)。
  - 插入一個字元 (sic 變成 sick)。
  - 調換兩個相鄰的字元 (act 變成 cat)。

  若要尋找類似的字詞，fuzzy 查詢會在指定的編輯距離內建立一組搜尋字詞所有可能的變異和擴展，然後傳回這些 variant 每一個的完全相符項目。
- term – 傳回在其中一個指定欄位中包含指定字詞完全相符項目的文件。

  您可以使用 `term` 查詢，根據價格、產品 ID 或使用者名稱等精確值尋找文件。

  > ⚠ Warning
  >
  > 文字欄位請避免使用 term 查詢。根據預設，OpenSearch 會在分析時變更文字欄位的值，這讓尋找文字欄位的完全相符項目變得很困難。
  > 若要搜尋文字欄位值，請改用 match 查詢。
- query_string – 使用解析器搭配嚴格語法 (Lucene 語法)，根據提供的查詢字串傳回文件。

  此查詢會根據 AND 或 NOT 等運算子，使用語法剖析和分割提供的查詢字串。然後，查詢會先個別分析各分割文字，再傳回相符的文件。

  您可以使用 `query_string` 查詢建立包含萬用字元、跨多欄位搜尋及更多的複雜搜尋。查詢雖然多樣化，但卻很嚴格，且若查詢字串包含任何無效的語法即會傳回錯誤。

  > ⚠ Warning
  >
  > 因為所有無效的語法都會傳回錯誤，所以我們不建議搜尋方塊使用 `query_string` 查詢。
  > 如果不需要支援查詢語法，請考慮使用 `match` 查詢。如果需要查詢語法的功能，請使用較不嚴格的 `simple_query_string` 查詢。

- **`field`** – OpenSearch 中要對其執行搜尋的欄位。只有在 `queryType` 允許時，才會忽略 (就像 `simple_query_string` 和 `query_string` 一樣)，在這種情況下，會針對所有欄位搜尋。在 Gremlin 中是隱含的。

  如果查詢允許，可指定多個欄位，如同執行 `simple_query_string` 和 `query_string` 一樣。
- **`query`** – (必要) 要針對 OpenSearch 執行的查詢。此欄位的內容可能會隨 queryType 而有所不同。不同的 queryTypes 接受不同的語法，例如，`Regexp`。在 Gremlin 中，`query` 是隱含的。
- **`maxResults`** – 要傳回的結果數量上限。預設值是 `index.max_result_window` OpenSearch 設定，本身預設為 10,000。`maxResults` 參數可以指定任何低於該值的數字。

  > **Important**
  >
  > 如果您將 `maxResults` 設為高於 OpenSearch `index.max_result_window` 值的值，並嘗試擷取超過 `index.max_result_window` 個結果，OpenSearch 會由於 `Result window is too large` 錯誤而失敗。不過，Neptune 會正常處理這個問題，不傳播此錯誤。如果您嘗試擷超過 `index.max_result_window` 的結果，請記住這一點。
- **`minScore`** – 搜尋結果必須傳回的最低分數。如需結果評分的說明，請參閱 OpenSearch 相關文件。
- **`batchSize`** – Neptune 一律批次擷取資料 (預設批次大小為 100)。您可以使用此參數調整效能。批次大小不能超過 `index.max_result_window` OpenSearch 設定，預設為 10,000。
- **`sortBy`** – 選用參數可讓您按以下任一種方式對 OpenSearch 傳回的結果排序：
  - 文件中的特定字串欄位 –

    例如，在 SPARQL 查詢中，您可以指定：

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name .
    ```

    在類似的 Gremlin 查詢中，您可以指定：

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name')
    ```
  - 文件中的特定非字串欄位 (*`long`*、*`double`* 等) –

    請注意，在非字串欄位上進行排序時，您需要將 `.value` 附加到欄位名稱，以區分其與字串欄位。

    例如，在 SPARQL 查詢中，您可以指定：

```
    neptune-fts:config neptune-fts:sortBy foaf:name.value .
```

在類似的 Gremlin 查詢中，您可以指定：

```
    .withSideEffect('Neptune#fts.sortBy', 'name.value')
```

- `score` – 按相符分數排序 (預設值)。

  如果 `sortOrder` 參數存在但 `sortBy` 不存在，則會依 `sortOrder` 指定的順序由 `score` 排序結果。

- `id` – 按 ID 排序，表示按 SPARQL 主旨 URI 或 Gremlin 頂點或邊緣 ID 排序。

  例如，在 SPARQL 查詢中，您可以指定：

```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
```

  在類似的 Gremlin 查詢中，您可以指定：

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
```

- `label` – 按標籤排序。

  例如，在 SPARQL 查詢中，您可以指定：

```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
```

  在類似的 Gremlin 查詢中，您可以指定：

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
```

- `doc_type` – 按文件類型 (亦即 SPARQL 或 Gremlin) 排序。

  例如，在 SPARQL 查詢中，您可以指定：

```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
```

  在類似的 Gremlin 查詢中，您可以指定：

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
```

根據預設，OpenSearch 結果不會排序，其順序不具確定性，這表示相同的查詢可能會在每次執行時傳回順序不同的項目。因此，如果結果集大於 `max_result_window`，則每次執行查詢時都會傳回完全不同的總結果子集。但透過排序，您可以更直接地比較不同執行的結果。

若無 `sortOrder` 參數伴隨 `sortBy`，則會採用從最大到最小的遞減 (DESC) 順序。

- **`sortOrder`** – 選用參數可讓您指定 OpenSearch 結果是從最小排到最大，還是從最大排到最小 (預設值)：

  - `ASC` – 遞增順序，從最小到最大。
  - `DESC` – 遞減順序，從最大到最小。

    此為預設值，可在 `sortBy` 參數存在，但未指定 `sortOrder` 的情況時使用。

  如果 `sortBy` 都 `sortOrder` 不存在，則根據預設，不會排序 OpenSearch 結果。

Amazon Neptune 中的非字串 OpenSearch 索引編製。

Amazon Neptune 中的非字串 OpenSearch 索引編製可讓您使用串流輸詢器，將述詞的非字串值複寫到 OpenSearch。所有可以安全地轉換為對應的 OpenSearch 對應或資料類型的述詞值，接著會複寫至 OpenSearch。

若要在新堆疊上啟用非字串索引編製，必須將 AWS CloudFormation 範本中的 `Enable Non-String Indexing` 旗標設為 `true`。這是預設設定。若要更新現有的堆疊以支援非字串索引編製，請參閱下面的 更新現有堆疊。

> **Note**
>
> - 最好不要在 **`1.0.4.2`** 之前的引擎版本上啟用非字串索引編製。
> - 針對符合多個欄位 (其中一些欄位包含字串值，而其他欄位則包含非字串值) 的欄位名稱使用規則表達式的 OpenSearch 查詢，會由於錯誤而失敗。如果 Neptune 中的全文檢索搜尋查詢屬於該類型，也會發生同樣的情況。
> - 按非字串欄位進行排序時，請將「.value」附加到欄位名稱，以區分其與字串欄位。

內容

更新現有的 Neptune 全文檢索搜尋堆疊以支援非字串索引編製

如果您已經在使用 Neptune 全文檢索搜尋，以下是支援非字串索引編製所需採取的步驟：

1. 停止串流輪詢器 Lambda 函數。這可確保在匯出期間不會複製任何新更新。若要執行此操作，請停用調用 Lambda 函數的雲端事件規則：

   - 在 AWS Management Console 中，導覽至 CloudWatch。
   - 選取規則。
   - 選擇具有 Lambda 串流輪詢器名稱的規則。
   - 選取停用以暫時停用規則。

2. 在 OpenSearch 中刪除目前的 Neptune 索引。使用下列 `curl` 查詢，從 OpenSearch 叢集中刪除 `amazon_neptune` 索引：

   ```
   curl -X DELETE "your OpenSearch endpoint/amazon_neptune"
   ```

3. 開始一次性從 Neptune 匯出至 OpenSearch。最好在此時設定新的 OpenSearch 堆疊，以便可為執行匯出的輪詢器挑選新的成品。

   請遵循 GitHub 中這裡列出的步驟，開始將您的 Neptune 資料一次性匯出至 OpenSearch。

4. 更新現有串流輪詢器的 Lambda 成品。在成功完成了將 Neptune 資料匯出至 OpenSearch 之後，請執行下列步驟：

- 在 AWS Management Console 中，導覽至 AWS CloudFormation。
- 選擇主父項 AWS CloudFormation 堆疊。
- 為該堆疊選取更新選項。
- 從選項中選取取代目前的範本。
- 對於範本來源，請選取 Amazon S3 URL。
- 對於 Amazon S3 URL，請輸入：

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_elastic_search.json
```

- 在不變更任何 AWS CloudFormation 參數的情況下選擇下一步。
- 選取更新堆疊。AWS CloudFormation 會將串流輪詢器的 Lambda 程式碼成品取代為最新的成品。

5. 重新啟動串流輪詢器。若要這樣做，請啟用適當的 CloudWatch 規則：

- 在 AWS Management Console 中，導覽至 CloudWatch。
- 選取規則。
- 選擇具有 Lambda 串流輪詢器名稱的規則。
- 選取啟用。

篩選 Neptune 全文檢索搜尋中要編製索引的欄位

AWS CloudFormation 範本詳細資訊中有兩個欄位，可讓您指定要從 OpenSearch 索引編製中排除的屬性或述詞索引鍵或資料類型：

按屬性或述詞名稱篩選

您可以使用名為 `Properties to exclude from being inserted into Elastic Search Index` 的選用 AWS CloudFormation 範本參數，提供逗號分隔清單，來列出要從 OpenSearch 索引編製中排除的屬性或述詞索引鍵。

例如，假設您將此參數設為 bob：

```
"Properties to exclude from being inserted into Elastic Search Index" : bob
```

在這種情況下，下列 Gemlin 更新查詢的串流記錄將會遭到捨棄，而不是進入索引：

```
g.V("1").property("bob", "test")
```

同樣地，您可以將此參數設為 http://my/example#bob：

```
"Properties to exclude from being inserted into Elastic Search Index" : http://my/
example#bob
```

在這種情況下，下列 Gemlin 更新查詢的串流記錄將會遭到捨棄，而不是進入索引：

```
PREFIX ex: <http://my/example#>
INSERT DATA { ex:s1 ex:bob "test"}.
```

如果您沒有在此 AWS CloudFormation 範本參數中輸入任何內容，則未以其他方式排除的所有屬性鍵將會編製索引。

按屬性或述詞值類型篩選

您可以使用名為 Datatypes to exclude from being inserted into Elastic Search Index 的選用 AWS CloudFormation 範本參數，提供逗號分隔清單，來列出要從 OpenSearch 索引編製中排除的屬性或述詞值資料類型。

對於 SPARQL，您不需要列出完整的 XSD 類型 URI，只需列出資料類型記號即可。您可以列出的有效資料類型記號如下：

- string
- boolean
- float
- double
- dateTime
- date
- time
- byte
- short
- int
- long

- decimal
- integer
- nonNegativeInteger
- nonPositiveInteger
- negativeInteger
- unsignedByte
- unsignedShort
- unsignedInt
- unsignedLong

對於 Gremlin，要列出的有效資料類型如下：

- string
- date
- bool
- byte
- short
- int
- long
- float
- double

例如，假設您將此參數設為 string：

```
"Datatypes to exclude from being inserted into Elastic Search Index" : string
```

在這種情況下，下列 Gemlin 更新查詢的串流記錄將會遭到捨棄，而不是進入索引：

```
g.V("1").property("myStringval", "testvalue")
```

同樣地，您可以將此參數設為 int：

```
"Datatypes to exclude from being inserted into Elastic Search Index" : int
```

在這種情況下，下列 Gemlin 更新查詢的串流記錄將會遭到捨棄，而不是進入索引：

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
INSERT DATA { ex:s1 ex:bob "11"^^xsd:int }.
```

如果您沒有在此 AWS CloudFormation 範本參數中輸入任何內容，則其值可以安全地轉換為 OpenSearch 對等項目的所有屬性都會編製索引。系統會忽略查詢語言不支援的列出類型。

將 SPARQL 和 Gremlin 資料類型對應至 OpenSearch

OpenSearch 中的新資料類型對應是根據屬性或物件中使用的資料類型而建立的。因為某些欄位包含不同類型的值，所以初始對應可能會排除欄位的某些值。

Neptune 資料類型會對應到 OpenSearch 資料類型，如下所示：

SPARQL 類型	Gremlin 類型	OpenSearch 類型
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
XSD:double	double	
XSD:decimal		

SPARQL 類型	Gremlin 類型	OpenSearch 類型
XSD:boolean	bool	boolean
XSD:datetime XSD:date	date	date
XSD:string XSD:time	string	text
自訂資料類型	無	text
任何其他資料類型	無	text

例如，下列 Gremlin 更新查詢會針對要新增至 OpenSearch 的「newField」產生新的對應，即 `{ "type" : "double" }`：

```
g.V("1").property("newField" 10.5)
```

同樣地，下列 SPARQL 更新查詢會針對要新增至 OpenSearch 的「ex:byte」產生新的對應，即 `{ "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**
>
> 如您所見，從 Neptune 對應到 OpenSearch 的項目，其最後結果可能是在 OpenSearch 具有的資料類型與在 Neptune 具有的資料類型不同。不過，OpenSearch 中有一個明確的文字欄位「資料類型」，可記錄項目在 Neptune 中具有的資料類型。

資料對應的驗證

使用下列程序將資料從 Neptune 複製到 OpenSearch：

- 如果有問題欄位的對應已存在於 OpenSearch 中：
  - 如果可以使用資料驗證規則，將資料安全地轉換為現有的對應，請將欄位儲存在 OpenSearch 中。
  - 如果沒有，請捨棄相應的串流更新記錄。
- 如果有問題的欄位中沒有現有的對應，請尋找與 Neptune 中欄位資料類型對應的 OpenSearch 資料類型。
  - 如果可以使用資料驗證規則，將欄位資料安全地轉換為 OpenSearch 資料類型，請將新對應和欄位資料儲存在 OpenSearch 中。
  - 如果沒有，請捨棄相應的串流更新記錄。

這些值是根據對等的 OpenSearch 類型或現有的 OpenSearch 對應 (而非 Neptune 類型) 進行驗證。例如，`"123"^^xsd:int` 中 `"123"` 值的驗證是針對 `long` 類型而非 `int` 類型執行的。

雖然 Neptune 嘗試將所有資料複製到 OpenSearch，但有些情況，OpenSearch 中的資料類型與 Neptune 中的資料類型完全不同，在這類情況下，會略過記錄，而不是在 OpenSearch 中編製索引。

例如，在 Neptune 中，一個屬性可以具有不同類型的多個值，而在 OpenSearch 中，一個欄位必須跨索引具有相同的類型。

透過啟用偵錯日誌，您可以檢視從 Neptune 匯出至 OpenSearch 期間已捨棄哪些記錄。除錯日誌項目的範例如下：

```
Dropping Record : Data type not a valid Gremlin type
<Record>
```

資料類型的驗證方式如下：

- **`text`** – Neptune 中的所有值都可以安全地對應到 OpenSearch 中的文字。
- **`long`** – OpenSearch 對應類型很長時，適用 Neptune 資料類型的下列規則 (在以下範例中，假設 `"testLong"` 具有 `long` 對應類型)：
  - `boolean` – 無效、無法轉換，並會捨棄對應的串流更新記錄。

    無效的 Gremlin 範例如下：

    ```
    "testLong" : true.
    ```

```
  "testLong" : false.
```

無效的 SPARQL 範例如下：

```
  ":testLong" : "true"^^xsd:boolean
  ":testLong" : "false"^^xsd:boolean
```

- `datetime` – 無效、無法轉換，並會捨棄對應的串流更新記錄。

  無效的 Gremlin 範例如下：

  ```
    ":testLong" :  datetime('2018-11-04T00:00:00').
  ```

  無效的 SPARQL 範例如下：

  ```
    ":testLong" : "2016-01-01"^^xsd:date
  ```

- `float`、`double`、或 `decimal` – 如果 Neptune 中的值是可以容納 64 位元的整數，則該值有效且會長時間儲存在 OpenSearch 中，但如果它具有小數部分，或是 NaN 或 INF，或是大於 9,223,372,036,854,775,807 或小於 -9,223,372,036,854,775,808，則它是無效的，且對應的串流更新記錄會遭到捨棄。

  有效的 Gremlin 範例如下：

  ```
    "testLong" :  145.0.
    ":testLong" :  123
    ":testLong" :  -9223372036854775807
  ```

  有效的 SPARQL 範例如下：

  ```
    ":testLong" : "145.0"^^xsd:float
    ":testLong" :  145.0
    ":testLong" : "145.0"^^xsd:double
    ":testLong" : "145.0"^^xsd:decimal
    ":testLong" : "-9223372036854775807"
  ```

  無效的 Gremlin 範例如下：

  ```
    "testLong" :  123.45
  ```

```
":testLong" :  9223372036854775900
```

無效的 SPARQL 範例如下：

```
":testLong" :  123.45
":testLong" :  9223372036854775900
":testLong" : "123.45"^^xsd:float
":testLong" : "123.45"^^xsd:double
":testLong" : "123.45"^^xsd:decimal
```

- `string` – 如果 Neptune 中的值是可以包含在 64 位元整數中的整數的字串表示法，則它是有效的，且會轉換為 OpenSearch 中的 `long`。對於 Elasticseearch `long` 對應，任何其他字串值都是無效的，且相應的串流更新記錄會遭到捨棄。

  有效的 Gremlin 範例如下：

  ```
  "testLong" :  "123".
  ":testLong" :  "145.0"
  ":testLong" :  "-9223372036854775807"
  ```

  有效的 SPARQL 範例如下：

  ```
  ":testLong" : "145.0"^^xsd:string
  ":testLong" : "-9223372036854775807"^^xsd:string
  ```

  無效的 Gremlin 範例如下：

  ```
  "testLong" :  "123.45"
  ":testLong" :  "9223372036854775900"
  ":testLong" :  "abc"
  ```

  無效的 SPARQL 範例如下：

  ```
  ":testLong" : "123.45"^^xsd:string
  ":testLong" : "abc"
  ":testLong" : "9223372036854775900"^^xsd:string
  ```

- **`double`** – 如果 OpenSearch 對應類型為 `double`，則會套用下列規則 (此處假設「testDouble」欄位在 OpenSearch 中具有 `double` 對應)：
  - `boolean` – 無效、無法轉換，並會捨棄對應的串流更新記錄。

無效的 Gremlin 範例如下：

```
  "testDouble" : true.
  "testDouble" : false.
```

無效的 SPARQL 範例如下：

```
  ":testDouble" : "true"^^xsd:boolean
  ":testDouble" : "false"^^xsd:boolean
```

- `datetime` – 無效、無法轉換，並會捨棄對應的串流更新記錄。

  無效的 Gremlin 範例如下：

```
  ":testDouble" :  datetime('2018-11-04T00:00:00').
```

  無效的 SPARQL 範例如下：

```
  ":testDouble" : "2016-01-01"^^xsd:date
```

- 浮點 `NaN` 或 `INF` – 如果 SPARQL 中的值是浮點 `NaN` 或 `INF`，則它是無效的，且對應的串流更新記錄會遭到捨棄。

  無效的 SPARQL 範例如下：

```
"  :testDouble" : "NaN"^^xsd:float
  ":testDouble" : "NaN"^^double
  ":testDouble" : "INF"^^double
  ":testDouble" : "-INF"^^double
```

- 數字或數值字串 – 如果 Neptune 中的值是任何其他數字，或是可以安全地表示為 `double` 之數字的數值字串表示法，則它是有效的，且會在 OpenSearch 中轉換為 `double`。對於 OpenSearch `double` 對應，任何其他字串值都是無效的，且相應的串流更新記錄會遭到捨棄。

  有效的 Gremlin 範例如下：

```
  "testDouble" :  123
  ":testDouble" :  "123"
  ":testDouble" :  145.67
```

```
  ":testDouble" :  "145.67"
```

有效的 SPARQL 範例如下：

```
  ":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
```

無效的 Gremlin 範例如下：

```
  ":testDouble" :  "abc"
```

無效的 SPARQL 範例如下：

```
  ":testDouble" : "abc"
```

- **`date`** – 如果 OpenSearch 對應類型為 `date`，Neptune `date` 和 `dateTime` 值都是有效的，任何可以成功剖析為 `dateTime` 格式的字串值也一樣有效。

  Gremlin 或 SPARQL 中的有效範例如下：

```
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"
```

無效的範例如下：

```
  123.45
  True
  "abc"
```

Neptune 中的範例非字串 OpenSearch 查詢

Neptune 目前不直接支援 OpenSearch 範圍查詢。不過，您可以使用 Lucene 語法和 query-type="query_string" 來達到相同的效果，正如您在下列範例查詢中看到的那樣。

1. 取得效齡大於 30 且名稱開頭為「Si」的所有頂點

在 Gremlin 中：

```
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*');
```

在 SPARQL 中：

```
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 .
  }
}
```

為了簡潔起見，這裡使用 "*foaf*age，而不是完整的 URI。此規則表達式將擷取在 URI 中同時具有 foaf 和 age 的所有欄位。

2. 取得效齡在 10 和 50 之間且名稱與「Ronka」模糊相符的所有節點

在 Gremlin 中：

```
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~');
```

在 SPARQL 中：

```
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. 取得時間戳記落在過去 25 天內的所有節點

在 Gremlin 中：

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>now-25d');
```

在 SPARQL 中：

```
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. 取得時間戳記落在給定年份和月份內的所有節點

在 Gremlin 中，於 Lucene 語法中使用日期數學運算式，表示 2020 年 12 月：

```
g.withSideEffect('Neptune#fts.endpoint', 'http://your-es-endpoint')
 .withSideEffect("Neptune#fts.queryType", "query_string")
 .V().has('*', 'Neptune#fts predicates.timestamp.value:>2020-12');
```

Gremlin 替代方案：

```
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]');
```

Amazon Neptune 中的全文檢索搜尋查詢執行

在包含全文搜尋的查詢中，Neptune 會先嘗試放置全文檢索搜尋呼叫，再放置查詢的其他部分。這可減少 OpenSearch 的呼叫次數，並在大多數情況下，大幅提升效能。不過，這絕不是硬性規則。例如，有些情況，PatternNode 或 UnionNode 可能先於全文檢索搜尋呼叫。

請考慮對資料庫進行下列 Gremlin 查詢，此資料庫包含 100,000 個 Person 執行個體：

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

如果此查詢按照步驟出現的順序執行，則 100,000 個解決方案將流入 OpenSearch，導致數百個 OpenSearch 呼叫。事實上，Neptune 會先呼叫 OpenSearch，然後結合產生的結果與 Neptune 結果。在大多數情況下，這樣做比依原始順序執行查詢快得多。

您可以使用 noReordering 查詢提示來防止重新排序查詢步驟執行：

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .withSideEffect('Neptune#noReordering', true)
```

```
  .hasLabel('Person')
  .has('name', 'Neptune#fts marcello~');
```

在這第二種情況下，會先執行 .hasLabel 步驟，再執行 .has('name', 'Neptune#fts marcello~') 步驟。

對於另一個範例，考慮對相同類型的資料進行 SPARQL 查詢：

```
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 .
  }
}
```

在這裡，Neptune 會再次先執行查詢的 SERVICE 部分，然後結合產生的結果與 Person 資料。您可以使用 joinOrder 查詢提示來隱藏此行為：

```
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 .
  }
}
```

再次，在第二個查詢中，這些部分會按照它們出現在查詢的順序執行。

Neptune 中使用全文檢索搜尋的範例 SPARQL 查詢

以下是在 Amazon Neptune 中使用全文檢索搜尋的一些範例 SPARQL 查詢。

SPARQL 比對查詢範例

```
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 .
  }
}
```

SPARQL 前置詞查詢範例

```
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 .
  }
}
```

SPARQL 模糊查詢範例

```
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 .
  }
}
```

SPARQL 術語查詢範例

```
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 .
  }
}
```

SPARQL query_string 查詢範例

此查詢可指定多個欄位。

```
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 .
  }
}
```

SPARQL simple_query_string 查詢範例

下列查詢使用萬用字元 ('*') 指定欄位。

```
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 .
  }
}
```

SPARQL 按字串欄位排序查詢範例

```
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 .
  }
}
```

SPARQL 按非字串欄位排序查詢範例

```
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 .
  }
}
```

SPARQL 按 ID 排序查詢範例

```
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 .
  }
}
```

SPARQL 按標籤排序查詢範例

```
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 .
  }
}
```

SPARQL 按 doc_type 排序查詢範例

```
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 .
  }
```

```
}
```

在 SPARQL 中使用 Lucene 語法的範例

僅對 OpenSearch 中的 `query_string` 查詢支援 Lucene 語法。

```
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 .
  }
}
```

在 Gremlin 查詢中使用 Neptune 全文檢索搜尋

`NeptuneSearchStep` 可以對未轉換為 Neptune 步驟的 Gremlin 周遊部分，進行全文搜尋查詢。例如，考慮使用以下查詢。

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")
  .V()
      .tail(100)
      .has("name", "Neptune#fts mark*")            <== # Limit the search on name
```

此查詢會在 Neptune 中轉換為以下最佳化的周遊。

```
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}
    }
}]
```

下列範例是針對航線資料的 Gremlin 查詢：

Gremlin 基本不區分大小寫的 **match** 查詢

```
g.withSideEffect("Neptune#fts.endpoint",
                 "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts dallas")

==>v[186]
==>v[8]
```

Gremlin **match** 查詢

```
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]
```

Gremlin **fuzzy** 查詢

```
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
```

Gremlin **query_string** 模糊查詢

```
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
```

Gremlin **query_string** 規則表達式查詢

```
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
```

Gremlin 混合查詢

此查詢會在同一個查詢中使用 Neptune 內部索引和 OpenSearch 索引。

```
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
```

簡單的 Gremlin 全文檢索搜尋範例

```
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]
```

使用 **query_string** 搭配「+」和「-」運算子的 Gremlin 查詢

雖然 query_string 查詢類型比預設的 simple_query_string 類型嚴格，但其確實能讓您執行較精確的查詢。下面的第一個查詢使用 query_string，第二個查詢使用預設的 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]
```

請注意在以下範例中，simple_query_string 如何不動聲色地忽略 '+' 和 '-' 運算子：

```
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]
```

具有 **AND** 和 **OR** 運算子的 Gremlin **query_string** 查詢

```
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]
```

Gremlin **term** 查詢

```
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]
```

Gremlin **prefix** 查詢

```
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]
```

在 Neptune Gremlin 中使用 Lucene 語法

在 Neptune Gremlin 中，您也可以使用 Lucene 查詢語法編寫非常強大的查詢條件。請注意，僅對 OpenSearch 中的 `query_string` 查詢支援 Lucene 語法。

假設下列資料：

```
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")
```

使用 (`queryType` 為 `query_string` 時所呼叫的) Lucene 語法，即可按名字和姓氏搜尋此資料，如下所示：

```
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]
```

請注意，在上述 `has()` 步驟中，此欄位由 "*" 取代)。實際上，置於該處的任何值都由您在查詢中存取的欄位所覆寫。您可以使用 `predicates.name.value`，存取名字欄位，因為這是資料模型的結構方式。

您可以按名字、姓氏和標籤進行搜尋，如下所示：

```
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]
```

標籤以 entity_type 加以存取，因為這同樣是資料模型的結構方式。

您也可以加入巢狀條件：

```
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]
```

插入現代 TinkerPop 圖形

```
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')
```

按字串欄位值排序範例

```
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')
```

按非字串欄位值排序範例

```
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')
```

按 ID 欄位值排序範例

```
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')
```

按標籤欄位值排序範例

```
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')
```

按 document_type 欄位值排序範例

```
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')
```

對 Neptune 全文檢索搜尋進行疑難排解

> **Note**
>
> 如果您已在 OpenSearch 叢集上啟用精細存取控制，則也需要在 Neptune 資料庫中啟用 IAM 身分驗證。

若要診斷從 Neptune 複寫到 OpenSearch 方面的問題，請參閱輪詢器 Lambda 函數的 CloudWatch Logs。這些日誌提供自串流讀取的詳細記錄數目，以及成功複寫到 OpenSearch 的詳細記錄數目。

您也可以透過變更 `LoggingLevel` 環境變數，以變更 Lambda 函數的 LOGGING 層級。

> **Note**
>
> 將 `LoggingLevel` 設為 DEBUG 後，您可以檢視其他詳細資訊，例如捨棄的串流記錄以及捨棄每筆記錄的原因，同時透過 StreamPoller 將資料從 Neptune 複寫到 OpenSearch。如果您發現遺失記錄，這樣做可能很有用。

Neptune 串流消費者應用程式會在 CloudWatch 上發佈兩個亦可協助您診斷問題的指標：

- `StreamRecordsProcessed` – 應用程式每時間單位所處理的記錄數目。有助於追蹤應用程式執行速率。
- `StreamLagTime` – 目前時間與正在處理之串流記錄遞交時間的時間差 (以毫秒為單位)。此指標會顯示消費者應用程式落後的程度。

此外，與複寫程序相關的所有指標都會公開在 CloudWatch 的儀表板中，與您使用 CloudWatch 範本執行個體化應用程式時所提供的 `ApplicationName` 同名。

您也可以選擇建立 CloudWatch 警示，每次輪詢失敗連續超過兩次即觸發。執行個體化應用程式時，透過將 `CreateCloudWatchAlarm` 欄位設為 `true` 以執行此作業。然後指定您希望觸發警示時收到通知的電子郵件地址。

故障診斷從串流讀取記錄時失敗的程序

如果程序在讀取串流的記錄時失敗，請確定下列事項：

- 已在叢集上啟用串流。
- Neptune 串流端點的格式正確：
  - 對於 Gremlin 或 openCypher：`https://`*`your cluster endpoint`*`:`*`your cluster port`*`/propertygraph/stream` 或其別名 (`https://`*`your cluster endpoint`*`:`*`your cluster port`*`/pg/stream`)
  - SPARQL：`https://`*`your cluster endpoint`*`:`*`your cluster port`*`/sparql/stream`
- DynamoDB 端點是針對 VPC 設定的。
- 已針對您的 VPC 子網路設定監控端點。

對將資料寫入 OpenSearch 時失敗的程序進行疑難排解

如果程序在將記錄寫入 OpenSearch 時失敗，請確定您有下列項目：

- 您的 Elasticsearch 版本為 7.1 或更高版本，或是 Opensearch 2.3 以上。
- 可從 VPC 中的輪詢器 Lambda 函數存取 OpenSearch。
- 附加至 OpenSearch 的安全政策允許傳入 HTTP/HTTPS 請求。

修正現有複寫設定上 Neptune 與 OpenSearch 之間的不同步問題

您可以使用下列步驟，讓 Neptune 資料庫和 OpenSearch 網域重新與最新資料同步，以防它們之間由於 `ExpiredStreamException` 或資料損毀而發生不同步問題。

請注意，此方法會刪除 OpenSearch 網域中的所有資料，並從 Neptune 資料庫的目前狀態重新同步它，因此不需要在 Neptune 資料庫中重新載入任何資料。

1. 停用複寫程序，如停用 (暫停) 串流輪詢器程序中所述。
2. 使用下列命令刪除 OpenSearch 網域上的 Neptune 索引：

```
curl -X DELETE "(your OpenSearch endpoint)/amazon_neptune"
```

3. 建立資料庫的複製 (請參閱 Neptune 中的資料庫複製)。
4. 針對串流 API 端點執行此類命令，以取得所複製資料庫上串流的最新 eventID (如需詳細資訊，請參閱呼叫 Neptune 串流 REST API)：

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   記下回應中 lastEventId 物件中 commitNum 和 opNum 欄位中的值。
5. 使用 github 上的 export-neptune-to-elasticsearch 工具，執行從複製的資料庫到 OpenSearch 網域的一次性同步。
6. 移至複寫堆疊的 DynamoDB 資料表。資料表的名稱將是您在 AWS CloudFormation 範本中指定的應用程式名稱 (預設值為 NeptuneStream)，尾碼為 -LeaseTable。換言之，預設資料表名稱為 NeptuneStream-LeaseTable。

   您可以透過掃描來探索資料表資料列，因為資料表中應該只有一個資料列。使用上面記錄的 commitNum 和 opNum 值進行以下變更：

   - 將資料表中 checkpoint 欄位的值變更為您針對 commitNum 記下的值。
   - 將資料表中 checkpointSubSequenceNumber 欄位的值變更為您針對 opNum 記下的值。
7. 重新啟用複寫程序，如重新啟用串流輪詢器程序中所述。
8. 刪除複製的資料庫和針對 export-neptune-to-elasticsearch 工具建立的 AWS CloudFormation 堆疊。

使用 Amazon Neptune 中的 AWS Lambda 函數

AWS Lambda 函數在 Amazon Neptune 應用程式中有很多用途。這裡我們會提供使用 Lambda 函數搭配任何熱門 Gemlin 驅動程式和語言變體的一般指引，以及以 Java、JavaScript 和 Python 撰寫之 Lambda 函數的特定範例。

> **Note**
>
> 使用 Lambda 函數搭配 Neptune 的最佳方式已隨著最新的引擎版本而變更。Neptune 曾經在 Lambda 執行內容回收之後讓閒置連線保持長時間的開啟狀態，這可能會導致伺服器上的資源洩漏。為了緩解這種情況，我們曾經建議在每次 Lambda 調用時開啟和關閉連線。不過，從引擎 1.0.3.0 版開始，閒置連線逾時已降低，以便在非作用中的 Lambda 執行內容回收之後，連線不再洩漏，因此我們建議您在執行內容期間使用單一連線。這應該包括一些錯誤處理以及退避和重試樣板程式碼，以處理非預期關閉的連線。

管理 AWS Lambda 函數中的 Gremlin WebSocket 連線

如果您使用 Gremlin 語言變體來查詢 Neptune，驅動程式會使用 WebSocket 連線來連線到資料庫。WebSockets 旨在支援長期用戶端-伺服器連線案例。另一方面，AWS Lambda 旨在支援相對短期和無狀態執行。使用 Lambda 查詢 Neptune 時，這種在設計理念上的不相符可能會導致一些非預期的問題。

一個 AWS Lambda 函數在執行內容中執行，此內容會將該函數與其他函數隔離。執行內容是在第一次調用函數時建立的，並且可在後續調用相同函數時重複使用。

不過，任何一個執行上下文永不會用來處理函數的多個並行調用。如果多個用戶端同時調用您的函數，Lambda 會為每個函數執行個體啟動額外的執行內容。所有這些新的執行內容可能轉而在後續的函數調用中重複使用。

在某些時候，Lambda 會回收執行內容，特別是如果它們已經處於非作用中一段時間。AWS Lambda 會透過 Lambda 延伸模組公開執行內容生命週期，包括 `Init`、`Invoke` 和 `Shutdown` 階段。使用這些延伸模組，您可以撰寫程式碼，在回收執行內容時清除外部資源，例如資料庫連線。

常見的最佳實務是在 Lambda 處理常式函數之外開啟資料庫連線，以便可在每次呼叫處理常式時重複使用。如果資料庫連線在某個時候中斷，您可以從處理常式內部重新連線。不過，使用此方法存在連線洩漏的危險。如果閒置連線在執行內容銷毀後長時間保持開啟狀態，則間歇性或突發性 Lambda 調用案例可能會逐漸洩漏連線並耗盡資料庫資源。

Neptune 連線限制和連線逾時已隨著較新的引擎版本而變更。以前，每個執行個體最多支援 60,000 個 WebSocket 連線。現在，每個 Neptune 執行個體的並行 WebSocket 連線數目上限會隨執行個體類型而有所不同。

此外，從引擎 1.0.3.0 版開始，Neptune 會將連線的閒置逾時從一小時縮減至大約 20 分鐘。如果用戶端未關閉連線，連線會在 20 到 25 分鐘的閒置逾時之後自動關閉。AWS Lambda 不會記載執行內容的生命週期，但實驗顯示新的 Neptune 連線逾時與非作用中的 Lambda 執行內容逾時一致。回收非作用中執行內容時，其連線很有可能已被 Neptune 關閉，或者之後很快就會關閉。

使用 AWS Lambda 搭配 Amazon Neptune Gremlin 的建議

現在，我們建議在 Lambda 執行內容的整個生命週期中使用單一連線和圖形周遊來源，而不是每個函數調用各有一個 (每個函數調用僅處理一個用戶端請求)。由於並行用戶端請求是由在個別執行內容中執行的不同函數執行個體處理的，因此不需要維護連線集區來處理函數執行個體內的並行請求。如果您使用的 Gemlin 驅動程式具有連線集區，請將其設定為只使用一個連線。

若要處理連線失敗，請對每個查詢使用重試邏輯。即使目標是在執行內容的生命週期內維護單一連線，但非預期的網路事件可能會導致該連線突然終止。這種連線失敗會顯示為不同的錯誤，取決於您使用的驅動程式。您應該編碼 Lambda 函數以處理這些連線問題，並在必要時嘗試重新連線。

一些 Gimlin 驅動程式會自動處理重新連線。例如，Java 驅動程式會自動嘗試代表您的用戶端程式碼重新建立與 Neptune 的連線。使用此驅動程式，您的函數程式碼只需要退避並重試查詢。相比之下，JavaScript 和 Python 驅動程式不會實作任何自動重新連線邏輯，因此使用這些驅動程式，您的函數程式碼必須在退避之後嘗試重新連線，並且僅在重新建立連線後才重新嘗試查詢。

這裡的程式碼範例確實包含重新連線邏輯，而不是假設用戶端正在處理它。

在 Lambda 中使用 Gremlin 寫入請求的建議

如果您的 Lambda 函數修改圖形資料，請考慮採用退避和重試策略來處理下列例外狀況：

- **`ConcurrentModificationException`** – Neptune 交易語義意味著寫入請求有時會由於 `ConcurrentModificationException` 而失敗。在這些情況下，請嘗試指數退避型重試機制。
- **`ReadOnlyViolationException`** – 由於叢集拓撲隨時可能因計劃或非計劃的事件而變更，因此寫入責任可能會從叢集中的某個執行個體轉移到另一個執行個體。如果您的函數程式碼嘗試將寫入請求傳送至不再是主要 (寫入器) 執行個體的執行個體，則請求會因 `ReadOnlyViolationException` 而失敗。發生這種情況時，請關閉現有連線、重新連線至叢集端點，然後重試請求。

此外，如果您使用退避和重試策略來處理寫入請求問題，請考慮為建立和更新請求實作等冪性查詢 (例如，使用 fold().coalesce().unfold())。

在 Lambda 中使用 Gremlin 讀取請求的建議

如果您的叢集中有一或多個僅供讀取複本，平衡這些複本之間的讀取要求是好主意。一種選項是使用讀取器端點。讀取器端點會平衡複本之間的連線，即使叢集拓撲在您新增或移除複本時發生變更，或提升複本以變成新的主要執行個體時也一樣。

不過，在某些情況下，使用讀取器端點可能會導致叢集資源的使用不均。讀取器端點的運作方式是定期變更 DNS 項目指向的主機。如果用戶端在 DNS 項目變更之前開啟了許多連線，則所有連線請求都會傳送至單一 Neptune 執行個體。在高輸送量 Lambda 案例中可能會發生這種情況，其中對 Lambda 函數的大量並行請求會導致建立多個執行內容，而每個內容都有自己的連線。如果這些連線幾乎全都同時建立，則連線很可能都指向叢集中的相同複本，並一直指向該複本，直到回收執行內容為止。

在執行個體之間分發請求的一種方式是，將 Lambda 函數設定為連線至執行個體端點 (從複本執行個體端點清單中隨機選擇)，而不是讀取器端點。這種方法的缺點是它需要 Lambda 程式碼來處理叢集拓撲中的變更，方法是監控叢集，並且每當叢集的成員資格變更時就更新端點清單。

如果您正要編寫一個 Java Lambda 函數，其需要平衡叢集中執行個體之間的讀取請求，則您可以使用適用於 Amazon Neptune 的 Gremlin 用戶端，這是 Java Gemlin 用戶端，它了解叢集拓撲，並可跨 Neptune 叢集中的一組執行個體 公平地分發連線和請求。這篇部落格文章包含一個範例 Java Lambda 函數，此函數會使用適用於 Amazon Neptune 的 Gremlin 用戶端。

可能會減慢 Neptune Gremlin Lambda 函數冷啟動速度的因素

第一次調用 AWS Lambda 函數稱為冷啟動。有幾個因素可能會增加冷啟動的延遲：

- 務必將足夠的記憶體指派您的 Lambda 函數。 – 對於 Lambda 函數，冷啟動期間的編譯速度可能會比 EC2 上的速度明顯慢很多，因為 AWS Lambda 會依您指派給函數的記憶體比例線性配置 CPU 週期。透過 1,792 MB 的記憶體，函數會接收相當於一個完整 vCPU 的項目 (每秒一個 vCPU 秒的額度)。對於以 Java 撰寫的大型 Lambda 函數，未指派足夠記憶體來接收足夠 CPU 週期的影響尤其明顯。
- 請注意，啟用 IAM 資料庫身分驗證可能會減慢冷啟動速度 – AWS Identity and Access Management (IAM) 資料庫身分驗證也會減慢冷啟動速度，尤其是在 Lambda 函數必須產生新的簽署金鑰時。此延遲只會影響冷啟動，而不會影響後續請求，因為一旦 IAM 資料庫身分驗證建立了連線憑證，Neptune 只會定期驗證它們是否仍然有效。

使用 AWS CloudFormation 建立要在 Neptune 中使用的 Lambda 函數

您可以使用 AWS CloudFormation 範本建立可存取 Neptune 的 AWS Lambda 函數。

1. 若要在 AWS CloudFormation 主控台上啟動 Lambda 函數堆疊，請在下表中選擇其中一個啟動堆疊按鈕。

區域	檢視	在設計工具中檢視	啟動
美國東部 (維吉尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國東部 (俄亥俄)	檢視	在設計工具中檢視	Launch Stack
美國西部 (加利佛尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國西部 (奧勒岡)	檢視	在設計工具中檢視	Launch Stack
加拿大 (中部)	檢視	在設計工具中檢視	Launch Stack
南美洲 (聖保羅)	檢視	在設計工具中檢視	Launch Stack
歐洲 (斯德哥爾摩)	檢視	在設計工具中檢視	Launch Stack
歐洲 (愛爾蘭)	檢視	在設計工具中檢視	Launch Stack
歐洲 (倫敦)	檢視	在設計工具中檢視	Launch Stack
歐洲 (巴黎)	檢視	在設計工具中檢視	Launch Stack
歐洲 (法蘭克福)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
中東 (巴林)	檢視	在設計工具中檢視	Launch Stack
中東 (阿拉伯聯合大公國)	檢視	在設計工具中檢視	Launch Stack
以色列 (特拉維夫)	檢視	在設計工具中檢視	Launch Stack
非洲 (開普敦)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (香港)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (東京)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (首爾)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (新加坡)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (雪梨)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (孟買)	檢視	在設計工具中檢視	Launch Stack
中國 (北京)	檢視	在設計工具中檢視	Launch Stack
中國 (寧夏)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國西部)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國東部)	檢視	在設計工具中檢視	Launch Stack

2. 在 Select Template (選取範本) 頁面上，請選擇 Next (下一步)。

3. 在 Specify Details (指定詳細資訊) 頁面上，設定下列選項：

   a. 選擇 Lambda 執行期，取決於您想要在 Lambda 函數中使用的語言。這些 AWS CloudFormation 範本目前支援下列語言：

      - Python 3.9 (對應到 Amazon S3 URL 中的 `python39`)
      - NodeJS 18 (對應到 Amazon S3 URL 中的 `nodejs18x`)
      - Ruby 2.5 (對應到 Amazon S3 URL 中的 `ruby25`)

   b. 提供適當的 Neptune 叢集端點及連接埠號碼。

   c. 提供適當的 Neptune 安全群組。

   d. 提供適當的 Neptune 子網路參數。

4. 選擇 Next (下一步)。

5. 在 Options (選項) 頁面上，選擇 Next (下一步)。

6. 在檢閱頁面上，選取第一個核取方塊以確認 AWS CloudFormation 將建立 IAM 資源。

   然後選擇 Create (建立)。

如果您需要對 Lambda 執行期自行進行變更，您可以從您區域中的 Amazon S3 位置下載一般版本：

```
https://s3.Amazon region.amazonaws.com/aws-neptune-customer-samples-Amazon region/
lambda/runtime-language/lambda_function.zip.
```

例如：

```
https://s3.us-west-2.amazonaws.com/aws-neptune-customer-samples-us-west-2/lambda/
python36/lambda_function.zip
```

Amazon Neptune 的 AWS Lambda 函數範例

下列範例 AWS Lambda 函數 (以 Java，JavaScript 和 Python 撰寫) 說明了使用 `fold().coalesce().unfold()` 慣用語搭配隨機產生的 ID 更新並插入單一頂點。

每個函數中的大部分程式碼都是樣板程式碼，負責管理連線並在發生錯誤時重試連線和查詢。真正的應用程式邏輯和 Gemlin 查詢是在 `doQuery()` 和 `query()` 方法中分別實作。如果您使用這些範例做為自己的 Lambda 函數的基礎，便可專注於修改 `doQuery()` 和 `query()`。

這些函數會設定為重試失敗的查詢 5 次，在重試之間等候 1 秒鐘。

這些函數需要一些值存在於下列 Lambda 環境變數中：

- **NEPTUNE_ENDPOINT** – 您的 Neptune 資料庫叢集端點。對於 Python，這應該是 `neptuneEndpoint`。
- **NEPTUNE_PORT** – Neptune 連接埠。對於 Python，這應該是 `neptunePort`。
- **USE_IAM** – (`true` 或 `false`) 如果您的資料庫已啟用 AWS Identity and Access Management (IAM) 資料庫身分驗證，請將 `USE_IAM` 環境變數設為 `true`。這會導致 Lambda 函數向 Neptune 提出 Sigv4-sign 連線請求。對於此類 IAM 資料庫身分驗證請求，請確保 Lambda 函數的執行角色已附加適當的 IAM 政策，其允許函數連線到 Neptune 資料庫叢集 (請參閱 IAM 政策的類型)。

Amazon Neptune 的 Java Lambda 函數範例

關於 Java AWS Lambda 函數，需要記住以下幾點：

- Java 驅動程式會維護自己的連線集區，您不需要這些連線集區，因此請使用 `minConnectionPoolSize(1)` 和 `maxConnectionPoolSize(1)` 設定您的 `Cluster` 物件。
- `Cluster` 物件的建置速度可能很慢，因為它會建立一個或多個序列化程序 (預設為 Gyro，再加上另一個序列化程序，如果您已針對其他輸出格式設定它的話，例如 `binary`)。這些需要一些時間才能執行個體化。
- 連接集區會連同第一個請求進行初始化。此時，驅動程式會設定 `Netty` 堆疊、配置位元組緩衝區，以及建立簽署金鑰 (如果您使用 IAM 資料庫身分驗證的話)。所有這些都會增加冷啟動延遲。
- Java 驅動程式的連線集區會監控伺服器主機的可用性，並在連線失敗時自動嘗試重新連線。它會啟動背景任務以嘗試重新建立連線。使用 `reconnectInterval( )` 設定重新連線嘗試之間的間隔。當驅動程式嘗試重新連線時，您的 Lambda 函數只需重試查詢即可。

  如果重試之間的間隔小於重新連線嘗試之間的間隔，則重試失敗的連線會再次失敗，因為主機被視為無法使用。這不適用於 `ConcurrentModificationException` 的重試。
- 使用 Java 8 而不是 Java 11。預設不會在 Java 11 中啟用 `Netty` 最佳化。
- 此範例使用 Retry4j 進行重試。
- 若要在 Java Lambda 函數中使用 `Sigv4` 簽署驅動程式，請參閱 搭配 Signature 第 4 版簽署，使用 Java 和 Gremlin 連線到 Neptune 中的相依性需求。

> **Warning**
>
> 來自 Retry4j 的 `CallExecutor` 可能不是執行緒安全的程式碼。考慮讓每個執行緒使用自己的 `CallExecutor` 執行個體。

```
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;
    }
  }
}
```

如果要在您的函數中包含重新連線邏輯，請參閱 Java 重新連線範例。

Amazon Neptune 的 JavaScript Lambda 函數範例

關於此範例的注意事項

- JavaScript 驅動程式不會維護連線集區。它一律開啟單一連線。
- 範例函數會使用來自 gremlin-aws-sigv4 的 Sigv4 簽署公用程式，將請求簽署至啟用 IAM 身分驗證的資料庫。
- 它會使用來自開放原始碼非同步公用程式模組中的 retry( ) 函數來處理「退避和重試」嘗試。
- Gremlin 終端步驟會傳回一個 JavaScript `promise` (請參閱 TinkerPop 文件)。對於 `next()`，這是 `{value, done}` 元組。
- 連線錯誤是在處理常式內引發的，並根據這裡列出的建議使用一些退避和重試邏輯進行處理，但有一個例外狀況。有一種連線問題是驅動程式不會被視為例外狀況，因此無法透過這種退避和重試邏輯來解決。

  問題是，如果在驅動程式傳送請求之後，但在驅動程式收到回應之前關閉連線，則查詢似乎已完成，但傳回 null 值。就 lambda 函數用戶端而言，該函數似乎成功完成，但回應是空的。

  此問題的影響取決於您的應用程式如何處理空白回應。某些應用程式可能會將來自讀取請求的空白回應視為錯誤，但其他應用程式可能會將其視為空白結果。

  遇到此連線問題的寫入請求也會傳回空白回應。空白回應的成功調用表示成功還是失敗？ 如果調用 write 函數的用戶端只是將函數的成功調用視為表示已遞交對資料庫的寫入，而不是檢查回應的本文，則系統可能似乎遺失了資料。

  此問題起因於驅動程式如何處理基礎通訊端發出的事件。當基礎網路通訊端由於 ECONNRESET 錯誤而關閉時，驅動程式使用的 WebSocket 會關閉並發出 `'ws close'` 事件。不過，驅動程式中沒有任何東西可以透過用來引發例外狀況的方式處理該事件。因此，查詢就會消失。

  若要解決此問題，這裡的範例 Lambda 函數會新增 `'ws close'` 事件處理常式，在建立遠端連線時將例外狀況擲回驅動程式。不過，此例外狀況並非沿著 Gemlin 查詢的要求-回應路徑引發，因此無

法用來觸發 lambda 函數本身內的任何退避和重試邏輯。'ws close' 事件處理常式所擲回的例外狀況會產生未處理的例外狀況，導致 lambda 調用失敗。這允許調用函數的用戶端處理錯誤，並在適當的情況下重試 lambda 調用。

我們建議您在 lambda 函數本身中實作退避和重試邏輯，以保護您的用戶端免於間歇性連線問題。不過，上述問題的因應措施也需要用戶端實作重試邏輯，以處理起因於這個特定連線問題的失敗。

Javascript 程式碼

```
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);
};
```

Amazon Neptune 的 Python Lambda 函數範例

以下是有關下列 Python AWS Lambda 範例函數的一些注意事項：

- 它會使用退避模組。
- 它會設定 `pool_size=1` 以防止建立不必要的連線集區。
- 它會設定 `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)
```

以下是範例結果，顯示重負載和輕負載的交替週期：


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

用於圖形上機器學習的 Amazon Neptune ML

在大型連線的資料集中通常會有珍貴的資訊，但很難僅根據人類的直覺使用查詢來擷取這些資訊。機器學習 (ML) 技術可以協助在具有數十億個關係的圖形中找出隱藏的關聯性。這些關聯性可能有助於推薦產品、預測信用度，識別詐欺等等。

Neptune ML 功能可讓您在數小時而不是數週的時間內，在大型圖形上建置並訓練有用的機器學習模型。為了達成這個目標，Neptune ML 會使用由 Amazon SageMaker 和 Deep Graph Library (DGL) (這是開放原始碼) 提供的圖形神經網路 (GNN)技術。圖形神經網路是人工智慧的新興領域 (例如，請參閱圖形神經網路的全面調查)。如需有關使用 GNN 搭配 DGL 的實作教學課程，請參閱使用 Deep Graph Library 學習圖形神經網路。

> **Note**
>
> 圖形頂點會在 Neptune ML 模型識別為「節點」。例如，頂點分類會使用節點分類機器學習模型，而頂點迴歸則會使用節點迴歸模型。

Neptune ML 可以做什麼

Neptune 同時支援直推式推論 (根據當時的圖形資料傳回訓練時預先計算的預測) 和直推式推論 (根據目前的資料，即時套用資料處理和模型評估)。請參閱 歸納推論與直推式推論之間的區別。

Neptune ML 可以訓練機器學習模型，以支援五種不同類別的推論：

Neptune ML 目前支援的推論任務類型

- 節點分類 – 預測頂點屬性的類別特徵。

  例如, 鑑於電影 The Shawshank Redemption，Neptune ML 可以從 `[story, crime, action, fantasy, drama, family, ...]` 的候選集將其 `genre` 屬性預測為 `story`。

  有兩種類型的節點分類任務：

  - 單一類別分類：在這種任務中，每個節點只有一個目標特徵。例如，`Alan Turing` 的屬性 `Place_of_birth` 具有值 UK。
  - 多類別分類：在這種任務中，每個節點可有多個目標特徵。例如，影片 The Godfather 的屬性 `genre` 具有值 `crime` 和 `story`。

- 節點迴歸 – 預測頂點的數值屬性。

  例如，鑑於電影 Avengers: Endgame，Neptune ML 可以預測其屬性 `popularity` 具有 `5.0` 一值。

- 邊緣分類 – 預測邊緣屬性的類別特徵。

  有兩種類型的邊緣分類任務：

  - 單一類別分類：在這種任務中，每個邊緣只有一個目標特徵。例如，使用者與電影之間的評分邊緣可能具有屬性 `liked`，其值為「Yes」或「No」。
  - 多類別分類：在這種任務中，每個邊緣可有多個目標特徵。例如，使用者與電影之間的評分可能會對「Funny」、「Heartwarming」、「Chilling」等屬性標籤具有多個值。

- 邊緣迴歸 – 預測邊緣的數值屬性。

  例如，使用者與電影之間的評分邊緣可能具有數值屬性 `score`，Neptune ML 可以鑑於使用者和電影針對該屬性預測一值。

- 連結預測 – 為特定來源節點和傳出邊緣預測最有可能的目的地節點，或為給定的目的地節點和傳入邊緣預測最有可能的來源節點。

  例如，透過藥物疾病知識圖譜，假設 `Aspirin` 作為來源節點，且 `treats` 作為傳出邊緣，Neptune ML 可以將最相關的目的地節點預測為 `heart disease`、`fever` 等等。

  或者，透過 Wikimedia 知識圖譜，假設 `President-of` 作為邊緣或關聯，且 `United-States` 作為目的地節點，Neptune ML 可以將最相關的前端預測為 `George Washington`、`Abraham Lincoln`、`Franklin D. Roosevelt` 等等。

> **Note**
>
> 節點分類和邊緣分類僅支援字串值。這表示不支援數值屬性值 (例如 `0` 或 `1`)，儘管字串等同於 `"0"` 和 `"1"`。同樣地，布林屬性值 `true` 和 `false` 不起作用，但 `"true"` 和 `"false"` 可以。

搭配 Neptune ML，您可以使用分為兩大類的機器學習模型：

Neptune ML 目前支援的機器學習模型類型

- 圖形神經網路（GNN）模型 – 其中包含 Relational Graph Convolutional Networks (R-GCNs)。GNN 模型適用於上述所有三種類型的任務。

- 知識圖譜嵌入 (KGE) 模型 – 其中包括 `TransE`、`DistMult` 和 `RotatE` 模型。它們僅適用於鏈接預測。

使用者定義的模型 – Neptune ML 也可讓您為上面列出的所有類型任務提供您自己的自訂模型實作。在使用 Neptune ML 訓練 API 搭配您的模型之前，您可以使用 Neptune ML 工具組，來開發並測試 Python 型自訂模型實作。如需如何建構和組織實作，以便其與 Neptune ML 的訓練基礎結構相容的詳細資訊，請參閱 Neptune ML 中的自訂模型。

設定 Neptune ML

開始使用 Neptune ML 的最簡單方式就是使用 AWS CloudFormation 快速入門範本。此範本會安裝所有必要的元件，包括新的 Neptune DB 叢集、所有必要的 IAM 角色，以及新的 Neptune 圖形筆記本，讓使用 Neptune ML 變得更輕鬆。

您也可以手動安裝 Neptune ML，如 在不使用快速入門 AWS CloudFormation 範本的情況下設定 Neptune ML 中所述。

使用 Neptune ML AWS CloudFormation 範本在新的資料庫叢集中快速開始

開始使用 Neptune ML 的最簡單方式就是使用 AWS CloudFormation 快速入門範本。此範本會安裝所有必要的元件，包括新的 Neptune DB 叢集、所有必要的 IAM 角色，以及新的 Neptune 圖形筆記本，讓使用 Neptune ML 變得更輕鬆。

建立 Neptune ML 快速入門堆疊

1. 若要在 AWS CloudFormation 主控台上啟動 AWS CloudFormation 堆疊，請在下表中選擇其中一個啟動堆疊按鈕。

區域	檢視	在設計工具中檢視	啟動
美國東部 (維吉尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國東部 (俄亥俄)	檢視	在設計工具中檢視	Launch Stack
美國西部 (加利佛尼亞北部)	檢視	在設計工具中檢視	Launch Stack
美國西部 (奧勒岡)	檢視	在設計工具中檢視	Launch Stack
加拿大 (中部)	檢視	在設計工具中檢視	Launch Stack
南美洲 (聖保羅)	檢視	在設計工具中檢視	Launch Stack
歐洲 (斯德哥爾摩)	檢視	在設計工具中檢視	Launch Stack
歐洲 (愛爾蘭)	檢視	在設計工具中檢視	Launch Stack
歐洲 (倫敦)	檢視	在設計工具中檢視	Launch Stack
歐洲 (巴黎)	檢視	在設計工具中檢視	Launch Stack

區域	檢視	在設計工具中檢視	啟動
歐洲 (法蘭克福)	檢視	在設計工具中檢視	Launch Stack
中東 (巴林)	檢視	在設計工具中檢視	Launch Stack
中東 (阿拉伯聯合大公國)	檢視	在設計工具中檢視	Launch Stack
以色列 (特拉維夫)	檢視	在設計工具中檢視	Launch Stack
非洲 (開普敦)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (香港)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (東京)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (首爾)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (新加坡)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (雪梨)	檢視	在設計工具中檢視	Launch Stack
亞太區域 (孟買)	檢視	在設計工具中檢視	Launch Stack
中國 (北京)	檢視	在設計工具中檢視	Launch Stack
中國 (寧夏)	檢視	在設計工具中檢視	Launch Stack
AWS GovCloud (美國西部)	檢視	在設計工具中檢視	Launch Stack

2. 在 Select Template (選取範本) 頁面上，請選擇 Next (下一步)。

3. 在指定詳細資訊頁面上，選擇下一步。
4. 在 Options (選項) 頁面上，選擇 Next (下一步)。
5. 在檢閱頁面上，有兩個您需要核取的核取方塊：

   - 第一個確認 AWS CloudFormation 可能使用自訂名稱建立 IAM 資源。
   - 第二個確認 AWS CloudFormation 可能需要新堆疊的 `CAPABILITY_AUTO_EXPAND` 功能。`CAPABILITY_AUTO_EXPAND` 明確允許 AWS CloudFormation 在建立堆疊時自動擴展巨集，而無需事先檢閱。

     客戶通常會在處理過的範本中建立變更集，以便在實際建立堆疊之前，檢閱巨集所做的變更。如需詳細資訊，請參閱 AWS CloudFormationCreateStack API。

   然後選擇 Create (建立)。

快速入門範本會建立並設定下列項目：

- Neptune 資料庫叢集。
- 必要的 IAM 角色 (並附加它們)。
- 必要的 Amazon EC2 安全群組。
- 必要的 SageMaker VPC 端點。
- Neptune ML 的資料庫叢集參數群組。
- 該參數群組中的必要參數。
- 預先填入 Neptune ML 筆記本範例的 SageMaker 筆記本。請注意，並非每個區域都提供所有執行個體大小，因此您必須確定選取的筆記型電腦執行個體大小是您所在地區支援的大小。
- Neptune-Export 服務。

當快速入門堆疊準備就緒時，請前往範本建立的 SageMaker 筆記本，並查看預先填入的範例。它們將協助您下載要用於嘗試 Neptune ML 功能的範例資料集。

當您使用 Neptune ML 時，它們還可以為您節省大量時間。例如，請參閱這些筆記本支援的 %neptune_ml 行魔法和 %%neptune_ml 儲存格魔法。

您也可以使用下列 AWS CLI 命令來執行快速入門 AWS CloudFormation 範本：

```
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)
```

在不使用快速入門 AWS CloudFormation 範本的情況下設定 Neptune ML

1. 從運作中的 Neptune 資料庫叢集開始

如果您不使用 AWS CloudFormation 快速入門範本來設定 Neptune ML，則需要現有的 Neptune 資料庫叢集才能使用。如果想要的話，您可以使用已有的叢集，或複製您已使用中的叢集，或者您可以建立新的叢集 (請參閱 建立資料庫叢集)。

2. 安裝 Neptune-Export 服務

如果您尚未安裝，請安裝 Neptune-Export 服務，如 使用 Neptune-Export 服務匯出 Neptune 資料 中所述。

使用下列設定，將傳入規則新增至安裝所建立的 NeptuneExportSecurityGroup 安全群組：

- Type (類型)：Custom TCP
- Protocol (通訊協定)：TCP
- 連接埠範圍：80 - 443
- 來源：*(Neptune ######### ID)*

3. 建立自訂 **NeptuneLoadFromS3** IAM 角色

如果您尚未建立，請建立自訂 NeptuneLoadFromS3 IAM 角色，如 建立 IAM 角色來存取 Amazon S3 中所述。

建立自訂 **NeptuneSageMakerIAMRole** 角色

使用 IAM 主控台搭配下列政策建立自訂 NeptuneSageMakerIAMRole：

```
{
  "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"
    }
  ]
}
```

建立此角色時，請編輯信任關係，以便其如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
```

```
          "ec2.amazonaws.com",
          "rds.amazonaws.com",
          "sagemaker.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

最後，複製指派給這個新 NeptuneSageMakerIAMRole 角色的 ARN。

> **Important**
>
> - 確定 NeptuneSageMakerIAMRole 中的 Amazon S3 許可符合上述的許可。
> - 在上述政策中，通用 ARN (arn:aws:s3:::*) 用於 Amazon S3 資源。如果由於某種原因無法使用通用 ARN，則 NeptuneML 命令將使用的任何其他客戶 Amazon S3 資源的 arn:aws:s3:::graphlytics* 和 ARN 必須新增到資源區段。

設定資料庫叢集以啟用 Neptune ML

為 Neptune ML 設定資料庫叢集

1. 在 Neptune 主控台中，導覽至參數群組，然後導覽至與您將要使用的資料庫叢集相關聯的資料庫叢集參數群組。將 neptune_ml_iam_role 參數設定為指派給您剛建立之 NeptuneSageMakerIAMRole 角色的 ARN。
2. 導覽至資料庫，然後選取將要用於 Neptune ML 的資料庫叢集。選取動作，然後選取管理 IAM 角色。
3. 在管理 IAM 角色頁面上，選取新增角色，然後新增 NeptuneSageMakerIAMRole。然後新增 NeptuneLoadFromS3 角色。
4. 重新啟動資料庫叢集的寫入器執行個體。

在 Neptune VPC 中建立兩個 SageMaker 端點

最後，若要讓 Neptune 引擎可以存取必要的 SageMaker 管理 API，您需要在 Neptune VPC 中建立兩個 SageMaker 端點，如 在 Neptune VPC 中為 SageMaker 建立兩個端點 中所述。

為 Neptune ML 手動設定 Neptune 筆記本

Neptune SageMaker 筆記本已預先載入各種適用於 Neptune ML 的範例筆記本。您可以在開放原始碼圖形筆記本 GitHub 儲存庫中預覽這些範例。

您可以使用其中一個現有的 Neptune 筆記本，或者如果想要的話，您可以遵循 使用 Neptune 工作台託管 Neptune 筆記本 中的指示建立自己的筆記本。

您也可以遵循下列步驟，設定預設 Neptune 筆記本以搭配 Neptune ML 使用：

修改 Neptune ML 的筆記本

1. 開啟位於 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 主控台。
2. 在左側的導覽窗格中，選擇筆記本，然後選擇筆記本執行個體。尋找您要用於 Neptune ML 的 Neptune 筆記本名稱，然後選取該名稱以前往其詳細資訊頁面。
3. 如果筆記本執行個體正在執行，請選取筆記本詳細資料頁面右上方的停止按鈕。
4. 在筆記本執行個體設定中的生命週期組態下，選取連結以開啟筆記本生命週期的頁面。
5. 選取右上方的編輯，然後選取繼續。
6. 在啟動筆記本索引標籤中，修改指令碼以包含其他匯出命令，並填入 Neptune ML IAM 角色和匯出服務 URI 的欄位 (如下所示)，取決於您的 Shell：

```
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. 選取 Update (更新)。
8. 返回筆記本執行個體頁面。在許可和加密下，IAM 角色 ARN 有一個欄位。選取此欄位中的連結，即可前往此筆記本執行個體搭配執行的 IAM 角色。
9. 建立新的內嵌政策，如下所示：

```
{
  "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. 儲存此新政策並在步驟 8 中將其附加至 IAM 角色。
11. 選取 SageMaker 筆記本執行個體詳細資訊頁面右上方的啟動，以啟動記事本執行個體。

使用 AWS CLI 在資料庫叢集上設定 Neptune ML

除了 AWS CloudFormation 快速入門範本和 AWS Management Console 之外，您也可以使用 AWS CLI 設定 Neptune ML。

1. 為新的 Neptune ML 叢集建立資料庫叢集參數群組

下列 AWS CLI 命令會建立新的資料庫叢集參數群組，並將其設定為使用 Neptune ML：

建立和設定 Neptune ML 的資料庫叢集參數群組

1. 建立新的資料庫叢集參數群組：

```
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. 為資料庫叢集建立設定為 SageMakerExcecutionIAMRole 之 ARN 的 neptune_ml_iam_role 資料庫叢集參數，以在呼叫 SageMaker 時用於建立工作並從託管的 ML 模型取得預測：

```
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)
```

   設定此參數可讓 Neptune 存取 SageMaker，而不必在每次呼叫時傳入角色。

   如需如何建立 SageMakerExcecutionIAMRole 的相關資訊，請參閱 建立自訂 NeptuneSageMakerIAMRole 角色。

3. 最後，使用 describe-db-cluster-parameters 檢查新資料庫叢集參數群組中的所有參數是否都按照您想要的方式進行設定：

```
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)
```

將新的資料庫叢集參數群組附加至您將搭配 Neptune ML 使用的資料庫叢集

現在，您可以使用下列命令，將剛建立的新資料庫叢集參數群組附加至現有的資料庫叢集：

```
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)
```

若要使所有參數生效，您可以接著重新啟動資料庫叢集：

```
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)
```

或者，如果您要建立新的資料庫叢集以搭配 Neptune ML 使用，則可以使用下列命令，建立附加新參數群組的叢集，然後建立新的主要 (寫入器) 執行個體：

```
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)
```

將 **NeptuneSageMakerIAMRole** 附加至您的資料庫叢集，以便它可以存取 SageMaker 和 Amazon S3 資源

最後，請遵循 建立自訂 NeptuneSageMakerIAMRole 角色 中的指示來建立 IAM 角色，讓您的資料庫叢集可與 SageMaker 和 Amazon S3 通訊。然後，使用以下命令將您建立的 NeptuneSageMakerIAMRole 角色附加至資料庫叢集：

```
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)
```

在 Neptune VPC 中為 SageMaker 建立兩個端點

Neptune ML 在 Neptune 資料庫叢集的 VPC 中需要兩個 SageMaker 端點：

- com.amazonaws.*(AWS region, like us-east-1)*.sagemaker.runtime
- com.amazonaws.*(AWS region, like us-east-1)*.sagemaker.api

如果您未使用快速入門 AWS CloudFormation 範本，自動為您建立這些端點，您可以使用下列 AWS CLI 命令來建立它們：

下列命令會建立 sagemaker.runtime 端點：

```
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
```

下列命令則會建立 sagemaker.api 端點：

```
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
```

您也可以使用 VPC 主控台建立這些端點。請參閱使用 AWS PrivateLink 保護 Amazon SageMaker 中的預測呼叫，以及使用 AWS PrivateLink 保護所有 Amazon SageMaker API 呼叫。

在資料庫叢集參數群組中建立 SageMaker 推論端點參數

若要避免指定模型的 SageMaker 推論端點，而您會在對其進行的每次查詢中使用此模型，請在 Neptune ML 的資料庫叢集參數群組中建立名為 `neptune_ml_endpoint` 的資料庫叢集參數。將參數設定為有問題執行個體端點的 `id`。

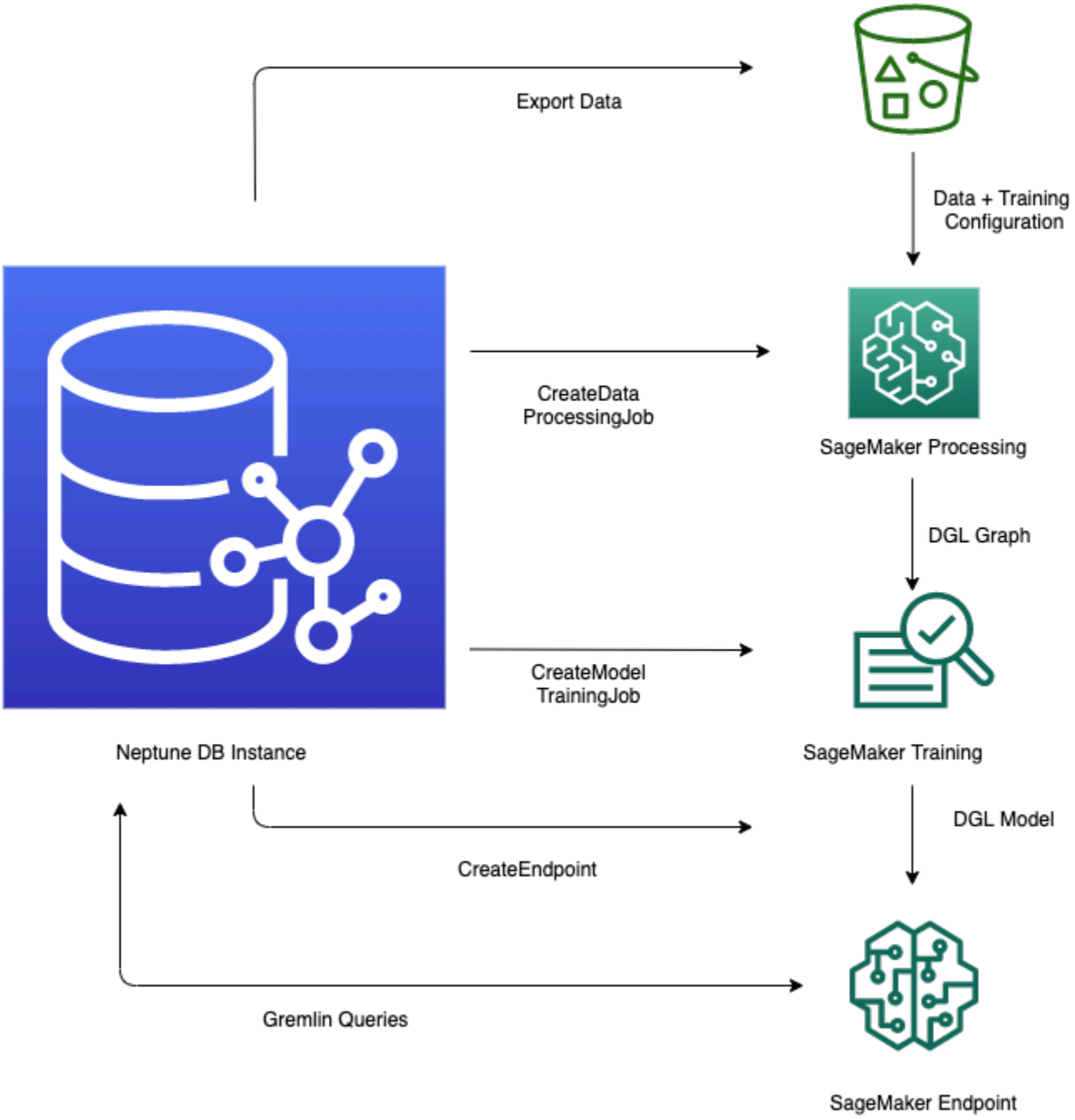
您可以使用下列 AWS CLI 命令來執行該操作：

```
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)
```

如何使用 Neptune ML 特徵的概觀

啟動使用 Neptune ML 的工作流程

在 Amazon Neptune 中使用 Neptune ML 功能通常涉及以下五個步驟以開始：



1. 資料匯出和組態 – 資料匯出步驟會使用 Neptune-Export 服務或 `neptune-export` 命令列工具，以 CSV 格式將資料從 Neptune 匯出至 Amazon Simple Storage Service (Amazon S3)。名為

`training-data-configuration.json` 的組態檔案會同時自動產生，指定如何將匯出的資料載入至可訓練的圖形。

2. 資料預先處理 – 在此步驟中，匯出的資料集會使用標準技術預先處理，以準備進行模型訓練。可針對數值資料執行特徵標準化，並可以使用 `word2vec` 編碼文字特徵。在此步驟結束時，會從匯出的資料集產生 DGL (Deep Graph Library) 圖形，供模型訓練步驟使用。

   此步驟是使用您帳戶中的 SageMaker 處理工作來實作，且產生的資料會儲存在您指定的 Amazon S3 位置中。

3. 模型訓練 – 模型訓練步驟會訓練將用於預測的機器學習模型。

   模型訓練分兩個階段完成：

   - 第一個階段使用 SageMaker 處理工作來產生模型訓練策略組態集，其會指定將用於模型訓練的模型類型和模型超參數範圍。
   - 然後，第二個階段使用 SageMaker 模型調整工作來嘗試不同的超參數組態，並選取產生效能最佳模型的訓練工作。調整工作會對已處理的資料執行預先指定數量的模型訓練工作試驗。在此階段結束時，會使用最佳訓練工作的訓練模型參數，產生用於推論的模型成品。

4. 在 Amazon SageMaker 中建立推論端點 – 推論端點是 SageMaker 端點執行個體，其會隨著最佳訓練工作產生的模型成品一起啟動。每個模型都繫結至單一端點。端點能夠接受來自圖形資料庫的傳入請求，並傳回請求中輸入的模型預測。在您建立端點之後，它會保持作用中狀態，直到您將其刪除為止。

5. 使用 Gremlin 查詢機器學習模型 – 您可以使用 Gremlin 查詢語言的延伸模組，從推論端點查詢預測。

> **Note**
>
> Neptune 工作台包含一個行魔法和一個儲存格魔法，其可以為您節省管理這些步驟的大量時間，即：
>
> - %neptune_ml
> - %%neptune_ml

根據不斷發展的圖形資料進行預測

使用持續變更的圖形，您可能想要使用全新資料定期建立新的批次預測。查詢預先計算的預測 (直推式推論) 比根據最新資料 (歸納推論) 即時產生新預測要快得多。這兩種方法都有自己的可取之處，取決於資料變更的速度和效能需求。

歸納推論與直推式推論之間的區別

執行直推式推論時，Neptune 會查詢並傳回訓練時預先計算的預測。

執行歸納推論時，Neptune 會建構相關的子圖形並擷取其屬性。然後，DGL GNN 模型會即時套用資料處理和模型評估。

因此，歸納推論可以產生涉及節點和邊緣的預測，而這些節點和邊緣在訓練時不存在，並會反映圖形的目前狀態。代價是更高的延遲。

如果您的圖形是動態的，您可能想要使用歸納推論來確保將最新資料納入考慮，但是如果您的圖形是靜態的，則直推式推論更快、更有效率。

歸納推論預設為停用。您可以在查詢中使用 Gremlin Neptune#ml.inductiveInference 述詞，啟用此推論進行查詢，如下所示：

```
.with( "Neptune#ml.inductiveInference")
```

增量直推式工作流程

儘管您只需重新執行步驟一到三 (從資料匯出和組態到模型轉換) 即可更新模型成品，但 Neptune ML 支援更簡單的方法，以使用新資料更新批次 ML 預測。一種是使用增量模型工作流程，而另一種則是使用透過暖啟動進行模型重新訓練。

增量模型工作流程

在此工作流程中，您可以更新 ML 預測，而無需要重新訓練 ML 模型。

> **Note**
>
> 只有當圖形資料已使用新的節點和/或邊緣進行更新時，您才能執行此操作。目前，當節點遭刪除時，它無法運作。

1. 資料匯出和組態 – 此步驟與主工作流程中的步驟相同。
2. 增量資料預先處理 – 此步驟類似於主工作流程中的資料預先處理步驟，但使用先前使用的相同處理組態，此組態對應至特定的訓練模型。
3. 模型轉換 – 此模型轉換步驟不是模型訓練步驟，而是從主工作流程和增量資料預先處理步驟的結果中取得訓練模型，並產生要用於推論的新模型成品。模型轉換步驟會啟動 SageMaker 處理工作，以執行產生更新模型成品的運算。
4. 更新 Amazon SageMaker 推論端點 – 如果您具有現有的推論端點，此步驟會使用模型轉換步驟所產生的新模型成品來更新端點。或者，您也可以使用新的模型成品建立新的推論端點。

透過暖啟動進行模型重新訓練

使用此工作流程，您可以訓練和部署新的 ML 模型，以使用增量圖形資料進行預測，但是從使用主工作流程產生的現有模型開始：

1. 資料匯出和組態 – 此步驟與主工作流程中的步驟相同。
2. 增量資料預先處理 – 此步驟與增量模型推論工作流程中的步驟相同。新的圖形資料應該使用先前用於模型訓練的相同處理方法來處理。
3. 透過暖啟動進行模型訓練 – 模型訓練類似於主工作流程中發生的情況，但是您可以利用來自先前模型訓練任務的資訊來加速模型超參數搜尋。
4. 更新 Amazon SageMaker 推論端點 – 此步驟與增量模型推論工作流程中的步驟相同。

Neptune ML 中自訂模型的工作流程

Neptune ML 可讓您針對 Neptune ML 支援的任何任務實作、訓練和部署您自己的自訂模型。開發和部署自訂模型的工作流程基本上與內建模型的工作流程相同，但有一些差異，如 自訂模型工作流 中所述。

Neptune ML 階段的執行個體選擇

Neptune ML 處理的不同階段會使用不同的 SageMaker 執行個體。在此，我們討論如何為每個階段選擇正確的執行個體類型。您可以在 Amazon SageMaker 定價中找到 SageMaker 執行個體類型和定價的相關資訊。

選取執行個體進行資料處理

SageMaker 資料處理步驟需要處理執行個體對輸入、中繼和輸出資料具有足夠的記憶體和磁碟儲存體。所需的特定記憶體和磁碟儲存體容量取決於 Neptune ML 圖形的特性及其匯出的特徵。

根據預設，Neptune ML 會選擇其記憶體十倍於磁碟上所匯出圖形資料大小的最小 `ml.r5` 執行個體。

選取執行個體進行模型訓練和模型轉換

選取正確的執行個體類型進行模型訓練或模型轉換，取決於任務類型、圖形大小和您的周轉需求。GPU 執行個體可提供最佳效能。我們通常建議 p3 和 g4dn 序列執行個體。您也可以使用 p2 或 p4d 執行個體。

根據預設，Neptune ML 會選擇其記憶體超過模型訓練和模型轉換所需的最小 GPU 執行個體。您可以在 Amazon S3 資料處理輸出位置的 `train_instance_recommendation.json` 檔案中找到該選擇的內容。以下是 `train_instance_recommendation.json` 檔案的內容範例：

```
{
  "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)"
}
```

為推論端點選取執行個體

為推論端點選取正確的執行個體類型，取決於任務類型、圖形大小和預算。根據預設，Neptune ML 會選擇其記憶體超過推論端點所需的最小`ml.m5d`執行個體。

> **Note**
>
> 如果需要超過 384 GB 的記憶體，Neptune ML 會使用 `ml.r5d.24xlarge` 執行個體。

您可以在 `infer_instance_recommendation.json` 檔案中查看 Neptune ML 建議的執行個體類型，此檔案位於您用於模型訓練的 Amazon S3 位置。以下是該檔案的內容範例：

```
{
  "instance" :   "(the recommended instance type for an inference endpoint)",
  "disk_size" :  "(the estimated disk space required)",
  "mem_size" :   "(the estimated memory required)"
}
```

使用 **neptune-export** 工具或 Neptune-Export 服務，從 Neptune ML 的 Neptune 中匯出資料

Neptune ML 需要您為 Deep Graph Library (DGL) 提供訓練資料，以建立和評估模型。

您可以使用 Neptune-Export 服務或 neptune-export 公用程式，從 Neptune 中匯出資料。此服務和命令列工具兩者都會以 CSV 格式將資料發佈至 Amazon Simple Storage Service (Amazon S3)，並使用 Amazon S3 伺服器端加密 (SSE-S3) 進行加密。請參閱 由 Neptune-Export 和 neptune-export 匯出的檔案。

此外，當您針對 Neptune ML 設定訓練資料的匯出時，匯出工作會建立並發佈加密的模型訓練組態檔案，以及匯出的資料。根據預設，此檔案會命名為 `training-data-configuration.json`。

使用 Neptune-Export 服務匯出 Neptune ML 訓練資料的範例

此請求會匯出節點分類任務的屬性圖訓練資料：

```
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"
              }
            ]
          }
        }
      }'
```

此請求會匯出節點分類任務的 RDF 訓練資料：

```
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"
              }
            ]
          }
        }
      }'
```

匯出訓練資料時要在 **params** 物件中設定的欄位

匯出請求中的 params 物件可以包含各種欄位，如 params 文件中所述。下列欄位與匯出機器學習訓練資料最相關：

- **endpoint** – 使用 endpoint 來指定資料庫叢集中 Neptune 執行個體的端點，匯出程序可以查詢該資料庫叢集來擷取資料。
- **profile** – 必須將 params 物件中的 profile 欄位設為 **neptune-ml**。

  這會導致匯出程序針對 Neptune ML 模型訓練適當地格式化匯出的資料，若是屬性圖資料採取 CSV 格式，若是 RDF 資料則以 N-Triples 表示。它也會導致系統建立 training-data-configuration.json 檔案，並將其寫入與匯出的訓練資料相同的 Amazon S3 位置。

- **cloneCluster** – 如果設為 `true`，匯出程序會複製您的資料庫叢集、從該複製中匯出，然後在完成時刪除該複製。
- **useIamAuth** – 如果您的資料庫叢集已啟用 IAM 身分驗證，則您必須包含這個設為 `true` 的欄位。

匯出程序還提供了幾種篩選您匯出之資料的方法 (請參閱這些範例)。

使用 additionalParams 物件來調整模型訓練資訊的匯出

`additionalParams` 物件包含若干欄位，您可以使用這些欄位，來指定機器學習類別標籤特徵以供訓練用途，並且指導訓練資料組態檔案的建立。

匯出程序無法自動推斷哪些節點和邊緣屬性應該是機器學習類別標籤，以作為訓練用途的範例。它也無法自動推斷數值、類別和文字屬性的最佳特徵編碼，因此您需要使用 `additionalParams` 物件中的欄位提供提示，來指定這些項目或覆寫預設編碼。

對於屬性圖資料，匯出要求中 `additionalParams` 的最上層結構可能如下所示：

```
{
  "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) ]
    }
  }
}
```

對於 RDF 資料，其最上層結構可能如下所示：

```
{
  "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) ]
    }
  }
}
```

您也可以使用 jobs 欄位來提供多個匯出組態：

```
{
  "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) ]
        }
      ]
    }
  }
}
```

additionalParams 中 neptune_ml 欄位內的最上層元素

neptune_ml中的 version 元素

指定要產生的訓練資料組態版本。

(選用)，類型：字串，預設值：「v2.0」。

如果確實包含 `version`，請將其設為 `v2.0`。

`neptune_ml` 中的 `jobs` 欄位

包含訓練資料組態物件的陣列，每個物件都會定義資料處理工作，並包含：

- **`name`** – 要建立的訓練資料組態名稱。

  例如，名稱為「job-number-1」的訓練資料組態會產生一個名為 `job-number-1.json` 的訓練資料組態檔案。
- **`targets`** – 節點和邊緣類別標籤目標的 JSON 陣列，這些目標代表供訓練用途的機器學習類別標籤。請參閱 neptune_ml 物件中的 targets 欄位。
- **`features`** – 節點屬性特徵的 JSON 陣列。請參閱neptune_ml 中的 features 欄位。

neptune_ml 物件中的 targets 欄位

JSON 訓練資料匯出組態中的 targets 欄位包含目標物件的陣列，這些物件會指定訓練任務，以及用於訓練此任務的機器學習類別標籤。目標物件的內容會有所不同，取決於您是在屬性圖資料還是 RDF 資料上進行訓練。

對於屬性圖節點分類和迴歸任務，陣列中的目標物件可能如下所示：

```
{
  "node": "(node property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

對於屬性圖邊緣分類，迴歸或連結預測任務，它們可以如下所示：

```
{
  "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": ","
}
```

對於 RDF 分類和迴歸任務，陣列中的目標物件可能如下所示：

```
{
  "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]
}
```

對於 RDF 連結預測任務，陣列中的目標物件可以如下所示：

```
{
  "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]
}
```

目標物件可以包含下列欄位：

內容

屬性圖目標物件中的欄位

目標物件中的 **node** (頂點) 欄位

目標節點 (頂點) 的屬性圖標籤。目標物件必須包含 node 元素或 edge 元素，但不能同時包含兩者。

node 可以採取單一值，如下所示：

```
  "node": "Movie"
```

或者，在多標籤頂點的情況下，它可以採取值陣列，如下所示：

```
  "node": ["Content", "Movie"]
```

屬性圖目標物件中的 edge 欄位

透過其起始節點標籤、自己的標籤及其結束節點標籤來指定目標邊緣。目標物件必須包含 edge 元素或 node 元素，但不能同時包含兩者。

edge 欄位的值是一個 JSON 陣列，由三個字串組成，這些字串代表起始節點的屬性圖標籤、邊緣本身的屬性圖標籤，以及結束節點的屬性圖標籤，如下所示：

```
  "edge": ["Person_A", "knows", "Person_B"]
```

如果起始節點和/或結束節點有多個標籤，請以陣列括住它們，如下所示：

```
  "edge": [ ["Admin", Person_A"], "knows", ["Admin", "Person_B"] ]
```

屬性圖目標物件中的 property 欄位

指定目標頂點或邊緣的屬性，如下所示：

```
  "property" : "rating"
```

除了目標任務為連結預測時，此欄位都為必要的。

屬性圖目標物件中的 type 欄位

指出要在 node 或 edge 上執行的目標任務類型，如下所示：

```
  "type" : "regression"
```

節點支援的任務類型如下：

- `classification`
- `regression`

邊緣支援的任務類型如下：

- `classification`
- `regression`
- `link_prediction`

此欄位為必填。

屬性圖目標物件中的 **split_rate** 欄位

(選用) 訓練、驗證和測試階段將分別使用的節點或邊緣比例估計。這些比例是以 JSON 陣列表示，該陣列由介於零與一之間的三個數字組成，加起來最多為 1：

```
"split_rate": [0.7, 0.1, 0.2]
```

如果您未提供選用 split_rate 欄位，則預設估計值為 [0.9, 0.1, 0.0]。

屬性圖目標物件中的 **separator** 欄位

(選用) 搭配分類任務使用。

separator 欄位會指定一個字元，用來在其用來將多個類別值儲存在字串中時，將目標屬性值分割為多個類別值。例如：

```
"separator": "|"
```

separator 欄位的存在表示任務是多目標分類任務。

RDF 目標物件中的欄位

RDF 目標物件中的 **node** 欄位

定義目標節點的節點類型。與節點分類任務或節點迴歸任務搭配使用。RDF 中節點的節點類型，其定義方式如下：

```
  node_id, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, node_type
```

RDF node 只能採取單一值，如下所示：

```
  "node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

RDF 目標物件中的 **subject** 欄位

對於連結預測任務，subject 定義目標邊緣的來源節點類型。

```
  "subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director"
```

> **Note**
>
> 對於連結預測任務，subject 應同時與 predicate 和 object 搭配使用。如果未提供這三個的任一個，則會將所有邊緣視為訓練目標。

RDF 目標物件中的 **predicate** 欄位

對於節點分類和節點迴歸任務，predicate 定義要使用哪個常值資料做為目標節點的目標節點特徵。

```
  "predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre"
```

> **Note**
>
> 如果目標節點只有一個定義目標節點特徵的述詞，則可以省略 predicate 欄位。

對於連結預測任務，predicate 定義目標邊緣的關聯類型：

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/direct"
```

> **Note**
>
> 對於連結預測任務，predicate 應同時與 subject 和 object 搭配使用。如果未提供這三個的任一個，則會將所有邊緣視為訓練目標。

RDF 目標物件中的 **object** 欄位

對於連結預測任務，object 定義目標邊緣的目的地節點類型：

```
  "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

> **Note**
>
> 對於連結預測任務，object 應同時與 subject 和 predicate 搭配使用。如果未提供這三個的任一個，則會將所有邊緣視為訓練目標。

RDF 目標物件中的 **type** 欄位

指示要執行的目標任務類型，如下所示：

```
  "type" : "regression"
```

RDF 資料支援的任務類型如下：

- `link_prediction`
- `classification`
- `regression`

此欄位為必填。

屬性圖目標物件中的 **split_rate** 欄位

(選用) 訓練、驗證和測試階段將分別使用的節點或邊緣比例估計。這些比例是以 JSON 陣列表示，該陣列由介於零與一之間的三個數字組成，加起來最多為 1：

```
"split_rate": [0.7, 0.1, 0.2]
```

如果您未提供選用 `split_rate` 欄位，則預設估計值為 `[0.9, 0.1, 0.0]`。

neptune_ml 中的 features 欄位

屬性值和 RDF 常值有不同的格式和資料類型。若要在機器學習中實現良好的效能，必須將這些值轉換為稱為特徵的數值編碼。

Neptune ML 會執行特徵擷取和編碼，做為資料匯出和資料處理步驟的一部分，如 Neptune ML 中的特徵編碼 中所述。

對於屬性圖資料集，匯出程序會自動針對字串屬性和包含多個值的數值屬性推斷 auto 特徵。對於包含單一值的數值屬性，它會推斷 numerical 特徵。對於日期屬性，它會推斷 datetime 特徵。

如果您想要覆寫自動推斷的特徵規格，或為屬性新增儲存貯體數值、TF-IDF、FastText 或 SBERT 規格，您可以使用特徵欄位控制特徵編碼。

> **Note**
>
> 您只能使用 features 欄位來控制屬性圖資料的特徵規格，而不能控制 RDF 資料的特徵規格。

對於自由格式的文字，Neptune ML 可以使用數個不同的模型，將字串屬性值中的記號序列轉換為固定大小的實數值向量：

- text_fasttext – 使用 fastText 編碼。對於使用 fastText 所支援五種語言的一種且只有一種的功能，這是建議的編碼方式。
- text_sbert – 使用句子 BERT (SBERT) 編碼模型。對於 text_fasttext 不支援的文字，這是建議的編碼方式。
- text_word2vec – 使用 Google 最初發佈的 Word2VEC 演算法來編碼文字。Word2Vec 只支援英文。
- text_tfidf – 使用詞頻-逆向文件頻率 (TF-IDF) 向量化程式來編碼文字。TF-IDF 編碼支援其他編碼不支援的統計特徵。

features 欄位包含節點屬性特徵的 JSON 陣列。陣列中的物件可以包含下列欄位：

內容

- 特徵的 type 欄位可能值
- norm 欄位
- language 欄位
- max_length 欄位
- separator 欄位
- range 欄位
- bucket_cnt 欄位
- slide_window_size 欄位
- imputer 欄位
- max_features 欄位
- min_df 欄位
- ngram_range 欄位
- datetime_parts 欄位

`features` 中的 `node` 欄位

node 欄位指定特徵頂點的屬性圖標籤。例如：

```
  "node": "Person"
```

如果一個頂點有多個標籤，請使用陣列來包含它們。例如：

```
  "node": ["Admin", "Person"]
```

`features` 中的 `edge` 欄位

edge 欄位指定特徵邊緣的邊緣類型。邊緣類型由陣列組成，此陣列包含來源頂點的屬性圖標籤、邊緣的屬性圖標籤，以及目的地頂點的屬性圖標籤。指定邊緣特徵時，您必須提供全部三個值。例如：

```
  "edge": ["User", "reviewed", "Movie"]
```

如果邊緣類型的來源或目標頂點有多個標籤，請使用另一個陣列來包含它們。例如：

```
  "edge": [["Admin", "Person"]. "edited", "Post"]
```

`features` 中的 `property` 欄位

使用屬性參數來指定由 node 參數所識別之頂點的屬性。例如：

```
  "property" : "age"
```

特徵的 `type` 欄位可能值

type 參數指定要定義的特徵類型。例如：

```
  "type": "bucket_numerical"
```

`type` 參數的可能值

- **`"auto"`** – 指定 Neptune ML 應該自動偵測屬性類型並套用適當的特徵編碼。auto 特徵也可以具有一個選用 separator 欄位。

  請參閱 Neptune ML 中的自動特徵編碼。
- **`"category"`** – 此特徵編碼將屬性值表示為數個類別的其中一個。換句話說，此特徵可以採取一個或多個離散值。category 特徵也可以具有一個選用 separator 欄位。

  請參閱 Neptune ML 中的類別特徵。
- **`"numerical"`** – 此特徵編碼會將數值屬性值表示為連續間隔中的數字，其中「大於」和「小於」都有意義。

  numerical 特徵也可以具有選用 norm、imputer 和 separator 欄位。

  請參閱 Neptune ML 中的數值特徵。
- **`"bucket_numerical"`** – 此特徵編碼會將數值屬性值分成一組儲存貯體或類別。

  例如，您可以將人們的年齡編碼為 4 個儲存貯體：小孩 (0-20)、年輕人 (20-40)、中年人 (40-60) 和長者 (60 歲以上)。

  bucket_numerical 特徵需要 range 和 bucket_cnt 欄位，也可以選擇性地包含 imputer 和/或 slide_window_size 欄位。

  請參閱 Neptune ML 中的儲存貯體數值特徵。
- **`"datetime"`** – 此特徵編碼會將 datetime 屬性值表示為下列類別特徵的陣列：年、月、工作日和小時。

可以使用 datetime_parts 參數消除這四個類別中的一個或多個類別。

請參閱 Neptune ML 中的日期時間特徵。

- **"text_fasttext"** – 此特徵編碼會使用 fastText 模型，將由句子或自由格式文字組成的屬性值轉換為數值向量。它支持五種語言，即英文 (en)、中文 (zh)、北印度文 (hi)、西班牙文 (es) 和法文 (fr)。對於這五種語言中任何一種的文字屬性值，text_fasttext 是建議的編碼方式。不過，它無法處理同一句子包含多種語言單字的情況。

  對於 fastText 所支援語言以外的其他語言，請使用 text_sbert 編碼。

  假設，如果您的屬性值文字字串長度超過 120 個記號，請使用 max_length 欄位來限制 "text_fasttext" 編碼的每個字串中的記號數目。

  請參閱 Neptune ML 中文字屬性值的 fastText 編碼。

- **"text_sbert"** – 此編碼會使用句子 BERT (SBERT) 模型，將文字屬性值轉換為數值向量。Neptune 支援兩種 SBERT 方法，即 text_sbert128，這是預設值，如果您僅指定 text_sbert 和 text_sbert512 的話。它們之間的差異在於進行編碼的文字屬性中的記號數目上限。text_sbert128 編碼僅會編碼前 128 個記號，而 text_sbert512 最多可編碼 512 個記號。因此，使用 text_sbert512 可能比 text_sbert128 需要更多的處理時間。這兩種方法都比 text_fasttext 慢。

  text_sbert* 方法支援多種語言，而且可以編碼包含多種語言的句子。

  請參閱 Neptune ML 中文字特徵的句子 BERT (SBERT) 編碼。

- **"text_word2vec"** – 此編碼會使用 Word2VEC 演算法，將文字屬性值轉換為數值向量。它僅支援英文。

  請參閱 Neptune ML 中文字特徵的 Word2Vec 編碼。

- **"text_tfidf"** – 此編碼會使用詞頻-逆向文件頻率 (TF-IDF) 向量化程式，將文字屬性值轉換為數值向量。

  您可以使用 ngram_range 欄位、min_df 欄位和 max_features 欄位來定義 text_tfidf 特徵編碼的參數。

  請參閱 Neptune ML 中文字特徵的 TF-IDF 編碼。

- **"none"** – 使用 none 類型會導致不發生任何特徵編碼。反而會剖析並儲存原始屬性值。

  僅在您計劃執行自己的自訂特徵編碼，做為自訂模型訓練的一部分時，才會使用 none。

`norm` 欄位

數值特徵需要此欄位。它會指定要在數值上使用的標準化方法：

```
"norm": "min-max"
```

支援以下標準化方法：

- 「min-max」 – 透過從中減去最小值，然後將其除以最大值與最小值之間的差值來標準化每個值。
- 「standard」 – 將每個值除以所有值的總和來標準化這些值。
- 「none」 – 不要在編碼期間標準化數值。

請參閱 Neptune ML 中的數值特徵。

`language` 欄位

語言欄位會指定文字屬性值中使用的語言。它的用法取決於文字編碼方法：

- 對於 `text_fasttext` 編碼，此欄位是必要的，且必須指定下列其中一種語言：
  - `en` (英文)
  - `zh` (中文)
  - `hi` (北印度文)
  - `es` (西班牙文)
  - `fr` (法文)
- 對於 `text_sbert` 編碼，不會使用此欄位，因為 SBERT 編碼是多語系的。
- 對於 `text_word2vec` 編碼，此欄位是選用的，因為 `text_word2vec` 僅支援英文。如果存在，它必須指定英文模型的名稱：

  ```
  "language" : "en_core_web_lg"
  ```
- 對於 `text_tfidf` 編碼，不會使用此欄位。

`max_length` 欄位

`max_length` 欄位是 `text_fasttext` 特徵的選用欄位，其中指定將編碼之輸入文字特徵中的記號數目上限。長度超過 `max_length` 的輸入文字會被截斷。例如，將 max_length 設定為 128，表示將忽略文字序列中第 128 之後的任何記號：

```
"max_length": 128
```

`separator` 欄位

此欄位可選擇性地與 `category`、`numerical` 和 `auto` 特徵搭配使用。它會指定一個字元，其可以用來將屬性值分成多個類別值或數值：

```
"separator": ";"
```

僅在屬性將多個分隔值儲存在單一字串 (例如 `"Actor;Director"` 或 `"0.1;0.2"`) 中時，才會使用 `separator` 欄位。

請參閱類別功能、數值特徵和自動編碼。

`range` 欄位

`bucket_numerical` 特徵需要此欄位。它會指定要分成儲存貯體的數值範圍，格式為 [*lower-bound*, *upper-bound*]：

```
"range" : [20, 100]
```

如果屬性值小於下限，則會將其指派給第一個儲存貯體，或者如果它大於上限，則會將其指派給最後一個儲存貯體。

請參閱 Neptune ML 中的儲存貯體數值特徵。

`bucket_cnt` 欄位

`bucket_numerical` 特徵需要此欄位。它會指定由 `range` 參數定義的數值範圍應分成的儲存貯體數目：

```
"bucket_cnt": 10
```

請參閱 Neptune ML 中的儲存貯體數值特徵。

`slide_window_size` 欄位

此欄位可選擇性地搭配 `bucket_numerical` 特徵使用，以將值指派給多個儲存貯體：

```
"slide_window_size": 5
```

滑動視窗的運作方式是 Neptune ML 採取視窗大小 **s**，並將屬性的每個數值 **v** 轉換為從 `v - s/2` 到 `v + s/2` 的範圍。然後，此值會指定給範圍重疊的每個儲存貯體。

請參閱 Neptune ML 中的儲存貯體數值特徵。

`imputer` 欄位

此欄位可選擇搭配 `numerical` 和 `bucket_numerical` 特徵使用，以提供填入缺失值的插補技術：

```
"imputer": "mean"
```

支援的插補技術如下：

- `"mean"`
- `"median"`
- `"most-frequent"`

如果您未包含 imputer 參數，資料預先處理會在遇到缺失值時停止並結束。

請參閱 Neptune ML 中的數值特徵 和 Neptune ML 中的儲存貯體數值特徵。

`max_features` 欄位

此欄位是 `text_tfidf` 特徵可選擇性地用來指定要編碼的詞數上限：

```
"max_features": 100
```

設定為 100 會導致 TF-IDF 向量化程式只編碼 100 個最常見的字詞。如果未包含 `max_features`，則預設值為 5,000。

請參閱 Neptune ML 中文字特徵的 TF-IDF 編碼。

`min_df` 欄位

`text_tfidf` 特徵可選擇性地使用此欄位，來指定要編碼之字詞的文件頻率上限：

```
"min_df": 5
```

設定 5 表示字詞必須出現在至少 5 個不同的屬性值中，才能進行編碼。

如果您未包含 `min_df` 參數，預設值為 2。

請參閱 Neptune ML 中文字特徵的 TF-IDF 編碼。

`ngram_range` 欄位

`text_tfidf` 特徵可選擇性地使用此欄位，指定應將單字或記號的哪些大小序列視為要編碼的潛在個別字詞：

```
"ngram_range": [2, 4]
```

值 `[2, 4]` 指定 2、3 和 4 個單字的序列應被視為潛在的個別字詞。

如果您沒有明確設定 `ngram_range`，則預設值為 `[1, 1]`，這表示只有單一單字或記號會被視為要編碼的字詞。

請參閱 Neptune ML 中文字特徵的 TF-IDF 編碼。

`datetime_parts` 欄位

`datetime` 特徵可選擇性地使用此欄位，來指定要以分類方式編碼 datetime 值的哪些部分：

```
"datetime_parts": ["weekday", "hour"]
```

如果您未包含 `datetime_parts`，Neptune ML 預設會編碼 datetime 值的年、月、工作日和小時部分。值 `["weekday", "hour"]` 表示只有 datetime 值的工作日和小時應在特徵中進行分類編碼。

如果其中一個部分在訓練集中沒有多個唯一值，則不會對其進行編碼。

請參閱Neptune ML 中的日期時間特徵。

在 `additionalParams` 內使用參數調整模型訓練組態的範例

內容

使用 **additionalParams** 的屬性圖範例

為模型訓練組態指定預設分割率

在下列範例中，split_rate 參數會設定模型訓練的預設分割率。如果未指定預設分割率，則訓練會使用 [0.9, 0.1, 0.0] 的值。您可以透過為每個目標指定 split_rate，覆寫每個目標上的預設值。

在下列範例中，default split_rate 欄位指示除非針對每個目標進行覆寫，否則應使用 [0.7,0.1,0.2] 的分割率：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ],
    "features": [
      (...)
    ]
  }
}
```

為模型訓練組態指定節點分類任務

若要指示哪個節點屬性包含標記的範例可供訓練用途，請使用 "type" : "classification" 將節點分類元素新增至 targets 陣列。如果想要覆寫預設分割率，請新增 split_rate 欄位。

在下列範例中，node 目標會指示應將每個 Movie 節點的 genre 屬性視為節點類別標籤。split_rate 值會覆寫預設分割率：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
```

```
    "features": [
      (...)
    ]
  }
}
```

為模型訓練組態指定多類別節點分類任務

若要指示哪個節點屬性包含多個標記的範例可供訓練用途，請使用 "type" : "classification" 和 separator 將節點分類元素新增至目標陣列，來指定可以用來將目標屬性值分割為多個類別值的字元。如果想要覆寫預設分割率，請新增 split_rate 欄位。

在下列範例中，node 目標會指示應將每個 Movie 節點的 genre 屬性視為節點類別標籤。separator 欄位指示每個類型屬性包含多個分號分隔值：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "separator": ";"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

為模型訓練組態指定節點迴歸任務

若要指示哪個節點屬性包含用於訓練用途的標記迴歸，請使用 "type" : "regression" 將節點迴歸元素新增至陣列。如果想要覆寫預設分割率，請新增 split_rate 欄位。

下列 node 目標指示應將每個 Movie 節點的 rating 屬性視為節點迴歸標籤。

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
```

```
    "targets": [
      {
        "node": "Movie",
        "property": "rating",
        "type" : "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      ...
    ]
  }
}
```

為模型訓練組態指定 edge-classification 任務

若要指示哪個邊緣屬性包含用於訓練用途的標記範例，請使用 `"type" : "regression"` 將邊緣元素新增至 `targets` 陣列。如果想要覆寫預設分割率，請新增 split_rate 欄位。

下列 edge 目標指示應將每個 knows 邊緣的 `metAtLocation` 屬性視為迴歸類別標籤：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "knows", "Person"],
        "property": "metAtLocation",
        "type": "classification"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

為模型訓練組態指定多類別邊緣分類任務

若要指示哪個邊緣屬性包含多個用於訓練用途的標記範例，請使用 `"type" : "classification"` 和 `separator` 欄位將邊緣元素新增至 `targets` 陣列，來指定用來將目標屬性值分割為多個類別值的字元。如果想要覆寫預設分割率，請新增 `split_rate` 欄位。

下列 edge 目標指示應將每個 repliedTo 邊緣的 sentiment 屬性視為迴歸類別標籤。分隔符號欄位會指示每個情緒屬性都包含多個逗號分隔值：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "repliedTo", "Message"],
        "property": "sentiment",
        "type": "classification",
        "separator": ","
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

為模型訓練組態指定邊緣迴歸

若要指示哪個邊緣屬性包含用於訓練用途的標記迴歸範例，請使用 "type" : "regression" 將 edge 元素新增至 targets 陣列。如果想要覆寫預設分割率，請新增 split_rate 欄位。

下列 edge 目標指示應將每個 reviewed 邊緣的 rating 屬性視為邊緣迴歸：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "reviewed", "Movie"],
        "property": "rating",
        "type" : "regression"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

為模型訓練組態指定連結預測任務

若要指示應將哪些邊緣用於連結預測訓練用途，請使用 "type" : "link_prediction" 將邊緣元素新增至目標陣列。如果想要覆寫預設分割率，請新增 split_rate 欄位。

下列 edge 目標指示應該使用 cites 邊緣進行連結預測：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Article", "cites", "Article"],
        "type" : "link_prediction"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

指定數值儲存貯體特徵

您可以透過將 "type": "bucket_numerical" 新增至 features 陣列來為節點屬性指定數值資料特徵。

下列 node 特徵指示應將每個 Person 節點的 age 屬性視為數值儲存貯體特徵：

```
  "additionalParams": {
  "neptune_ml": {
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Person",
        "property": "age",
        "type": "bucket_numerical",
        "range": [1, 100],
        "bucket_cnt": 5,
        "slide_window_size": 3,
```

```
        "imputer": "median"
      }
    ]
  }
}
```

指定 **Word2Vec** 特徵

您可以透過將 "type": "text_word2vec" 新增至 features 陣列來為節點屬性指定 Word2Vec 特徵。

下列 node 特徵指示應將每個 Movie 節點的 description 屬性視為 Word2Vec 特徵：

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_word2vec",
        "language": "en_core_web_lg"
      }
    ]
  }
}
```

指定 **FastText** 特徵

您可以透過將 "type": "text_fasttext" 新增至 features 陣列來為節點屬性指定 FastText 特徵。language 欄位為必要的，且必須指定下列其中一個語言代碼：

- en (英文)
- zh (中文)
- hi (北印度文)
- es (西班牙文)
- fr (法文)

請注意，text_fasttext 編碼無法在特徵中一次處理多種語言。

下列 node 特徵指示應將每個 Movie 節點的法文 description 屬性視為 FastText 特徵：

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_fasttext",
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
}
```

指定 Sentence BERT 特徵

您可以透過將 "type": "text_sbert" 新增至 features 陣列來為節點屬性指定 Sentence BERT 特徵。您不需要指定語言，因為方法會使用多語系模型自動編碼文字特徵。

下列 node 特徵指示應將每個 Movie 節點的 description 屬性視為 Sentence BERT 特徵：

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_sbert128",
      }
    ]
  }
```

```
}
```

指定 TF-IDF 特徵

您可以透過將 "type": "text_tfidf" 新增至 features 陣列來為節點屬性指定 TF-IDF 特徵。

下列 node 特徵指示應將每個 Person 節點的 bio 屬性視為 TF-IDF 特徵：

```
  "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
      }
    ]
  }
}
```

指定 datetime 特徵

匯出程序會自動推斷日期屬性的 datetime 特徵。不過，如果要限制用於 datetime 特徵的 datetime_parts，或覆寫特徵規格，以便將通常被視為 auto 特徵的屬性明確地視為 datetime 特徵，則您可以透過將特徵陣列新增至 "type": "datetime" 來執行此操作。

下列 node 特徵指示應將每個 Post 節點的 createdAt 屬性視為 datetime 特徵：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
```

```
      {
        "node": "Post",
        "property": "createdAt",
        "type": "datetime",
        "datetime_parts": ["month", "weekday", "hour"]
      }
    ]
  }
}
```

指定 **category** 特徵

匯出程序會自動針對字串屬性和包含多個值的數值屬性推斷 auto 特徵。對於包含單一值的數值屬性，它會推斷 numerical 特徵。對於日期屬性，它會推斷 datetime 特徵。

如果您想要覆寫特徵規格，以便將屬性視為類別特徵，請將 "type": "category" 新增至特徵陣列。如果屬性包含多個值，請包括一個 separator 欄位。例如：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "tag",
        "type": "category",
        "separator": "|"
      }
    ]
  }
}
```

指定 **numerical** 特徵

匯出程序會自動針對字串屬性和包含多個值的數值屬性推斷 auto 特徵。對於包含單一值的數值屬性，它會推斷 numerical 特徵。對於日期屬性，它會推斷 datetime 特徵。

如果您想要覆寫特徵規格，以便將屬性視為 numerical 特徵，請將 "type": "numerical" 新增至特徵陣列。如果屬性包含多個值，請包括一個 separator 欄位。例如：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Recording",
        "property": "duration",
        "type": "numerical",
        "separator": ","
      }
    ]
  }
}
```

指定 **auto** 特徵

匯出程序會自動針對字串屬性和包含多個值的數值屬性推斷 auto 特徵。對於包含單一值的數值屬性，它會推斷 numerical 特徵。對於日期屬性，它會推斷 datetime 特徵。

如果您想要覆寫特徵規格，以便將屬性視為 auto 特徵，請將 "type": "auto" 新增至特徵陣列。如果屬性包含多個值，請包括一個 separator 欄位。例如：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "User",
        "property": "role",
        "type": "auto",
        "separator": ","
      }
    ]
  }
}
```

使用 **additionalParams** 的 RDF 範例

為模型訓練組態指定預設分割率

在下列範例中，split_rate 參數會設定模型訓練的預設分割率。如果未指定預設分割率，則訓練會使用 [0.9, 0.1, 0.0] 的值。您可以透過為每個目標指定 split_rate，覆寫每個目標上的預設值。

在下列範例中，default split_rate 欄位指示除非針對每個目標進行覆寫，否則應使用 [0.7,0.1,0.2] 的分割率：

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ]
  }
}
```

為模型訓練組態指定節點分類任務

若要指示哪個節點屬性包含訓練用途的標記範例，請使用 "type" : "classification" 將節點分類元素新增至 targets 陣列。新增節點欄位以指示目標節點的節點類型。新增 predicate 欄位以定義使用哪個常值資料做為目標節點的目標節點特徵。如果想要覆寫預設分割率，請新增 split_rate 欄位。

在下列範例中，node 目標會指示應將每個 Movie 節點的 genre 屬性視為節點類別標籤。split_rate 值會覆寫預設分割率：

```
  "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]
      }
    ]
  }
```

```
}
```

為模型訓練組態指定節點迴歸任務

若要指示哪個節點屬性包含用於訓練用途的標記迴歸，請使用 "type" : "regression" 將節點迴歸元素新增至陣列。新增 node 欄位以指示目標節點的節點類型。新增 predicate 欄位以定義使用哪個常值資料做為目標節點的目標節點特徵。如果想要覆寫預設分割率，請新增 split_rate 欄位。

下列 node 目標指示應將每個 Movie 節點的 rating 屬性視為節點迴歸標籤。

```
    "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]
      }
    ]
  }
}
```

為特定邊緣指定連結預測任務

若要指示應將哪些邊緣用於連結預測訓練用途，請使用 "type" : "link_prediction" 將邊緣元素新增至目標陣列。新增 subject、predicate 和 object 欄位以指定邊緣類型。如果想要覆寫預設分割率，請新增 split_rate 欄位。

下列 edge 目標指示將 Directors 連線至 Movies 的 directed 邊緣應該用於連結預測：

```
  "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"
      }
```

```
    ]
  }
}
```

為所有邊緣指定連結預測任務

若要指示應將所有邊緣用於連結預測訓練用途，請使用 "type" : "link_prediction" 將 edge 元素新增至目標陣列。請不要新增 subject、predicate 或 object 欄位。如果想要覆寫預設分割率，請新增 split_rate 欄位。

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "type" : "link_prediction"
      }
    ]
  }
}
```

處理從 Neptune 匯出用於訓練的圖形資料

資料處理步驟會取得匯出程序所建立的 Neptune 圖形資料，並建立 Deep Graph Library (DGL) 在訓練期間使用的資訊。這包括執行各種資料對應和轉換：

- 解析節點和邊緣以建構 DGL 所需的圖形和 ID 對應檔案。
- 將節點和邊緣屬性轉換為 DGL 所需的節點和邊緣特徵。
- 將資料分割為訓練、驗證和測試集。

管理 Neptune ML 的資料處理步驟

在從 Neptune 匯出了您要用於模型訓練的資料之後，您可以使用 `curl` (或 `awscurl`) 命令啟動資料處理工作，如下所示：

```
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"
      }'
```

如何使用此命令的詳細資訊會在 dataprocessing 命令 中加以說明，伴隨如何取得執行中工作狀態、如何停止執行中工作，以及如何列出所有執行中工作的相關資訊。

處理 Neptune ML 的更新圖形資料

您也可以將 `previousDataProcessingJobId` 提供給 API，以確保新的資料處理工作使用與先前工作相同的處理方法。當您想要透過對新資料重新訓練舊模型，或對新資料上重新計算模型成品，以在 Neptune 中取得更新圖形資料的預測時，這是必要的。

您可以使用 `curl` (或 `awscurl`) 命令來執行此操作，如下所示：

```
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)"}'
```

將 previousDataProcessingJobId 參數值設定為對應至訓練模型之先前資料處理工作的工作 ID。

> **Note**
>
> 目前不支援更新圖形中的節點刪除。如果節點已在更新圖形中移除，您必須啟動全新的資料處理工作，而不是使用 previousDataProcessingJobId。

Neptune ML 中的特徵編碼

屬性值有不同的格式和資料類型。若要在機器學習中實現良好的效能，必須將這些值轉換為稱為特徵的數值編碼。

Neptune ML 會使用這裡描述的特徵編碼技術，執行特徵擷取和編碼，做為資料匯出和資料處理步驟的一部分。

> **Note**
>
> 如果您計劃在自訂模型實作中實作自己的特徵編碼，則可以選取 none 做為特徵編碼類型，來停用資料預先處理階段中的自動特徵編碼。然後，不會對該節點或邊緣屬性進行任何特徵編碼，而是剖析原始屬性值並將其儲存在字典中。資料預先處理仍會從匯出的資料集建立 DGL 圖形，但是建構的 DGL 圖形沒有預先處理的訓練特徵。
> 僅在您計劃執行自訂特徵編碼，做為自訂模型訓練的一部分時，才應使用此選項。如需詳細資訊，請參閱 Neptune ML 中的自訂模型。

Neptune ML 中的類別特徵

可以從可能值的固定清單中獲得一個或多個不同值的屬性，就是類別特徵。在 Neptune ML 中，類別特徵會使用獨熱編碼進行編碼。以下範例顯示了不同食物的屬性名稱如何根據其類別進行獨熱編碼：

```
 Food        Veg.   Meat   Fruit    Encoding
---------    ----   ----   -----    --------
 Apple         0      0      1         001
 Chicken       0      1      0         010
 Broccoli      1      0      0         100
```

> **Note**
>
> 任何類別特徵中的類別數目上限為 100。如果一個屬性具有超過 100 種類別的值，則僅會將其中最常見的 99 個放置在不同的類別中，其餘則會放置在名為 OTHER 的特殊類別中。

Neptune ML 中的數值特徵

其值為實數的任何屬性都可以編碼為 Neptune ML 中的數值特徵。數值特徵會使用浮點數進行編碼。

您可以指定要在編碼數值特徵時使用的資料標準化方法，如下所示："norm": "*normalization technique*"。支援以下標準化技術：

- 「none」 – 不要在編碼期間標準化數值。
- 「min-max」 – 透過從中減去最小值，然後將其除以最大值與最小值之間的差值來標準化每個值。
- 「standard」 – 將每個值除以所有值的總和來標準化這些值。

Neptune ML 中的儲存貯體數值特徵

您可以將數值壓縮為類別，而不是使用原始數字表示數值屬性。例如，您可以將人們的年齡分成兒童 (0-20 歲)、年輕人 (20-40 歲)、中年人 (40-60 歲) 和老年人 (從 60 歲開始)。使用這些數值儲存貯體，您可以將數值屬性轉換為一種類別特徵。

在 Neptune ML 中，您可能會導致數值屬性編碼為儲存貯體數值特徵。您必須提供兩個項目：

- 格式為 "range": [*a*, *b*] 的數值範圍，其中 a 和 b 是整數。
- 格式為 "bucket_cnt": *c* 的儲存貯體計數，其中 c 是儲存貯體的數目，也是一個整數。

然後，Neptune ML 會將每個儲存貯體的大小計算為 (b - a) / c ，並將每個數值編碼為其落入的任何儲存貯體的數目。任何小於 a 的值都會被視為屬於第一個儲存貯體，而任何大於 b 的值都會被視為屬於最後一個儲存貯體。

您也可以選擇性地使數值落入多個儲存貯體，方法是指定滑動視窗大小，如下所示：
"slide_window_size": *s* ，其中 s 是一個數字。Neptune ML 接著會將屬性的每個數值 v 轉換為從 v - s/2 到 v + s/2 的範圍，並將值 v 指派給範圍涵蓋的每個儲存貯體。

最後，您還可以選擇性地提供一種為數值特徵和儲存貯體數值特徵填入缺失值的方法。您可以使用 "imputer": "*imputation technique* " 執行此操作，其中插補技術是 "mean"、"median" 或 "most-frequent" 之一。如果您未指定 imputer，缺失值可能會導致處理停止。

Neptune ML 中的文字特徵編碼

對於自由格式的文字，Neptune ML 可以使用數個不同的模型，將屬性值字串中的記號序列轉換為固定大小的實數值向量：

- text_fasttext – 使用 fastText 編碼。對於使用 fastText 所支援五種語言的一種且只有一種的特徵，這是建議的編碼方式。

- `text_sbert` – 使用句子 BERT (SBERT) 編碼模型。對於 `text_fasttext` 不支援的文字，這是建議的編碼方式。
- `text_word2vec` – 使用 Google 最初發佈的 Word2VEC 演算法來編碼文字。Word2Vec 只支援英文。
- `text_tfidf` – 使用詞頻-逆向文件頻率 (TF-IDF) 向量化程式來編碼文字。TF-IDF 編碼支援其他編碼不支援的統計特徵。

Neptune ML 中文字屬性值的 fastText 編碼

Neptune ML 可以使用 fastText 模型，將文字屬性值轉換為固定大小的實數值向量。對於 FastText 支援的五種語言中任何一種語言中的文字屬性值，這是建議的編碼方法：

- `en` (英文)
- `zh` (中文)
- `hi` (北印度文)
- `es` (西班牙文)
- `fr` (法文)

請注意，fastText 無法處理包含多種語言單字的句子。

`text_fasttext` 方法可以選擇性地採取 `max_length` 欄位，指定將編碼之文字屬性值中的記號數上限，之後字串會遭截斷。當文字屬性值包含長字串時，這可以改善效能，因為如果未指定 `max_length`，fastText 會編碼所有記號，而不論字串長度為何。

這個範例指定法文電影標題使用 fastText 進翰編碼：

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_fasttext"],
        "language": "fr",
        "max_length": 1024
      }
    ]
```

```
  }
```

Neptune ML 中文字特徵的句子 BERT (SBERT) 編碼

Neptune ML 可以使用句子 BERT (SBERT) 模型，將字串屬性值中的記號序列轉換為固定大小的實數值向量。Neptune 支援兩種 SBERT 方法：`text_sbert128`，這是預設值，如果您僅指定 `text_sbert` 和 `text_sbert512` 的話。兩者之間的差異是編碼的文字屬性值字串的長度上限。`text_sbert128` 編碼會在編碼 128 個記號之後截斷文字字串，而 `text_sbert512` 會在編碼 512 個記號之後截斷文字字串。因此，`text_sbert512` 可能比 `text_sbert128` 需要更多的處理時間。這兩種方法都比 `text_fasttext` 慢。

SBERT 編碼是多語系，因此無需為您正要編碼的屬性值文字指定語言。SBERT 方法支援多種語言，而且可以編碼包含多種語言的句子。如果您要編碼的屬性值包含以 FastText 不支援的語言撰寫的文字，則 SBERT 是建議的編碼方法。

下列範例會指定影片標題編碼為 SBERT，最多可有 128 個記號：

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      { "feature": ["title", "title", "text_sbert128"] }
    ]
  }
```

Neptune ML 中文字特徵的 Word2Vec 編碼

Neptune ML 可以將字串屬性值編碼為 Word2Vec 特徵 (Word2Vec 演算法最初是由 Google 發布)。`text_word2vec` 方法會使用其中一個 spaCy 訓練模型，將字串中的記號編碼為密集向量。這只支援使用 en_core_web_lg 模型的英文。

下列範例會指定影片標題使用 Word2VEC 進行編碼：

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
```

```
        "feature": ["title", "title", "text_word2vec"],
        "language": "en_core_web_lg"
      }
    ]
  }
```

請注意，語言欄位是選用的，因為英文 `en_core_web_lg` 模型是 Neptune 支援的唯一模型。

Neptune ML 中文字特徵的 TF-IDF 編碼

Neptune ML 可以將文字屬性值編碼為 `text_tfidf` 特徵。這種編碼會使用詞頻-逆向文件頻率 (TF-IDF) 向量化程式，接著進行維數縮減操作，將文字中的單字序列轉換為數值向量。

TF-IDF (詞頻 - 逆向文件頻率) 是旨在測量單字在文件集中有多重要的數值。它的計算方法是將單字在給定屬性值中出現的次數，除以其出現所在的此類屬性值的總數。

例如，如果在給定電影標題中出現兩次「kiss」一字 (例如，「kiss kiss bang bang」)，並且「kiss」出現在所有 4 部電影的標題中，則「kiss kiss bang bang」標題中「kiss」的 TF-IDF 值將是 `2 / 4`。

最初建立的向量具有 d 維度，其中 d 是該類型之所有屬性值中獨特字詞的數目。維數縮減操作會使用隨機稀疏投影，將該數目減少至上限 (100)。然後，透過合併其中的所有 `text_tfidf` 特徵來產生圖形的詞彙。

您可以透過多種方式控制 TF-IDF 向量化程式：

- **`max_features`** – 使用 `max_features` 參數，您可以將 `text_tfidf` 特徵中的字詞數限制為最常用的字詞。例如，如果您將 `max_features` 設為 100，則只會包含前 100 個最常用的字詞。如果您未明確地設定它，則 `max_features` 的預設值為 5,000。
- **`min_df`** – 使用 `min_df` 參數，您可以將 `text_tfidf` 特徵中的字詞數限制為至少具有所指定文件頻率的字詞。例如，如果將 `min_df` 設為 5，則只會使用出現在至少 5 個不同屬性值的字詞。如果您未明確地設定它，則 `min_df` 的預設值為 2。
- **`ngram_range`** – `ngram_range` 參數決定要將哪些單字組合會被視為字詞。例如，如果您將 `ngram_range` 設為 `[2, 4]`，則將會在「kiss kiss bang bang」標題中找到以下 6 個字詞：
  - 2 字字詞：「kiss kiss」、「kiss bang」和「bang bang」。
  - 3 字字詞：「kiss kiss bang」和「kiss bang bang」。
  - 4 字字詞：「kiss kiss bang bang」。

  `ngram_range` 的預設設定為 `[1, 1]`。

Neptune ML 中的日期時間特徵

Neptune ML 可以將 datetime 屬性值的一部分編碼為類別特徵，方法是將它們轉換為獨熱陣列。使用 datetime_parts 參數指定下列一或多個要編碼 ["year", "month", "weekday", "hour"] 的部分：如果未設定 datetime_parts，預設為編碼所有四個部分。

例如，如果日期時間值的範圍橫越 2010 到 2012 年，則日期時間項目 2011-04-22 01:16:34 的四個部分如下：

- year – [0, 1, 0]。

  由於跨度 (2010、2011 和 2012) 只有 3 年，因此獨熱陣列有三個項目，每年一個。

- month – [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]。

  在這裡，獨熱陣列對於一年中的每個月都有一個項目。

- weekday – [0, 0, 0, 0, 1, 0, 0]。

  ISO 8601 標準規定，星期一是一周的第一天，並且由於 2011 年 4 月 22 日是星期五，所以對應的獨熱工作日陣列在第五位置是熱的。

- 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]。

  凌晨 1 點設定在 24 個成員組成的獨熱陣列中。

幾月幾日、分鐘和秒不會進行類別編碼。

如果有問題的總計 datetime 範圍僅包含一年內的日期，則不會編碼任何 year 陣列。

您可以使用 imputer 參數和可用於數值特徵的其中一個策略，來指定插補策略以填入缺失的 datetime 值。

Neptune ML 中的自動特徵編碼

您可以將 auto 設定為特徵編碼方法，而不是手動指定要用於圖形中屬性的特徵編碼方法。Neptune ML 接著會嘗試根據其基礎資料類型推斷每個屬性的最佳特徵編碼。

以下是 Neptune ML 在選取適當特徵編碼時使用的一些啟發式法：

- 如果屬性只有數值，並且可以轉換為數值資料類型，則 Neptune ML 通常會將其編碼為數值。不過，如果內容的唯一值數目小於值總數的 10%，且這些唯一值的基數小於 100，則 Neptune ML 會使用類別編碼。
- 如果屬性值可以轉換為 `datetime` 類型，則 Neptune ML 將它們編碼為 `datetime` 特徵。
- 如果屬性值可以強制轉換為布林值 (1/0 或 True /False)，則 Neptune ML 會使用類別編碼。
- 如果屬性是其中 10% 以上是唯一值的字串，且每個值的平均記號數大於或等於 3，則 Neptune ML 會推斷屬性類型為文字，並自動偵測正在使用的語言。如果偵測到的語言是 fastText 支援其中一種語言，即英文、中文、北印度文、西班牙語和法文，則 Neptune ML 會使用 `text_fasttext` 來編碼文字。否則，Neptune ML 會使用 `text_sbert`。
- 如果屬性是未分類為文字特徵的字串，則 Neptune ML 會假定它是類別特徵，並使用類別編碼。
- 如果每個節點對於推斷為類別特徵的屬性都有自己的唯一值，Neptune ML 會從訓練圖表中捨棄屬性，因為它可能是不會對學習提供資訊的 ID。
- 如果已知屬性包含有效的 Neptune 分隔符號，例如分號 (「;」)，則 Neptune ML 只能將該屬性視為 `MultiNumerical` 或 `MultiCategorical`。
  - Neptune ML 首先嘗試將值編碼為數值特徵。如果成功，Neptune ML 會使用數值編碼來建立數值向量特徵。
  - 否則，Neptune ML 會將這些值編碼為多重類別。
- 如果 Neptune ML 無法推斷屬性值的資料類型，Neptune ML 會從訓練圖中捨棄屬性。

編輯訓練資料組態檔案

Neptune 匯出程序會將 Neptune ML 資料從 Neptune 資料庫叢集匯出至 S3 儲存貯體。它將節點和邊緣分別匯出至 nodes/ 和 edges/ 資料夾。根據預設，它也會建立 JSON 訓練資料組態檔案，名為 training-data-configuration.json。此檔案包含有關圖形結構描述、其特徵類型、特徵轉換和標準化操作，以及分類或迴歸任務之目標特徵的資訊。

可能有些情況，您想要直接修改組態檔案。這類情況是，您想要變更特徵的處理方式或建構圖形的方式，而不需要在每次您想要針對您要解決的機器學習任務修改其規格時重新執行匯出。

編輯訓練資料組態檔案

1. 將檔案下載到您的本機電腦。

   除非您在傳遞至匯出程序的 additionalParams/neptune_ml 參數中指定了一或多個具名工作，否則檔案將具有預設名稱，即 training-data-configuration.json。您可以使用如下的 AWS CLI 命令來下載檔案：

   ```
   aws s3 cp \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json \
     ./
   ```

2. 使用文字編輯器編輯檔案。
3. 上傳修改後的檔案。使用如下的 AWS CLI 命令，將修改後的檔案上傳回 Amazon S3 中您之前下載它的同一位置：

   ```
   aws s3 cp \
     training-data-configuration.json \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json
   ```

JSON 訓練資料組態檔案的範例

以下是描述節點分類任務圖形的範例訓練資料組態檔案：

```
{
  "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" : [ ]
  ]
}
```

JSON 訓練資料組態檔案的結構

訓練組態檔案是指匯出程序儲存在 nodes/ 和 edges/ 資料夾中的 CSV 檔案。

nodes/ 下的每個檔案都會儲存具有相同屬性圖節點標籤之節點的相關資訊。節點檔案中的每一資料行都會儲存節點 ID 或節點屬性。檔案的第一行包含一個標頭，為每一資料行指定 ~id 或屬性名稱。

edges/ 下的每個檔案都會儲存具有相同屬性圖邊緣標籤之節點的相關資訊。節點檔案中的每一資料行都會儲存節點 ID、目的地節點 ID 或邊緣屬性。檔案的第一行包含一個標頭，為每一資料行指定 ~from、~to 或屬性名稱。

訓練資料組態檔案具有三個最上層元素：

```
{
```

```
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [ ... ]
}
```

- `version` – (字串) 正在使用的組態檔案版本。
- `query_engine` – (字串) 用於匯出圖形資料的查詢語言。目前，僅「gremlin」是有效的。
- `graph` – (JSON 陣列) 列出一或多個組態物件，其中包含每個將使用之節點和邊緣的模型參數。

  圖形陣列中的組態物件具有下一節所述的結構。

`graph` 陣列中列出之組態物件的內容

`graph` 陣列中的組態物件可以包含三個最上層節點：

```
   {
     "edges"    : [ ... ],
     "nodes"    : [ ... ],
     "warnings" : [ ... ],
   }
```

- `edges` – (JSON 物件陣列) 每個 JSON 物件都會指定一組參數，以定義在模型處理和訓練期間將如何處理圖形中的邊緣。這僅會與 Gremlin 引擎搭配使用。
- `nodes` – (JSON 物件陣列) 每個 JSON 物件都會指定一組參數，以定義在模型處理和訓練期間將如何處理圖形中的節點。這僅會與 Gremlin 引擎搭配使用。
- `warnings` – (JSON 物件陣列) 每個物件都包含在資料匯出過程中產生的警告。

`edges` 陣列中列出之邊緣組態物件的內容

`edges` 陣列中列出的邊緣組態物件可以包含下列最上層欄位：

```
     {
       "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** – 字串，其會指定 CSV 檔案的路徑，而此檔案會儲存具有相同屬性圖標籤之邊緣的相關資訊。

  該檔案的第一行包含資料行標籤的標頭行。

  前兩個資料行標籤為 ~from 和 ~to。第一個資料行 (~from 資料行) 儲存邊緣起始節點的 ID，而第二個資料行 (~to 資料行) 儲存邊緣結束節點的 ID。

  標頭行中的其餘資料行標籤會為每個剩餘的資料行指定其值已匯出至該資料行的邊緣屬性名稱。
- **separator** – 字串，其中包含分隔該 CSV 檔案中資料行的分隔符號。
- **source** – JSON 陣列，其中包含兩個指定邊緣起始節點的字串。第一個字串包含起始節點 ID 儲存所在資料行的標頭名稱。第二個字串指定節點類型。
- **relation** – JSON 陣列，其中包含兩個指定邊緣關聯類型的字串。第一個字串包含關聯名稱 (relname) 儲存所在資料行的標頭名稱。第二個字串包含關聯名稱 (prefixname) 的字首。

  完整的關聯類型由兩個字串組合而成，它們之間有一個連字號，如下所示：*prefixname*-*relname*。

  如果第一個字串是空的，則所有邊緣都具有相同的關聯類型，即 prefixname 字串。
- **dest** – JSON 陣列，其中包含兩個指定邊緣結束節點的字串。第一個字串包含節點 ID 儲存所在資料行的標頭名稱。第二個字串指定節點類型。
- **features** – 屬性值特徵物件的 JSON 陣列。每個屬性值特徵物件包含下列欄位：
  - feature – 由三個字串組成的 JSON 陣列。第一個字串包含資料行的標頭名稱，而該資料行包含屬性值。第二個字串包含特徵名稱。第三個字串包含特徵類型。
  - norm – (選用) 指定要套用至屬性值的標準化方法。
- **labels** – 物件的 JSON 陣列。每個物件都會定義邊緣的目標特徵，並指定訓練和驗證階段應採取的邊緣比例。每個物件包含下列欄位：
  - label – 由兩個字串組成的 JSON 陣列。第一個字串包含資料行的標頭名稱，而該資料行包含目標特徵屬性值。第二個字串會指定下列其中一個目標任務類型：
    - "classification" – 邊緣分類任務。label 陣列中第一個字串所識別之資料行中提供的屬性值會被視為類別值。對於邊緣分類任務，label 陣列中的第一個字串不能是空的。

  - `"regression"` – 邊緣迴歸任務。`label` 陣列中第一個字串所識別之資料行中提供的屬性值會被視為數值。對於邊緣迴歸任務，`label` 陣列中的第一個字串不能是空的。
  - `"link_prediction"` – 連結預測任務。不需要任何屬性值。對於連結預測任務，會忽略 `label` 陣列中的第一個字串。
- **`split_rate`** – JSON 陣列，其中包含三個介於零和一之間的數字，這些數字加起來最多為一，並代表訓練、驗證和測試階段將分別使用的節點比例估計值。可以定義此欄位或 `custom_split_filenames`，但不能同時定義這兩者。請參閱 split_rate。
- **`custom_split_filenames`** – JSON 物件，其會指定用來定義訓練、驗證和測試母體之檔案的名稱。可以定義此欄位或 `split_rate`，但不能同時定義這兩者。如需詳細資訊，請參閱自訂訓練-驗證-測試比例。

`nodes` 陣列中列出之節點組態物件的內容

`nodes` 陣列中列出的節點組態物件可以包含下列欄位：

```
    {
      "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`** – 字串，其會指定 CSV 檔案的路徑，而此檔案會儲存具有相同屬性圖標籤之節點的相關資訊。

  該檔案的第一行包含資料行標籤的標頭行。

  第一個資料行標籤是 `~id`，且第一個資料行 (`~id` 資料行) 會儲存節點 ID。

  標頭行中的其餘資料行標籤會為每個剩餘的資料行指定其值已匯出至該資料行的節點屬性名稱。
- **`separator`** – 字串，其中包含分隔該 CSV 檔案中資料行的分隔符號。
- **`node`** – 包含兩個字串的 JSON 陣列。第一個字串包含儲存節點 ID 之資料行的標頭名稱。第二個字串指定圖形中的節點類型，該類型對應至節點的屬性圖標籤。
- **`features`** – 節點特徵物件的 JSON 陣列。請參閱 列示在節點或邊緣之 `features` 陣列中的特徵物件內容。
- **`labels`** – 節點標籤物件的 JSON 陣列。請參閱 節點 `labels` 陣列中列出之節點標籤物件的內容。

列示在節點或邊緣之 **features** 陣列中的特徵物件內容

節點 features 陣列中列出的節點特徵物件可以包含下列最上層欄位：

- **feature** – 由三個字串組成的 JSON 陣列。第一個字串包含資料行的標頭名稱，而該資料行包含特徵的屬性值。第二個字串包含特徵名稱。

  第三個字串包含特徵類型。有效的特徵類型列示在 特徵的 type 欄位可能值 中。
- **norm** – 數值特徵需要此欄位。它會指定要在數值上使用的標準化方法。有效值為 "none"、"min-max" 和「standard」。如需詳細資訊，請參閱 norm 欄位。
- **language** – 語言欄位會指定文字屬性值中使用的語言。它的用法取決於文字編碼方法：
  - 對於 text_fasttext 編碼，此欄位是必要的，且必須指定下列其中一種語言：
    - en (英文)
    - zh (中文)
    - hi (北印度文)
    - es (西班牙文)
    - fr (法文)

    不過，text_fasttext 一次無法處理多種語言。
  - 對於 text_sbert 編碼，不會使用此欄位，因為 SBERT 編碼是多語系的。
  - 對於 text_word2vec 編碼，此欄位是選用的，因為 text_word2vec 僅支援英文。如果存在，它必須指定英文模型的名稱：

    ```
    "language" : "en_core_web_lg"
    ```
  - 對於 tfidf 編碼，不會使用此欄位。
- **max_length** – 此欄位是 text_fasttext 特徵的選用欄位，其中指定將編碼之輸入文字特徵中的記號數目上限。在到達 max_length 之後，會忽略輸入文字。例如，將 max_length 設定為 128，表示將忽略文字序列中第 128 個之後的任何記號：
- **separator** – 此欄位可選擇性地與 category、numerical 和 auto 特徵搭配使用。它會指定一個字元，其可以用來將屬性值分成多個類別值或數值。

  請參閱 separator 欄位。
- **range** – bucket_numerical 特徵需要此欄位。它會指定要分成儲存貯體的數值範圍。

  請參閱 range 欄位。

- **bucket_cnt** – bucket_numerical 特徵需要此欄位。它會指定由 range 參數定義的數值範圍應分成的儲存貯體數目。

  請參閱 Neptune ML 中的儲存貯體數值特徵。

- **slide_window_size** – 此欄位可選擇性地與 bucket_numerical 特徵搭配使用，以將值指派給多個儲存貯體。

  請參閱 slide_window_size 欄位。

- **imputer** – 此欄位可選擇性地與 numerical、bucket_numerical 和 datetime 特徵搭配使用，以提供填入缺失值的插補技術。支援的插補技術為 "mean"、"median" 和 "most_frequent"。

  請參閱 imputer 欄位。

- **max_features** – text_tfidf 特徵可選擇性地使用此欄位，來指定要編碼的字詞數上限。

  請參閱 max_features 欄位。

- **min_df** – text_tfidf 特徵可選擇性地使用此欄位，來指定要編碼之字詞的文件頻率下限

  請參閱 min_df 欄位。

- **ngram_range** – text_tfidf 特徵可選擇性地使用此欄位，來指定哪個範圍或多少單字或記號要被視為要編碼的潛在個別字詞

  請參閱 ngram_range 欄位。

- **datetime_parts** – datetime 特徵可選擇性地使用此欄位，來指定日期時間值的哪些部分要進行類別編碼。

  請參閱 datetime_parts 欄位。

節點 **labels** 陣列中列出之節點標籤物件的內容

節點 labels 陣列中列出的標籤物件會定義節點目標特徵，並會指定訓練、驗證和測試階段將使用的節點比例。每個物件可以包含下列欄位：

```
    {
      "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** – 包含兩個字串的 JSON 陣列。第一個字串包含資料行的標頭名稱，而該資料行會儲存特徵的屬性值。第二個字串指定目標任務類型，其可以是：
  - "classification" – 節點分類任務。所指定資料行中的屬性值用來建立類別特徵。
  - "regression" – 節點迴歸任務。所指定資料行中的屬性值用來建立數值特徵。
- **split_rate** – JSON 陣列，其中包含三個介於零和一之間的數字，這些數字加起來最多為一，並代表訓練、驗證和測試階段將分別使用的節點比例估計值。請參閱 split_rate。
- **custom_split_filenames** – JSON 物件，其會指定用來定義訓練、驗證和測試母體之檔案的名稱。可以定義此欄位或 split_rate，但不能同時定義這兩者。如需詳細資訊，請參閱自訂訓練-驗證-測試比例。
- **separator** – 包含分隔符號的字串，此分隔符號會分隔分類任務的類別特徵值。

> **Note**
>
> 如果未對邊緣和節點兩者提供標籤物件，則會自動假設任務為連結預測，且邊緣會隨機分割成 90% 用於訓練，10% 用於驗證。

自訂訓練-驗證-測試比例

根據預設，Neptune ML 會使用 split_rate 參數，以使用此參數中定義的比例，將圖形隨機分割成訓練、驗證和測試母體。若要更精確地控制在這些不同母體中使用哪些實體，可以建立明確定義它們的檔案，然後可以編輯訓練資料組態檔案，將這些索引檔案對應至母體。此對應是由 JSON 物件針對訓練組態檔案中的 custom_split_filesnames 索引鍵所指定。如果使用此選項，則必須為 train 和 validation 索引鍵提供檔案名稱，而對於 test 索引鍵則是選用的。

這些檔案的格式應符合 Gremlin 資料格式。特別是，對於節點層級任務，每個檔案應包含其 ~id 標頭列出節點 ID 的資料行，而對於邊緣層級任務，檔案應分別指定 ~from 和 ~to，以指示邊緣的來源和目的地節點。這些檔案必須放置在與用於資料處理之匯出資料相同的 Amazon S3 位置 (請參閱：outputS3Path)。

對於屬性分類或迴歸任務，這些檔案可以選擇性地定義機器學習任務的標籤。在此情況下，檔案必須具有一個屬性資料行，其標頭名稱與訓練資料組態檔案中定義的相同。如果在匯出的節點和邊緣檔案以及自訂分割檔案中都定義了屬性標籤，則會優先考慮自訂分割檔案。

使用 Neptune ML 訓練模型

在處理了您從 Neptune 匯出以用於模型訓練的資料之後，您可以使用如下的 curl (或 awscurl) 命令啟動模型訓練工作：

```
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"
      }'
```

如何使用此命令的詳細資訊會在 modeltraining 指令 中加以說明，伴隨如何取得執行中工作狀態、如何停止執行中工作，以及如何列出所有執行中工作的相關資訊。

您也可以提供一個 previousModelTrainingJobId，使用來自已完成 Neptune ML 模型訓練工作的資訊，以在新的訓練工作中加速超參數搜尋。在對新圖形資料進行模型重新訓練期間，以及在對相同圖形資料進行增量訓練期間，這樣做很有用。使用像這樣的命令：

```
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)"
      }'
```

您可以提供 customModelTrainingParameters 物件，在 Neptune ML 訓練基礎結構上訓練自己的模型實作，如下所示：

```
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)"
        }
      }'
```

如需詳細資訊，例如有關如何取得執行中工作的狀態、如何停止執行中工作，以及如何列出所有執行中工作，請參閱 modeltraining 指令。如需如何實作並使用自訂模型的相關資訊，請參閱 Neptune ML 中的自訂模型。

主題

- Amazon Neptune ML 中的模型和模型訓練
- 在 Neptune ML 中自訂模型超參數組態
- 模型訓練最佳實務

Amazon Neptune ML 中的模型和模型訓練

Neptune ML 會使用圖形神經網路 (GNN) 為各種機器學習任務建立模型。圖形神經網路已被證明可為圖形機器學習任務取得最先進的結果，並且在從圖形結構化資料中擷取資訊模式方面非常出色。

Neptune ML 中的圖形神經網路 (GNN)

圖形神經網路 (GNN) 屬於一系列神經網路，它們透過考慮附近節點的結構和特徵來計算節點表示法。GNN 補充了其他不是很適合圖形資料的傳統機器學習和神經網路方法。

GNN 用來解決機器學習任務，例如節點分類和迴歸 (預測節點的屬性)，以及邊緣分類和迴歸 (預測邊緣的屬性) 或連結預測 (預測圖形中的兩個節點是否應該連接)。

一般來說，將 GNN 用於機器學習任務涉及兩個階段：

- 編碼階段，其中 GNN 會計算圖形中每個節點的 d 維向量。這些向量也稱為表示法或嵌入。
- 解碼階段，其中會根據編碼的表示法做出預測。

對於節點分類和迴歸，節點表示法可以直接用於分類和迴歸任務。對於邊緣分類和迴歸，邊緣上事件節點的節點表示法會用作分類或迴歸的輸入。對於連結預測，計算邊緣可能性分數的方式為使用一對節點表示法和一個邊緣類型表示法。

Deep Graph Library (DGL) 可為這些任務促進 GNN 的有效定義和訓練。

不同的 GNN 模型在訊息傳遞的構想下得到了統一。在此檢視中，圖形中節點的表示法是使用節點鄰居的表示法 (訊息)，以及節點的初始表示法來計算。在 Neptune ML 中，節點的初始表示法是從擷取自其節點屬性的特徵衍生的，或是可學習的，並且取決於節點的身分。

Neptune ML 還提供了選項，讓您串連節點特徵和可學習節點表示法，以充當原始節點表示法。

對於 Neptune ML 中涉及圖形與節點屬性的各種任務，我們會使用關聯式圖形卷積網路 (R-GCN) 來執行編碼階段。R-GCN 是 GNN 架構，非常適合具有多個節點和邊緣類型的圖形 (這些圖形稱為異質圖)。

R-GCN 網絡由固定數量的層組成，一個接一個地堆疊。R-GCN 的每一層都會使用其可學習模型參數來彙總資訊，而這些資訊來自節點的鄰近 1 躍點鄰域。由於後續層會使用前一層的輸出表示法做為輸入，因此影響節點最終嵌入之圖形鄰域的半徑取決於 R-GCN 網路的層數 (`num-layer`)。

例如，這表示雙層網路使用的資訊來自兩個躍點之外的節點。

若要深入了解 GNN，請參閱圖形神經網路的全面調查。如需 Deep Graph Library (DGL) 的詳細資訊，請造訪 DGL 網頁。如需有關使用 DGL 搭配 GNN 的實作教學課程，請參閱使用 Deep Graph Library 學習圖形神經網路。

訓練圖形神經網路

在機器學習中，取得模型以了解如何對任務做出良好預測的程序稱為模型訓練。通常執行此操作的方式為指定要最佳化的特定目標，以及用來執行此最佳化的演算法。

此程序也會用於訓練 GNN，以了解下游任務的合適表示法。我們會針對該任務建立目標函數，而該任務會在模型訓練期間最小化。例如，對於節點分類，我們會使用 CrosssentroPyLoss 做為目標，這會懲罰錯誤分類，而對於節點迴歸，我們會將 MeanSquareError 降至最低。

目標通常是一個損失函數，其會取得特定資料點的模型預測，並將它們與該資料點的基準真相值進行比較。它會傳回損失值，其會顯示模型預測的偏差程度。訓練程序的目標是將損失降到最低，並確保模型預測接近基準真相。

深度學習中用於訓練程序的最佳化演算法通常是梯度下降的變體。在 Neptune ML 中，我們會使用 Adam，這是一種演算法，用於隨機目標函數的一階梯度型最佳化，而這些函數是以低階時刻的自適應估計為基礎。

雖然模型訓練程序會嘗試確保學到的模型參數接近目標函數的最小值，但模型的整體效能也取決於模型的超參數，而這些超參數是訓練演算法未學習的模型設定。例如，學到的節點表示法 (num-hidden) 的維數是影響模型效能的超參數。因此，在機器學習中，執行超參數最佳化 (HPO) 來選擇合適的超參數是很常見的。

Neptune ML 會使用 SageMaker 超參數，調校工作來啟動多個具有不同超參數組態的模型訓練執行個體，以嘗試為一系列超參數設定尋找最佳模型。請參閱 在 Neptune ML 中自訂模型超參數組態。

Neptune ML 中的知識圖譜嵌入模型

知識圖譜 (KG) 是編碼不同實體 (節點) 及其關聯 (邊緣) 之相關資訊的圖形。在 Neptune ML 中，當圖形不包含節點屬性，只包含與其他節點的關聯時，預設會套用知識圖譜嵌入模型以執行連結預測。雖然, R-GCN 模型與可學習嵌入也可以透過將模型類型指定為 `"rgcn"` 來用於這些圖形,但知識圖譜嵌入模型更簡單，且被設計為對大型知識圖譜的學習表示法有效。

知識圖譜嵌入模型用於連結預測任務，以預測完成三重 (**h**, **r**, **t**) 的節點或關聯，其中 **h** 是來源節點、**r** 是關聯類型，而 **t** 是目的地節點。

在 Neptune ML 中實作的知識圖譜嵌入模型是 `distmult`、`transE` 和 `rotatE`。若要深入了解知識圖譜嵌入模型，請參閱 DGL-KE。

在 Neptune ML 中訓練自訂模型

Neptune ML 可讓您針對特定案例定義和實作自己的自訂模型。如需如何實作自訂模型，以及如何使用 Neptune ML 基礎結構來訓練該模型的相關資訊，請參閱 Neptune ML 中的自訂模型。

在 Neptune ML 中自訂模型超參數組態

當您啟動 Neptune ML 模型訓練工作時，Neptune ML 會自動使用從先前資料處理工作推斷出的資訊。它會使用此資訊來產生超參數組態範圍，而這些範圍用來建立 SageMaker 超參數調校工作，以針對您的工作訓練多個模型。如此一來，您就不必為要訓練的模型指定很長的超參數值清單。相反，根據任務類型、圖形類型和調校工作設定來選取模型超參數範圍和預設值。

不過，您也可以修改資料處理工作所產生的 JSON 組態檔，來覆寫預設超參數組態並提供自訂超參數。

使用 Neptune ML modelTraining API，您可以控制數個高階超參數調校工作設定，例如 `maxHPONumberOfTrainingJobs`、`maxHPOParallelTrainingJobs` 和 `trainingInstanceType`。若要對模型超參數進行更精細的控制，您可以自訂資料處理工作所產生的 `model-HPO-configuration.json` 檔案。此檔案會儲存在您針對處理工作輸出所指定的 Amazon S3 位置。

您可以下載此檔案、進行編輯以覆寫預設的超參數組態，然後將其上傳回相同的 Amazon S3 位置。請不要變更檔案名稱，編輯時請小心遵循這些指示。

若要從 S3 下載檔案：

```
aws s3 cp \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json \
  ./
```

完成編輯後，請將檔案上傳回原來的位置：

```
aws s3 cp \
  model-HPO-configuration.json \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json
```

`model-HPO-configuration.json` 檔案的結構

`model-HPO-configuration.json` 檔案會指定要訓練的模型、機器學習 `task_type`，以及對於各種模型訓練執行應變動或固定的超參數。

超參數會分類為屬於不同層，這些層表示調用超參數調校工作時給與超參數的優先順序：

- 第 1 層超參數具有最高優先順序。如果您將 `maxHPONumberOfTrainingJobs` 設定為小於 10 的值，則只會調校第 1 層超參數，其餘的則採取預設值。

- 第 2 層超參數的優先順序較低，因此，如果對於調校工作，您有總數超過 10 個，但少於 50 個的訓練工作，則會同時調校第 1 層和第 2 層超參數。
- 僅在您有總數超過 50 個的訓練工作時，第 3 層超參數才會與第 1 層和第 2 層一起進行調校。
- 最後，固定的超參數完全不會進行調校，並且始終採取其預設值。

model-HPO-configuration.json 檔案的範例

以下是範例 model-HPO-configuration.json 檔案：

```
{
  "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
        }
      ]
    }
  ]
}
```

model-HPO-configuration.json 檔案的元素

此檔案包含一個 JSON 物件，其中包含名為 `models` 的單一最上層陣列，該陣列包含單一模型組態物件。自訂檔案時，請確定 `models` 陣列中只有一個模型組態物件。如果您的檔案包含多個模型組態物件，調校工作將會失敗並出現警告。

模型組態物件包含下列最上層元素：

- **`model`** – (字串) 要訓練的模型類型 (請勿修改)。有效值為：
  - `"rgcn"` – 這是節點分類和迴歸任務以及異質連結預測任務的預設值。
  - `"transe"` – 這是 KGE 連結預測任務的預設值。
  - `"distmult"` – 這是 KGE 連結預測任務的替代模型類型。
  - `"rotate"` – 這是 KGE 連結預測任務的替代模型類型。

規則是不要直接修改 model 值，因為不同的模型類型通常具有截然不同的適用超參數，這可能會在訓練工作開始之後導致剖析錯誤。

若要變更模型類型，請使用 modelTraining API 中的 modelName 參數，而不是在 model-HPO-configuration.json 檔案中變更它。

變更模型類型並進行精細超參數變更的方法是，為您要使用的模型複製預設模型組態範本，然後將其貼入 model-HPO-configuration.json 檔案中。如果推斷的任務類型支援多個模型，則在與 model-HPO-configuration.json 檔案相同的 Amazon S3 位置中會有一個名為 hpo-configuration-templates 的資料夾。此資料夾包含適用於任務之其他模型的所有預設超參數組態。

例如，如果您想要將 KGE 連結預測任務的模型和超參數組態從預設 transe 模型變更為 distmult 模型，只需將 hpo-configuration-templates/distmult.json 檔案內容貼入 model-HPO-configuration.json 檔案中，然後視需要編輯超參數即可。

> **Note**
>
> 如果您在 modelTraining API 中設定 modelName 參數，同時變更 model-HPO-configuration.json 檔案中的 model 和超參數規格，且這些規格不同，則 model-HPO-configuration.json 檔案中的 model 值優先採用，且會忽略 modelName 值。

- **task_type** – (字串) 由資料處理工作推斷或直接傳遞至資料處理工作的機器學習任務類型 (請勿修改)。有效值為：
  - "node_class"
  - "node_regression"
  - "link_prediction"

  資料處理工作會推斷任務類型，方法是檢查匯出的資料集和資料集屬性所產生的訓練工作組態檔案，以取得資料集的屬性。

  此值不得變更。如果您想要訓練不同的任務，則需要執行新的資料處理工作。如果 task_type 值不是您預期的值，您應該檢查資料處理工作的輸入，以確定它們是正確的。這包括 modelTraining API 的參數，以及資料匯出程序所產生的訓練工作組態檔案中的參數。
- **eval_metric** – (字串) 評估指標應用於評估模型效能，以及用於跨 HPO 執行選取效能最佳的模型。有效值為：

- "acc" – 標準分類正確性。這是單一標籤分類任務的預設值，除非在資料處理期間發現不平衡的標籤，在此情況下，預設值為 "F1"。
- "acc_topk" – 正確標籤位列排名最高 **k** 預測的次數。您也可以透過傳入 topk 做為額外索引鍵來設定值 **k**。
- "F1" – F1 分數。
- "mse" – 迴歸任務的均方誤差指標。
- "mrr" – 平均倒數排名指標。
- "precision" – 模型精確度，計算方式為真正陽性與預測陽性之比：= true-positives / (true-positives + false-positives)。
- "recall" – 模型回收，計算方式為真正陽性與實際陽性之比：= true-positives / (true-positives + false-negatives)。
- "roc_auc" – ROC 曲線下方的區域。這是多標籤分類的預設值。

例如，若要將指標變更為 F1，請變更 eval_metric 值，如下所示：

```
"  eval_metric": {
    "metric": "F1",
  },
```

或者，若要將指標變更為 topk 正確性分數，您將變更 eval_metric，如下所示：

```
  "eval_metric": {
    "metric": "acc_topk",
    "topk": 2
  },
```

- **eval_frequency** – (物件) 指定在訓練期間應檢查驗證集上模型效能的頻率。根據驗證效能，可以接著啟動提前停止，並儲存最佳模型。

  eval_frequency 物件包含兩個元素，即 "type" 和 "value"。例如：

```
  "eval_frequency": {
    "type":  "evaluate_every_pct",
    "value":  0.1
  },
```

  有效的 type 值如下：

- **evaluate_every_pct** – 指定每個評估要完成的訓練百分比。

  對於 evaluate_every_pct，"value" 欄位包含一個介於零與一之間的浮點數，表示該百分比。

- **evaluate_every_batch** – 指定每個評估要完成的訓練批次數目。

  對於 evaluate_every_batch，"value" 欄位包含表示該批次計數的整數。

- **evaluate_every_epoch** – 指定每個評估的 epoch 數目，其中新 epoch 從午夜開始。

  對於 evaluate_every_epoch，"value" 欄位包含表示該 epoch 計數的整數。

  eval_frequency 的預設設定如下：

```
"eval_frequency": {
  "type":  "evaluate_every_epoch",
  "value":  1
},
```

- **1-tier-param** – (必要) 第 1 層超參數的陣列。

  如果您不想要調校任何超參數，則可以將此設定為空陣列。這不會影響 SageMaker 超參數調校工作啟動的訓練工作總數。其只是意味著所有的訓練作業 (如果超過 1 個，但少於 10 個) 將搭配相同的一組超參數執行。

  另一方面，如果您想要以同等重要的方式處理所有可調校的超參數，則可以將所有的超參數放在這個陣列中。

- **2-tier-param** – (必要) 第 2 層超參數的陣列。

  僅在 maxHPONumberOfTrainingJobs 具有大於 10 的值時，才會調校這些參數。否則，它們會固定為預設值。

  如果您的訓練預算最多為 10 個訓練工作，或者因為任何其他原因而不想要第 2 層超參數，但是您想要調校所有可調校的超參數，則可以將此設定為空陣列。

- **3-tier-param** – (必要) 第 3 層超參數的陣列。

  僅在 maxHPONumberOfTrainingJobs 具有大於 50 的值時，才會調校這些參數。否則，它們會固定為預設值。

  如果您不想要第 3 層超參數，則可以將此設定為空陣列。

- **fixed-param** – (必要) 固定超參數的陣列，這些超參數僅採取其預設值，且在不同的訓練工作中不會有所變動。

  如果想要改變所有超參數，您可以將其設定為空陣列，然後將 maxHPONumberOfTrainingJobs 的值設定為足以改變所有層的大小，或使所有超參數成為第 1 層。

代表 1-tier-param、2-tier-param、3-tier-param 和 fixed-param 中每個超參數的 JSON 物件包含下列元素：

- **param** – (字串) 超參數的名稱 (請勿變更)。

  請參閱 Neptune ML 中有效的超參數名稱清單。

- **type** – (字串) 超參數類型 (請勿變更)。

  有效類型為：bool、int 和 float。

- **default** – (字串) 超參數的預設值。

  您可以設定新的預設值。

可調校的超參數也可以包含下列元素：

- **range** – (陣列) 連續可調校超參數的範圍。

  這應該是具有兩個值的陣列，即範圍的最小值和最大值 ([min, max])。

- **options** – (陣列) 類別可調校超參數的選項。

  這個陣列應該包含所有要考慮的選項：

  ```
  "options" : [value1, value2, ... valuen]
  ```

- **inc_strategy** – (字串) 連續可調校超參數範圍的增量變更類型 (請勿變更)。

  有效值為 log、linear 和 power2。這僅在設定了範圍索引鍵時才適用。

  修改此選項可能會導致未使用完整範圍的超參數進行調校。

- **inc_val** – (浮動) 連續增量對連續可調校超參數的差異量 (請勿變更)。

  這僅在設定了範圍索引鍵時才適用。

修改此選項可能會導致未使用完整範圍的超參數進行調校。

- **node_strategy** – (字串) 表示此超參數的有效範圍應根據圖形中的節點數目而變更 (請勿變更)。

  有效值為 "perM" (每百萬)、"per10M" (每千萬) 和 "per100M" (每億)。

  不是變更此值，而是變更 range。
- **edge_strategy** – (字串) 表示此超參數的有效範圍應根據圖形中的邊緣數目而變更 (請勿變更)。

  有效值為 "perM" (每百萬)、"per10M" (每千萬) 和 "per100M" (每億)。

  不是變更此值，而是變更 range。

Neptune ML 中所有超參數的清單

下列清單包含可在 Neptune ML 中任何位置針對任何模型類型和任務設定的所有超參數。因為它們並非全部適用於每種模型類型，所以請務必只針對您正在使用的模型設定 model-HPO-configuration.json 檔案中出現在範本的超參數。

- **batch-size** – 在一個正向傳遞中使用之目標節點的批次大小。類型：int。

  將此值設定為更大的值可能會導致 GPU 執行個體訓練時出現記憶體問題。
- **concat-node-embed** – 指示是否透過串連節點處理的特徵與可學習的初始節點嵌入來取得節點的初始表示法，以便增加模型的表達能力。類型：bool。
- **dropout** – 套用至退出層的退出機率。類型：float。
- **edge-num-hidden** – 邊緣特徵模組的隱藏層大小或單位數量。僅在 use-edge-features 設為 True 時才使用。類型：浮動。
- **enable-early-stop** – 切換是否使用提前停止功能。類型：bool。預設：true。

  使用此布林參數來關閉提前停止功能。
- **fanout** – 鄰近取樣期間要為目標節點取樣的鄰近節點數。類型：int。

  此值與 num-layers 緊密結合，並且應始終位於相同的超參數層中。這是因為您可以為每個潛在的 GNN 層指定扇出。

  因為這個超參數可能會造成模型效能發生很大變化，所以其應該固定或設定為第 2 層或第 3 層超參數。將其設定為更大的值可能會導致 GPU 執行個體訓練時出現記憶體問題。
- **gamma** – 分數函數中的邊距值。類型：float。

這僅適用於 KGE 連結預測模型。

- **l2norm** – 最佳化工具中使用的權重衰減值，該值會對權重施加 L2 標準化懲罰。類型：bool。
- **layer-norm** – 指示是否要對 rgcn 模型使用層標準化。類型：bool。
- **low-mem** – 指示是否以犧牲速度為代價，使用關聯訊息傳遞函數的低記憶體實作。類型：bool。
- **lr** – 學習率。類型：float。

  這應該設為第 1 層超參數。
- **neg-share** – 在連結預測中，指示正取樣邊緣是否可以共用負邊緣取樣。類型：bool。
- **num-bases** – rgcn 模型中基礎分解的基數。使用小於圖形中邊緣類型數目的 num-bases 值，做為 rgcn 模型的正規化器。類型：int。
- **num-epochs** – 要執行之訓練的 epoch 數量。類型：int。

  一個 epoch 是通過圖形的完整訓練。
- **num-hidden** – 隱藏層大小或單位數量。類型：int。

  這也會設定無特徵節點的初始嵌入大小。

  在沒有減少 batch-size 的情況下將此值設定為更大的值可能會導致 GPU 執行個體訓練時出現記憶體用光問題。
- **num-layer** – 模型中 GNN 層的數量。類型：int。

  此值與扇出參數緊密結合，並且應該在同一超參數層中設定扇出之後出現。

  因為此值可能會造成模型效能發生很大變化，所以其應該固定或設定為第 2 層或第 3 層超參數。
- **num-negs** – 在連結預測中，每個正樣本的負樣本數量。類型：int。
- **per-feat-name-embed** – 指示是否在組合特徵之前透過獨立轉換來嵌入每個特徵。類型：bool。

  當設定為 true 時，在串連節點的所有轉換特徵並進一步轉換為 num_hidden 維度之前，將每個節點的每個特徵獨立轉換為固定維度大小。

  當設定為 false 時，會串連特徵，而不會進行任何特徵特定的轉換。
- **regularization-coef** – 在連結預測中，正規化損失的係數。類型：float。
- **rel-part** – 指示是否使用關聯分割區進行 KGE 連結預測。類型：bool。
- **sparse-lr** – 可學習節點嵌入的學習率。類型：float。

可學習的初始節點嵌入用於沒有特徵或設定 concat-node-embed 時的節點。稀疏可學習節點嵌入層的參數會使用個別的最佳化工具進行訓練，該最佳化工具可以具有個別的學習率。

- **use-class-weight** – 指示是否針對不平衡的分類任務套用類別權重。如果設定為 true，則會使用標籤計數來設定每個類別標籤的權重。類型：bool。
- **use-edge-features** – 指示是否要在訊息傳遞期間使用邊緣特徵。如果設定為 true，則對於具有特徵的邊緣類型，會將自訂邊緣特徵模組新增至 RGCN 層。類型：bool。
- **use-self-loop** – 指示是否要在訓練 rgcn 模型時包含自我迴圈。類型：bool。
- **window-for-early-stop** – 控制要平均的最新驗證分數，以決定是否要提前停止。預設值為 3。type=int。另請參閱 提前停止 Neptune ML 中的模型訓練程序。類型：int。預設：3。

請參閱「」。

在 Neptune ML 中自訂超參數

當您編輯 model-HPO-configuration.json 檔案時，以下是最常進行的變更種類：

- 編輯 range 超參數的最小值和/或最大值。
- 將超參數設定為固定值，方法是將該超參數移至 fixed-param 區段，並將其預設值設定為您想要其採取的固定值。
- 變更超參數的優先順序，方法是將該超參數置於特定層中、編輯其範圍，並確定已適當地設定其預設值。

模型訓練最佳實務

您可以採取一些措施來改善 Neptune ML 模型的效能。

選擇正確的節點屬性

並非圖形中的所有屬性都有意義或與您的機器學習任務相關。在資料匯出期間，應排除任何不相關的屬性。

以下是一些最佳實務：

- 使用領域專家來協助評估特徵的重要性，以及使用它們進行預測的可行性。
- 移除您確定是多餘或無關的特徵，以減少資料中的干擾和不重要的相關性。
- 在建置模型時進行迭代。在進行時調整特徵、特徵組合和調校目標。

Amazon Machine Learning 開發人員指南中的特徵處理為與 Neptune ML 相關的特徵處理提供其他指導方針。

處理異常值資料點

異常值是與剩餘資料明顯不同的資料點。資料異常值可能會破壞或誤導訓練程序，導致訓練時間更長或模型準確度降低。除非它們真的很重要，否則您應該在匯出資料之前消除異常值。

移除重複的節點和邊緣

儲存在 Neptune 的圖形可能具有重複的節點或邊緣。這些冗餘元素將為 ML 模型訓練帶來干擾。在匯出資料之前消除重複的節點或邊緣。

調校圖形結構

匯出圖形時，您可以變更特徵的處理方式及圖形的建構方式，以改善模型效能。

以下是一些最佳實務：

- 當邊緣屬性具有邊緣類別的意義時，在某些情況下，值得將其轉換為邊緣類型。
- 用於數值屬性的預設標準化政策是 `min-max`，但在某些情況下，其他標準化政策效果更好。您可以預先處理屬性並變更標準化政策，如 `model-HPO-configuration.json` 檔案的元素 中所述。
- 匯出程序會根據屬性類型自動產生特徵類型。例如，它會將 `String` 屬性視為類別特徵，並將 `Float` 和 `Int` 屬性視為數值特徵。如果需要的話，您可在匯出後修改特徵類型 (請參閱 `model-HPO-configuration.json` 檔案的元素)。

調校超參數範圍和預設值

資料處理操作會從圖形推斷超參數組態範圍。如果產生的模型超參數範圍和預設值不適用於圖形資料，您可以編輯 HPO 組態檔案，以建立您自己的超參數調校策略。

以下是一些最佳實務：

- 當圖形變大時，預設隱藏的維度大小可能不夠大，無法包含所有資訊。您可以變更 num-hidden 超參數來控制隱藏的維度大小。
- 對於知識圖譜嵌入（KGE）模型，您可能想要根據圖形結構和預算變更正在使用的特定模型。

  TrainsE 模型在處理一對多 (1-N)、多對一 (N-1) 和多對多 (N-N) 關係時遇到困難。DistMult 模型在處理對稱關係時遇到困難。RotatE 擅長建模各種關係，但在訓練期間比 TrainsE 和 DistMult 更昂貴。
- 在某些情況下，當節點識別和節點特徵資訊都很重要時，您應該使用 `concat-node-embed` 告知 Neptune ML 模型，透過將節點的特徵與初始嵌入串連來取得節點的初始表示法。
- 當您對某些超參數取得相當好的效能時，可以根據這些結果調整超參數搜尋空間。

提前停止 Neptune ML 中的模型訓練程序

提前停止可大幅縮短模型訓練執行時間和相關聯成本，而不會降低模型效能。它還可以防止模型在訓練資料上過度擬合。

提前停止取決於驗證集效能的定期測量。最初，效能隨著訓練的進行而改善，但是當模型開始過度擬合時，效能會再次開始下降。提前停止功能會識別模型開始過度擬合的點，並在該點停止模型訓練。

Neptune ML 會監控驗證指標呼叫，並將最新的驗證指標與最後 **n** 個評估所得到的驗證指標平均值進行比較，其中 **n** 是使用 window-for-early-stop 參數設定的數字。一旦驗證指標比該平均值差，Neptune ML 就會停止模型訓練並儲存迄今的最佳模型。

您可以使用下列參數控制提前停止：

- **window-for-early-stop** – 此參數的值為整數，其指定在決定提前停止時要平均的最新驗證分數。預設值為 3。
- **enable-early-stop** – 使用此布林參數來關閉提前停止功能。依預設，其值為 true。

提前停止 Neptune ML 中的 HPO 程序

Neptune ML 中的提前停止功能也會使用 SageMaker HPO 暖啟動功能，停止相較於其他訓練工作未妥善執行的訓練工作。這也可以降低成本並提高 HPO 的品質。

如需此運作方式的描述，請參閱執行暖啟動超參數調校工作。

暖啟動可讓您將從先前訓練工作中學到的資訊傳遞給後續訓練工作，並提供了兩個明顯的優勢：

- 首先，先前訓練工作的結果是用來選取好的超參數組合，以在新的調校工作中進行搜尋。
- 其次，它允許提前停止存取更多的模型執行，從而減少調校時間。

此功能會在 Neptune ML 中自動啟用，並可讓您在模型訓練時間與效能之間取得平衡。如果您對目前模型的效能感到滿意，則可以使用該模型。否則，您會執行更多由於先前執行結果而暖啟動的 HPO，以便探索更好的模型。

取得專業支援服務

AWS 會提供專業支援服務，協助您解決 Neptune 專案上機器學習的問題。如果您遇到困難，請尋求 AWS 支援。

使用訓練後的模型來產生新的模型成品

使用 Neptune ML 模型轉換命令，您可以使用預先訓練的模型參數計算模型成品，例如在處理後的圖形資料上進行節點嵌入。

用於增量推論的模型轉換

在增量模型推論工作流程中，於處理了您從 Neptune 匯出的更新圖形資料之後，您可以使用如下的 curl (或 awscurl) 命令啟動模型轉換工作：

```
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/"
      }'
```

然後，您可以將此工作的 ID 傳遞至 create-endpoints API 呼叫，以建立新端點，或使用此工作產生的新模型成品更新現有端點。這可讓新的或更新的端點，為更新的圖形資料提供模型預測。

任何訓練工作的模型轉換

您也可以提供 `trainingJobName` 參數，為 Neptune ML 模型訓練期間啟動的任何 SageMaker 訓練工作產生模型成品。由於 Neptune ML 模型訓練工作可能會啟動許多 SageMaker 訓練工作，因此這可讓彈性地根據其中任何一個 SageMaker 訓練工作建立推論端點。

例如：

```
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/"
```

```
    }'
```

如果原始訓練工作是針對使用者提供的自訂模型，您必須在調用模型轉換時包含 `customModelTransformParameters` 物件。如需如何實作並使用自訂模型的相關資訊，請參閱 Neptune ML 中的自訂模型。

> **Note**
>
> `modeltransform` 命令一律會在該訓練的最佳 SageMaker 訓練工作上執行模型轉換。

如需模型轉換工作的詳細資訊，請參閱 modeltransform 命令。

Neptune ML 中模型訓練產生的成品

在模型訓練之後，Neptune ML 會使用最佳訓練模型參數，來產生啟動推論端點並提供模型預測所需的模型成品。這些成品由訓練工作封裝，並儲存在最佳 SageMaker 訓練任務的 Amazon S3 輸出位置中。

以下各節描述了各種任務的模型成品中包含什麼內容，以及模型轉換命令如何使用預先存在的訓練模型，即使在新的圖形資料上也能產生成品。

針對不同任務產生的成品

訓練程序所產生的模型成品內容取決於目標機器學習任務：

- 節點分類與迴歸 – 對於節點屬性預測，成品包括模型參數、來自 GNN 編碼器的節點嵌入、訓練圖中節點的模型預測，以及推論端點的一些組態檔案。在節點分類和節點迴歸任務中，會針對訓練期間存在的節點預先計算模型預測，以減少查詢延遲。
- 邊緣分類和迴歸 – 對於邊緣屬性預測，成品還包含模型參數和節點嵌入。模型解碼器的參數對於推論尤為重要，因為我們會透過將模型解碼器套用至邊緣的來源和目標頂點嵌入，來計算邊緣分類或邊緣迴歸預測。
- 連結預測 – 對於連結預測，除了針對邊緣屬性預測產生的成品外，也會包含 DGL 圖形作為成品，因為連結預測需要訓練圖才能執行預測。連結預測的目標是預測可能與來源頂點結合的目的地頂點，以形成圖形中特定類型的邊緣。為了執行此操作，來源頂點的節點嵌入和學到的邊緣類型表示法會與所有可能目的地頂點的節點嵌入結合，以產生每個目的地頂點的邊緣可能性分數。然後，對分數進行排序以排名潛在的目的地頂點並傳回頂級候選者。

對於每個任務類型，來自 DGL 的圖形神經網路模型權重會儲存在模型成品中。除了使用預先計算的預測和嵌入 (直推式推論) 來減少延遲之外，這還可讓 Neptune ML 在圖形變更時計算全新的模型輸出 (歸納推論)。

產生新的模型成品

Neptune ML 中於模型訓練之後產生的模型成品會直接繫結至訓練程序。這意味著預先計算的嵌入和預測僅存在於原始訓練圖中的實體。雖然 Neptune ML 端點的歸納推論模式可以即時計算新實體的預測，但是您可能想要在新實體上產生批次預測，而不需要查詢端點。

對於已新增至圖形的新實體，為了取得其批次模型預測，需要針對新圖形資料重新計算新的模型成品。這是使用 `modeltransform` 命令來完成的。當您只想要批次預測而不設定端點時，或者當您想要產生所有預測，以便可以將其寫回圖形時，請使用 `modeltransform` 命令。

由於模型訓練會在訓練程序結束時隱含地執行模型轉換，因此訓練工作一律會在訓練圖資料上重新計算模型成品。不過，`modeltransform` 命令也可以在未用於訓練模型的圖形資料上計算模型成品。為此，必須使用與原始圖形資料相同的特徵編碼來處理新的圖形資料，並且必須遵守相同的圖形結構描述。

您可以達成此目標，方法是首先建立新的資料處理工作 (這是在原始訓練圖資料上執行的資料處理工作的複製)，接著在新的圖形資料上執行它 (請參閱 處理 Neptune ML 的更新圖形資料)。然後，使用新的 `dataProcessingJobId` 和舊的 `modelTrainingJobId` 呼叫 `modeltransform` 命令，以在更新的圖形資料上重新計算模型成品。

對於節點屬性預測，節點嵌入和預測會在新的圖形資料上重新計算，即使是原始訓練圖中存在的節點也是如此。

對於邊緣屬性預測和連結預測，也會重新計算節點嵌入，並會類似地覆蓋任何現有的節點嵌入。為了重新計算節點嵌入，Neptune ML 會將從先前訓練的模型中學到的 GNN 編碼器套用至具有其新特徵之新圖形資料的節點。

對於沒有特徵的節點，會重複使用從原始模型訓練中學到的初始表現法。對於沒有特徵且不存在於原始訓練圖中的新節點，Neptune ML 會將其表示法初始化為該節點類型 (存在於原始訓練圖中) 學到的初始節點表示法的平均值。如果您有許多沒有特徵的新節點，這可能會導致模型預測的效能下降，因為它們都會初始化為該節點類型的平均初始嵌入。

如果您的模型是在 `concat-node-embed` 設定為 true 的情況下進行訓練，則會透過串連節點特徵與可學習的初始表示法來建立初始節點表示法。因此，對於更新的圖形，新節點的初始節點表示法也會使用平均初始節點嵌入，與新的節點特徵串連。

此外，目前不支援節點刪除。如果已在更新的圖形中移除節點，則必須在更新的圖形資料上重新訓練模型。

重新計算模型成品會在新圖形上重複使用學到的模型參數，並且僅應在新圖形與舊圖形非常相似時才進行。如果您的新圖形不夠相似，則您需要重新訓練模型，才能在新圖形資料上取得類似的模型效能。構成足夠相似的內容取決於圖形資料的結構，但根據經驗法則，如果您的新資料與原始訓練圖形資料不同之處超過 10-20%，則您應該重新訓練模型。

對於所有節點都有特徵的圖形，會套用閾值的較高端 (20% 不同)，但對於許多節點沒有特徵，且新增至圖形的新節點沒有屬性的圖形，則較低端 (10% 不同) 甚至可能太高。

如需模型轉換工作的詳細資訊，請參閱 modeltransform 命令。

Neptune ML 中的自訂模型

Neptune ML 可讓您使用 Python 定義自己的自訂模型實作。您可以使用 Neptune ML 基礎結構訓練和部署自訂模型，就像對內建模型所做一樣，然後使用它們透過圖形查詢取得預測。

> **Note**
>
> 自訂模型目前不支援即時歸納推論。

您可以遵循 Neptune ML 工具組範例，並使用 Neptune ML 工具組中提供的模型元件，開始在 Python 中實作您自己的自訂模型。下列各節將詳細說明。

內容

Neptune ML 中自訂模型的概觀

何時在 Neptune ML 中使用自訂模型

Neptune ML 的內建模型會處理 Neptune ML 支援的所有標準任務，但是在有些情況，您可能想要對特定任務的模型進行更精細的控制，或者需要自訂模型訓練程序。例如，自訂模型適用於下列情況：

- 對於非常大文字模型的文字特徵，必須在 GPU 上執行其特徵編碼。
- 您想要使用自己的自訂圖形神經網路（GNN）模型，這是在 Deep Graph Library (DGL) 中開發的模型。
- 您想要使用表格式模型或整體模型進行節點分類和迴歸。

在 Neptune ML 中開發和使用自訂模型的工作流程

Neptune ML 中的自訂模型支援旨在無縫整合至現有的 Neptune ML 工作流程。它的運作方式是在 Neptune ML 基礎結構上的來源模組中執行自訂程式碼，以訓練模型。就像內建模式一樣，Neptune ML 會自動啟動 SageMaker HyperParameter 調校工作，並根據評估指標選取最佳模型。然後，它會使用來源模組中提供的實作來產生模型成品進行部署。

自訂模型的資料匯出、訓練組態和資料預先處理與內建模型相同。

在資料預先處理之後，您就可以反覆且以互動方式使用 Python，來開發並測試您的自訂模型實作。當您的模型已準備好生產時，您可以將產生的 Python 模組上傳到 Amazon S3，如下所示：

```
aws s3 cp --recursive (source path to module) s3://(bucket name)/(destination path for
 your module)
```

然後，您可以使用一般預設或增量資料工作流程，將模型部署至生產環境，但有一些差異。

對於使用自訂模型的模型訓練，您必須將 `customModelTrainingParameters` JSON 物件提供給 Neptune ML 模型訓練 API，以確保使用您的自訂程式碼。`customModelTrainingParameters` 物件中的欄位如下所示：

- **`sourceS3DirectoryPath`** – (必要) 此路徑通往實作您模型之 Python 模組所在的 Amazon S3 位置。這必須指向有效的現有 Amazon S3 位置，其中至少包含訓練指令碼、轉換指令碼和 `model-hpo-configuration.json` 檔案。
- **`trainingEntryPointScript`** – (選用) 指令碼模組中的進入點名稱，該指令碼會執行模型訓練，並接受超參數作為命令列引數 (包括固定的超參數)。

預設：training.py。

- **transformEntryPointScript** – (選用) 指令碼模組中的進入點名稱，該指令碼應在識別了超參數搜尋中的最佳模型之後執行，以計算模型部署所需的模型成品。它應該能夠在沒有命令列參數的情況下執行。

  預設：transform.py。

例如：

```
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)"
        }
      }'
```

同樣地，若要啟用自訂模型轉換，您必須將 customModelTransformParameters JSON 物件提供給 Neptune ML 模型轉換 API，其欄位值與訓練工作中儲存的模型參數相容。customModelTransformParameters 物件包含下列欄位：

- **sourceS3DirectoryPath** – (必要) 此路徑通往實作您模型之 Python 模組所在的 Amazon S3 位置。這必須指向有效的現有 Amazon S3 位置，其中至少包含訓練指令碼、轉換指令碼和 model-hpo-configuration.json 檔案。
- **transformEntryPointScript** – (選用) 指令碼模組中的進入點名稱，該指令碼應在識別了超參數搜尋中的最佳模型之後執行，以計算模型部署所需的模型成品。它應該能夠在沒有命令列參數的情況下執行。

預設：transform.py。

例如：

```
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)"
        }
      }'
```

Neptune ML 中的自訂模型開發

開始自訂模型開發的好方法是，遵循 Neptune ML 工具組範例來建構和撰寫訓練模組。Neptune ML 工具組也會在 modelzoo 中實作模組化圖形 ML 模型元件，您可以堆疊並使用這些元件來建立自訂模型。

此外，此工具組還提供公用程式函數，協助您在模型訓練和模型轉換期間產生必要的成品。您可以在自訂實作中匯入此 Python 套件。此工具組中提供的任何函數或模組也可在 Neptune ML 訓練環境中使用。

如果您的 Python 模組有額外的外部相依性，則您可以透過在模組的目錄中建立 `requirements.txt` 檔案，來包含這些額外的相依性。然後，會在訓練指令碼執行之前安裝 `requirements.txt` 檔案中列出的套件。

實作自訂模型的 Python 模組至少需要包含以下內容：

- 訓練指令碼進入點
- 轉換指令碼進入點
- `model-hpo-configuration.json` 檔案

Neptune ML 中的自訂模型訓練指令碼開發

您的自訂模型訓練指令碼應該是可執行的 Python 指令碼，例如 Neptune ML 工具組的 `train.py` 範例。它必須接受超參數名稱和值作為命令列參數。在模型訓練期間，會從 `model-hpo-configuration.json` 檔案取得超參數名稱。超參數值會落在有效的超參數範圍內 (如果超參數是可調校的話)，或採取預設的超參數值 (如果無法調校的話)。

您的訓練指令碼會使用如下語法在 SageMaker 訓練執行個體上執行：

```
python3 (script entry point) --(1st parameter) (1st value) --(2nd parameter) (2nd
 value) (...)
```

對於所有工作，除了您指定的超參數之外，Neptune ML AutoTrainer 還會將數個必要的參數傳送至您的訓練指令碼，而且您的指令碼必須能夠處理這些額外的參數，才能正常運作。

這些額外的必要參數會因任務而有所不同：

若是節點分類或節點迴歸

- **`task`** – Neptune ML 內部使用的任務類型。對於節點分類，這是 `node_class`，對於節點迴歸，它是 `node_regression`。

- **model** – Neptune ML 內部使用的模組名稱，在此情況下，該模組為 custom。
- **name** – Neptune ML 內部使用的任務名稱，在此情況下該任務是 node_class-custom (用於節點分類)，以及 node_regression-custom (用於節點迴歸)。
- **target_ntype** – 用於分類或迴歸的節點類型名稱。
- **property** – 用於分類或迴歸的節點屬性名稱。

若是連結預測

- **task** – Neptune ML 內部使用的任務類型。對於連結預測，這是 link_predict。
- **model** – Neptune ML 內部使用的模組名稱，在此情況下，該模組為 custom。
- **name** – Neptune ML 內部使用的任務名稱，在此情況下，該任務為 link_predict-custom。

若是邊緣分類或邊緣迴歸

- **task** – Neptune ML 內部使用的任務類型。對於邊緣分類，這是 edge_class，對於邊緣迴歸，它是 edge_regression。
- **model** – Neptune ML 內部使用的模組名稱，在此情況下，該模組為 custom。
- **name** – Neptune ML 內部使用的任務名稱，在此情況下該任務是 edge_class-custom (用於邊緣分類)，以及 edge_regression-custom (用於邊緣迴歸)。
- **target_etype** – 用於分類或迴歸的邊緣類型名稱。
- **property** – 用於分類或迴歸的邊緣屬性名稱。

您的指令碼應該儲存模型參數，以及訓練結束時將需要的任何其他成品。

您可以使用 Neptune ML 工具組公用程式函數，來判斷所處理圖形資料的位置、應儲存模型參數的位置，以及訓練執行個體上可用的 GPU 裝置。如需如何使用這些公用程式函數的範例，請參閱 train.py 範例訓練指令碼。

Neptune ML 中的自訂模型轉換指令碼開發

需要轉換指令碼，才能利用 Neptune ML 增量工作流程，在不重新訓練模型的情況下，對不斷發展的圖形進行模型推論。即使模型部署所需的所有成品都是由訓練指令碼產生，但如果您想要在不重新訓練模型的情況下產生更新的模型，您仍然需要提供轉換指令碼。

> **Note**
>
> 自訂模型目前不支援即時歸納推論。

您的自訂模型轉換指令碼應該是可執行的 Python 指令碼，例如 Neptune ML 工具組的 transform.py 範例指令碼。因為這個指令碼是在模型訓練期間調用，沒有命令列引數，所以指令碼接受的任何命令列引數都必須具有預設值。

指令碼會在 SageMaker 訓練執行個體上執行，語法如下：

```
python3 (your transform script entry point)
```

您的轉換指令碼將需要各項資訊，例如：

- 所處理圖形資料的位置。
- 儲存模型參數的位置，以及應儲存新模型成品的位置。
- 可在執行個體上使用的裝置。
- 已產生最佳模型的超參數。

這些輸入是使用指令碼可呼叫的 Neptune ML 公用程式函數來取得。如需如何執行此操作的範例，請參閱工具組的範例 transform.py 指令碼。

指令碼應儲存節點嵌入、節點 ID 對應，以及每項任務的模型部署所需的任何其他成品。如需不同 Neptune ML 任務所需之模型成品的詳細資訊，請參閱模型成品文件。

Neptune ML 中的自訂 `model-hpo-configuration.json` 檔案

`model-hpo-configuration.json` 檔案會針對您的自訂模型定義超參數。它的格式同於與 Neptune ML 內建模型搭配使用的 `model-hpo-configuration.json` 檔案，而且優先於 Neptune ML 自動產生並上傳到所處理資料位置的版本。

將新的超參數新增至您的模型時，您也必須為此檔案中的超參數新增一個項目，以便該超參數會傳遞至您的訓練指令碼。

如果您想要超參數可調校，則必須為這個超參數提供範圍，並將其設定為 `tier-1`、`tier-2`、或 `tier-3` 參數。如果設定的訓練工作總數允許調整其層中的超參數，則會調整超參數。對於不可調校的參數，您必須提供預設值，並將超參數新增至檔案的 `fixed-param` 區段。如需如何執行此操作的範例，請參閱工具組的範例 `model-hpo-configuration.json` 檔案範例。

您也必須提供指標定義，SageMaker 超參數最佳化工作將用來評估已訓練的候選模型。若要這樣做，您可以將 `eval_metric` JSON 物件新增至 `model-hpo-configuration.json` 檔案，如下所示：

```
"eval_metric": {
  "tuning_objective": {
      "MetricName": "(metric_name)",
      "Type": "Maximize"
  },
  "metric_definitions": [
    {
      "Name": "(metric_name)",
      "Regex": "(metric regular expression)"
    }
  ]
},
```

對於每一個您想要 SageMaker 從訓練執行個體擷取的指標，`metric_definitions` 物件中的 `eval_metric` 陣列會列出其指標定義物件。每個指標定義物件都有一個 `Name` 索引鍵，可讓您提供指標的名稱 (例如「accuracy」、「f1」等)。`Regex` 索引鍵可讓您提供一個規則表達式字串，與該特定指標在訓練日誌中列印的方式相符。如需如何定義指標的詳細資訊，請參閱 SageMaker HyperParameter 調校頁面。

然後，`eval_metric` 中的 `tuning_objective` 物件可讓您指定 `metric_definitions` 中的哪個指標應用作評估指標，做為超參數最佳化的目標指標。`MetricName` 的值必須與 `metric_definitions` 內其中一個定義中 `Name` 的值相符。`Type` 的值應該是「Maximize」或「Minimize」，取決於是否應將指標解譯為更大更好 (例如「accuracy」) 或更少更好 (例如「mean-squared-error」)。

`model-hpo-configuration.json` 檔案的這個區段中的錯誤可能會導致 Neptune ML 模型訓練 API 工作失敗，因為 SageMaker 超參數調校工作將無法選取最佳模型。

在 Neptune ML 中對您的自訂模型實作進行本機測試

您可以使用 Neptune ML 工具組 Conda 環境，在本機執行程式碼，以便測試並驗證您的模型。如果您是在 Neptune 筆記本執行個體上進行開發，則此 Conda 環境將預先安裝在 Neptune 筆記本執行個體執行個體上。如果您是在不同的執行個體上進行開發，則需要遵循 Neptune ML 工具組中的本機設定指示。

當您呼叫模型訓練 API 時，Conda 環境會準確地重現模型將在其中執行的環境。所有範例訓練指令碼和轉換指令碼都可讓您傳遞命令列 `--local` 旗標，在本機環境中執行指令碼，進行輕鬆的偵錯。

在開發自己的模型時，這是一個很好的做法，因為它可以讓您以互動方式並反覆測試模型實作。在 Neptune ML 生產訓練環境中進行模型訓練期間，會省略此參數。

建立要查詢的推論端點

推論端點可讓您查詢模型訓練程序所建構的特定模型。端點會附加至屬於給定類型且訓練程序能夠產生的最佳表現模型。然後，端點能夠接受來自 Neptune 的 Gemlin 查詢，並傳回該模型對查詢中輸入的預測。在您建立了推論端點之後，它會保持作用中狀態，直到您將其刪除為止。

管理 Neptune ML 的推論端點

在您從 Neptune 匯出的資料上完成了模型訓練之後，您可以使用 `curl` (或 `awscurl`) 命令建立推論端點，如下所示：

```
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)"
      }'
```

您也可以從完成的模型轉換工作所建立的模型建立推論端點，方式大致相同：

```
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)"
      }'
```

如何使用這些命令的詳細資訊會在 endpoints 命令 加以說明，其中還有如何取得端點狀態、如何刪除端點，以及如何列出所有推論端點的相關資訊。

Neptune ML 中的推論查詢

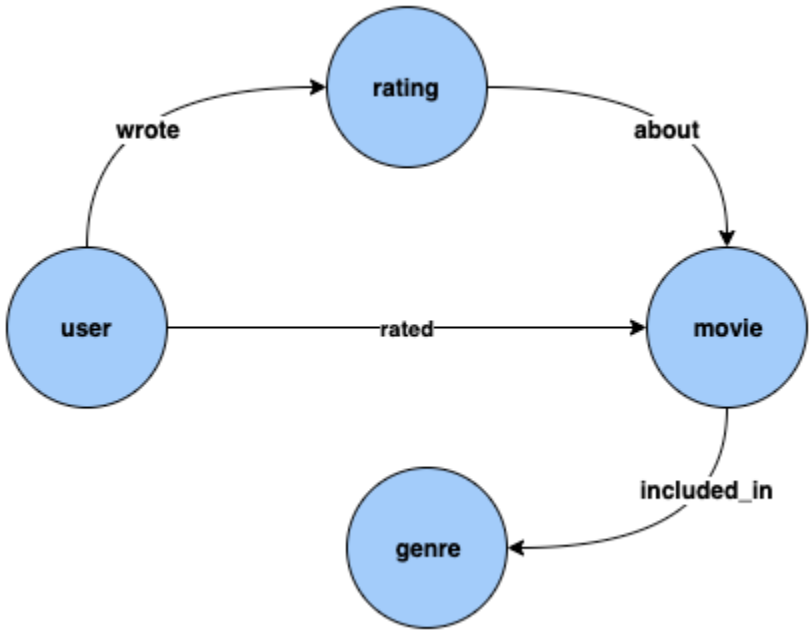
您可以使用 Gremlin 或 SPARQL，查詢 Neptune ML 推論端點。不過，目前僅 Gemlin 查詢支援即時歸納推論。

Neptune ML 中的 Gremlin 推論查詢

如 Neptune ML 功能 中所述，Neptune ML 支援可以執行下列各種推論任務的訓練模型：

- 節點分類 – 預測頂點屬性的類別特徵。
- 節點迴歸 – 預測頂點的數值屬性。
- 邊緣分類 – 預測邊緣屬性的類別特徵。
- 邊緣迴歸 – 預測邊緣的數值屬性。
- 連結預測 – 鑑於來源節點和傳出邊緣預測目的地節點，或鑑於目的地節點和傳入邊緣指定來源節點。

我們可以透過使用 GroupLens Research 所提供的 MovieLens 100k 資料集的範例來說明這些不同的任務。此資料集由電影、使用者和使用者對電影的評分組成，我們從中建立了一個屬性圖，如下所示：



節點分類：在上述資料集中，Genre 是由邊緣 included_in 連線至頂點類型 Movie 的頂點類型。但是，如果我們調整資料集，使 Genre 成為頂點類型 Movie 的類別特徵，則為新增至我們知識圖譜的新電影推斷 Genre 的問題可以使用節點分類模型來解決。

節點迴歸：如果我們考慮頂點類型 Rating，它具有 timestamp 和 score 等屬性，則為 Rating 推斷數值 Score 的問題可以使用節點迴歸模型來解決。

邊緣分類：同樣地，對於 Rated 邊緣，如果我們有一個屬性 Scale 可以具有 Love、Like、Dislike、Neutral、Hate 其中一個值，則為新電影/評分的 Rated 邊緣推斷 Scale 的問題可以使用邊緣分類模型來解決。

邊緣迴歸：同樣地，對於相同的 Rated 邊緣，如果我們有一個屬性 Score 保留評分的數值，則這可以從邊緣迴歸模型推斷出來。

連結預測：尋找最有可能對給定電影評分的前十名使用者，或尋找給定使用者最有可能評分的前十部電影之類的問題屬於連結預測。

> **Note**
>
> 對於 Neptune ML 使用案例，我們有一組非常豐富的筆記本，旨在讓您實際了解每個使用案例。使用 Neptune ML AWS CloudFormation 範本建立 Neptune ML 叢集時，您可以建立這些筆記本以及 Neptune 叢集。這些筆記本也可以在 github 上取得。

主題

- Gremlin 推論查詢中使用的 Neptune ML 述詞
- Neptune ML 中的 Gremlin 節點分類查詢
- Neptune ML 中的 Gremlin 節點迴歸查詢
- Neptune ML 中的 Gremlin 邊緣分類查詢
- Neptune ML 中的 Gremlin 邊緣迴歸查詢
- 在 Neptune ML 中使用連結預測模型進行 Gremlin 連結預測查詢
- Neptune ML Gremlin 推論查詢的例外狀況清單

Gremlin 推論查詢中使用的 Neptune ML 述詞

`Neptune#ml.deterministic`

這個述詞是針對歸納推論查詢的選項，也就是，針對包含 `Neptune#ml.inductiveInference` 述詞的查詢。

使用歸納推斷時，Neptune 引擎會建立適當的子圖來評估訓練後的 GNN 模型，而這個子圖的需求取決於最終模型的參數。尤其，`num-layer` 參數會決定來自目標節點或邊緣的周遊躍點數目，而 `fanouts` 參數則會指定要在每個躍點上取樣的鄰近項目數 (請參閱 HPO 參數)。

根據預設，歸納推論查詢會在非確定性模式下執行，其中 Neptune 會隨機建置鄰域。在進行預測時，這種正常的隨機鄰近項目抽樣有時會導致不同的預測。

當您在歸納推論查詢中包含 Neptune#ml.deterministic 時，Neptune 引擎會嘗試以確定性的方式對鄰近項目進行取樣，以便相同查詢的多次調用，每次都會傳回相同的結果。不過，不能保證結果是完全確定的，因為對分散式系統的基礎圖形和成品的變更仍然可能會引起波動。

您可以在查詢中包含 Neptune#ml.deterministic 述詞，如下所示：

```
.with("Neptune#ml.deterministic")
```

如果包含 Neptune#ml.deterministic 述詞的查詢中未同時包含 Neptune#ml.inductiveInference，則只需將其忽略即可。

Neptune#ml.disableInductiveInferenceMetadataCache

這個述詞是針對歸納推論查詢的選項，也就是，針對包含 Neptune#ml.inductiveInference 述詞的查詢。

對於歸納推論查詢，Neptune 會使用儲存在 Amazon S3 中的中繼資料檔案，以在建立鄰域時決定躍點數和扇出數。Neptune 通常會快取此模型中繼資料，以避免重複從 Amazon S3 擷取檔案。可在查詢中包含 Neptune#ml.disableInductiveInferenceMetadataCache 述詞來停用快取。雖然 Neptune 直接從 Amazon S3 擷取中繼資料可能會比較慢，但是當 SageMaker 端點在重新訓練或轉換之後更新了且快取過時時，這會很有用。

您可以在查詢中包含 Neptune#ml.disableInductiveInferenceMetadataCache 述詞，如下所示：

```
.with("Neptune#ml.disableInductiveInferenceMetadataCache")
```

以下是查詢範例在 Jupyter 筆記本中的可能樣子：

```
%%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

Neptune#ml.endpoint 述詞會在 with() 步驟中用來指定推論端點 (如有必要)：

```
.with("Neptune#ml.endpoint", "the model's SageMaker inference endpoint")
```

您可以透過端點 id 或其 URL 來識別端點。例如：

```
.with( "Neptune#ml.endpoint", "node-classification-movie-lens-endpoint" )
```

或者：

```
.with( "Neptune#ml.endpoint", "https://runtime.sagemaker.us-east-1.amazonaws.com/
endpoints/node-classification-movie-lens-endpoint/invocations" )
```

> **Note**
>
> 如果您將 Neptune 資料庫叢集參數群組中的 neptune_ml_endpoint 參數設定為端點 id 或 URL，則不需要在每個查詢中包含 Neptune#ml.endpoint 述詞。

Neptune#ml.iamRoleArn

Neptune#ml.iamRoleArn 會在 with() 步驟中用來指定 SageMaker 執行 IAM 角色的 ARN (如有必要)：

```
.with("Neptune#ml.iamRoleArn", "the ARN for the SageMaker execution IAM role")
```

如需如何建立 SageMaker 執行 IAM 角色的相關資訊，請參閱 建立自訂 NeptuneSageMakerIAMRole 角色。

> **Note**
>
> 如果您將 Neptune 資料庫叢集參數群組中的 neptune_ml_iam_role 參數設定為 SageMaker 執行 IAM 角色的 ARN，則不需要在每個查詢中包含 Neptune#ml.iamRoleArn 述詞。

Neptune#ml.inductiveInference

預設會在 Gremlin 中啟用直推式推論。若要進行即時直推式推論查詢，請包含如下 `Neptune#ml.inductiveInference` 述詞：

```
.with("Neptune#ml.inductiveInference")
```

如果您的圖形是動態的，歸納推論通常是最佳選擇，但是如果您的圖形是靜態的，則直推式推論更快、更有效率。

Neptune#ml.limit

`Neptune#ml.limit` 述詞可以選擇性地限制每個實體傳回的結果數量：

```
 .with( "Neptune#ml.limit", 2 )
```

根據預設，限制為 1，但可以設定的數量上限為 100。

Neptune#ml.threshold

`Neptune#ml.threshold` 述詞可以選擇性地建立結果分數的截止閾值：

```
 .with( "Neptune#ml.threshold", 0.5D )
```

這可讓您捨棄分數低於指定閾值的所有結果。

Neptune#ml.classification

`Neptune#ml.classification` 述詞會附加至 `properties()` 步驟，以確定需要從節點分類模型的 SageMaker 端點擷取屬性：

```
 .properties( "property key of the node classification
model" ).with( "Neptune#ml.classification" )
```

Neptune#ml.regression

`Neptune#ml.regression` 述詞會附加至 `properties()` 步驟，以確定需要從節點迴歸模型的 SageMaker 端點擷取屬性：

```
 .properties( "property key of the node regression
model" ).with( "Neptune#ml.regression" )
```

Neptune#ml.prediction

Neptune#ml.prediction 述詞會附加到 in() 和 out() 步驟，以確定這是連結預測查詢：

```
.in("edge label of the link prediction
model").with("Neptune#ml.prediction").hasLabel("target node label")
```

Neptune#ml.score

Neptune#ml.score 述詞用於 Gemlin 節點或邊緣分類查詢中，以擷取機器學習可信度分數。Neptune#ml.score 述詞應該與 properties() 步驟中的查詢述詞一起傳遞，以取得節點或邊緣分類查詢的 ML 可信度分數。

您可以在邊緣分類區段中找到具有其他節點分類範例的節點分類範例，以及邊緣分類範例。

Neptune ML 中的 Gremlin 節點分類查詢

對於 Neptune ML 中的 Gremlin 節點分類：

- 模型根據頂點的一個屬性進行訓練。這個屬性的唯一值集被稱為節點類別集，或者簡稱為類別。
- 頂點屬性的節點類別屬性值可從節點分類模型中推斷出來。這在此屬性尚未附加至頂點時很有用。
- 為了從節點分類模型中擷取一個或多個類別，您需要使用 with() 步驟搭配述詞 Neptune#ml.classification 來設定 properties() 步驟。如果輸出格式是頂點屬性，則輸出格式與您所期望的格式相似。

> **Note**
>
> 節點分類僅會使用字串屬性值。這表示不支援數值屬性值 (例如 0 或 1)，儘管字串等同於 "0" 和 "1"。同樣地，布林屬性值 true 和 false 不起作用，但 "true" 和 "false" 可以。

以下是範例節點分類查詢：

```
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")
```

此查詢的輸出如下所示：

```
==>vp[genre->Action]
==>vp[genre->Crime]
==>vp[genre->Comedy]
```

在上述查詢中，V() 和 properties() 步驟的使用方式如下：

V() 步驟包含您要從節點分類模型中擷取其類別的頂點集：

```
  .V( "movie_1", "movie_2", "movie_3" )
```

properties() 步驟包含模型訓練時根據的索引鍵，而且具有 .with("Neptune#ml.classification") 指示這是節點分類 ML 推論查詢。

properties().with("Neptune#ml.classification") 步驟中目前不支援多個內容索引鍵。例如，下列查詢會導致例外狀況：

```
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")
```

如需特定錯誤訊息，請參閱 Neptune ML 例外狀況清單。

properties().with("Neptune#ml.classification") 步驟可與下列任一步驟搭配使用：

- value()
- value().is()
- hasValue()
- has(value,"")
- key()
- key().is()
- hasKey()
- has(key,"")
- path()

其他節點分類查詢

如果推論端點和對應的 IAM 角色都已儲存在您的資料庫叢集參數群組中，則節點分類查詢可以簡單的如下所示：

```
g.V("movie_1", "movie_2",
 "movie_3").properties("genre").with("Neptune#ml.classification")
```

您可以使用 union() 步驟，在查詢中混合頂點屬性和類別：

```
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")
 )
```

您也可以建立無限制的查詢，如下所示：

```
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")
```

您可以使用 select() 步驟搭配 as() 步驟來擷取節點類別以及頂點：

```
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")
```

您也可以根據節點類別篩選，如下列範例所示：

```
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"))
```

您可以使用 Neptune#ml.score 述詞，取得節點分類可信度分數：

```
 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")
```

回應看起來像這樣：

```
==>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]
```

在節點分類查詢中使用歸納推論

假設您要在 Jupyter 筆記本中將新節點新增至現有圖形，如下所示：

```
%%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')
```

您接著可以使用歸納推論查詢，取得反映了新節點的類型和可信度分數：

```
%%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")
```

不過，如果您執行了多次查詢，則可能會得到有些不同的結果：

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.21365921]
```

你可以使相同的查詢具有確定性：

```
%%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")
```

在這種情況下，每次結果將大致相同：

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
```

Neptune ML 中的 Gremlin 節點迴歸查詢

節點迴歸類似於節點分類，不同之處在於從每個節點的迴歸模型推斷出的值是數值。您可以使用與節點分類相同的 Gremlin 查詢進行節點迴歸，但以下差異除外：

- 同樣，在 Neptune ML 中，節點指的是頂點。

- properties() 步驟接受形式 properties().with("Neptune#ml.regression") 而不是 properties().with("Neptune#ml.classification")。
- "Neptune#ml.limit" 和 "Neptune#ml.threshold" 述詞不適用。
- 根據值進行篩選時，您必須指定一個數值。

以下是範例頂點分類查詢：

```
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")
```

您可以根據使用迴歸模型推斷出的值進行篩選，如下列範例所示：

```
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))
```

在節點迴歸查詢中使用歸納推論

假設您要在 Jupyter 筆記本中將新節點新增至現有圖形，如下所示：

```
%%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')
```

然後，您可以使用歸納推論查詢，來取得考慮到新節點的評分：

```
%%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")
```

因為查詢不具確定性，所以如果根據鄰域多次執行該查詢，其可能會傳回有些不同的結果：

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->8.9]
```

如果需要更一致的結果，您可以使查詢具有確定性：

```
%%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")
```

現在每次結果都會大致相同：

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->9.1]
```

Neptune ML 中的 Gremlin 邊緣分類查詢

對於 Neptune ML 中的 Gremlin 邊緣分類：

- 模型根據邊緣的一個屬性進行訓練。這個屬性的唯一值集被稱為類別集。
- 邊緣的類別屬性值可從邊緣分類模型推斷出來，這在此屬性尚未附加至邊緣時很有用。
- 為了從邊緣分類模型中擷取一個或多個類別，您需要使用 `with()` 步驟搭配述詞 `"Neptune#ml.classification"` 來設定 `properties()` 步驟。如果輸出格式是邊緣邊緣屬性，則輸出格式與您所期望的格式相似。

> **Note**
>
> 邊緣分類僅會使用字串屬性值。這表示不支援數值屬性值 (例如 `0` 或 `1`)，儘管字串等同於 `"0"` 和 `"1"`。同樣地，布林屬性值 `true` 和 `false` 不起作用，但 `"true"` 和 `"false"` 可以。

以下是邊緣分類查詢的範例，其會使用 `Neptune#ml.score` 述詞請求可信度分數：

```
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")
```

回應看起來像這樣：

```
==>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]
```

Gremlin 邊緣分類查詢的語法

對於簡單圖形，其中 `User` 是前端和尾端節點，且 `Relationship` 是連線它們的邊緣，範例邊緣分類查詢如下：

```
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")
```

此查詢的輸出如下所示：

```
==>p[knows_by->"Family"]
==>p[knows_by->"Friends"]
==>p[knows_by->"Colleagues"]
```

在上述查詢中，`E()` 和 `properties()` 步驟的使用方式如下：

- E() 步驟包含您要從邊緣分類模型中擷取其類別的邊緣集：

```
.E("relationship_1","relationship_2","relationship_3")
```

- properties() 步驟包含模型訓練時根據的索引鍵，而且具有 .with("Neptune#ml.classification") 指示這是邊緣分類 ML 推論查詢。

properties().with("Neptune#ml.classification") 步驟中目前不支援多個內容索引鍵。例如，下列查詢會導致擲出例外狀況：

```
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")
```

如需特定錯誤訊息，請參閱 Neptune ML Gremlin 推論查詢的例外狀況清單。

properties().with("Neptune#ml.classification") 步驟可與下列任一步驟搭配使用：

- value()
- value().is()
- hasValue()
- has(value,"")
- key()
- key().is()
- hasKey()
- has(key,"")
- path()

在邊緣分類查詢中使用歸納推論

假設您要在 Jupyter 筆記本中將新邊緣新增至現有圖形，如下所示：

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
```

```
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

然後，您可以使用歸納推論查詢，來取得考慮到新邊緣的範圍：

```
%%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")
```

因為查詢不具確定性，所以如果根據隨機鄰域多次執行該查詢，結果將會有些不同：

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.21365921]
```

如果需要更一致的結果，您可以使查詢具有確定性：

```
%%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")
```

現在，每次執行查詢時，結果都會或多或少相同：

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]
```

Neptune ML 中的 Gremlin 邊緣迴歸查詢

邊緣迴歸類似於邊緣分類，不同之處在於從 ML 模型推斷出的值是數值。對於邊緣迴歸，Neptune ML 支援與分類相同的查詢。

需要注意的要點如下：

- 您必須使用 ML 述詞 `"Neptune#ml.regression"`，針對此使用案例設定 `properties()` 步驟。
- `"Neptune#ml.limit"` 和 `"Neptune#ml.threshold"` 述詞不適用於此使用案例。
- 如需根據值進行過濾，您需要將該值指定為數值。

Gremlin 邊緣迴歸查詢的語法

對於簡單圖形，其中 `User` 是前端節點、`Movie` 是尾端節點，以及 `Rated` 是連接它們的邊緣，以下是一個範例邊緣迴歸查詢，其會尋找邊緣 `Rated` 的數值評分值，在這裡稱為分數：

```
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")
```

您也可以根據從 ML 迴歸模型推斷出的值進行篩選。對於由 `"rating_1"`、`"rating_2"` 和 `"rating_3"` 所識別的現有 `Rated` 邊緣 (從 `User` 到 `Movie`)，其中這些評分不存在邊緣屬性 `Score`，您可以使用如下查詢來推斷邊緣的 `Score`，其中它大於或等於 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))
```

在邊緣迴歸查詢中使用歸納推論

假設您要在 Jupyter 筆記本中將新邊緣新增至現有圖形，如下所示：

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

然後，您可以使用歸納推論查詢，來取得考慮到新邊緣的分數：

```
%%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")
```

因為查詢不具確定性，所以如果根據隨機鄰域多次執行該查詢，結果將會有些不同：

```
# First time
==>ep[score->96]

# Second time
==>ep[score->91]
```

如果需要更一致的結果，您可以使查詢具有確定性：

```
%%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")
```

現在，每次執行查詢時，結果都會或多或少相同：

```
# First time
==>ep[score->96]

# Second time
==>ep[score->96]
```

在 Neptune ML 中使用連結預測模型進行 Gremlin 連結預測查詢

連結預測模型可以解決如下問題：

- 前端節點預測：鑑於頂點和邊緣類型，該頂點可能從哪些頂點連結？
- 尾端節點預測：鑑於頂點和邊緣標籤，該頂點可能連結至哪些頂點？

> **Note**
>
> Neptune ML 中尚未支援邊緣預測。

對於下面範例，考慮其中頂點 `User` 和 `Movie` 是由邊緣 `Rated` 連結的簡單圖形。

以下是範例前端節點預測查詢，用來預測最有可能評分電影 (`"movie_1"`、`"movie_2"` 和 `"movie_3"`) 的前五名使用者：

```
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")
```

以下是用於尾端節點預測的類似查詢，用來預測使用者 `"user_1"` 可能評分的前五名電影：

```
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")
```

邊緣標籤和預測的頂點標籤都是必要的。如果省略其中一個，則會擲出例外狀況。例如，以下查詢沒有預測的頂點標籤，其會擲回例外狀況：

```
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")
```

同樣地，沒有邊緣標籤的下列查詢也會擲回例外狀況：

```
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")
```

如需這些例外狀況傳回的特定錯誤訊息，請參閱 Neptune ML 例外狀況清單。

其他連結預測查詢

您可以使用 `select()` 步驟搭配 `as()` 步驟，以同時輸出預測的頂點與輸入頂點：

```
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")
```

您可以進行無限制查詢，如下所示：

```
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")
```

在連結預測查詢中使用歸納推論

假設您要在 Jupyter 筆記本中將新節點新增至現有圖形，如下所示：

```
%%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')
```

然後，您可以使用歸納推論查詢來預測前端節點，同時考慮到新節點：

```
%%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")
```

結果：

```
==>V[2]
```

同樣地，您可以使用歸納推論查詢來預測尾端節點，同時考慮到新節點：

```
%%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")
```

結果：

```
==>V[1]
```

Neptune ML Gremlin 推論查詢的例外狀況清單

- **BadRequestException** – 無法載入所提供角色的憑證。

  訊息：`Unable to load credentials for role: the specified IAM Role ARN.`
- **BadRequestException** – 指定的 IAM 角色未獲得授權，無法調用 SageMaker 端點。

  訊息：`User: the specified IAM Role ARN is not authorized to perform: sagemaker:InvokeEndpoint on resource: the specified endpoint.`
- **BadRequestException** – 指定的端點不存在。

  訊息：`Endpoint the specified endpoint not found.`
- **InternalFailureException** – 無法從 Amazon S3 擷取 Neptune ML 即時歸納推論中繼資料。

訊息：`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 無法在 Amazon S3 中找到用於即時歸納推論的中繼資料檔案。

  訊息：`Neptune ML cannot find the metadata file for Real-Time Inductive Inference in S3.`

- **InvalidParameterException** – 指定的端點在語法上無效。

  訊息：`Invalid endpoint provided for external service query.`

- **InvalidParameterException** – 指定的 SageMaker 執行 IAM 角色 ARN 在語法上無效。

  訊息：`Invalid IAM role ARN provided for external service query.`

- **InvalidParameterException** – 在查詢的 `properties()` 步驟中指定多個屬性索引鍵。

  訊息：`ML inference queries are currently supported for one property key.`

- **InvalidParameterException** – 在查詢中指定多個邊緣標籤。

  訊息：`ML inference are currently supported only with one edge label.`

- **InvalidParameterException** – 在查詢中指定多個頂點標籤限制條件。

  訊息：`ML inference are currently supported only with one vertex label constraint.`

- **InvalidParameterException** – `Neptune#ml.classification` 和 `Neptune#ml.regression` 述詞都存在於相同的查詢中。

  訊息：`Both regression and classification ML predicates cannot be specified in the query.`

- **InvalidParameterException** – 已在連結預測查詢的 `in()` 或 `out()` 步驟中指定多個邊緣標籤。

  訊息：`ML inference are currently supported only with one edge label.`

- **InvalidParameterException** – 已使用 Neptune#ml.score 指定多個屬性索引鍵。

  訊息：`Neptune ML inference queries are currently supported for one property key and one Neptune#ml.score property key.`

- **MissingParameterException** – 未在查詢中指定端點，或未將其指定為資料庫叢集參數。

  訊息：`No endpoint provided for external service query.`
- MissingParameterException – 未在查詢中指定 SageMaker 執行 IAM 角色，或未將其指定為資料庫叢集參數。

  訊息：`No IAM role ARN provided for external service query.`
- **MissingParameterException** – 查詢中的 `properties()` 步驟遺失內容索引鍵。

  訊息：`Property key needs to be specified using properties() step for ML inference queries.`
- **MissingParameterException** – 未在連結預測查詢的 `in()` 或 `out()` 步驟中指定任何邊緣標籤。

  訊息：`Edge label needs to be specified while using in() or out() step for ML inference queries.`
- **MissingParameterException** – 未使用 Neptune#ml.score 指定屬性索引鍵。

  訊息：`Property key needs to be specified along with Neptune#ml.score property key while using the properties() step for Neptune ML inference queries.`
- **UnsupportedOperationException** – `both()` 步驟用於連結預測查詢中。

  訊息：`ML inference queries are currently not supported with both() step.`
- **UnsupportedOperationException** – 未在 `has()` 步驟以及連結預測查詢中的 `in()` 或 `out()` 步驟中指定預測的頂點標籤。

  訊息：`Predicted vertex label needs to be specified using has() step for ML inference queries.`
- **UnsupportedOperationException** – 未最佳化的步驟目前不支援 Gremlin ML 歸納推論查詢。

  訊息：`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** – `repeat` 步驟內目前不支援 Neptune ML 推論查詢。

  訊息：`Neptune ML inference queries are currently not supported inside a repeat step.`

- **UnsupportedOperationException** – 每個 Grinlin 查詢目前不支援多個 Neptune ML 推論查詢。

  訊息：`Neptune ML inference queries are currently supported only with one ML inference query per gremlin query.`

Neptune ML 中的 SPARQL 推論查詢

Neptune ML 會將 RDF 圖形對應到屬性圖，對 ML 任務進行建模。目前，它支援下列使用案例：

- 物件分類 – 預測物件的類別特徵。
- 物件迴歸 – 預測物件的數值屬性。
- 物件預測 – 預測獲給與主旨和關係的物件。
- 主旨預測 – 預測獲給與物件和關係的物件。

> **Note**
>
> Neptune ML 不支援主旨分類和迴歸使用案例搭配 SPARQL。

SPARQL 推論查詢中使用的 Neptune ML 述詞

下列述詞與 SPARQL 推論搭配使用：

`neptune-ml:timeout` 述詞

針對與遠端伺服器的連線指定逾時。不應與查詢請求逾時混淆，這是伺服器滿足請求所需的時間量上限。

請注意，如果在 `neptune-ml:timeout` 述詞指定的服務逾時發生之前發生查詢逾時，也會取消服務連線。

`neptune-ml:outputClass` 述詞

`neptune-ml:outputClass` 述詞僅用來定義所預測物件的類別進行物件預測，或定義所預測主旨的類別進行主旨預測。

`neptune-ml:outputScore` 述詞

`neptune-ml:outputScore` 述詞是正數，代表機器學習模型輸出正確的可能性。

`neptune-ml:modelType` 述詞

`neptune-ml:modelType` 述詞指定正在訓練的機器學習模型類型：

- `OBJECT_CLASSIFICATION`

- OBJECT_REGRESSION
- OBJECT_PREDICTION
- SUBJECT_PREDICTION

neptune-ml:input 述詞

neptune-ml:input 述詞是指用作 Neptune ML 輸入的 URI 清單。

neptune-ml:output 述詞

neptune-ml:output 述詞是指繫結集的清單，而 Neptune ML 會在這些繫結中傳回結果。

neptune-ml:predicate 述詞

neptune-ml:predicate 述詞的使用方式會有所不同，取決於正在執行的任務：

- 對於物件或主旨預測：定義述詞的類型 (邊緣或關係類型)。
- 對於物件分類和迴歸：定義要預測的常值 (屬性)。

neptune-ml:batchSize 述詞

neptune-ml:batchSize 指定遠端服務呼叫的輸入大小。

SPARQL 物件分類範例

對於 Neptune ML 中的 SPARQL 物件分類，模型會根據其中一個述詞值進行訓練。在該述詞尚未與給定主旨一起存在時，這很有用。

只能使用物件分類模型推斷類別述詞值。

下列查詢試圖為類型 foaf:Person 的所有輸入預測 <http://www.example.org/team> 述詞值：

```
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 .
  }
}
```

此查詢可以自訂如下：

```
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 .
  }
}
```

SPARQL 物件迴歸範例

物件迴歸類似於物件分類，不同之處在於從每個節點的迴歸模型推斷出一個數值述詞值。如同物件分類一般，您可以將相同的 SPARQL 查詢用於物件迴歸，但 the `Neptune#ml.limit` 與 `Neptune#ml.threshold` 述詞不適用。

下列查詢試圖為類型 `foaf:Person` 的所有輸入預測 <http://www.example.org/accountbalance> 述詞值：

```
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 .
  }
}
```

此查詢可以自訂如下：

```
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 .
  }
}
```

SPARQL 物件預測範例

「物件預測」會預測給定主旨和述詞的物件值。

下列物件預測查詢試圖預測類型 `foaf:Person` 的輸入會喜歡什麼電影：

```
?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> .
  }
}
```

查詢本身可以自訂如下：

```
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> .
  }
}
```

SPARQL 主旨預測範例

主旨預測會預測指定述詞和物件的主旨。

例如，下列查詢預測誰 (類型 `foaf:User`) 將觀看給定的電影：

```
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> ;        }
}
```

Neptune ML SPARQL 推論查詢的例外狀況清單

- **BadRequestException** – 訊息：`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** – 訊息：`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** – 訊息：Invalid predicate *(predicate name)* provided for external service http://aws.amazon.com/neptune/vocab/v01/services/ml#inference query.
- **BadRequestException** – 訊息：The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the predicate *(predicate name)* to be defined。
- **BadRequestException** – 訊息：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** – 訊息：The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the input *(parameter name)* to be a constant, found: *(type)*。
- **BadRequestException** – 訊息：The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference is expected to return only 1 value。
- **BadRequestException** – 訊息："The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference only allows StatementPatternNodes。
- **BadRequestException** – 訊息：The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference does not allow the predicate *(predicate name)*。
- **BadRequestException** – 訊息：The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates cannot be variables, found: *(type)*。
- **BadRequestException** – 訊息：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)*。

Neptune ML 管理 API 參考

內容

使用 **dataprocessing** 命令進行資料處理

您可以使用 Neptune ML `dataprocessing` 命令來建立資料處理工作、檢查其狀態、停止該工作，或列出所有作用中的資料處理工作。

使用 Neptune ML **dataprocessing** 命令建立資料處理工作

用於建立新工作的典型 Neptune ML `dataprocessing` 命令如下所示：

```
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)"
      }'
```

要啟動增量重新處理的命令如下所示：

```
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)"
}'
```

用於建立 **dataprocessing** 工作的參數

- **id** – (選用) 新工作的唯一識別符。

  類型：字串 預設值：自動產生的 UUID。
- **previousDataProcessingJobId** – (選用) 在舊版資料上執行且已完成之資料處理工作的工作 ID。

類型：字串 預設值：none。

注意：使用此項進行增量資料處理，以在圖形資料變更時更新模型 (但在資料已遭刪除時不適用)。

- **`inputDataS3Location`** – (必要) Amazon S3 位置的 URI，您想要 SageMaker 在該位置下載執行資料處理工作所需的資料。

  類型：字串

- **`processedDataS3Location`** – (必要) Amazon S3 位置的 URI，您想要 SageMaker 在該位置儲存資料處理工作的結果。

  類型：字串

- **`sagemakerIamRoleArn`** – (選用) 用於 SageMaker 執行之 IAM 角色的 ARN。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- **`neptuneIamRoleArn`** – (選用) IAM 角色的 Amazon Resource Name (ARN)，SageMaker 可以擔任這個角色來代表您執行任務。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- **`processingInstanceType`** – (選用) 資料處理期間所使用的 ML 執行個體類型。它的記憶體應該大到足以保留處理後的資料集。

  類型：字串 預設值：最小的 `ml.r5` 類型，其記憶體十倍於磁碟上所匯出圖形資料的大小。

  注意：Neptune ML 可以自動選取執行個體類型。請參閱 選取執行個體進行資料處理。

- **`processingInstanceVolumeSizeInGB`** – (選用) 處理執行個體的磁碟區大小。輸入資料和處理後的資料都會儲存在磁碟上，因此磁碟區大小必須大到足以保留這兩個資料集。

  類型：整數。預設：`0`。

  注意：如果未指定或指定 0，Neptune ML 會根據資料大小自動選擇磁碟區大小。

- **`processingTimeOutInSeconds`** – (選用) 資料處理工作的逾時 (以秒為單位)。

  類型：整數。預設值：`86,400` (1 天)。

- **`modelType`** – (選用) Neptune ML 目前支援的兩種模型類型之一：異質圖形模型 (`heterogeneous`) 和知識圖譜 (`kge`)。

  類型：字串 預設值：none。

注意：如果未指定，Neptune ML 會根據資料自動選擇模型類型。

- **`configFileName`** – (選用) 描述如何載入所匯出圖形資料進行訓練的資料規格檔案。Neptune 匯出工具組會自動產生此檔案。

  類型：字串 預設：`training-data-configuration.json`。

- **`subnets`** – (選用) Neptune VPC 中子網路的 ID。

  類型：字串清單。預設值：none。

- **`securityGroupIds`** – (選用) VPC 安全群組 ID。

  類型：字串清單。預設值：none。

- **`volumeEncryptionKMSKey`** – (選用) SageMaker 用來加密資料旳 AWS Key Management Service (AWS KMS) 金鑰，這些資料位於附加至執行處理工作之 ML 運算執行個體的儲存磁碟區。

  類型：字串 預設值：none。

- **`enableInterContainerTrafficEncryption`** – (選用) 在訓練或超參數調校工作中啟用或停用容器間流量加密。

  類型：布林值。預設值：true。

  > **Note**
  >
  > `enableInterContainerTrafficEncryption` 參數僅適用於引擎版本 1.2.0.2.R3。

- **`s3OutputEncryptionKMSKey`** – (選用) SageMaker 用來加密訓練工作輸出的 AWS Key Management Service (AWS KMS) 金鑰。

  類型：字串 預設值：none。

使用 Neptune ML **`dataprocessing`** 命令取得資料處理工作的狀態

工作狀態的範例 Neptune ML `dataprocessing` 命令如下所示：

```
curl -s \
  "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)" \
  | python -m json.tool
```

dataprocessing 工作狀態的參數

- **id** – (必要) 資料處理工作的唯一識別符。

  類型：字串

- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

使用 Neptune ML **dataprocessing** 命令停止資料處理工作

用於停止工作的範例 Neptune ML `dataprocessing` 命令如下所示：

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)"
```

或如下所示：

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/dataprocessing/(the job ID)?clean=true"
```

dataprocessing 停止工作的參數

- **id** – (必要) 資料處理工作的唯一識別符。

  類型：字串

- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- **clean** – (選用) 此旗標指定在工作停止時應刪除所有 Amazon S3 成品。

  類型：布林值。預設：FALSE。

使用 Neptune ML **dataprocessing** 命令列出作用中的資料處理工作

用於列出作用中工作的範例 Neptune ML `dataprocessing` 命令如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing"
```

或如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/dataprocessing?maxItems=3"
```

dataprocessing 列出工作的參數

- **maxItems** – (選用) 要傳回的項目數上限。

  類型：整數。預設：10。允許的最大值：1024。
- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

使用 **modeltraining** 命令進行模型訓練

您可以使用 Neptune ML `modeltraining` 命令來建立模型訓練工作、檢查其狀態、停止該工作，或列出所有作用中的模型訓練工作。

使用 Neptune ML **modeltraining** 命令建立模型訓練工作

用於建立全新工作的 Neptune ML `modeltraining` 命令如下所示：

```
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"
      }'
```

用於為增量模型訓練建立更新工作的 Neptune ML `modeltraining` 命令如下所示：

```
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)",
      }'
```

透過使用者提供的自訂模型實作建立新工作的 Neptune ML `modeltraining` 命令如下所示：

```
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)"
        }
      }'
```

用於建立 **modeltraining** 工作的參數

- **id** – (選用) 新工作的唯一識別符。

  類型：字串 預設值：自動產生的 UUID。
- **dataProcessingJobId** – (必要) 已完成資料處理工作的工作 ID，該工作已建立訓練將使用的資料。

  類型：字串
- **trainModelS3Location** – (必要) Amazon S3 中要儲存模型成品的位置。

  類型：字串
- **previousModelTrainingJobId** – (選用) 已完成模型訓練工作的工作 ID，您想要根據更新的資料以增量方式更新此工作。

  類型：字串 預設值：none。
- **sagemakerIamRoleArn** – (選用) 用於 SageMaker 執行之 IAM 角色的 ARN。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。
- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。
- **modelName** – (選用) 用於訓練的模型類型。根據預設，ML 模型會是自動以資料處理中使用的 `modelType` 為基礎，但您可以在這裡指定不同的模型類型。

類型：字串 預設值：rgcn 用於異質圖和 kge 用於知識圖譜。有效值：若為異質圖：rgcn。若為 kge 圖形：transe、distmult 或 rotate。若為自訂模型實作：custom。

- **baseProcessingInstanceType** – (選用) 用於準備和管理 ML 模型訓練的 ML 執行個體類型。

  類型：字串 注意：這是根據記憶體需求選擇的 CPU 執行個體，用於處理訓練資料和模型。請參閱 選取執行個體進行模型訓練和模型轉換。

- **trainingInstanceType** – (選用) 用於模型訓練的 ML 執行個體類型。所有 Neptune ML 模型都支援 CPU、GPU 和多 GPU 訓練。

  類型：字串 預設：ml.p3.2xlarge。

  注意：選擇適合訓練的執行個體類型取決於工作類型、圖形大小和您的預算。請參閱 選取執行個體進行模型訓練和模型轉換。

- **trainingInstanceVolumeSizeInGB** – (選用) 訓練執行個體的磁碟區大小。輸入資料和輸出模型都會儲存在磁碟上，因此磁碟區大小必須大到足以保留這兩個資料集。

  類型：整數。預設：0。

  備註：如果未指定或指定 0，Neptune ML 會根據資料處理步驟中產生的建議選取磁碟區大小。請參閱 選取執行個體進行模型訓練和模型轉換。

- **trainingTimeOutInSeconds** – (選用) 訓練工作的逾時 (以秒為單位)。

  類型：整數。預設值：86,400 (1 天)。

- **maxHPONumberOfTrainingJobs** – 要對超參數調校工作啟動的訓練工作總數上限。

  類型：整數。預設：2。

  注意：Neptune ML 會自動調校機器學習模型的超參數。若要取得效能良好的模型，請至少使用 10 個工作 (換句話說，將 maxHPONumberOfTrainingJobs 設為 10)。一般來說，調校執行越多，結果越好。

- **maxHPOParallelTrainingJobs** – 要對超參數調校工作啟動的並行訓練工作數目上限。

  類型：整數。預設：2。

  注意：您可以執行的並行工作數目受制於訓練執行個體上可用的資源。

- **subnets** – (選用) Neptune VPC 中子網路的 ID。

  類型：字串清單。預設值：none。

- **securityGroupIds** – (選用) VPC 安全群組 ID。

  類型：字串清單。預設值：none。

- **volumeEncryptionKMSKey** – (選用) SageMaker 用來加密資料的 AWS Key Management Service (AWS KMS) 金鑰，這些資料位於附加至執行訓練工作之 ML 運算執行個體的儲存磁碟區。

  類型：字串 預設值：none。

- **s3OutputEncryptionKMSKey** – (選用) SageMaker 用來加密處理工作輸出的 AWS Key Management Service (AWS KMS) 金鑰。

  類型：字串 預設值：none。

- **enableInterContainerTrafficEncryption** – (選用) 在訓練或超參數調校工作中啟用或停用容器間流量加密。

  類型：布林值。預設值：true。

  > **Note**
  >
  > enableInterContainerTrafficEncryption 參數僅適用於引擎版本 1.2.0.2.R3。

- **enableManagedSpotTraining** – (選用) 使用 Amazon Elastic Compute Cloud Spot 執行個體，將訓練機器學習模型的成本最佳化。如需詳細資訊，請參閱 Amazon SageMaker 中的受管 SageMaker 訓練。

  類型：布林值。預設值：false。

- **customModelTrainingParameters** – (選用) 自訂模型訓練的組態。這是具有下列欄位的 JSON 物件：

  - **sourceS3DirectoryPath** – (必要) 此路徑通往實作您模型之 Python 模組所在的 Amazon S3 位置。這必須指向有效的現有 Amazon S3 位置，其中至少包含訓練指令碼、轉換指令碼和 model-hpo-configuration.json 檔案。
  - **trainingEntryPointScript** – (選用) 指令碼模組中的進入點名稱，該指令碼會執行模型訓練，並接受超參數作為命令列引數 (包括固定的超參數)。

    預設：training.py。
  - **transformEntryPointScript** – (選用) 指令碼模組中的進入點名稱，該指令碼應在識別了超參數搜尋中的最佳模型之後執行，以計算模型部署所需的模型成品。它應該能夠在沒有命令列參數的情況下執行。

預設：transform.py。

- **maxWaitTime** – (選用) 使用 Spot 執行個體來執行模型訓練時，要等待的時間上限 (以秒為單位)。應大於 trainingTimeOutInSeconds。

  類型：整數。

使用 Neptune ML **modeltraining** 命令取得模型訓練工作的狀態

工作狀態的範例 Neptune ML modeltraining 命令如下所示：

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)" \
  | python -m json.tool
```

modeltraining 工作狀態的參數

- **id** – (必要) 模型訓練工作的唯一識別符。

  類型：字串

- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

使用 Neptune ML **modeltraining** 命令停止模型訓練工作

用於停止工作的範例 Neptune ML modeltraining 命令如下所示：

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)"
```

或如下所示：

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltraining/(the job ID)?clean=true"
```

modeltraining 停止工作的參數

- **id** – (必要) 模型訓練工作的唯一識別符。

  類型：字串

- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- **clean** – (選用) 此旗標指定在工作停止時應刪除所有 Amazon S3 成品。

  類型：布林值。預設：FALSE。

使用 Neptune ML **modeltraining** 命令列出作用中的模型訓練工作

用於列出作用中工作的範例 Neptune ML `modeltraining` 命令如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining" | python -m json.tool
```

或如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/modeltraining?maxItems=3" | python -m
 json.tool
```

modeltraining 列出工作的參數

- **maxItems** – (選用) 要傳回的項目數上限。

  類型：整數。預設：10。允許的最大值：1024。

- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

使用 **modeltransform** 命令進行模型轉換

您可以使用 Neptune ML `modeltransform` 命令來建立模型轉換工作、檢查其狀態、停止該工作，或列出所有作用中的模型轉換工作。

使用 Neptune ML **modeltransform** 命令建立模型轉換工作

用於建立增量轉換工作而不需模型重新訓練的 Neptune ML `modeltransform` 命令如下所示：

```
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"
      }'
```

用於從已完成的 SageMaker 訓練工作建立工作的 Neptune ML `modeltransform` 命令如下所示：

```
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" : ""
      }'
```

用於建立一個工作，使用自訂模型實作的 Neptune ML `modeltransform` 命令如下所示：

```
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)"
        }
      }'
```

用於建立 **modeltransform** 工作的參數

- **id** – (選用) 新工作的唯一識別符。

  類型：字串 預設值：自動產生的 UUID。
- **dataProcessingJobId** – 已完成資料處理工作的工作 ID。

  類型：字串

  附註：您必須同時包含 dataProcessingJobId 和 mlModelTrainingJobId 或 trainingJobName。
- **mlModelTrainingJobId** – 已完成模型訓練工作的工作 ID。

  類型：字串

  附註：您必須同時包含 dataProcessingJobId 和 mlModelTrainingJobId 或 trainingJobName。
- **trainingJobName** – 已完成 SageMaker 訓練工作的名稱。

  類型：字串

  注意：您必須同時包含 dataProcessingJobId 和 mlModelTrainingJobId 參數或包含 trainingJobName 參數。
- **sagemakerIamRoleArn** – (選用) 用於 SageMaker 執行之 IAM 角色的 ARN。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。
- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- **customModelTransformParameters** – (選用) 使用自訂模型進行模型轉換的組態資訊。customModelTransformParameters 物件包含下列欄位，其具有的值必須與訓練工作中儲存的模型參數相容：
  - **sourceS3DirectoryPath** – (必要) 此路徑通往實作您模型之 Python 模組所在的 Amazon S3 位置。這必須指向有效的現有 Amazon S3 位置，其中至少包含訓練指令碼、轉換指令碼和 model-hpo-configuration.json 檔案。
  - **transformEntryPointScript** – (選用) 指令碼模組中的進入點名稱，該指令碼應在識別了超參數搜尋中的最佳模型之後執行，以計算模型部署所需的模型成品。它應該能夠在沒有命令列參數的情況下執行。

    預設：transform.py。
- **baseProcessingInstanceType** – (選用) 用於準備和管理 ML 模型訓練的 ML 執行個體類型。

  類型：字串 注意：這是根據記憶體需求選擇的 CPU 執行個體，用於處理轉換資料和模型。請參閱選取執行個體進行模型訓練和模型轉換。
- **baseProcessingInstanceVolumeSizeInGB** – (選用) 訓練執行個體的磁碟區大小。輸入資料和輸出模型都會儲存在磁碟上，因此磁碟區大小必須大到足以保留這兩個資料集。

  類型：整數。預設：0。

  備註：如果未指定或指定 0，Neptune ML 會根據資料處理步驟中產生的建議選取磁碟區大小。請參閱 選取執行個體進行模型訓練和模型轉換。
- **subnets** – (選用) Neptune VPC 中子網路的 ID。

  類型：字串清單。預設值：none。
- **securityGroupIds** – (選用) VPC 安全群組 ID。

  類型：字串清單。預設值：none。
- **volumeEncryptionKMSKey** – (選用) SageMaker 用來加密資料的 AWS Key Management Service (AWS KMS) 金鑰，這些資料位於附加至執行轉換工作之 ML 運算執行個體的儲存磁碟區。

  類型：字串 預設值：none。
- **enableInterContainerTrafficEncryption** – (選用) 在訓練或超參數調校工作中啟用或停用容器間流量加密。

  類型：布林值。預設值：true。

> **Note**
>
> `enableInterContainerTrafficEncryption` 參數僅適用於引擎版本 1.2.0.2.R3。

- **`s3OutputEncryptionKMSKey`** – (選用) SageMaker 用來加密處理工作輸出的 AWS Key Management Service (AWS KMS) 金鑰。

  類型：字串 預設值：none。

使用 Neptune ML **`modeltransform`** 命令取得模型轉換工作的狀態

工作狀態的範例 Neptune ML `modeltransform` 命令如下所示：

```
curl -s \
  "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)" \
  | python -m json.tool
```

`modeltransform` 工作狀態的參數

- **`id`** – (必要) 模型轉換工作的唯一識別符。

  類型：字串

- **`neptuneIamRoleArn`** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

使用 Neptune ML **`modeltransform`** 命令停止模型轉換工作

用於停止工作的範例 Neptune ML `modeltransform` 命令如下所示：

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)"
```

或如下所示：

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/modeltransform/(the job ID)?clean=true"
```

`modeltransform` 停止工作的參數

- **`id`** – (必要) 模型轉換工作的唯一識別符。

  類型：字串

- **`neptuneIamRoleArn`** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- **`clean`** – (選用) 此旗標指定在工作停止時應刪除所有 Amazon S3 成品。

  類型：布林值。預設：FALSE。

使用 Neptune ML **`modeltransform`** 命令列出作用中的模型轉換工作

用於列出作用中工作的範例 Neptune ML `modeltransform` 命令如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform" | python -m json.tool
```

或如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/modeltransform?maxItems=3" | python -m
 json.tool
```

`modeltransform` 列出工作的參數

- **`maxItems`** – (選用) 要傳回的項目數上限。

  類型：整數。預設：10。允許的最大值：1024。

- **`neptuneIamRoleArn`** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

使用 **endpoints** 命令管理推論端點

您可以使用 Neptune ML `endpoints` 命令來建立推論端點、檢查其狀態、將其刪除，或列出現有的推論端點。

使用 Neptune ML **endpoints** 命令建立推論端點

如下 Neptune ML `endpoints` 命令可從訓練工作所建立的模型建立推論端點：

```
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)"
      }'
```

如下 Neptune ML `endpoints` 命令可從訓練工作所建立的模型更新現有的推論端點：

```
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)"
      }'
```

如下 Neptune ML `endpoints` 命令可從模型轉換工作所建立的模型建立推論端點：

```
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)"
      }'
```

如下 Neptune ML `endpoints` 命令可從模型轉換工作所建立的模型更新現有的推論端點：

```
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)"
      }'
```

用於建立 **endpoints** 推論端點的參數

- **id** – (選用) 新推論端點的唯一識別符。

  類型：字串 預設值：自動產生的時間戳記名稱。
- **mlModelTrainingJobId** – 已完成模型訓練工作的工作 ID，該工作已建立推論端點將指向的模型。

  類型：字串

  注意：您必須提供 `mlModelTrainingJobId` 或 `mlModelTransformJobId`。
- **mlModelTransformJobId** – 已完成模型轉換工作的工作 ID。

  類型：字串

  注意：您必須提供 `mlModelTrainingJobId` 或 `mlModelTransformJobId`。
- **update** – (選用) 如果存在，此參數指示這是更新請求。

  類型：布林值。預設：`false`

  注意：您必須提供 `mlModelTrainingJobId` 或 `mlModelTransformJobId`。
- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會擲回錯誤。
- **modelName** – (選用) 用於訓練的模型類型。根據預設，ML 模型會是自動以資料處理中使用的 `modelType` 為基礎，但您可以在這裡指定不同的模型類型。

  類型：字串 預設值：`rgcn` 用於異質圖和 `kge` 用於知識圖譜。有效值：若為異質圖：`rgcn`。若為知識圖譜：`kge`、`transe`、`distmult` 或 `rotate`。
- **instanceType** – (選用) 用於線上服務的 ML 執行個體類型。

  類型：字串 預設：`ml.m5.xlarge`。

注意：為推論端點選擇 ML 執行個體，取決於任務類型、圖形大小以及您的預算。請參閱為推論端點選取執行個體。

- **instanceCount** – (選用) 要部署到端點進行預測的 Amazon EC2 執行個體數量下限。

  類型：整數。預設：1。

- **volumeEncryptionKMSKey** – (選用) SageMaker 用來加密資料的 AWS Key Management Service (AWS KMS) 金鑰，這些資料位於附加至執行端點之 ML 運算執行個體的儲存磁碟區。

  類型：字串 預設值：none。

使用 Neptune ML **endpoints** 命令取得推論端點的狀態

執行個體端點狀態的範例 Neptune ML `endpoints` 命令如下所示：

```
curl -s \
  "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)" \
  | python -m json.tool
```

endpoints 執行個體-端點狀態的參數

- **id** – (必要) 推論端點的唯一識別符。

  類型：字串

- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會擲回錯誤。

使用 Neptune ML **endpoints** 命令刪除執行個體端點

用於刪除執行個體端點的範例 Neptune ML `endpoints` 命令如下所示：

```
curl -s \
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)"
```

或如下所示：

```
curl -s \
```

```
  -X DELETE "https://(your Neptune endpoint)/ml/endpoints/(the inference endpoint ID)?
clean=true"
```

endpoints 刪除推論端點的參數

- **id** – (必要) 推論端點的唯一識別符。

  類型：字串
- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會擲回錯誤。
- **clean** – (選用) 指示也應刪除與此端點相關的所有成品。

  類型：布林值。預設：FALSE。

使用 Neptune ML **endpoints** 命令列出推論端點

用於列出推論端點的 Neptune ML `endpoints` 命令如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/endpoints" \
  | python -m json.tool
```

或如下所示：

```
curl -s "https://(your Neptune endpoint)/ml/endpoints?maxItems=3" \
  | python -m json.tool
```

dataprocessing 列出推論端點的參數

- **maxItems** – (選用) 要傳回的項目數上限。

  類型：整數。預設：10。允許的最大值：1024。
- **neptuneIamRoleArn** – (選用) IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。

  類型：字串 注意：這必須列示在您的資料庫叢集參數群組中，否則會擲回錯誤。

Neptune ML 管理 API 錯誤和例外狀況

所有 Neptune ML 管理 API 例外狀況都會傳回 400 HTTP 代碼。在收到任何這些例外狀況之後，不應重試產生例外狀況的命令。

- **MissingParameterException** – 錯誤訊息：

  ```
  Required credentials are missing. Please add IAM role to the cluster or pass as a parameter to this request.
  ```

- **InvalidParameterException** – 錯誤訊息：
  - `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** – 錯誤訊息：
  - `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.`

Neptune ML 限制

- 目前支援的推論類型為節點分類、節點迴歸、邊緣分類、邊緣迴歸和連結預測 (請參閱 Neptune ML 功能)。
- Neptune ML 可支援的圖形大小上限取決於資料準備、模型訓練和推論期間所需的記憶體和儲存體數量。
  - SageMaker 資料處理執行個體的記憶體大小上限為 768 GB。因此，資料處理階段若需要超過 768 GB 的記憶體，就會失敗。
  - SageMaker 訓練執行個體的記憶體大小上限為 732 GB。因此，訓練階段若需要超過 732 GB 的記憶體，就會失敗。
- SageMaker 端點的推論承載大小上限為 6 MiB。因此，如果子圖承載超過此大小，歸納推論就會失敗。
- 目前僅在 Neptune 及其所依賴的其他服務 (例如 AWS Lambda、Amazon API Gateway 和 Amazon SageMaker) 都受到支援的區域中，才能使用 Neptune ML。

  中國 (北京) 和中國 (寧夏) 中有關於預設使用 IAM 身分驗證的差異，如這裡所述以及其他差異。
- Neptune ML 所啟動的連結預測推論端點目前只能預測訓練期間存在於圖形中之節點的可能連結。

  例如，考慮具有 `User` 和 `Movie` 頂點以及 `Rated` 邊緣的圖形。使用對應的 Neptune ML 連結預測建議模型，您可以將新的使用者新增至圖形，並讓模型為他們預測電影，但模型只能建議模型訓練期間存在的電影。雖然 `User` 節點嵌入是使用其本機子圖和 GNN 模型即時計算的，因此可在使用者對電影進行評分時隨時間變化，但會將其與靜態、預先計算的電影嵌入進行比較，以取得最終建議。
- Neptune ML 支援的 KGE 模型僅適用於連結預測任務，而且表示法是在訓練期間存在於圖形中的頂點和邊緣類型特有的。這表示在訓練期間，推論查詢中提及的所有頂點和邊緣類型都必須存在於圖形中。若未重新訓練模型，則無法對新邊緣類型或頂點進行預測。

SageMaker 資源限制

根據一段時間後的活動和資源使用量，您可能會收到錯誤訊息，指出您已超過配額 (ResourceLimited)，因此您需要縱向擴展 SageMaker 資源，請遵循此頁面上請求增加 SageMaker 資源的服務配額程序中的步驟，向 AWS Support 請求增加配額。

SageMaker 資源名稱對應至 Neptune ML 階段，如下所示：

- Neptune 資料處理、模型訓練和模型轉換工作會使用 SageMaker `ProcessingJob`。
- Neptune 模型訓練工作會使用 SageMaker `HyperParameterTuningJob`。

- Neptune 模型訓練工作會使用 SageMaker `TrainingJob`。
- Neptune 推論端點會使用 SageMaker `Endpoint`。

監控 Amazon Neptune 資源

Amazon Neptune 支援各種監控資料庫效能和用量的方法。

- 執行個體狀態 – 檢查 Neptune 叢集圖形資料庫引擎的運作狀態、了解已安裝的引擎版本，以及使用執行個體狀態 API 取得其他執行個體相關的資訊。
- 圖形摘要 API – 圖形摘要 API 可讓您快速深入了解圖形資料大小和內容。

> **Note**
>
> 因為圖形摘要 API 依賴 DFE 統計資料，所以只有在啟用統計資料時才能使用，而 T3 和 T4G 執行個體類型並非如此。

- Amazon CloudWatch — Neptune 自動將指標發送到警報， CloudWatch 並支持 CloudWatch 警報。如需詳細資訊，請參閱 the section called “使用 CloudWatch”。
- 稽核日誌檔 – 使用 Neptune 主控台檢視、下載或監看資料庫日誌檔。如需詳細資訊，請參閱 the section called “稽核日誌搭配 Neptune”。
- 將日誌發佈到 Amazon CloudWatch 日誌 — 您可以設定 Neptune 資料庫叢集，將稽核日誌資料發佈到 Amazon CloudWatch 日誌中的日誌群組。使用 CloudWatch 日誌，您可以對日誌數據進行實時分析，用 CloudWatch 於創建警報和查看指標，以及使用 CloudWatch 日誌將日誌記錄存儲在高度耐用的存儲中。請參閱Neptune CloudWatch 日誌。
- AWS CloudTrail— Neptune 支持使用 API 日誌記錄 CloudTrail. 如需詳細資訊，請參閱 the section called “使用記錄 Neptune API 呼叫 AWS CloudTrail”。
- 事件通知訂閱 – 訂閱 Neptune 事件以隨時了解正在發生的事情。如需詳細資訊，請參閱 the section called “事件通知”。
- 標記 – 使用標籤將中繼資料新增到 Neptune 資源，並根據標籤追蹤使用情形。如需詳細資訊，請參閱 the section called “標記 Neptune 資源”。

主題

- 檢查 Neptune 執行個體的運作狀態
- 使用 Amazon 監控 Neptune CloudWatch
- 使用稽核日誌搭配 Amazon Neptune 叢集
- 將 Neptune 日誌發佈到 Amazon CloudWatch 日
- 為 Neptune 筆記本啟用 Amazon CloudWatch 日誌

- 使用 Amazon Neptune 慢查詢記錄
- 使用記錄 Amazon Neptune API 呼叫 AWS CloudTrail
- 使用 Neptune 事件通知
- 標記 Amazon Neptune 資源

檢查 Neptune 執行個體的運作狀態

Amazon Neptune 會提供一種機制，用來檢查主機上圖形資料庫的狀態。這也是確認您可以連線到執行個體的好方法。

使用 `curl` 檢查執行個體的運作狀態，並取得資料庫叢集狀態：

```
curl -G https://your-neptune-endpoint:port/status
```

或者，從引擎版本 1.2.1.0.R6 開始，您也可以改用下列 CLI 指令：

```
aws neptunedata get-engine-status
```

如果執行個體運作狀態良好，`status` 命令會一併傳回 JSON 物件及下列欄位：

- **`status`** – 如果執行個體沒有發生問題，則設定為 `"healthy"`。

  如果執行個體正在從當機復原，或是正在重新開機，而仍有從最近一次伺服器關機執行的作用中交易，則 `status` 會設定為 `"recovery"`。
- **`startTime`** – 設定為目前伺服器程序啟動的 UTC 時間。
- **`dbEngineVersion`** – 設定為資料庫叢集上執行的 Neptune 引擎版本。

  若此引擎版本自發行以來已手動進行修補，則版本號碼會加上 `"Patch-"` 前綴。
- **`role`** – 如果執行個體是僅供讀取複本，則設定為 `"reader"`，或者，如果執行個體是主要執行個體，則設定為 `"writer"`。
- **`dfeQueryEngine`** – 如果 DFE 引擎已完全啟用，則設定為 `"enabled"`，或者，如果 DFE 引擎僅與 `useDFE` 查詢提示設定為 `true` 的查詢搭配使用，則設定為 `viaQueryHint` (`viaQueryHint` 是預設值)。
- **`gremlin`** – 包含叢集上可用之 Grimlin 查詢語言的相關資訊。具體而言，它包含一個`version`字段，用於指定引擎正在使用的當前 TinkerPop 版本。

- **sparql** – 包含叢集上可用之 SPARQL 查詢語言的相關資訊。具體而言，它包含一個 `version` 欄位，指定引擎正在使用的目前 SPARQL 版本。
- **opencypher** – 包含叢集上可用之 openCypher 查詢語言的相關資訊。具體而言，它包含一個 `version` 欄位，指定引擎正在使用的目前 openCypher 版本。
- **labMode** – 包含引擎正在使用的 實驗室模式 設定。
- **rollingBackTrxCount** – 如果有要復原的交易，則此欄位會設定為此類交易的數目。如果沒有，則欄位根本不會出現。
- **rollingBackTrxEarliestStartTime** – 設定為正在復原之最早交易的開始時間。如果沒有任何交易正在復原，則欄位根本不會出現。
- **features** – 包含有關資料庫叢集上啟用之功能的狀態資訊：
  - **lookupCache** – 查詢快取 目前的狀態。此欄位只會出現在 R5d 執行個體類型上，因為這些是查詢快取可在其中存在的唯一執行個體。此欄位是 JSON 物件，格式如下：

    ```
    "lookupCache": {
      "status": "current lookup cache status"
    }
    ```

    在 R5d 執行個體上：
    - 如果已啟用查詢快取，則狀態會列示為 `"Available"`。
    - 如果已停用查詢快取，則狀態會列示為 `"Disabled"`。
    - 如果執行個體上已達到磁碟限制，則狀態會列示為 `"Read Only Mode - Storage Limit Reached"`。
  - **ResultCache** – 快取查詢結果 目前的狀態。此欄位是 JSON 物件，格式如下：

    ```
    "ResultCache": {
      "status": "current results cache status"
    }
    ```

    - 如果已啟用結果快取，則狀態會列示為 `"Available"`。
    - 如果已停用快取，則狀態會列示為 `"Disabled"`。
  - **IAMAuthentication**— 指定是否已在資料庫叢集上啟用 AWS Identity and Access Management (IAM) 身份驗證：
    - 如果已啟用 IAM 身分驗證，則狀態會列示為 `"enabled"`。
    - 如果已停用 IAM 身分驗證，則狀態會列示為 `"disabled"`。

  - **Streams** – 指定是否已在資料庫叢集上啟用 Neptune 串流：
    - 如果已啟用串流，則狀態會列示為 "enabled"。
    - 如果已停用串流，則狀態會列示為 "disabled"。
  - **AuditLog** – 如果已啟用稽核日誌，等於 enabled，否則等於 disabled。
  - **SlowQueryLogs** – 如果已啟用慢查詢記錄，等於 info 或 debug，否則等於 disabled。
  - **QueryTimeout** – 查詢逾時的值 (以毫秒為單位)。
- **settings** – 套用至執行個體的設定：
  - **clusterQueryTimeoutInMs** – 針對整個叢集設定的查詢逾時值 (以毫秒為單位)。
  - **SlowQueryLogsThreshold** – 針對整個叢集設定的查詢逾時值 (以毫秒為單位)。
- **serverlessConfiguration** – 叢集的無伺服器設定 (如果叢集以無伺服器形式執行)：
  - **minCapacity** – 資料庫叢集中無伺服器執行個體可以縮小至的最小大小，以 Neptune 容量單位 (NCU) 表示。
  - **maxCapacity** – 資料庫叢集中無伺服器執行個體可以增長至的最大大小，以 Neptune 容量單位 (NCU) 表示。

來自執行個體狀態命令的輸出範例

以下是來自執行個體狀態命令的輸出範例 (在此情況下，於 R5d 執行個體上執行)：

```
{
  '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'
  }
}
```

若執行個體發生問題，status 命令會傳回 HTTP 500 錯誤碼。如果主機無法連線，請求將逾時。確認您是從虛擬私有雲端 (VPC) 內部存取執行個體，並且您的安全群組允許您存取它。

使用 Amazon 監控 Neptune CloudWatch

整合了 Amazon Neptune 和 Amazon CloudWatch ，因此您可以收集和分析效能指標。您可以使用 CloudWatch 主控台、 AWS Command Line Interface (AWS CLI) 或 CloudWatch API 監視這些指標。

CloudWatch 也可讓您設定警示，以便在指標值違反您指定的閾值時收到通知。您甚至可以設置 CloudWatch 事件以在發生違規時採取糾正措施。如需使用 CloudWatch 和警示的詳細資訊，請參閱CloudWatch 文件。

主題

- 檢視 CloudWatch 資料 (主控台)
- 檢視 CloudWatch 資料 (AWS CLI)
- 檢視 CloudWatch 資料 (API)
- 用 CloudWatch 來監視 Neptune 中的資料庫執行個體效
- Neptune CloudWatch 度量
- Neptune CloudWatch 尺寸

檢視 CloudWatch 資料 (主控台)

若要檢視 Neptune 叢集 (主控台) 的 CloudWatch 資料

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。

2. 在導覽窗格中，選擇 指標。
3. 在「所有測量結果」窗格中，選擇 Neptune，然後選擇「資料庫」ClusterIdentifier。
4. 在上方窗格中，向下捲動以檢視叢集的完整指標清單。可用的 Neptune 指標選項會出現在檢視清單中。

若要選取或取消選取個別指標，請在結果窗格中，選擇資源名稱和指標旁的核取方塊。圖表顯示所選取項目的指標顯示於主控台底部。若要進一步了解 CloudWatch 圖形，請參閱 Amazon CloudWatch 使用者指南中的圖形指標。

檢視 CloudWatch 資料 (AWS CLI)

若要檢視 Neptune 叢集的 CloudWatch 資料 (AWS CLI)

1. 安裝 AWS CLI. 如需指示，請參閱 AWS Command Line Interface 使用者指南。
2. 使用 AWS CLI 來擷取資訊。中列出了 Neptune 的相關 CloudWatch 參數Neptune CloudWatch 度量。

   下列範例會 CloudWatch 擷取叢集每秒 Grinlin 要求數目的度量。`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
```

檢視 CloudWatch 資料 (API)

CloudWatch 也支援`Query`動作，以便您可以透過程式設計方式要求資訊。如需詳細資訊，請參閱CloudWatch 查詢 API 文件和 Amazon CloudWatch API 參考資料。

當 CloudWatch 動作需要 Neptune 監視專用的參數時 (例如)`MetricName`，請使用中列出的值Neptune CloudWatch 度量。

下列範例會示範使用下列參數的低階 CloudWatch 要求：

- `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

這是 CloudWatch 請求的樣子。不過，這只是顯示請求的形式；您必須根據您的指標與時間範圍建構自己的請求。

```
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
```

用 CloudWatch 來監視 Neptune 中的資料庫執行個體效

您可以使用 Neptune 中的 CloudWatch 指標來監視資料庫執行個體上發生的情況，並追蹤資料庫觀察到的查詢佇列長度。以下指標特別有用：

- **CPUUtilization** – 顯示 CPU 使用率。
- **VolumeWriteIOPs** – 顯示叢集磁碟區的磁碟 I/O 寫入平均數目，每隔 5 分鐘回報一次。
- **MainRequestQueuePendingRequests** – 顯示輸入佇列中等待執行的請求數目。

您也可以搭配 `includeWaiting` 參數使用 Gremlin 查詢狀態端點，了解伺服器上有多少請求待定。這將為您提供所有等待中查詢的狀態。

下列指標可協助您調整 Neptune 佈建和查詢策略，以改善效率和效能：

- 一致的延遲、高 CPUUtilization、高 VolumeWriteIOPs 和低 MainRequestQueuePendingRequests 合起來說明伺服器正在積極參與以可持續的速率處理並行寫入請求，I/O 等待很少發生。
- 一致的延遲、低 CPUUtilization、低 VolumeWriteIOPs 及沒有 MainRequestQueuePendingRequests 合起來說明您在主要資料庫執行個體上有過多的容量用於處理寫入請求。
- 高 CPUUtilization 和 高 VolumeWriteIOPs，但可變的延遲和 MainRequestQueuePendingRequests 合起來說明您正在傳送的工作量超過伺服器可在給定間隔內處理的工作量。考慮建立批次請求或調整其大小，以便能以較少的交易負荷執行相同的工作量，和/或縱向擴展主要執行個體，以增加能夠同時處理寫入要求的查詢執行緒數目。
- 低 CPUUtilization 與高 VolumeWriteIOPs 表示查詢執行緒正在等待儲存層的 I/O 操作完成。如果您看到可變的延遲，而且 MainRequestQueuePendingRequests 中有些增加，請考慮建立批次請求或調整其大小，以便能以較少的交易負荷來執行相同的工作量。

Neptune CloudWatch 度量

> **Note**
>
> Amazon Neptune 只會在指標具有非零值時， CloudWatch 才會傳送指標給這些指標。對於所有 Neptune 指標，彙總粒度為 5 分鐘。

主題

Neptune CloudWatch 度量

下表列出 Neptune 支援的 CloudWatch 測量結果。

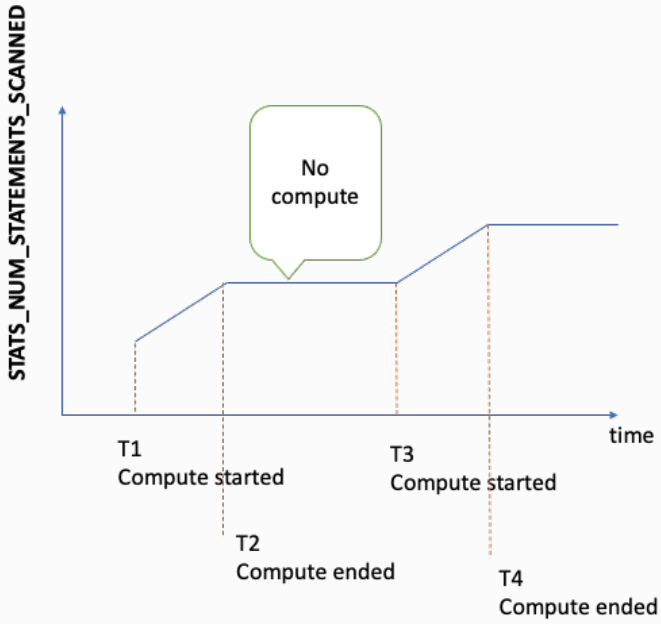
> **Note**
>
> 每當伺服器重新啟動時，所有累積指標都會重設為零，無論是進行維護、重新開機或從當機復原都會如此。

Neptune CloudWatch 度量

指標	描述
BackupRetentionPeriodStorageUsed	用來支援 Neptune 資料庫叢集備份保留時段的備份儲存總量 (以位元組為單位)。包含在 TotalBackupStorageBilled 指標報告的總數中。
BufferCacheHitRatio	由緩衝區快取提供服務的請求的百分比。此指標在診斷查詢延遲時很有用，因為快取遺漏會造成顯著延遲。如果快取命中率低於 99.9，請考慮升級執行個體類型，以在記憶體中快取更多資料。
ClusterReplicaLag	如果是僅供讀取複本，則為從主要執行個體複寫更新時的延遲量，以毫秒單位。
ClusterReplicaLagMaximum	主要執行個體與資料庫叢集中各個 Neptune 資料庫執行個體之間的最大延遲量，以毫秒單位。
ClusterReplicaLagMinimum	主要執行個體與資料庫叢集中各個 Neptune 資料庫執行個體之間的最小延遲量，以毫秒單位。
CPUUtilization	CPU 使用率。
EngineUptime	執行個體已執行的時間量，以秒為單位。
FreeableMemory	可用的隨機存取記憶體的數量，以位元組為單位。
GlobalDbDataTransferBytes	Neptune 全域資料庫中，從主要資料庫傳輸 AWS 區域 到次 AWS 區域 要重做日誌資料的位元組數目。
GlobalDbReplicatedWriteIO	從全球資料庫中主要 AWS 區域 複寫到次要 AWS 區域中叢集磁碟區的寫入 I/O 操作次數。 Neptune 全球資料庫中每個資料庫叢集的計費計算會使用 VolumeWriteIOPS ，來說明該叢集

指標	描述
	內執行的寫入。對於主要資料庫叢集，計費計算會使用 GlobalDbReplicatedWriteIO 來說明對次要資料庫叢集的跨區域複寫。
GlobalDbProgressLag	對於使用者交易和系統交易，次要叢集在主要叢集後面的毫秒數。
GremlinRequestsPerSec	每秒對 Gremlin 引擎的請求數量。
GremlinWebSocketOpenConnections	開啟至 Neptune 的 WebSocket 連線數目。
LoaderRequestsPerSec	每秒載入器請求的數量。
MainRequestQueuePendingRequests	輸入佇列中等待執行的請求數目。Neptune 會在請求超過最大佇列容量時開始限流這些請求。
NCUUtilization	僅適用於 Neptune Serverless 資料庫執行個體或資料庫叢集。在執行個體層級，報告百分比，其計算方式是有問題執行個體目前正在使用的 Neptune 容量單位 (NCU) 數目，除以叢集的最大 NCU 容量設定。NCU 或 Neptune 容量單位 (NCU) 包含 2 GiB 記憶體 (RAM)，以及相關聯的虛擬處理器容量 (vCPU) 和網路。 在叢集層級，NCUUtilization 會報告叢集整體使用的最大容量百分比。
NetworkThroughput	Neptune 資料庫叢集中的各個執行個體從用戶端接收及傳輸至用戶端的網路輸送量，以位元組/秒為單位。此輸送量不包含資料庫叢集中的執行個體與叢集磁碟區之間的網路流量。
NetworkTransmitThroughput	Neptune 資料庫叢集中的各個執行個體傳輸至用戶端的傳出網路輸送量，以位元組/秒為單位。此輸送量不包含資料庫叢集中的執行個體與叢集磁碟區之間的網路流量。

指標	描述
NumTxCommitted	每秒成功遞交的交易數量。
NumTxOpened	每秒在伺服器上開啟的交易數量。
NumTxRolledBack	對於寫入查詢，伺服器上由於錯誤而每秒復原的交易數目。對於唯讀查詢，此指標等於每秒完成的唯讀交易數目。
OpenCypherRequestsPerSec	每秒 (HTTPS 和 Bolt) 對 openCypher 引擎的請求數目。
OpenCypherBoltOpenConnections	開放式 Bolt 與 Neptune 的連線數目。
ServerlessDatabaseCapacity	做為執行個體層級指標，ServerlessDatabaseCapacity 會報告給定 Neptune Serverless 執行個體的目前執行個體容量，以 NCU 表示。NCU 或 Neptune 容量單位 (NCU) 包含 2 GiB 記憶體 (RAM)，以及相關聯的虛擬處理器容量 (vCPU) 和網路。 在叢集層級，ServerlessDatabaseCapacity 會報告叢集中資料庫執行個體的所有 ServerlessDatabaseCapacity 值的平均值。
SnapshotStorageUsed	Neptune 資料庫叢集在其備份保留期間外，由所有快照取用的備份儲存總量 (以位元組為單位)。包含在 TotalBackupStorageBilled 指標報告的總數中。
SparqlRequestsPerSec	每秒對 SPARQL 引擎的請求數量。

指標	描述
StatsNumStatementsScanned	自伺服器啟動以來針對 DEF 統計資料掃描的陳述式總數。 每次觸發統計資料計算時，這個數字都會增加，但當沒有計算發生時，它會保持靜態。因此，如果您隨著時間對其繪製圖形，則可以知道計算何時發生以及何時不發生：  透過查看指標增加期間的圖形斜率，您還可以知道計算的速度。 如果沒有此類指標，則表示已在您的資料庫叢集上停用統計資料功能，或者您正在執行的引擎版本沒有統計資料功能。如果指標值為零，表示沒有發生統計資料計算。
TotalBackupStorageBilled	指定 Neptune 資料庫叢集要計費的備份儲存總量 (以位元組為單位)。包括以 BackupRetentionPeriodStorageUsed 和 SnapshotStorageUsed 指標衡量的備份儲存。

指標	描述
TotalRequestsPerSec	每秒來自所有來源對伺服器的請求總數。
TotalClientErrorsPerSec	每秒由於用戶端問題而傳回錯誤的請求總數。
TotalServerErrorsPerSec	伺服器上每秒由於內部失敗而傳回錯誤的請求總數。
UndoLogListSize	復原日誌清單中復原日誌的計數。 復原日誌包含已遞交交易的記錄，這些記錄會在所有作用中交易比遞交時間更新時到期。過期的記錄會定期清除。刪除操作的記錄在清除時需要的時間可能比其他交易類型的記錄更長。 清除僅能由資料庫叢集的寫入器執行個體完成，因此清除率取決於寫入器執行個體類型。如果 UndoLogListSize 很高，且在資料庫叢集中不斷增長，請升級寫入器執行個體以提高清除率。 此外，如果您要從早於 1.2.0.0 的版本升級到引擎版本 1.2.0.0 或更新版本，請首先確定 UndoLogListSize 值接近 0。因為引擎版本 1.2.0.0 及更新版本會針對復原日誌使用不同的格式，所以只有在完全清除先前的復原日誌之後才能開始升級。如需詳細資訊，請參閱升級至 1.2.0.0 或更新版本。
VolumeBytesUsed	配置給 Neptune 資料庫叢集的儲存體總量 (位元組)。這是您收取費用的儲存空間量。它是分配給資料庫叢集群的最大儲存量 (在叢集存在時)，而不是您當前使用的數量 (請參閱 Neptune 儲存計費)。

指標	描述
VolumeReadIOPs	從叢集磁碟區計費的讀取 I/O 作業總數，每隔 5 分鐘回報一次。計費的讀取操作以叢集磁碟區層級計算，彙整來自 Neptune 資料庫叢集中的所有執行個體，每隔 5 分鐘回報一次。
VolumeWriteIOPs	叢集磁碟區的寫入磁碟 I/O 作業總數，每隔 5 分鐘報告一次。

CloudWatch Neptune 目前已棄用的量度

現在已棄用這些 Neptune 指標。它們仍然受到支援，但在未來可能會被淘汰，因為有新的和更好的指標可用。

指標	描述
GremlinHttp1xx	每秒 Gremlin 端點的 HTTP 1xx 回應數量。 建議您改為使用新的 Http1xx 組合指標。
GremlinHttp2xx	每秒 Gremlin 端點的 HTTP 2xx 回應數量。 建議您改為使用新的 Http2xx 組合指標。
GremlinHttp4xx	每秒 Gremlin 端點的 HTTP 4xx 錯誤數量。 建議您改為使用新的 Http4xx 組合指標。
GremlinHttp5xx	每秒 Gremlin 端點的 HTTP 5xx 錯誤數量。 建議您改為使用新的 Http5xx 組合指標。
GremlinErrors	在 Gremlin 周遊中的錯誤數量。
GremlinRequests	對 Gremlin 引擎的請求數量。
GremlinWebSocketSuccess	每秒成功 WebSocket 連線到 Gemlin 端點的數目。

指標	描述
GremlinWebSocketClientErrors	每秒 Gemlin 端點上的 WebSocket 用戶端錯誤數目。
GremlinWebSocketServerErrors	每秒 Gemlin 端點上的 WebSocket 伺服器錯誤數目。
GremlinWebSocketAvailableConnections	目前可用的潛在 WebSocket 連線數目。
Http100	每秒端點的 HTTP 100 回應數量。 建議您改為使用新的 Http1xx 組合指標。
Http101	每秒端點的 HTTP 101 回應數量。 建議您改為使用新的 Http1xx 組合指標。
Http1xx	每秒端點的 HTTP 1xx 回應數量。
Http200	每秒端點的 HTTP 200 回應數量。 建議您改為使用新的 Http2xx 組合指標。
Http2xx	每秒端點的 HTTP 2xx 回應數量。
Http400	每秒端點的 HTTP 400 錯誤數量。 建議您改為使用新的 Http4xx 組合指標。
Http403	每秒端點的 HTTP 403 錯誤數量。 建議您改為使用新的 Http4xx 組合指標。
Http405	每秒端點的 HTTP 405 錯誤數量。 建議您改為使用新的 Http4xx 組合指標。
Http413	每秒端點的 HTTP 413 錯誤數量。 建議您改為使用新的 Http4xx 組合指標。

指標	描述
Http429	每秒端點的 HTTP 429 錯誤數量。 建議您改為使用新的 Http4xx 組合指標。
Http4xx	每秒端點的 HTTP 4xx 錯誤數量。
Http500	每秒端點的 HTTP 500 錯誤數量。 建議您改為使用新的 Http5xx 組合指標。
Http501	每秒端點的 HTTP 501 錯誤數量。 建議您改為使用新的 Http5xx 組合指標。
Http5xx	每秒端點的 HTTP 5xx 錯誤數量。
LoaderErrors	來自載入器請求的錯誤數量。
LoaderRequests	載入器請求的數量。
SparqlHttp1xx	每秒 SPARQL 端點的 HTTP 1xx 回應數量。 建議您改為使用新的 Http1xx 組合指標。
SparqlHttp2xx	每秒 SPARQL 端點的 HTTP 2xx 回應數量。 建議您改為使用新的 Http2xx 組合指標。
SparqlHttp4xx	每秒 SPARQL 端點的 HTTP 4xx 錯誤數量。 建議您改為使用新的 Http4xx 組合指標。
SparqlHttp5xx	每秒 SPARQL 端點的 HTTP 5xx 錯誤數量。 建議您改為使用新的 Http5xx 組合指標。
SparqlErrors	SPARQL 查詢中的錯誤數量。
SparqlRequests	對 SPARQL 引擎的請求數量。

指標	描述
StatusErrors	來自狀態端點的錯誤數量。
StatusRequests	向狀態端點請求的數量。

Neptune CloudWatch 尺寸

Amazon Neptune 的指標是由帳戶的值、圖形名稱或操作進行限定。您可以使用 Amazon CloudWatch 主控台擷取 Neptune 資料以及下表中的任何維度。

維度	描述
DBInstanceIdentifier	篩選您為叢集中的特定資料庫執行個體請求的資料。
DBClusterIdentifier	篩選您對特定 Neptune 資料庫叢集所請求的資料。
DBClusterIdentifier ，EngineName	依照叢集篩選資料。所有 Neptune 執行個體的引擎名稱為 neptune。
DBClusterIdentifier ，Role	篩選您為特定 Neptune 資料庫叢集請求的資料，依據執行個體角色 (WRITER/READER) 彙總指標。例如，您可以為屬於某個叢集的所有讀取者執行個體彙總指標。
DBClusterIdentifier ，SourceRegion	依全球資料庫主要區域中的主要叢集篩選資料。
DatabaseClass	篩選您為某個資料庫類別中的所有執行個體請求的資料。例如，您可以為屬於資料庫類別 db.r4.large 的所有執行個體彙總指標
EngineName	所有 Neptune 執行個體的引擎名稱為 neptune。
GlobalDbDBClusterIdentifier ，SecondaryRegion	依次要區域中所指定全球資料庫的次要叢集篩選資料。

使用稽核日誌搭配 Amazon Neptune 叢集

若要稽核 Amazon Neptune 資料庫叢集活動，請設定資料庫叢集參數來啟用稽核日誌集合。當稽核日誌啟用時，可用來記錄任意組合的支援事件。您可以檢視或下載稽核日誌來進行檢閱。

啟用 Neptune 稽核日誌

使用 `neptune_enable_audit_log` 參數以啟用 (`1`) 或停用 (`0`) 稽核日誌。

在資料庫叢集所用的參數群組中設定此參數。您可以使用中顯示的程序編輯資料庫叢集參數群組或資料庫參數群組來修改參數 AWS Management Console，或使用修改 -db-叢集-參數群組指令或 ModifyDB Group API AWS CLI 指令，以程式設計方式修改ClusterParameter參數。

在修改此參數之後，您必須重新啟動資料庫執行個體，才能套用變更。

使用主控台檢視 Neptune 稽核日誌

您可以使用 AWS Management Console來檢視和下載稽核日誌。在 Instances (執行個體) 頁面上，選擇資料庫執行個體以顯示其詳細資訊，然後捲動至 Logs (日誌) 區段。

若要下載日誌檔案，請在 Logs (日誌) 區段中選取該檔案，然後選擇 Download (下載)。

Neptune 稽核日誌詳細資訊

日誌檔案為 UTF-8 格式。日誌會寫入多個檔案中，數量隨著執行個體大小而不同。若要查看最新事件，您可能必須檢閱所有稽核日誌檔案。

日誌項目不會循序排列。您可以使用 `timestamp` 值來排序它們。

日誌檔案合計達到 100 MB 時會輪換。無法設定此限制。

稽核日誌檔案的資料列依指定順序包含以下逗號分隔的資訊：

欄位	描述
時間戳記	所記錄事件的 Unix 時間戳記 (精確度達到微秒)。
ClientHost	連線使用者來自的主機名稱或 IP。
ServerHost	記錄事件所針對之執行個體的主機名稱或 IP。

欄位	描述
ConnectionType	連線類型。可以是 `Websocket` 、`HTTP_POST` 、`HTTP_GET` 或 `Bolt`。

呼叫者的 IAM ARN	用來簽署請求之 IAM 使用者或 IAM 角色的 ARM。如果停用 IAM 身分驗證會空白。格式為：
	`arn:partition :service:region:account:resource`
	例如：
	`arn:aws:iam::123456789012:user/Anna`
	`arn:aws:sts::123456789012:assumed-role/AWSNeptuneNotebookRole/SageMaker`

Auth Context	包含具備身分驗證資訊的序列化 JSON 物件。如果使用者已經驗證，則 `authenticationSucceeded` 欄位為 `True`。 如果停用 IAM 身分驗證會空白。
HttpHeader	HTTP 標頭資訊。可包含一個查詢。空白表示 WebSocket和螺栓連接。
承載	Gremlin、SPARQL 或 openCypher 查詢。

將 Neptune 日誌發佈到 Amazon CloudWatch 日

您可以設定 Neptune DB 叢集，將稽核記錄資料和/或緩慢查詢日誌資料發佈到 Amazon CloudWatch Logs 中的日誌群組。使用 CloudWatch Logs，您可以對記錄資料執行即時分析，並用 CloudWatch 來建立警示和檢視指標。您可以使用 CloudWatch 日誌將日誌記錄存儲在高度耐用的存儲中。

若要將稽核記錄發佈到 CloudWatch 記錄，必須明確啟用稽核記錄 (請參閱啟用稽核日誌)。同樣地，若要將慢速查詢記錄檔發佈至記 CloudWatch 錄檔，必須明確啟用慢速查詢記錄檔 (請參閱)。使用 Amazon Neptune 慢查詢記錄

> **Note**
>
> 請注意以下事項：

- 將記錄發佈至時需支付額外費用 CloudWatch。如需詳細資訊，請參閱CloudWatch 定價頁面。
- 您無法將記錄發佈至中國 (北京) 或中國 (寧夏) 區域的 CloudWatch 記錄檔。
- 如果匯出日誌資料已停用，Neptune 不會刪除現有的日誌群組或日誌串流。如果停用匯出記錄資料， CloudWatch 記錄檔中會保留現有的記錄資料 (視記錄保留而定)，而且您仍會對儲存的稽核記錄資料產生費用。您可以使用 CloudWatch 記錄主控台、或記錄 API 刪除記錄串流和記 CloudWatch 錄群組。 AWS CLI

使用主控台將 Neptune 記錄發佈至記 CloudWatch 錄檔

從主控台將 Neptune 記 CloudWatch 錄檔發佈至記錄檔

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 選擇您要發佈日誌資料的 Neptune 資料庫叢集。
4. 在 Actions (動作) 中，選擇 Modify (修改)。
5. 在 [記錄匯出] 區段中，選擇您要開始發佈至 CloudWatch 記錄的記錄檔。
6. 選擇 Continue (繼續)，然後在摘要頁面上選擇 Modify DB Cluster (修改資料庫叢集)。

使用 CLI 將 Neptune 稽核記錄發佈至 CloudWatch 記錄

您可以使用具有下列參數的 AWS CLI `create-db-cluster`命令來建立新的資料庫叢集，將稽核 CloudWatch 記錄發佈至記錄檔：

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["audit"]'
```

您可以使用具有下列參數的 AWS CLI `modify-db-cluster`命令，將現有的資料庫叢集設定為將稽核 CloudWatch 記錄發佈至記錄：

```
aws neptune modify-db-cluster \
```

```
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

使用 CLI 將 Neptune 慢速查詢記錄檔發佈至記錄檔 CloudWatch

您也可以使用具有下列參數的 AWS CLI `create-db-cluster`命令，建立新的資料庫叢集，將慢速查詢 CloudWatch 記錄檔發佈至記錄檔：

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["slowquery"]'
```

同樣地，您可以使用具有下列參數的 AWS CLI `modify-db-cluster`命令，將現有的資料庫叢集設定為將慢速查詢 CloudWatch 記錄檔發佈至記錄檔：

```
aws neptune modify-db-cluster --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["slowquery"]}'
```

在 Amazon 監控 Neptune 日誌事件 CloudWatch

啟用 Neptune 日誌後，您可以在 Amazon CloudWatch 日誌中監控日誌事件。系統會自動在下列字首下方為 Neptune 資料庫叢集建立新的日誌群組，其中 *`cluster-name`* 代表資料庫叢集名稱，而 *`log_type`* 代表日誌類型。

```
/aws/neptune/cluster-name/log_type
```

例如，若您設定匯出功能來包含 `mydbcluster` 資料庫叢集的稽核日誌，則日誌資料會存放在 `/aws/neptune/mydbcluster/audit` 日誌群組中。

來自資料庫叢集的所有資料庫執行個體的所有事件，將會透過不同的日誌串流推送至日誌群組。

如果存在指定名稱的日誌群組，Neptune 會使用該日誌群組來匯出 Neptune 資料庫叢集的日誌資料。您可以使用自動化設定，例如 AWS CloudFormation，建立具有預先定義的記錄保留期間、指標篩選器和客戶存取權限的記錄群組。否則，系統會使用 [記錄檔] 中的預設記錄保留期間 [永不過期] 自動建立新的 CloudWatch 記錄群組。

您可以使用 CloudWatch 記錄主控台 AWS CLI、或 CloudWatch 記錄 API 來變更記錄保留期間。如需變更記錄檔中記錄保留期間的詳細資訊，請參閱變更 CloudWatch 記錄檔中的 CloudWatch 記錄檔資料保留。

您可以使用 CloudWatch 記錄主控台 AWS CLI、或 CloudWatch 記錄 API 來搜尋資料庫叢集的記錄事件中的資訊。如需搜尋和篩選日誌資料的詳細資訊，請參閱搜尋和篩選日誌資料。

為 Neptune 筆記本啟用 Amazon CloudWatch 日誌

CloudWatch Neptune 筆記本的記錄預設為停用。請遵循下列步驟啟用它們，用於偵錯或其他目的：

使用啟 AWS Management Console 用 Neptune 筆記本的記 CloudWatch 錄檔

1. 在以下位置打開 Amazon SageMaker 控制台 https://console.aws.amazon.com/sagemaker/。
2. 在左側的導覽窗格中，選擇筆記本，然後選擇筆記本執行個體。尋找您要為其啟用日誌的 Neptune 筆記本名稱。
3. 選擇上述步驟中提到的筆記本執行個體名稱，前往詳細資訊頁面。
4. 如果筆記本執行個體正在執行，請選取筆記本詳細資料頁面右上方的停止按鈕。
5. 在許可和加密下，IAM 角色 ARN 有一個欄位。選取此欄位中的連結，即可前往此筆記本的 IAM 角色。
6. 建立下列政策：

```
{
  "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. 儲存此新政策並在步驟 4 中將其附加至 IAM 角色。
8. 選取 SageMaker筆記本執行個體詳細資訊頁面右上角的 [啟動]。
9. 一旦日誌開始流動，您就應該會在詳細資料頁面的筆記本執行個體設定區段左下方附近，於標示為生命週期組態的欄位下方看到檢視日誌連結。

如果您的筆記型電腦無法啟動， SageMaker 主控台的筆記型電腦詳細資料頁面會顯示訊息，指出筆記型電腦執行個體啟動時間超過 5 分鐘。您可以在以下名稱下找到與此問題相關的 CloudWatch 記錄：*(your-notebook-name)*/LifecycleConfigOnStart。

如有必要，請參閱使用 Amazon 記錄 Amazon SageMaker 事件以 CloudWatch獲取更多詳細資訊

使用 Amazon Neptune 慢查詢記錄

識別、偵錯和最佳化執行緩慢的查詢可能很困難。當 Neptune 的慢查詢記錄啟用時，系統會自動記錄所有長時間執行之查詢的屬性，讓這個程序更容易進行。

> **Note**
>
> 慢查詢記錄是在 Neptune 引擎 1.2.1.0 版中引進的。

您可以使用 neptune_enable_slow_query_log 資料庫叢集參數，啟用慢查詢記錄。根據預設，此參數會設定為 `disabled`。將其設定為 `info` 或 `debug` 可啟用慢查詢記錄。`info` 設定會記錄每個緩慢執行之查詢的一些有用屬性，而 `debug` 設定則會記錄所有可用的屬性。

若要設定被視為執行緩慢之查詢的閾值，請使用 neptune_slow_query_log_threshold 資料庫叢集參數，指定多少毫秒後，執行中查詢會被視為緩慢，並在慢查詢記錄啟用時加以記錄。預設值為 5000 毫秒 (5 秒)。

您可以在 AWS Management Console，或使用修改 -db-叢集參數群組命令或 ModifyDB 群組管理功能，設定這些資料庫叢AWS CLI 集參數。ClusterParameter

> **Note**
>
> 慢查詢記錄參數是動態的，這表示變更其值不需要也不會導致重新啟動資料庫叢集。

若要檢視中的慢速查詢記錄檔 AWS Management Console

您可以在中檢視和下載慢速查詢記錄檔 AWS Management Console，如下所示：

在執行個體頁面上，選擇資料庫執行個體，然後捲動至日誌區段。您可以接著在那裡選擇一個日誌檔，然後選擇下載以下載它。

由 Neptune 慢查詢日誌產生的檔案

Neptune 中由慢查詢記錄所產生的日誌檔具有下列特性：

- 檔案會編碼為 UTF-8。
- 查詢及其屬性是以 JSON 格式記錄。
- 除了 `queryTime` 資料外，不會記錄 Null 和空白屬性。
- 日誌檔會跨越多個檔案，其數目隨著執行個體大小而不同。
- 日誌項目不會循序排列。您可以使用其 `timestamp` 值來排序它們。
- 若要查看最新事件，您可能必須檢閱所有慢查詢日誌檔。
- 日誌檔合計達到 100 MB 時便會輪換。無法設定此限制。

在 **`info`** 模式中記錄的查詢屬性

當 `neptune_enable_slow_query_log` 資料庫叢集參數已設定為 `info` 時，系統會記錄慢查詢的下列屬性：

群組	屬性	描述
請求 ResponseMetadata	`requestId`	查詢的請求 ID。
	`requestType`	請求類型，如 HTTP 或 WebSocket.
	`responseStatusCode`	查詢回應狀態碼，如 200。

群組	屬性	描述
	exceptionClass	查詢執行後所傳回錯誤的例外狀況類別。
queryStats	query	查詢字串。
	queryFingerprint	查詢的指紋。
	queryLanguage	查詢語言，如 Gremlin、SPARQL 或 openCypher。
memoryStats	allocatedPermits	分配給查詢的許可。
	approximateUsedMemoryBytes	執行期間查詢所使用的大約記憶體。
queryTime	startTime	查詢開始時間 (UTC)。
	overallRunTimeMs	查詢總執行時間，以毫秒為單位。
	parsingTimeMs	查詢剖析時間，以毫秒為單位。
	waitingTimeMs	查詢 Gremlin/SPARQL/openCypher 佇列等待時間，以毫秒為單位
	executionTimeMs	查詢執行時間，以毫秒為單位。
	serializationTimeMs	查詢序列化時間，以毫秒為單位。
statementCounters	scanned	已掃描的陳述式數目。
	written	已寫入的陳述式數目。
	deleted	已刪除的陳述式數目。

群組	屬性	描述
transactionCounters	committed	已完成的交易數目。
	rolledBack	已復原的交易數目。
vertexCounters	added	已新增的頂點數目。
	removed	已移除的頂點數目。
	propertiesAdded	已新增的頂點屬性數目。
	propertiesRemoved	已移除的頂點屬性數目。
edgeCounters	added	已新增的邊緣數目。
	removed	已移除的邊緣數目。
	propertiesAdded	已新增的邊緣屬性數目。
	propertiesRemoved	已移除的邊緣屬性數目。
resultCache	hitCount	結果快取命中計數。
	missCount	結果快取遺漏計數。
	putCount	結果快取放置計數。
concurrentExecution	acceptedQueryCountAtStart	開始時目前查詢執行接受的平行查詢。
	runningQueryCountAtStart	開始時與目前查詢執行一起執行的平行查詢。
	acceptedQueryCountAtEnd	結束時目前查詢執行接受的平行查詢。
	runningQueryCountAtEnd	結束時與目前查詢執行一起執行的平行查詢。

群組	屬性	描述
queryBatch	queryProcessingBatchSize	查詢處理期間的批次大小。
	querySerialisationBatchSize	查詢序列化期間的批次大小。

在 **debug** 模式中記錄的查詢屬性

當 neptune_enable_slow_query_log 資料庫叢集參數已設定為 debug 時，除了在 info 模式中記錄的屬性之外，還會記錄下列儲存體計數器屬性：

屬性	描述
statementsScannedInAllIndexes	在所有索引中掃描的陳述式。
statementsScannedSPOGIndex	在 SPOG 索引中掃描的陳述式。
statementsScannedPOGSIndex	在 POGS 索引中掃描的陳述式。
statementsScannedGPSOIndex	在 GPSO 索引中掃描的陳述式。
statementsScannedOSGPIndex	在 OSGP 索引中掃描的陳述式。
statementsScannedInChunk	在區塊中一起掃描的陳述式。
postFilteredStatementScans	掃描後篩選後留下的陳述式。
distinctStatementScans	已掃描的不同陳述式。
statementsReadInAllIndexes	在所有索引中掃描後置篩選之後讀取的陳述式。
statementsReadSPOGIndex	在 SPOG 索引中掃描後置篩選之後讀取的陳述式。
statementsReadPOGSIndex	在 POGS 索引中掃描後置篩選之後讀取的陳述式。

屬性	描述
statementsReadGPSOIndex	在 GPSO 索引中掃描後置篩選之後讀取的陳述式。
statementsReadOSGPIndex	在 OSGP 索引中掃描後置篩選之後讀取的陳述式。
accessPathSearches	存取路徑搜尋次數。
fullyBoundedAccessPathSearches	完全界限金鑰存取路徑搜尋次數。
accessPathSearchedByPrefix	按字首搜尋的存取路徑數目。
searchesWhereRecordsWereFound	有 1 筆或多筆記錄做為輸出的搜尋次數。
searchesWhereRecordsWereNotFound	沒有記錄做為輸出的搜尋次數。
totalRecordsFoundInSearches	從所有搜尋中找到的記錄總數。
statementsInsertedInAllIndexes	在所有索引中插入的陳述式數目。
statementsUpdatedInAllIndexes	在所有索引中更新的陳述式數目。
statementsDeletedInAllIndexes	在所有索引中刪除的陳述式數目。
predicateCount	述詞數目。
dictionaryReadsFromValueToIdTable	字典從值讀取到 ID 資料表的次數。
dictionaryReadsFromIdToValueTable	字典 ID 讀取到值資料表的次數。
dictionaryWritesToValueToIdTable	字典寫入到值再到 ID 資料表的次數。
dictionaryWritesToIdToValueTable	字典寫入到 ID 再到值資料表的次數。
rangeCountsInAllIndexes	所有索引中範圍計數的數目。
deadlockCount	查詢中的死結數目。

屬性	描述
singleCardinalityInserts	已執行的單一基數插入數目。
singleCardinalityInsertDeletions	單一基數插入期間刪除的陳述式數目。

慢查詢的偵錯記錄範例

下列 Gemlin 查詢所需的執行時間可能會比針對慢查詢設定的閾值更長：

```
gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by('
```

然後，如果在偵錯模式下啟用了慢查詢記錄，將為查詢記錄下列屬性，其格式如下所示：

```
{
  "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
  }
}
```

使用記錄 Amazon Neptune API 呼叫 AWS CloudTrail

Amazon Neptune 整合了這項服務 AWS CloudTrail，可提供 Neptune 中使用者、角色或服務所採取之動作記錄的 AWS 服務。 CloudTrail 擷取 Neptune 的 API 呼叫做為事件，包括來自 Neptune 主控台的呼叫，以及從對 Neptune API 的程式碼呼叫。

CloudTrail 僅記錄 Neptune 管理 API 呼叫的事件，例如建立執行個體或叢集。若您希望稽核您圖表的變更，您可以使用稽核日誌。如需詳細資訊，請參閱 [使用稽核日誌搭配 Amazon Neptune 叢集](#)。

> **Important**
>
> Amazon Neptune 主控 AWS CLI台和 API 呼叫會記錄為對 Amazon Relational Database Service 服務 (Amazon RDS) API 的呼叫。

如果您建立追蹤，您可以啟用持續傳遞 CloudTrail 事件到 Amazon S3 儲存貯體，包括 Neptune 的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷向 Neptune 發出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。

Neptune 信息 CloudTrail

CloudTrail 在您創建 AWS 帳戶時，您的帳戶已啟用。當 Amazon Neptune 中發生活動時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以在帳戶中查看，搜索和下載最近的事 AWS 件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中持續的事件記錄 (包括 Neptune 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定的 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔

如果使用 Neptune 主控台、Neptune 命令列界面或 Neptune SDK API 代表您的 AWS 帳戶採取動作，則會將該動作 AWS CloudTrail 記錄為對 Amazon RDS API 發出的呼叫。例如，如果您使用 Neptune 主控台修改資料庫執行個體或呼叫修 AWS CLI改 db 執行個體命令，則 AWS CloudTrail 記錄會顯示對 Amazon RDS API 修改資料庫執行個體動作的呼叫。如需記錄的 Neptune API 動作清單 AWS CloudTrail，請參閱 Neptune API 參考。

> **Note**
>
> AWS CloudTrail 僅記錄 Neptune 管理 API 呼叫的事件，例如建立執行個體或叢集。若您希望稽核您圖表的變更，您可以使用稽核日誌。如需詳細資訊，請參閱 使用稽核日誌搭配 Amazon Neptune 叢集。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或 IAM 使用者憑證提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

了解 Neptune 日誌檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示建立資料庫執行個體快照，然後使用 Neptune 主控台刪除該執行個體的使用者 CloudTrail 記錄。主控台由 `userAgent` 元素識別。透過主控台提出的請求 API 呼叫 (`CreateDBSnapshot`) 和 (`DeleteDBInstance`) 可在各記錄的 `eventName` 元素中找到。使用者 (`Alice`) 的相關資訊則可在 `userIdentity` 元素中找到。

```
{
  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"
    }
  }
  ]
}
```

使用 Neptune 事件通知

主題

- Amazon Neptune 事件類別和事件訊息
- 訂閱 Neptune 事件通知

- 管理 Neptune 事件通知訂閱

Amazon Neptune 會使用 Amazon Simple Notification Service (Amazon SNS)，在 Neptune 事件發生時提供通知。這些通知可以是 Amazon SNS 在某個 AWS 區域支援的任何形式，例如電子郵件、文字訊息或呼叫 HTTP 端點。

Neptune 會將這些事件分組幾個類別讓您可以訂閱，在該類別的事件發生時，您將會收到通知。您可以訂閱資料庫執行個體、資料庫叢集、資料庫快照、資料庫叢集快照或資料庫參數群組的事件類別。例如，如果您訂閱指定資料庫執行個體的備份類別，當發生會影響資料庫執行個體的備份相關事件時，您將會收到通知。當事件通知訂閱變更時，您也會收到通知。

事件發生於資料庫叢集和資料庫執行個體層級，因此，您只要訂閱資料庫叢集或資料庫執行個體，就會收到事件。

事件通知會傳送到您在建立訂閱時提供的地址。您可能想要建立多個不同的訂閱，例如一個訂閱接收所有事件通知，而另一個訂閱僅包含您的生產資料庫執行個體的重要事件。您可以輕鬆關閉通知，而無需刪除訂閱。若要這麼做，請在 Neptune 主控台中將已啟用選項按鈕設定為否。

> **Important**
>
> Amazon Neptune 不保證事件串流中傳送事件的順序。事件順序可能會改變。

Neptune 會使用 Amazon SNS 主題的 Amazon Resource Name (ARN) 來識別每個訂閱。Neptune 主控台會在您建立訂閱時為您建立 ARN。

Neptune 事件通知的帳單是透過 Amazon SNS 寄送。使用事件通知需要支付 Amazon SNS 費用。如需詳細資訊，請參閱 Amazon Simple Notification Service 定價。

Amazon Neptune 事件類別和事件訊息

Neptune 會產生各類別的大量事件，您可以使用 Neptune 主控台訂閱這些類別。每個類別皆套用至某個來源類型，可以是資料庫執行個體、資料庫快照或資料庫參數群組。

> **Note**
>
> Neptune 使用現有的 Amazon RDS 事件定義和 ID。

源自資料庫執行個體的 Neptune 事件

下表顯示當資料庫執行個體為來源類型時的事件清單 (依事件類別排列)。

類別	Amazon RDS 事件 ID	描述
可用性	RDS-EVENT-0006	資料庫執行個體已重新啟動。
	RDS-EVENT-0004	資料庫執行個體關機。
	RDS-EVENT-0022	重新啟動 Neptune 引擎時發生了錯誤。
備份	RDS-EVENT-0001	備份資料庫執行個體。
	RDS-EVENT-0002	已完成資料庫執行個體備份。
組態變更	RDS-EVENT-0009	資料庫執行個體已新增至安全群組。
	RDS-EVENT-0024	資料庫執行個體正在轉換為多可用區域資料庫執行個體。
	RDS-EVENT-0030	資料庫執行個體正在轉換為單一可用區資料庫執行個體。
	RDS-EVENT-0012	套用修改至資料庫執行個體類別。
	RDS-EVENT-0018	此資料庫執行個體目前的儲存設定正在變更。

類別	Amazon RDS 事件 ID	描述
	RDS-EVENT-0011	此資料庫執行個體的參數群組已變更。
	RDS-EVENT-0092	此資料庫執行個體的參數群組已完成更新。
	RDS-EVENT-0028	此資料庫執行個體的自動備份已停用。
	RDS-EVENT-0032	此資料庫執行個體的自動備份已啟用。
	RDS-EVENT-0025	資料庫執行個體已轉換為多可用區域資料庫執行個體。
	RDS-EVENT-0029	資料庫執行個體已轉換為單一可用區資料庫執行個體。
	RDS-EVENT-0014	此資料庫執行個體的資料庫執行個體類別已變更。
	RDS-EVENT-0017	此資料庫執行個體的儲存設定已變更。
	RDS-EVENT-0010	資料庫執行個體已從安全群組移除。
建立	RDS-EVENT-0005	已建立資料庫執行個體。
刪除	RDS-EVENT-0003	資料庫執行個體已遭刪除。

類別	Amazon RDS 事件 ID	描述
容錯移轉	RDS-EVENT-0034	Neptune 不會因為資料庫執行個體最近發生的容錯移轉而嘗試請求的容錯移轉。
	RDS-EVENT-0013	導致備用執行個體提升的多可用區域容錯移轉已啟動。
	RDS-EVENT-0015	導致備用執行個體提升的多可用區域容錯移轉已完成。DNS 傳輸到新的主資料庫執行個體可能需要幾分鐘的時間。
	RDS-EVENT-0065	執行個體已從部分容錯移轉中復原。
	RDS-EVENT-0049	多可用區域容錯移轉已完成。
	RDS-EVENT-0050	多可用區域啟用已在執行個體成功復原之後開始。
	RDS-EVENT-0051	多可用區域啟用已完成。現在應可存取您的資料庫。
	RDS-EVENT-0031	因為不相容的組態或基礎儲存問題，資料庫執行個體已失敗。開始資 point-in-time-restore 料庫執行個體。

類別	Amazon RDS 事件 ID	描述
	RDS-EVENT-0036	資料庫執行個體位於不相容的網路中。部分指定的子網路 ID 無效或不存在。
	RDS-EVENT-0035	資料庫執行個體有無效的參數。例如，如果此執行個體類別與記憶體相關的參數設定過高造成資料庫叢集無法啟動，則客戶動作將為修改記憶體參數並重新啟動資料庫執行個體。
	RDS-EVENT-0082	Neptune 無法從 Amazon S3 儲存貯體複製備份資料。可能是 Neptune 存取 Amazon S3 儲存貯體的許可設定不正確。
低儲存	RDS-EVENT-0089	資料庫執行個體已消耗超過其分配儲存容量的 90%。您可以使用 Free Storage Space (可用儲存空間) 指標來監控資料庫執行個體的儲存空間。

類別	Amazon RDS 事件 ID	描述
	RDS-EVENT-0007	資料庫執行個體的分配儲存空間已耗盡。若要解決此問題，您應為此資料庫執行個體分配額外儲存空間。
維護	RDS-EVENT-0026	資料庫執行個體的離線維護正在進行。資料庫執行個體目前無法使用。
	RDS-EVENT-0027	資料庫執行個體的離線維護已完成。資料庫執行個體現在可用。
	RDS-EVENT-0047	資料庫執行個體的修補作業已完成。
notification	RDS-EVENT-0044	操作者發出的通知。如需更多詳細資訊，請參閱事件訊息。
	RDS-EVENT-0048	資料庫執行個體的修補作業已延遲。
	RDS-EVENT-0087	資料庫執行個體已停止。
	RDS-EVENT-0088	資料庫執行個體已啟動。
	RDS-EVENT-0154	資料庫執行個體由於超過允許停止的時間上限，正在啟動。

類別	Amazon RDS 事件 ID	描述
	RDS-EVENT-0158	資料庫執行個體處於無法升級的狀態。
	RDS-EVENT-0173	已修補資料庫執行個體。
僅供讀取複本	RDS-EVENT-0045	僅供讀取複寫程序發生錯誤。如需更多詳細資訊，請參閱事件訊息。
	RDS-EVENT-0046	僅供讀取複本已恢復複寫。此訊息會在您首次建立僅供讀取複本時出現，或做為確認複寫正常運作的監控訊息顯示。如果此訊息出現在 RDS-EVENT-0045 通知之後，則在錯誤或複寫停止之後已恢復複寫。
	RDS-EVENT-0057	僅供讀取複本上的複寫已終止。
	RDS-EVENT-0062	僅供讀取複本上的複寫已手動停止。
	RDS-EVENT-0063	僅供讀取複本上的複寫已重設。
復原	RDS-EVENT-0020	資料庫執行個體的復原已啟動。復原時間依據恢復的資料量而有不同。

類別	Amazon RDS 事件 ID	描述
	RDS-EVENT-0021	資料庫執行個體的復原已完成。
	RDS-EVENT-0023	已請求手動備份，但 Neptune 目前正在建立資料庫快照。在 Neptune 完成了資料庫快照之後，再次提交請求。
	RDS-EVENT-0052	多可用區域執行個體的復原已啟動。復原時間依據恢復的資料量而有不同。
	RDS-EVENT-0053	多可用區域執行個體的復原已完成。
還原	RDS-EVENT-0008	資料庫執行個體已從資料庫快照還原。
	RDS-EVENT-0019	資料庫執行個體已從point-in-time備份還原。

源自資料庫叢集的 Neptune 事件

下表顯示當資料庫執行個體為來源類型時的事件清單 (依事件類別排列)。

類別	RDS 事件 ID	描述
容錯移轉	RDS-EVENT-0069	資料庫叢集的容錯移轉已失敗。
	RDS-EVENT-0070	資料庫叢集的容錯移轉已重新啟動。

類別	RDS 事件 ID	描述
	RDS-EVENT-0071	資料庫叢集的容錯移轉已結束。
	RDS-EVENT-0072	資料庫叢集的容錯移轉已在相同的可用區域中開始。
	RDS-EVENT-0073	資料庫叢集的容錯移轉已跨越可用區域開始。
	RDS-EVENT-0083	Neptune 無法從 Amazon S3 儲存貯體複製備份資料。可能是 Neptune 存取 Amazon S3 儲存貯體的許可設定不正確。
維護	RDS-EVENT-0156	資料庫叢集有可用的資料庫引擎次要版本升級。
則通知	RDS-EVENT-0076	遷移至 Neptune 資料庫叢集失敗。
	RDS-EVENT-0077	在遷移至 Neptune 資料庫叢集期間，嘗試從來源資料庫將資料表轉換為資料庫形式失敗。
	RDS-EVENT-0150	資料庫叢集已停止。
	RDS-EVENT-0151	資料庫叢集已啟動。
	RDS-EVENT-0152	資料庫叢集停止失敗。

類別	RDS 事件 ID	描述
	RDS-EVENT-0153	資料庫叢集由於超過允許停止的時間上限，正在啟動。

源自資料庫叢集快照的 Neptune 事件

下表顯示當 Neptune 資料庫叢集快照為來源類型時的事件類別和事件清單。

類別	RDS 事件 ID	描述
備份	RDS-EVENT-0074	已開始建立手動資料庫叢集快照。
備份	RDS-EVENT-0075	手動資料庫叢集快照已建立。
則通知	RDS-EVENT-0162	資料庫叢集快照匯出工作失敗。
則通知	RDS-EVENT-0163	已取消資料庫叢集快照匯出工作。
則通知	RDS-EVENT-0164	已完成資料庫叢集快照匯出工作。
備份	RDS-EVENT-0168	建立自動叢集快照。
備份	RDS-EVENT-0169	已建立自動叢集快照。
建立	RDS-EVENT-0170	已建立資料庫叢集。
刪除	RDS-EVENT-0171	資料庫叢集已刪除。
則通知	RDS-EVENT-0172	已將資料庫叢集從 [舊的資料庫叢集名稱] 重

類別	RDS 事件 ID	描述
		新命名為 [新的資料庫叢集名稱]。

源自資料庫叢集參數群組的 Neptune 事件

下表顯示當資料庫叢集參數群組為來源類型時的事件類別和事件清單。

類別	RDS 事件 ID	描述
組態變更	RDS-EVENT-0037	參數群組已修改。

源自安全群組的 Neptune 事件

下表顯示當安全群組為來源類型時的事件類別和事件清單。

類別	RDS 事件 ID	描述
組態變更	RDS-EVENT-0038	安全群組已修改。
失敗	RDS-EVENT-0039	由 [使用者] 擁有的安全群組不存在，安全群組的授權已撤回。

訂閱 Neptune 事件通知

您可以使用 Neptune 主控台來訂閱事件通知，如下所示：

訂閱 Neptune 事件通知

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在導覽窗格中，選擇 Event subscriptions (事件訂閱)。
3. 在 Event subscriptions (事件訂閱) 窗格中，選擇 Create event subscription (建立事件訂閱)。
4. 在 Create event subscription (建立事件訂閱) 對話方塊中，執行下列動作：

a. 在 Name (名稱) 中，輸入事件通知訂閱的名稱。
b. 在 Send notifications to (傳送通知至) 中，選擇 Amazon SNS 主題的現有 Amazon SNS ARN，或選擇 create topic (建立主題) 以輸入主題名稱及收件人清單。
c. 在 Source type (來源類型) 中選擇來源類型。
d. 選擇 Yes (是) 以啟用訂閱。如果您要建立訂閱，但還不要傳送通知，選擇 No (否)。
e. 根據您選擇的來源類型，選擇事件類別以及您要從中接收事件通知的來源。
f. 選擇建立。

管理 Neptune 事件通知訂閱

如果您在 Neptune 主控台的導覽窗格中選擇事件訂閱，則可以檢視訂閱類別和目前訂閱的清單。

您也可以修改或刪除特定訂閱。

修改 Neptune 事件通知訂閱

修改您目前的 Neptune 事件通知訂閱

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在導覽窗格中，選擇 Event subscriptions (事件訂閱)。Event subscriptions (事件訂閱) 窗格顯示所有事件通知訂閱。
3. 在 Event subscriptions (事件訂閱) 窗格中，選擇您要修改的訂閱，然後選擇 Edit (編輯)。
4. 在 Target (目標) 或 Source (來源) 區段中，對訂閱進行變更。您可以新增或移除來源識別符，方法是在來源區段中選取或取消選取它們。
5. 選擇編輯。Neptune 主控台指出正在修改訂閱。

刪除 Neptune 事件通知訂閱

您可以刪除不再需要的訂閱。該主題的所有訂閱者將不會再收到該訂閱指定的事件通知。

刪除 Neptune 事件通知訂閱

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。

2. 在導覽窗格中，選擇 Event subscriptions (事件訂閱)。
3. 在事件訂閱窗格中，選擇您要刪除的訂閱。
4. 選擇刪除。
5. Neptune 主控台指出正在刪除訂閱。

標記 Amazon Neptune 資源

您可以使用 Neptune 標籤，將中繼資料新增至您的 Neptune 資源。此外，您可以搭配 AWS Identity and Access Management (IAM) 政策使用標籤來管理 Neptune 資源的存取，並控制可套用至這些資源的動作。最後，您可以使用標籤將類似的標記資源分類為群組以追蹤成本。

所有 Neptune 管理資源都可以加上標籤，包括下列項目：

- 資料庫執行個體
- 資料庫叢集
- 僅供讀取複本
- 資料庫快照
- 資料庫叢集快照
- 事件訂閱
- 資料庫參數群組
- 資料庫叢集參數群組
- 資料庫子網路群組

Neptune 資源標籤概觀

Amazon Neptune 標籤是您定義並將其與 Neptune 資源建立關聯的名稱值組。此名稱叫做金鑰。為金鑰提供值是選用的。您可以使用標籤，將任意資訊指派給 Neptune 資源。例如，您可以使用標籤金鑰來定義類別，此標籤值可以是該類別中的項目。例如，您可以定義標籤金鑰 “project”及標籤值 “Salix”，以指示 Neptune 資源指派給 Salix 專案。您也可以使用金鑰 (例如 `environment=test` 或 `environment=production`)，以使用標籤來指定用於測試或生產的 Neptune 資源。我們建議您使用一組一致的標籤金鑰，讓您更輕鬆地追蹤與 Neptune 資源相關聯的中繼資料。

使用標籤來組織帳 AWS 單，以反映您自己的成本結構。要做到這一點，註冊以獲取包含標籤鍵值的 AWS 帳戶 帳單。接著，若要查看合併資源的成本，請根據具有相同標籤鍵值的資源來整理您的帳單資

訊。例如，您可以使用特定應用程式名稱來標記數個資源，然後整理帳單資訊以查看該應用程式跨數項服務的總成本。如需詳細資訊，請參閱《AWS Billing 》使用者指南中的使用成本分配標籤。

每個 Neptune 資源皆有標籤集，其中包含指派給該 Neptune 資源的所有標籤。標籤集最多可以包含 10 個標籤，也可以是空的。如果您將標籤新增至 Neptune 資源，而它有與資源上的現有標籤相同的金鑰，則新值會覆寫舊值。

AWS 不會將任何語義意義應用於您的標籤; 標籤嚴格解釋為字符串。Neptune 可在資料庫執行個體或其他 Neptune 資源上設定標籤，這取決於您建立資源時所使用的設定。例如，Neptune 可能新增標籤以表示資料庫執行個體用於生產或測試。

- 標籤金鑰是標籤必要的名稱。字串值長度可以是 1 到 128 個 Unicode 字元，不可在前面加上「`aws:`」或「`rds:`」。此字串只能包含一組 Unicode 字母、數字、空格、「_」、「.」、「/」、「=」、「+」、「–」(Java regex："^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")。
- 標籤值即為選用的標籤字串值，字串值長度可以是 1 到 256 個 Unicode 字元，不可在前面加上「`aws:`」。此字串只能包含一組 Unicode 字母、數字、空格、「_」、「.」、「/」、「=」、「+」、「–」(Java regex："^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")。

  標籤組中的值不必是唯一的，並且可以是 null。例如，您可以在 `project/Trinity` 和 `cost-center/Trinity` 標籤集中有一個鍵值對。

> **Note**
>
> 您可以將標籤新增到快照。不過，您的帳單不會反映這個分組。

您可以使用 AWS Management Console AWS CLI、或 Neptune API 來新增、列出和刪除 Neptune 資源上的標籤。使用 AWS CLI 或 Neptune API 時，您必須為要使用的 Neptune 資源提供 Amazon 資源名稱 (ARN)。如需建構 ARN 的詳細資訊，請參閱 建構 Neptune 的 ARN。

標籤是快取用於授權之用。因此，新增和更新 Neptune 資源的標籤可能需要幾分鐘之後才可以使用。

複製 Neptune 中的標籤

當您建立或還原資料庫執行個體時，您可以指定標籤從資料庫執行個體複製到資料庫執行個體的快照。複製標籤可確保資料庫快照的中繼資料符合來源資料庫執行個體的中繼資料，以及資料庫快照的任何存取政策也符合來源資料庫執行個體的存取政策。根據預設，不會複製標籤。

您可以指定標籤複製到資料庫快照以用於下列動作：

- 建立資料庫執行個體。
- 還原資料庫執行個體。
- 建立僅供讀取複本。
- 複製資料庫快照。

> **Note**
>
> 如果您包含建立-db-cluster-snapshot AWS CLI 命令的`--tag-key`參數值 (或為 CreateDBClusterSnapshot API 動作提供至少一個標籤)，Neptune 不會將標籤從來源資料庫執行個體複製到新的資料庫快照集。即使來源資料庫執行個體啟用 `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) 選項，也是如此。
> 這表示您可以從資料庫快照建立資料庫執行個體的複本，避免新增不適用於新資料庫執行個體的標籤。使用 AWS CLI `create-db-cluster-snapshot`命令 (或 `CreateDBClusterSnapshot` Neptune API 動作) 建立資料庫快照集之後，您就可以依本主題稍後所述新增標籤。

在 Neptune 中使用標記 AWS Management Console

設定 Amazon Neptune 資源標籤的程序類似為所有資源設定標籤的程序。下列程序說明如何設定 Neptune 資料庫執行個體標籤。

新增標籤至資料庫資訊個體

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在導覽窗格中，選擇執行個體。

   > **Note**
   >
   > 若要過濾 Instances (執行個體) 窗格中的資料庫執行個體清單，請在 Filter instances (過濾執行個體) 方塊中輸入文字字串。只有包含此字串的資料庫執行個體會顯示出來。

3. 選擇您要設定標籤的資料庫執行個體。
4. 選擇 Instance actions (執行個體動作)，然後選擇 See details (請參閱詳細資訊)。
5. 在詳細內容區段，向下捲動至 Tags (標籤) 區段。

6. 選擇 Add (新增)。此時將會顯示 Add tags (新增標籤) 視窗。
7. 在 Tag key (標籤金鑰) 和 Value (值) 中輸入值。
8. 若要新增另一個標籤，您可以選擇 Add another Tag (新增另一個標籤)，然後在它的 Tag key (標籤金鑰) 和 Value (值) 中輸入值。

   視需要重複此步驟。
9. 選擇 Add (新增)。

從資料庫資訊個體刪除標籤

1. 登入 AWS 管理主控台，然後開啟 Amazon Neptune 主控台，網址為 https://console.aws.amazon.com/neptune/home。
2. 在導覽窗格中，選擇執行個體。

   > **Note**
   >
   > 若要過濾 Instances (執行個體) 窗格中的資料庫執行個體清單，請在 Filter instances (過濾執行個體) 方塊中輸入文字字串。只有包含此字串的資料庫執行個體會顯示出來。
3. 選擇您要設定標籤的資料庫執行個體。
4. 選擇 Instance actions (執行個體動作)，然後選擇 See details (請參閱詳細資訊)。
5. 在詳細內容區段，向下捲動至 Tags (標籤) 區段。
6. 選擇您要刪除的標籤。
7. 選擇 Remove (移除)，然後選擇 Remove tags (移除標籤) 視窗中的 Remove (移除)。

在 Neptune 中使用標記 AWS CLI

您可以使用 AWS CLI在 Neptune 中新增、列出或移除資料庫執行個體的標籤。

- 若要將一或多個標籤新增至 Neptune 資源，請使用 AWS CLI 命令`add-tags-to-resource`。
- 若要列出 Neptune 資源上的標籤，請使用 AWS CLI 命令`list-tags-for-resource`。
- 若要從 Neptune 資源中移除一或多個標籤，請使用 AWS CLI 命令`remove-tags-from-resource`。

若要進一步了解如何建構必要的 Amazon Resource Name (ARN)，請參閱建構 Neptune 的 ARN。

使用 API 在 Neptune 中進行標記

您可以使用 Neptune API，新增、列出或移除資料庫執行個體的標籤。

- 若要將標籤新增到 Neptune 資源，請使用 AddTagsToResource 操作。
- 若要列出指派給 Neptune 資源的標籤，請使用 ListTagsForResource。
- 若要從 Neptune 資源中移除標籤，請使用 RemoveTagsFromResource 操作。

若要進一步了解如何建構必要的 ARN，請參閱建構 Neptune 的 ARN。

搭配 XML 使用 Neptune API 時，標籤會使用以下結構描述：

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

下表列出允許的 XML 標籤及其特性。Key 和 Value 的值區分大小寫。例如，project=Trinity 和 PROJECT=Trinity 是兩個不同的標籤。

標籤元素	描述
TagSet	標籤集是指派給 Neptune 資源之所有標籤的容器。每個資源只能有一個標籤集。您只能透過 Neptune API 使用 TagSet。
Tag	標籤是使用者定義的鍵值組。標籤集內可以有 1 到 50 個標籤。
Key	鍵是標籤的必要名稱。字串值長度可以是 1 到 128 個 Unicode 字元，不可在前面加上「rds:」或「aws:」。此字串只能包含一組 Unicode 字母、數字、空格、「_」、「.」、「/」、「=」、「+」、「–」(Java regex："^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$ ")。

標籤元素	描述
	鍵在標籤集內必須是唯一的。例如，標籤集內的鍵組不能有相同的鍵但使用不同的值，例如 `project/Trinity` 和 `project/Xanadu` 。
Value	值是標籤的選用值。字串值長度可以是 1 到 256 個 Unicode 字元，不可在前面加上「`rds:`」或「`aws:`」。此字串只能包含一組 Unicode 字母、數字、空格、「_」、「.」、「/」、「=」、「+」、「–」(Java regex：`"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$ `)。 標籤集內的值不必是唯一的，並且可以是 null。例如，您可以在 `project/Trinity` 和 `cost-center/Trinity` 標籤集中有一個鍵值對。

在 Amazon Neptune 中使用管理 ARN

在 Amazon Web Services 中建立的資源，都是用 Amazon Resource Name (ARN) 做為唯一識別符。對於特定 Amazon Neptune 操作，您必須指定 ARN 以唯一識別 Neptune 資源。

> **Important**
>
> Amazon Neptune 共用 Amazon RDS ARN 的格式，用於使用 管理 API 參考 的管理動作. Neptune 管理 ARN 包含 `rds`，但不包含 `neptune-db`。如需識別 Neptune 資料資源的資料平面 ARN，請參閱指定資料資源。

主題

- 建構 Neptune 的 ARN
- 在 Amazon Neptune 中取得現有的 ARN

建構 Neptune 的 ARN

您可以使用下列語法，建構 Amazon Neptune 資源的 ARN。請注意，Neptune 共用 Amazon RDS ARN 的格式。

`arn:aws:rds:`*`<region>`*`:`*`<account number>`*`:`*`<resourcetype>`*`:`*`<name>`*

下表顯示您在為特定 Neptune 管理資源類型建構 ARN 時應使用的格式。

資源類型	ARN 格式
資料庫執行個體	arn:aws:rds:*<region>*:*<account>* :db:*<name>* 例如： `arn:aws:rds: us-east-2 :123456789012 :db:my-instance-1`
資料庫叢集	arn:aws:rds:*<region>*:*<account>* :cluster: *<name>* 例如： `arn:aws:rds: us-east-2 :123456789012 :cluster: my-cluster-1`
事件訂閱	arn:aws:rds:*<region>*:*<account>* :es:*<name>* 例如： `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB parameter group (資料庫參數群組)	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* 例如： `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
DB cluster parameter group (資料庫叢集參數群組)	arn:aws:rds:*<region>*:*<account>* :cluster-pg: *<name>* 例如： `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
資料庫叢集快照	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot: *<name>*

資源類型	ARN 格式
	例如： `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-snap-20160809`
資料庫子網路群組	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* 例如： `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`

在 Amazon Neptune 中取得現有的 ARN

您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 Neptune API 來取得 Neptune 資源的 ARN。

使用取得現有的 ARN AWS Management Console

若要使用主控台取得 ARN，請瀏覽至您需要 ARN 的資源，然後檢視該資源的詳細資訊。例如，若要取得資料庫執行個體的 ARN，請選擇導覽面板中的 Instances (執行個體)，然後從清單中選擇所要的執行個體。ARN 位在 Instance Details (執行個體詳細資訊) 部分。

使用取得現有的 ARN AWS CLI

若要使用取 AWS CLI 得特定 Neptune 資源的 ARN，請使用該資源的`describe`命令。下表顯示每個 AWS CLI 命令和 ARN 屬性，這些屬性與命令搭配使用以取得 ARN。

AWS CLI 命令	ARN 屬性
describe-event-subscriptions	EventSubscription阿恩
describe-certificates	CertificateArn
describe-db-parameter-groups	DB ParameterGroup 阿恩

AWS CLI 命令	ARN 屬性
describe-db-cluster-parameter-groups	資料庫 ClusterParameter GroupArn
describe-db-instances	資料庫 InstanceArn
describe-events	SourceArn
describe-db-subnet-groups	DB SubnetGroup 阿恩
describe-db-clusters	資料庫 ClusterArn
describe-db-cluster-snapshots	DB ClusterSnapshot 阿恩

例如，下列 AWS CLI 命令會取得資料庫執行個體的 ARN。

Example

對於 Linux、OS X 或 Unix：

```
aws neptune describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

針對 Windows：

```
aws neptune describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

使用 API 取得現有 ARN

若要取得特定 Neptune 資源的 ARN，請呼叫以下 API 動作，並使用如下所示的 ARN 屬性。

Neptune API 動作	ARN 屬性
DescribeEvent訂閱	EventSubscription阿恩
DescribeCertificates	CertificateArn

Neptune API 動作	ARN 屬性
描述 B ParameterGroups	DB ParameterGroup 阿恩
描述 B 群組 ClusterParameter	資料庫 ClusterParameter GroupArn
DescribeDBInstances	資料庫 InstanceArn
DescribeEvents	SourceArn
描述 B SubnetGroups	DB SubnetGroup 阿恩
DescribeDBClusters	資料庫 ClusterArn
描述 B ClusterSnapshots	DB ClusterSnapshot 阿恩

備份和還原 Amazon Neptune 資料庫叢集

本節顯示如何備份和還原 Neptune 資料庫叢集。

主題

備份和還原 Neptune 資料庫叢集的概觀

本節提供有關在 Amazon Neptune 備份和還原資料的高階資訊。

主題

Neptune 資料庫叢集的容錯能力

Neptune 資料庫叢集特意設計為具備容錯能力。叢集磁碟區跨單一 AWS 區域中的多個可用區域，每個可用區域包含叢集磁碟區資料的副本。此功能意味著資料庫叢集可承受可用區域故障，完全不會遺失資料，服務只會短暫中斷。

如果資料庫叢集中的主要執行個體失敗，Neptune 可透過兩種方式自動容錯移轉至新的主要執行個體：

- 將現有的 Neptune 複本提升為新的主要執行個體
- 建立新的主要執行個體

如果資料庫叢集有一或多個 Neptune 複本，則在失敗事件期間會將 Neptune 複本提升為主要執行個體。失敗事件會導致短暫中斷，在此期間，讀取和寫入操作會失敗，並引發例外狀況。不過，服務通常會在 120 秒之內恢復，往往不超過 60 秒。若要提高資料庫叢集的可用性，建議在兩個以上不同的可用區域中建立至少一或多個 Neptune 複本。

您可以指派每個 Neptune 複本的優先順序，以自訂複本在失敗之後提升為主要執行個體的順序。優先順序從 0 (代表最高優先順序) 到 15 (代表最低優先順序)。如果主要執行個體失敗，Neptune 會將優先順序最高的 Neptune 複本提升為新的主要執行個體。您隨時都可以修改 Neptune 複本的優先順序。修改優先順序不會觸發容錯移轉。

您可以使用 AWS CLI 來設定資料庫執行個體的容錯移轉優先順序，如下所示：

```
aws neptune modify-db-instance --db-instance-identifier (the instance ID) --promotion-
tier (the failover priority value)
```

多個 Neptune 複本可以擁有相同的優先順序，形成提升層。如果兩個以上的 Neptune 複本共用相同的優先順序，Neptune 會提升最大的複本。如果兩個以上的 Neptune 複本擁有相同的優先順序和大小，Neptune 會提升相同提升層中的任意複本。

如果資料庫叢集不包含任何 Neptune 複本，則會在失敗事件期間重建主要執行個體。失敗事件會導致中斷，在此期間，讀取和寫入操作會失敗，並引發例外狀況。建立新的主要執行個體後，服務就會恢復，通常不超過 10 分鐘。將 Neptune 複本提升為主要執行個體，比建立新的主要執行個體快得多。

Neptune 備份

Neptune 會自動備份您的叢集磁碟區，並在備份保留期保留還原資料。Neptune 備份具有連續性和增量性，因此，您可以快速還原到備份保留期內的任何時間點。寫入備份資料時不會影響資料庫服務的效能或中斷服務。當您建立或修改資料庫叢集時，您可以指定 1 到 35 天的備份保留期。

若要控制您的備份儲存用量，您可以降低備份保留間隔、移除不再需要的舊手動快照，或同時進行兩者。若要協助管理您的成本，您可以監控連續備份和超出保留期的手動快照所耗用的儲存量。您可以降低備份保留間隔，並移除不再需要的手動快照。

如果希望備份可以保留超過備份保留期，您也可以在叢集磁碟區中建立資料快照。儲存快照需要支付 Neptune 的標準儲存費用。如需 Neptune 儲存體定價的詳細資訊，請參閱 Amazon Neptune 定價。

Neptune 會保留整個備份保留期的增量還原資料。因此，您只需要建立希望在備份保留期後仍能保留的資料快照。您可以從快照建立新的資料庫叢集。

Important

如果您刪除資料庫叢集，其所有自動備份都會同時刪除，且無法將其復原。這表示，除非您選擇手動建立最後的資料庫快照，否則稍後無法將資料庫執行個體還原至其最後狀態。刪除叢集時，並不會刪除手動快照。

Note

- 無論資料庫叢集的建立方式為何，Amazon Neptune 資料庫叢集的預設備份保留期為一天。
- 您無法停用 Neptune 上的自動備份。Neptune 的備份保留期是由資料庫叢集管理。

有助於管理 Neptune 備份儲存體的 CloudWatch 指標

您可以使用 Amazon CloudWatch 指標 `TotalBackupStorageBilled`、`SnapshotStorageUsed` 和 `BackupRetentionPeriodStorageUsed`，來檢閱和監控 Neptune 備份所使用的儲存用量，如下所示：

- `BackupRetentionPeriodStorageUsed` 表示目前用於儲存連續備份的備份儲存量 (以位元組為單位)。此值取決於叢集磁碟區的大小，以及您在保留期中所做的變更量。不過，為了方便計費，該值不會超過保留期內累積的叢集磁碟區大小。例如，如果您叢集的 `VolumeBytesUsed` 大小為 107,374,182,400 位元組 (100GiB)，而保留期間為兩天，則 `BackupRetentionPeriodStorageUsed` 的最大值為 214,748,364,800 位元組 (100 GiB + 100 GiB)。
- `SnapshotStorageUsed` 表示用於儲存在備份保留期間外的手動快照的備份儲存量 (以位元組為單位)。手動快照不會計入快照備份儲存，手動快照的建立時間戳記在保留期間內。所有自動快照也不會計入您的快照備份儲存。每個快照的大小是您取得快照時的叢集磁碟區大小。`SnapshotStorageUsed` 值取決於您保留的快照數量和每個快照的大小。例如，假設您在保留期間之外有一個手動快照，且叢集在拍攝快照時的 `VolumeBytesUsed` 大小為 100 GiB。SnapshotStorageUsed 的量是 107,374,182,400 位元組 (100 GiB)。
- `TotalBackupStorageBilled` 表示 `BackupRetentionPeriodStorageUsed` 和 `SnapshotStorageUsed` 的總和 (以位元組為單位)，減去相當於叢集磁碟區大小的一天免費備份儲存量。可用備份儲存等同於最新的磁碟區大小。例如，如果您叢集的 `VolumeBytesUsed` 大小是 100 GiB，您的保留期間是兩天，而您在保留期間之外有一個手動快照，則 `TotalBackupStorageBilled` 是 214,748,364,800 個位元組 (200 GiB + 100 GiB - 100 GiB)。

您可以透過 CloudWatch 主控台使用 CloudWatch 指標，來監控 Neptune 叢集和建置報告。如需如何使用 CloudWatch 指標的詳細資訊，請參閱監控 Neptune和Neptune CloudWatch 度量中的指標表格。

從 Neptune 備份還原資料

您可以從 Neptune 保留的備份資料或您已儲存的資料庫叢集快照來建立新的 Neptune 資料庫叢集，以復原資料。您可以利用從備份資料所建立之資料庫叢集的新副本，快速還原至備份保留期之內的任何時間點。Neptune 在備份保留期的連續和增量性質，意味著您不需要經常建立資料的快照來改善還原時間。

若要判斷資料庫執行個體的最晚或最早可還原時間，請在 Neptune 主控台尋找 `Latest Restorable Time` 或 `Earliest Restorable Time` 值。資料庫叢集的最晚可還原時間是指可讓您還原資料庫叢

集的最近時間點，通常在目前時間的 5 分鐘內。最早可還原時間指定您在備份保留期之內可將叢集磁碟區還原到多久以前。

您可以檢查 `Latest Restorable Time` 和 `Earliest Restorable Time` 值，以判斷資料庫叢集還原何時完成。在還原操作完成之前，`Latest Restorable Time` 和 `Earliest Restorable Time` 值會傳回 NULL。如果 `Latest Restorable Time` 或 `Earliest Restorable Time` 傳回 NULL，則您無法要求備份或還原操作。

使用 AWS Management Console 將資料庫執行個體還原至指定的時間

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Instances (執行個體)。選擇您要還原之資料庫叢集的主要執行個體。
3. 選擇 Instance actions (執行個體動作)，然後選擇 Restore to point in time (還原至時間點)。

   在啟動資料庫執行個體視窗中，選擇還原時間下的自訂。
4. 在 Custom (自訂) 下指定您要還原至何時的日期和時間。
5. 針對設定下的資料庫執行個體識別符，輸入新還原資料庫執行個體的名稱。
6. 選擇 Launch DB Instance (啟動資料庫執行個體)，以啟動還原的資料庫執行個體。

   將會以您指定的名稱建立新的資料庫執行個體，並建立新的資料庫叢集。資料庫叢集名稱是新的資料庫執行個體名稱，後面加上 `-cluster`。例如，若新的資料庫執行個體名稱是 `myrestoreddb`，則新的資料庫叢集名稱為 `myrestoreddb-cluster`。

Neptune 中的備份時段

自動備份會每天在偏好的備份時段內執行。如果備份需要的時間超過所分配的備份時段，備份會在時段結束後繼續執行直到完成。備份時段不可與每週的資料庫執行個體維護時段重疊。

在自動備份期間，儲存輸入/輸出可能會在備份程序初始化時短暫暫停 (通常在幾秒內)。在備份異地同步備份部署時，可能會有幾分鐘的延遲增加情形。

備份時段通常由以 Neptune 為基礎的 Amazon RDS 控制平面，從每個區域的八小時時間區塊隨機選擇。從中指派預設備份時段之每個區域的時間區塊記載於《Amazon RDS 使用者指南》的備份時段一節中。

在 Neptune 中建立資料庫叢集快照

Neptune 會建立資料庫叢集的儲存體磁碟區快照，備份整個資料庫叢集，而不只是個別的資料庫。當您建立資料庫叢集快照時，您需要識別要備份的資料庫叢集。接著，為您的資料庫叢集快照命名，以便稍後可從該快照還原。建立資料庫叢集快照所需的時間長短隨資料庫的大小而異。快照包含整個儲存磁碟區。因此，檔案大小 (例如暫存檔案) 也會影響建立快照所需的時間。

您可以使用 AWS Management Console、AWS CLI 或 Neptune API 建立資料庫叢集快照。

使用主控台建立資料庫叢集快照

建立資料庫叢集快照

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Databases (資料庫)。
3. 在資料庫執行個體清單，選取資料庫叢集的主要執行個體。
4. 選擇 Instance actions (執行個體動作)，然後選擇 Take snapshot (擷取快照)。

   Take DB Snapshot (建立資料庫快照) 視窗隨即顯示。
5. 在 Snapshot name (快照名稱) 方塊中輸入資料庫叢集快照的名稱。
6. 選擇 Take Snapshot (擷取快照)。

從資料庫叢集快照還原

當您建立資料庫叢集的 Amazon Neptune 快照時，Neptune 會建立該叢集的儲存體磁碟區快照，備份其所有資料，而不只是個別的執行個體。您稍後可以從這個資料庫叢集快照進行還原來建立新的資料庫叢集。還原資料庫叢集時，您需提供要做為還原來源的資料庫叢集快照名稱，然後為還原時所建立的新資料庫叢集提供一個名稱。

內容

從快照還原 Neptune 資料庫叢集時要謹記的事項

您無法還原至現有的資料庫叢集

還原程序一律會建立新的資料庫叢集，因此您無法還原至已存在的資料庫叢集。

不會還原任何執行個體

還原時所建立的新資料庫叢集沒有與其相關聯的執行個體。

一旦還原完成且您的新資料庫叢集可供使用，請明確建立您將需要的執行個體。您可以在 Neptune 主控台上或使用 CreateDBInstance API 來執行此動作。

不會還原任何自訂參數群組

還原時建立的新資料庫叢集會自動具有與其相關聯的預設資料庫參數群組。

一旦還原完成且您的新資料庫叢集可供使用，請為您從中進行還原的執行個體正在使用的任何自訂資料庫參數群組建立關聯。若要這麼做，請在 Neptune 主控台或 ModifyDBInstance API 上使用修改命令。

> **Important**
>
> 建議您儲存要建立其快照的資料庫叢集中使用的自訂參數群組。然後，從該快照進行還原時，您可以輕鬆地將正確的參數群組與還原的資料庫叢集建立關聯。

不會還原任何自訂安全群組

還原時建立的新資料庫叢集會自動具有與其相關聯的預設安全群組。

一旦還原完成且您的新資料庫叢集可供使用，請為您從中進行還原的執行個體正在使用的任何自訂安全群組建立關聯。若要這麼做，請在 Neptune 主控台或 ModifyDBInstance API 上使用修改命令。

您無法從共用的加密快照進行還原。

您無法從已共用又加密的資料庫叢集快照來還原資料庫叢集。

相反，建立非共用的快照複本，然後從該副本進行還原。

還原的資料庫叢集會使用與以前一樣多的儲存體

當您從資料庫叢集快照還原資料庫叢集時，配置給新叢集的儲存空間數量與配置給建立快照的資料庫叢集相同，無論該配置的儲存空間實際使用量有多少。

換句話說，您需要付費的「高水位線」不會變更。重設高水位需要從圖形中導出資料，然後將其重新載入新資料庫叢集 (請參閱Neptune 儲存計費)。

如何從快照進行還原

您可以使用 AWS Management Console、AWS CLI 或 Neptune API，從資料庫叢集快照還原資料庫叢集。

使用主控台從快照還原

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Snapshots (快照)。
3. 選擇想要從中還原的資料庫叢集快照。
4. 選擇 Actions (動作) 和 Restore Snapshot (還原快照)。

5. 在 Restore DB Instance (還原資料庫執行個體) 頁面上，於 DB Instance Identifier (資料庫執行個體識別符) 方塊中輸入已還原的資料庫叢集名稱。
6. 選擇 Restore DB Instance (還原資料庫執行個體)。
7. 如果您想要將資料庫叢集的功能還原為建立快照來源的資料庫叢集的功能，您必須修改資料庫叢集，才能使用安全群組。後續步驟假設您的資料庫叢集位於虛擬私有雲端 (VPC) 中。如果您的資料庫叢集不在 VPC 中，請使用 Amazon EC2 主控台找到資料庫叢集所需的安全群組。

   a. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
   b. 在導覽窗格中，選擇 Security Groups (安全群組)。
   c. 選擇您想要用於資料庫叢集的安全群組。如果必要，請新增規則以將安全群組連結至 EC2 執行個體的安全群組。

複製資料庫叢集快照

使用 Neptune，您可以複製自動或手動的資料庫叢集快照。複製快照之後，副本即為手動快照。

您可以在相同 AWS 區域內和跨 AWS 區域複製快照。

將自動快照複製到另一個 AWS 帳戶分成兩個步驟：首先，從自動快照建立手動快照，然後再將手動快照複製到另一個帳戶。

除了複製，您也可以將手動快照共用給其他 AWS 帳戶。如需更多詳細資訊，請參閱 共享資料庫叢集快照。

主題

- 複製快照時的限制
- 保留資料庫叢集快照副本
- 複製快照時處理加密
- 跨 AWS 區域複製快照
- 使用主控台複製資料庫叢集快照
- 使用 AWS CLI 複製資料庫叢集快照

複製快照時的限制

以下是複製快照時的一些限制：

- 您可以在中國 (北京) 與中國 (寧夏) 之間複製快照，但無法在這些中國區域與其他 AWS 區域之間複製快照。
- 您可以在 AWS GovCloud (US-East) 與 AWS GovCloud (US-West) 之間複製快照，但無法在這些 AWS GovCloud (US) 區域與其他 AWS 區域之間複製快照。
- 如果您在目標快照變成可用之前刪除來源快照，則快照副本可能會失敗。刪除來源快照之前，請確認目標快照的狀態為 AVAILABLE。
- 每一帳戶的單一區域最多可有 5 個進行中的快照副本請求。
- 根據涉及的區域和要複製的資料量而定，跨區域快照複製可能需要數小時才會完成。

  如果有大量跨區域快照複製請求來自某個來源 AWS 區域，在一些進行中的複製完成之前，Neptune 可能會將該來源 AWS 區域新提出的跨區域複製請求排入佇列。位於該佇列中的複製請求不會顯示進度資訊。只有在複製開始後才會顯示進度資訊。

保留資料庫叢集快照副本

Neptune 會刪除自動快照，如下所示：

- 在其保留期結束時。
- 當您停用資料庫叢集的自動快照時。
- 當您刪除資料庫叢集時。

如果想要讓自動快照保留期間更長，請複製自動快照來建立手動快照，然後即可保留到您刪除為止。如果手動快照超出預設儲存空間，則其可能產生 Neptune 儲存成本。

如需備份儲存成本的詳細資訊，請參閱 Neptune 定價。

複製快照時處理加密

您可以複製以 AWS KMS 加密金鑰所加密的快照。如果您複製加密快照，則快照的副本也必須加密。您可以使用與原始快照相同的 AWS KMS 加密金鑰來加密該副本，或者您可以指定不同的 AWS KMS 加密金鑰。

複製時，您不能加密未加密的資料庫叢集快照。

針對 Amazon Neptune 資料庫叢集快照，您也可以維持不加密資料庫叢集快照，改在還原時指定 AWS KMS 加密金鑰。還原的資料庫叢集會以指定的金鑰加密。

跨 AWS 區域複製快照

> **Note**
>
> 此功能從 Neptune 引擎版本 1.0.2.1 版開始提供。

將快照複製到一個 AWS 區域時，若該區域並非來源快照的 AWS 區域，則第一個副本是完整的快照副本，即使您複製的是增量式快照，也是如此。完整的快照副本內含所有還原資料庫執行個體所需的資料及中繼資料。完成第一個快照副本之後，您可以將相同資料庫執行個體的增量快照，複製至相同 AWS 帳戶內的相同目的地區域。

增量式快照僅含相同資料庫執行個體最近一次拍攝快照之後所變更的資料。增量式快照複製速度較快，而且儲存成本較完整的快照複製低。加密與未加密快照均適用跨 AWS 區域的增量式快照複製。

> **Important**
>
> 對於共用快照，不支援複製增量式快照。對於共用快照，即使位於相同區域內，所有副本都是完整快照。

根據涉及的 AWS 區域和要複製的資料量，跨區域快照複製可能需要數小時才會完成。

使用主控台複製資料庫叢集快照

如果來源資料庫引擎是 Neptune，則快照即為資料庫叢集快照。對於每個 AWS 帳戶，您一次可在每一 AWS 區域中最多複製五個資料庫叢集快照。支援複製加密和未加密的資料庫叢集快照。

如需資料傳輸定價的詳細資訊，請參閱 Neptune 定價。

若要取消已開始進行的複製操作，請在目標資料庫叢集快照處於 copying (複製中) 狀態時，刪除其快照。

下列程序適用於複製加密或未加密的資料庫叢集快照：

複製資料庫叢集快照

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Snapshots (快照)。
3. 選取您要複製之資料庫叢集快照的核取方塊。
4. 選擇 Actions (動作)，然後選擇 Copy Snapshot (複製快照)。Make Copy of DB Snapshot (複製資料庫快照) 頁面隨即顯示。
5. 在 New DB Snapshot Identifier (新的資料庫快照識別符) 中輸入資料庫叢集快照副本的名稱。
6. 若要將快照中的標籤和值複製到快照的副本，請選擇 Copy Tags (複製標籤)。
7. 在 Enable Encryption (啟用加密) 中，選擇以下其中一項：

   - 如果資料庫叢集快照未加密，且您不要將副本加密，請選擇 Disable encryption (停用加密)。
   - 如果資料庫叢集快照未加密，但您想要將副本加密，請選擇 Enable encryption (啟用加密)。在此情況下，針對主金鑰指定用來加密資料庫叢集快照副本的 AWS KMS 金鑰識別符。
   - 如果資料庫叢集快照已加密，請選擇 Enable encryption (啟用加密)。在此情況下，您必須加密副本，所以已選取 Yes (是)。針對主金鑰，指定用來加密資料庫叢集快照副本的 AWS KMS 金鑰識別符。

8. 選擇 Copy Snapshot (複製快照)。

使用 AWS CLI 複製資料庫叢集快照

您可以使用 copy-db-cluster-snapshot AWS CLI 命令來複製資料庫快照。

如果您要將快照複製到新的 AWS 區域，請在新區域中執行此命令。

使用以下參數說明和範例來決定使用 AWS CLI 複製快照時要使用哪些參數。

- `--source-db-cluster-snapshot-identifier` – 來源資料庫快照的識別符。
  - 如果來源快照和副本位於相同的 AWS 區域，請指定有效的資料庫快照識別符，例如 `neptune:instance1-snapshot-20130805`。
  - 如果來源快照和副本位於不同的 AWS 區域，請指定有效的資料庫快照 ARN，例如 `arn:aws:neptune:us-west-2:123456789012:snapshot:instance1-snapshot-20130805`。
  - 如果您要從共用的手動資料庫快照複製，此參數必須是共用資料庫快照的 Amazon 資源名稱 (ARN)。
  - 如果您要複製加密快照，此參數必須是來源 AWS 區域的 ARN 格式，且必須符合 `SourceDBSnapshotIdentifier` 參數中的 `PreSignedUrl`。
- `--target-db-cluster-snapshot-identifier` – – 加密資料庫快照之新副本的識別符。
- `--kms-key-id` – – 加密資料庫快照的 AWS KMS 金鑰 ID。AWS KMS 金鑰 ID 是 AWS KMS 加密金鑰的 Amazon Resource Name (ARN)、AWS KMS 金鑰識別符或 AWS KMS 金鑰別名。
  - 如果您從 AWS 帳戶複製加密資料庫快照，您可以指定此參數的值，以使用新的 AWS KMS 加密金鑰來加密副本。如果您不指定此參數的值，則會使用與來源資料庫快照相同的 AWS KMS 金鑰，以加密資料庫快照的副本。
  - 您無法使用此參數，建立未加密快照的加密副本。嘗試這樣做將會產生錯誤。
  - 如果您將加密快照複製到不同的 AWS 區域，則必須針對目的地 AWS 區域指定 AWS KMS 金鑰。AWS KMS 加密金鑰是其建立所在 AWS 區域特有的，且您無法將來自某個 AWS 區域的加密金鑰用於另一個 AWS 區域。
- `--source-region` - - 來源資料庫快照所在之 AWS 區域的 ID。如果您將加密快照複製到不同的 AWS 區域，則必須指定此選項。
- `--region` - - 快照要複製到其中之 AWS 區域的 ID。如果您將加密快照複製到不同的 AWS 區域，則必須指定此選項。

Example 從未加密的快照到相同區域

下列程式碼會使用新名稱 mydbsnapshotcopy, 建立從 us-east-1 AWS 區域到 us-west-2 區域的快照複本。

對於 Linux、OS X 或 Unix：

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

針對 Windows：

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Example 從未加密的快照跨區域

下列程式碼會使用新名稱 mydbsnapshotcopy, 建立從 us-east-1 AWS 區域到 us-west-2 區域的快照複本。在 us-west-2 區域中執行命令。

對於 Linux、OS X 或 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
```

針對 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 從已加密的快照跨區域

下列程式碼範例會將加密的資料庫快照從 us-east-1 AWS 區域複製到 us-west-2 區域。在 us-west-2 區域中執行命令。

對於 Linux、OS X 或 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
```

針對 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
```

共享資料庫叢集快照

使用 Neptune 時，您可以透過下列方式共用手動資料庫叢集快照：

- 共用手動資料庫叢集快照 (無論加密或未加密) 可讓授權的 AWS 帳戶複製快照。
- 分享手動資料庫叢集快照 (無論加密或未加密) 可讓授權的 AWS 帳戶直接從快照還原資料庫叢集，而不需要先複製再還原。

> **Note**
>
> 若要共用自動資料庫叢集快照，請複製該自動快照來建立手動資料庫叢集快照，然後共用該複本。

如需從資料庫叢集快照還原資料庫叢集的詳細資訊，請參閱如何從快照進行還原。

您可以將手動快照最多共用給其他 20 個 AWS 帳戶。您也可以將未加密的手動快照以公有形式共用，讓所有 AWS 帳戶都可使用此快照。將快照以公有形式共用時，請小心不要在任何公有快照中包含您的私人資訊。

> **Note**
>
> 使用 AWS Command Line Interface (AWS CLI) 或 Neptune API 從共用快照還原資料庫叢集時，您必須指定共用快照的 Amazon Resource Name (ARN) 做為快照識別符。

主題

共用加密的資料庫叢集快照。

您可以共用已使用 AES-256 加密演算法「靜態」加密的資料庫叢集快照。如需更多詳細資訊，請參閱靜態加密 Neptune 資源。若要這樣做，您必須採取下列步驟：

1. 將用於對快照進行加密的 AWS Key Management Service (AWS KMS) 加密金鑰，與您希望能夠存取此快照的任何帳戶共享。

您可以將其他帳戶新增到 KMS 金鑰政策，以將 AWS KMS 加密金鑰與另一個 AWS 帳戶共用。如需有關更新金鑰政策的詳細資訊，請參閱 https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html 開發人員指南中的AWS KMS金鑰政策。有關建立金鑰政策的範例，請參閱本主題稍後的建立 IAM 政策以允許複製加密快照。

2. 使用 AWS Management Console、AWS CLI 或 Neptune API，與其他帳戶共用加密快照。

共用加密快照有下列限制：

- 您無法將加密快照以公有形式共用。
- 如果快照是以共用快照之 AWS 帳戶的預設 AWS KMS 加密金鑰進行加密，則您無法共用此快照。

允許存取 AWS KMS 加密金鑰

若要讓另一個 AWS 帳戶複製從您的帳戶共用的已加密資料庫叢集快照，您與之共用快照的帳戶必須具有加密該快照之 KMS 金鑰的存取權。若要允許另一個 AWS 帳戶存取 AWS KMS 金鑰，請使用您在 KMS 金鑰政策中以 `Principal` 身分共用之 AWS 帳戶的 ARN，更新 KMS 金鑰的金鑰政策。然後允許 `kms:CreateGrant` 動作。如需一般指示，請參閱《AWS Key Management Service 開發人員指南》中的允許其他帳戶中的使用者使用 KMS 金鑰。

在您授權 AWS 帳戶存取您的 KMS 加密金鑰之後，若要複製您的加密快照，該 AWS 帳戶必須建立 IAM 使用者 (如果還沒有的話)。KMS 安全限制不允許將根 AWS 帳戶身分識別用於此情況。AWS 帳戶還必須將 IAM 政策附加到該 IAM 使用者，以允許 IAM 使用者以您的 KMS 金鑰來複製加密資料庫叢集快照。

在以下金鑰政策範例中，使用者 `111122223333` 是 KMS 加密金鑰的擁有者，使用者 `444455556666` 是共用金鑰的帳戶。這個更新的金鑰政策將使用者 `444455556666` 之 AWS 根帳戶身分的 ARN 納入為政策的 `Principal`，並允許 `kms:CreateGrant` 動作，以授予 AWS 帳戶存取 KMS 金鑰的權利。

```
{
  "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}}
    }
  ]
}
```

建立 IAM 政策以允許複製加密快照

在外部 AWS 帳戶可以存取您的 KMS 金鑰之後，該帳戶的擁有者就可以建立政策，允許針對該帳戶建立 IAM 使用者，複製以該 KMS 金鑰加密的加密快照。

下列範例顯示可以連接到 AWS 帳戶 444455556666 之 IAM 使用者的政策。它可讓 IAM 使用者從 AWS 帳戶 111122223333 複製共用快照，此帳戶已在 us-west-2 區域中以 KMS 金鑰 c989c1dd-a3f2-4a5d-8d96-e793d082ab26 加密。

```
{
    "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
                }
            }
        }
    ]
}
```

如需有關更新金鑰政策的詳細資訊，請參閱 https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html 開發人員指南中的AWS Key Management Service金鑰政策。

共享資料庫叢集快照

您可以使用 AWS Management Console、AWS CLI 或 Neptune API 共用資料庫叢集快照。

使用主控台共享資料庫叢集快照

使用 Neptune 主控台，您可以將手動資料庫叢集快照與最多 20 個 AWS 帳戶共用。您也可以停止與一或多個帳戶共享手動快照。

共享手動資料庫叢集快照

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Snapshots (快照)。
3. 選擇您要共享的手動快照。
4. 選擇 Actions (動作) 和 Share Snapshot (共享快照)。
5. 在 DB snapshot visibility (資料庫快照可見度) 中，選擇下列其中一個選項。

   - 如果來源未加密，請選擇公有，來允許所有 AWS 帳戶從您的手動資料庫叢集快照還原資料庫叢集。或者選擇私有，只允許您指定的 AWS 帳戶從您的手動資料庫叢集快照還原資料庫叢集。

     > **Warning**
     >
     > 如果您將 DB Snapshot Visibility (資料庫快照可見度) 設為 Public (公有)，則所有 AWS 帳戶皆可從您的手動資料庫叢集快照還原資料庫叢集，而且可以存取您的資料。請勿將任何包含私人資訊的手動資料庫叢集快照以 Public (公有) 形式共用。

   - 如果來源資料庫叢集已加密，DB snapshot visibility (資料庫快照可見度) 會設為 Private (私有)，因為加密快照無法以公有形式共用。
6. 針對 AWS 帳戶 ID，輸入帳戶的 AWS 帳戶識別符，這是您想要允許從手動快照還原資料庫叢集的帳戶。接著選擇 Add (新增)。重複上一步來包含其他 AWS 帳戶識別符，最多 20 個 AWS 帳戶。

   如果您誤將 AWS 帳戶識別符新增至允許的帳戶清單，您可以選擇錯誤 AWS 帳戶識別符右邊的 Delete (刪除)，以從清單中刪除此識別符。
7. 在為您想要允許還原手動快照的所有 AWS 帳戶新增識別符之後，請選擇儲存。

如要停止與 AWS 帳戶共享手動資料庫叢集快照

1. 在 https://console.aws.amazon.com/neptune/home 開啟 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Snapshots (快照)。
3. 選擇您希望停止共享的手動快照。
4. 選擇 Actions (動作)，然後選擇 Share Snapshot (共享快照)。
5. 若要移除 AWS 帳戶的許可，請從授權帳戶清單中，對該帳戶的 AWS 帳戶識別符選擇 Delete (刪除)。
6. 選擇 Save (儲存)。

刪除 Neptune 快照。

您可以使用 AWS Management Console、AWS CLI 或 Neptune 管理 API 刪除資料庫快照：

使用主控台刪除

1. 登入 AWS 管理主控台，然後開啟位於 https://console.aws.amazon.com/neptune/home 的 Amazon Neptune 主控台。
2. 在導覽窗格中，選擇 Snapshots (快照)。
3. 選擇想要刪除的資料庫快照。
4. 針對 Actions (動作)，選擇 Delete Snapshot (刪除快照)。
5. 在確認頁面上，選擇 Delete (刪除)。

使用 AWS CLI 刪除

您也可以使用 AWS CLI delete_db_cluster_snapshot 命令，刪除資料庫快照，使用 `--db-snapshot-identifier` 參數以識別您要刪除的快照：

對於 Linux、OS X 或 Unix：

```
aws neptune delete-db-cluster-snapshot \
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

針對 Windows：

```
aws neptune delete-db-cluster-snapshot ^
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

使用 Neptune 管理 API 刪除

您可以使用其中一個 SDK，藉由呼叫 DeleteDBClusterSnapshot API 來刪除資料庫快照，並使用 `DBSnapshotIdentifier` 參數以識別要刪除的資料庫快照。

最佳實務：發揮 Neptune 的最大功用

以下是使用 Amazon Neptune 的一些一般建議。使用這裡的參考資訊，快速找到使用 Amazon Neptune 並將效能發揮到最高的建議。

內容

- 在容錯移轉之後建立新連線
- 將 maxInProcessPerConnection 和 maxSimultaneousUsagePerConnection 設定為相同值。
- 將查詢以位元碼形式而非字串形式傳送至伺服器
- 始終完全消耗查詢返回的 ResultSet 或迭代器
- 在批次中大量新增頂點和邊緣
- 在 Java 虛擬機器中停用 DNS 快取
- 或者，也可以在每個查詢層級設定逾時
- java.util.concurrent.TimeoutException 疑難排解

- 使用 openCypher 和 Bolt 的 Neptune 最佳實務
  - 偏好查詢中的定向至雙向邊緣
  - Neptune 不支援交易中的多個並行查詢
  - 在容錯移轉之後建立新連線
  - 適用於長期命應用的連線處理
  - 連線處理 AWS Lambda
  - 完成後關閉驅動程式物件
  - 使用明確的交易模式進行讀取和寫入
    - 唯讀交易
    - 唯讀交易
  - 例外狀況的重試邏輯
  - 使用單個 SET 子句一次設置多個屬性
  - 使用 SET 子句一次移除多個屬性
  - 使用參數化查詢
  - 在 RELUSE 子句中使用扁平化的映射而不是嵌套
  - 在可變長度路徑 (VLP) 運算式的左側放置更嚴格的節點
  - 使用細微關係名稱，避免多餘的節點標籤檢查
  - 盡可能指定邊標示
  - 盡可能避免使用 WITH 子句
  - 儘早在查詢中放置限制性篩選器
  - 明確檢查屬性是否存在
  - 不要使用命名路徑（除非它是必需的）

Amazon Neptune 基本操作準則

下列是使用 Neptune 時，您應該遵循的基本操作準則。

- 了解 Neptune 資料庫執行個體，以便您可以基於效能和使用案例需求適當調地整其大小。請參閱 Amazon Neptune 資料庫叢集和執行個體。
- 監控您的 CPU 和記憶體用量。這可協助您了解何時遷移到有更大 CPU 或記憶體容量的資料庫執行個體類別，以達到您需要的查詢效能。您可以設定 Amazon CloudWatch 在使用模式變更或接近部署容量時通知您。這樣做有助於維持系統效能和可用性。如需詳細資訊，請參閱監控執行個體和監控 Neptune。

  因為 Neptune 擁有自己的記憶體管理員，因此在 CPU 用量相當高時，記憶體用量仍相對較低的情況非常正常。在執行查詢時發生記憶體不足異常，便是您需要增加可釋放記憶體的最佳指示。
- 啟用自動備份並設定備份時段，在方便的時間進行備份。
- 測試資料庫執行個體的容錯移轉，以了解您的使用案例執行此程序需時多長。它也有助於確保可存取您資料庫執行個體的應用程式，可以在容錯移轉後，自動連線到新的資料庫執行個體。
- 如果可能，請在相同區域和 VPC 中執行用戶端與 Neptune 叢集，因為使用 VPC 對等互連進行跨區域連線可能會導致查詢回應時間的延遲。對於個位數毫秒的查詢回應，必須將用戶端和 Neptune 叢集保留在相同區域和 VPC 中。
- 當您建立僅供讀取複本執行個體時，它應該至少與主要寫入器執行個體一樣大。這有助於在檢查時保持複寫延遲，並避免複本重新啟動。請參閱 避免叢集中的不同執行個體類別。
- 在升級到新的主要引擎版本之前，請務必在其上測試您的應用程式，然後再升級。您可以透過複製資料庫叢集來執行此動作，以便複製叢集可執行新的引擎版本，然後在該複製上測試您的應用程式。

- 為了便於容錯移轉，所有執行個體最好都應該是相同的大小。

主題

Amazon Neptune 安全最佳實務

使用 AWS Identity and Access Management (IAM) 帳戶控制 Neptune API 動作的存取。控制建立、修改或刪除 Neptune 資源的動作 (例如資料庫執行個體、安全群組、選項群組或參數群組)，以及執行常見管理動作的動作 (例如備份和還原資料庫執行個體)。

- 盡可能使用臨時憑證而不是永久憑證。
- 將個別的 IAM 帳戶指派給管理 Amazon Relational Database Service (Amazon RDS) 資源的每個人員。永不使用 AWS 帳戶根使用者來管理 Neptune 資源。為每個人 (包含您) 建立 IAM 使用者。
- 授予每個使用者執行其職責所需的最低許可集。
- 使用 IAM 群組來有效管理多個使用者的許可。
- 定期輪替您的 IAM 登入資料。

如需使用 IAM 存取 Neptune 資源的詳細資訊，請參閱 Amazon Neptune 中的安全。如需有關使用 IAM 的一般資訊，請參閱《IAM 使用者指南》中的 AWS Identity and Access Management 和 IAM 最佳實務。

避免叢集中的不同執行個體類別

當您的資料庫叢集包含不同類別的執行個體時，可能會在一段時間後發生問題。最常見的問題是，由於複寫延遲，小型讀取器執行個體可能會進入重複重新啟動的週期。如果讀取器節點具有的資料庫執行個體類別組態比寫入器資料庫執行個體的組態還弱，則變更量可能太大，以致讀取器無法跟上。

> **Important**
>
> 若要避免複寫延遲所導致的重複重新啟動，請設定資料庫叢集，讓所有執行個體都具有相同的執行個體類別 (大小)。

您可以使用 Amazon CloudWatch 中的 `ClusterReplicaLag` 指標，查看資料庫叢集中寫入器執行個體 (主要執行個體) 與讀取器之間的延遲。`VolumeWriteIOPs` 指標也可讓您偵測叢集中可能會造成複寫延遲的大量寫入活動。

避免在大量載入期間重複重新啟動

如果您由於大量載入期間的複寫延遲而遭遇重複僅供讀取複本重新啟動的週期，則您的複本可能無法跟上資料庫叢集中的寫入器。

將讀取器擴展為大於寫入器，或在大量載入期間暫時將其移除，然後在完成後重新建立這些讀取器。

如果您有大量述詞，請啟用 OSGP 索引

如果您的資料模型包含大量不同的述詞 (在大多數情況下，超過一千個)，您可能會遭遇效能降低，而操作成本提高。

若是這種情況，您可以啟用 OSGP 索引來改善效能。請參閱 OSGP 索引。

盡可能避免長時間執行的交易

長時間執行的交易 (唯讀或讀寫) 可能會造成下列種類的非預期問題：

在讀取器執行個體或具有並行寫入的寫入器執行個體上，長時間執行的交易可能會導致不同版本的資料大量累積。這可能會對讀取查詢引入更高的延遲，從而篩選掉其大部分的結果。

在某些情況下，數小時的累積版本可能會導致新的寫入遭到限流。

如果執行個體重新啟動，具有許多寫入的長時間執行的讀寫交易也可能會造成問題。如果執行個體從維護事件或當機重新啟動，則所有未遞交的寫入都會加以復原。這類復原操作通常會在背景執行，而且不會封鎖執行個體回復，但是任何與要回復之操作衝突的新寫入都會失敗。

例如，如果在上次執行時連線中斷之後重試相同的查詢，則在重新啟動執行個體時可能會失敗。

復原操作所需的時間與涉及的變更大小成正比。

使用 Neptune 指標的最佳實務

若要識別由於資源不足和其他常見瓶頸造成的效能問題，您可以監控 Neptune 資料庫叢集的可用指標。

定期監控效能指標以蒐集各種時間範圍的平均值、最大值和最小值。效能降級時您便可以得知。使用此資料，您可以為特定指標閾值設定 Amazon CloudWatch 警示，讓您可以在到達閾值時收到提醒。

設定新的資料庫叢集並讓它以一般工作負載執行時，嘗試擷取不同間隔數量 (例如，一小時、24 小時、一週、兩週) 所有效能指標的平均值、最大值和最小值。這可讓您了解正常運作情況為何。這有助於比較尖峰與離峰時段的操作。您接著可以使用此資訊來識別效能下降到標準層級之下的時機，並設定警示來應對。

請參閱 使用 Amazon 監控 Neptune CloudWatch，以取得如何檢視 Neptune 指標的相關資訊。

以下是您開始的最重要指標：

- BufferCacheHitRatio - 由緩衝區快取提供服務的請求百分比。快取遺漏會將顯著延遲新增至查詢執行。如果快取命中率低於 99.9%，而延遲對應用程式造成問題，請考慮升級執行個體類型，以在記憶體中快取更多資料。
- CPU 使用率 - 已使用的電腦處理容量百分比。取決於您的查詢效能目標，較高的 CPU 使用量值有可能是適當的。
- 可釋放的記憶體 – 資料庫執行個體可用的 RAM (以 MB 為單位)。Neptune 有自己的記憶體管理員，因此這個指標可能低於您預期的。指出您應考慮將執行個體類別升級至擁有較多 RAM 類別的一項良好指標，就是若查詢時常拋出記憶體不足的異常時。

Monitoring (監控) 標籤指標的 CPU 及記憶體指標在 75% 會以紅線標記。如果執行個體記憶體的使用量經常超過該線，請檢查您的工作負載，並考慮將您的執行個體升級以改善查詢效能。

調校 Neptune 查詢的最佳實務

其中一種改善 Neptune 效能的最佳方法，便是調校您最常使用及最資源密集的查詢，降低執行它們的成本。

如需如何調校 Gremlin 查詢的資訊，請參閱 Gremlin 查詢提示 和 調校 Gremlin 查詢。如需如何調校 SPARQL 查詢的資訊，請參閱 SPARQL 查詢提示。

在各個僅供讀取複本之間平衡負載

讀取器端點循環配置資源路由的運作方式是改變 DNS 項目指向的主機。用戶端必須建立新連線並解析 DNS 記錄，以取得與僅供讀取新複本的連線，因為 WebSocket 連線通常會長時間保持活動狀態。

若要針對連續請求取得不同的僅供讀取複本，請確保用戶端在每次連線時都能解析 DNS 項目。這可能需要關閉連線並重新連接到讀取者端點。

您也可以明確連接到執行個體端點，在各個僅供讀取複本之間對請求進行負載平衡。

使用臨時的較大執行個體，載入速度更快

使用較大的執行個體可提高您的載入效能。如果您不是使用大型的執行個體類型，但您想要更高的載入速度，可以使用較大的執行個體來進行載入，然後將其刪除。

> **Note**
>
> 下列程序是用於新的叢集。如果您有現有的叢集，您可以新增較大的執行個體，然後將其提升至主要資料庫執行個體。

使用較大的執行個體大小將資料載入

1. 建立具有單一 `r5.12xlarge` 執行個體的叢集。這個執行個體是主要資料庫執行個體。
2. 建立大小 (`r5.12xlarge`) 相同的一個或多個僅供讀取複本。

   您可以建立更小的僅供讀取複本，但如果其大小不足以跟上主執行個體的寫入，則可能必須經常重新啟動。產生的停機時間會大幅降低效能。
3. 在大量載入器命令中，包括 `“parallelism” : “OVERSUBSCRIBE”` 以告訴 Neptune 使用所有可用的 CPU 資源進行載入 (請參閱 Neptune 載入器請求參數)。然後，載入操作將以 I/O 允許的一樣快速度進行，這通常需要 60-70% 的 CPU 資源。

4. 使用 Neptune 載入器載入資料。載入工作會在主要資料庫執行個體上執行。
5. 在完成資料載入之後，請務必將叢集中的所有執行個體縮減至相同的執行個體類型，以避免額外的費用和重複的重新啟動問題 (請參閱 避免不同的執行個體大小)。

容錯移轉至僅供讀取複本來調整您的寫入器執行個體大小

調整資料庫叢集中執行個體大小 (包括寫入器執行個體) 的最佳方法，就是建立或修改僅供讀取複本執行個體，以便其具有您所需的大小，然後刻意容錯移轉至該僅供讀取複本。您的應用程式看到的停機時間只是變更寫入器 IP 地址所需的時間，應該是 3 到 5 秒左右。

您用來刻意將目前寫入器執行個體容錯移轉至僅供讀取複本執行個體的 Neptune 管理 API 為 FailoverDBCluster。如果您是使用 Gremlin Java 用戶端，則可能需要在容錯移轉之後建立新的用戶端物件，以挑選新的 IP 地址，如這裡所述。

確定將所有執行個體變更為相同的大小，讓您避免重複重新啟動的週期，如下所述。

在資料預先擷取任務中斷錯誤之後重試上傳

當您使用大量載入器將資料載入 Neptune 時，結果有時會出現 `LOAD_FAILED` 狀態，而回應中會報告 `PARSING_ERROR` 和 `Data prefetch task interrupted` 訊息以要求詳細資訊，如下所示：

```
"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
  }
]
```

當此錯誤發生時，只要重試一次大量上傳請求。

當出現通常原因不在於您的請求或資料的暫時中斷時，就會出現這個錯誤，而且通常再次執行大量上傳請求就能加以解決。

如果您使用的是預設設定，即 `"mode":"AUTO"` 和 `"failOnError":"TRUE"`，這時載入器會略過其已成功載入的檔案，並恢復其在中斷發生時尚未載入的檔案載入。

使用 Gremlin 搭配 Neptune 的一般最佳實務

使用 Gremlin 圖形周遊語言搭配 Neptune 時，請遵循這些建議。如需使用 Gremlin 搭配 Neptune 的詳細資訊，請參閱 the section called “Gremlin”。

Important

已在 TinkerPop 3.4.11 版中進行變更，這可提高查詢處理方式的正確性，但目前有時可能會嚴重影響查詢效能。
例如，此類查詢的執行速度可能會明顯變慢：

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

由於 TinkerPop 3.4.11 變更，限制步驟之後的頂點現在會以非最佳方式擷取。若要避免這種情況，您可以在 `order().by()` 之後的任何點新增 barrier() 步驟來修改查詢。例如：

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 已在 Neptune 引擎 1.0.5.0 版中啟用。

主題

- 在您將部署 Girmlin 程式碼的內容中測試該程式碼
- 建造 upsert 查詢以利用 DFE 引擎
- 建立有效率的多執行緒 Gremlin 寫入
- 利用建立時間屬性清除記錄
- 對 Groovy 時間資料使用 datetime( ) 方法
- 在 GLV 時間資料使用原生日期和時間

在您將部署 Girmlin 程式碼的內容中測試該程式碼

在 Gemlin 中，有多種方式可讓用戶端將查詢提交至伺服器：使用 WebSocket 或位元碼 GLV，或透過使用字串型指令碼的 Gemlin 控制台。

務必認識到 Gemlin 查詢執行可能會有所不同，取決於您提交查詢的方式。如果在位元碼模式下提交，則傳回空結果的查詢可能會被視為成功，但如果在指令碼模式下提交，則可能會被視為失敗。例如，如果您在指令碼模式查詢中包含 `next()`，則 `next()` 會傳送至伺服器，但是用戶端通常會使用位元碼來處理 `next()` 本身。在第一種情況下，如果找不到任何結果，查詢就會失敗，但在第二種情況下，無論結果集是否為空，查詢都會成功。

如果您在某個內容中開發並測試您的程式碼 (例如，通常以文字形式提交查詢的 Gemlin 控制台)，但是接著在不同的內容中部署程式碼 (例如，透過使用位元碼的 Java 驅動程式)，則您可能會遇到下列問題：程式碼在生產環境中的行為與在開發環境中的行為不同。

> **Important**
>
> 務必在將要部署 Gremlin 程式碼的 GLV 內容中測試該程式碼，以避免發生非預期的結果。

建造 upsert 查詢以利用 DFE 引擎

您可以盡可能充分利用 Neptune DFE 引擎，來大幅改善 upsert 查詢的效能。

使用 Gremlin `mergeV()` 和 `mergeE()` 步驟進行有效的 upsert 說明如何建造 upsert 查詢，以盡可能有效地使用 DFE 引擎。

建立有效率的多執行緒 Gremlin 寫入

使用 Gremlin 以多執行緒的方式將資料載入 Neptune，有幾項準則。

如有可能，請提供每個執行緒一組頂點或邊緣，以插入或修改不衝突的項目。例如，執行緒 1 的地址 ID 範圍為 1–50,000，執行緒 2 地址 ID 範圍為 50,001–100,000，以此類推。這可減少出現 `ConcurrentModificationException` 的機率。為安全起見，所有寫入請圍以 `try/catch` 區塊。若任何一個項目失敗，您便可以在短暫的延遲後重試它們。

以介於 50 和 100 (頂點或邊緣) 間的批次大小批次寫入，通常效果很好。若您要為每個頂點新增許多屬性，接近 50 的數字可能比接近 100 的數字更好。這很值得進行一些實驗。因此針對批次寫入，您可以使用如下的內容：

```
g.addV('test').property(id,'1').as('a').
  addV('test').property(id,'2').
  addE('friend').to('a').
```

接著會在每一個批次操作中重複這段過程。

使用批次的效率，遠比在每一次 Gremlin 與伺服器的來回行程中新增一個頂點或邊緣高。

如果您使用的是 Gremlin 語言變體 (GLV) 用戶端，您可以先建立一個周遊，再以程式設計方式建立一個批次。然後，為它新增內容，最後再對它反覆運算，例如：

```
  t.addV('test').property(id,'1').as('a')
  t.addV('test').property(id,'2')
  t.addE('friend').to('a')
  t.iterate()
```

如果可能，最好使用 Gremlin 語言變體用戶端。但是，您可以串連字串來建立批次，用將查詢提交為文字字串的用戶端執行類似的作業。

若您使用其中一個 Gremlin 用戶端程式庫，而非基本的 HTTP 來進行查詢，則執行緒應該全部都會共享相同的用戶端、叢集或連線集區。您可能需要調校設定，來盡可能取得最佳的輸送量，像是 Gremlin 用戶端所使用的連線集區大小及工作者執行緒數量等設定。

利用建立時間屬性清除記錄

您可以把建立時間儲存為頂點上的屬性，然後定期丟棄它們，藉此清除過時記錄。

如果需要讓資料存放一段特定時間，然後從圖形中移除 (頂點存留時間)，您可以在建立頂點時儲存時間戳記屬性。然後，您可以定期對所有在特定時間前建立的頂點發出 `drop()` 查詢，例如：

```
g.V().has("timestamp", lt(datetime('2018-10-11')))
```

對 Groovy 時間資料使用 **`datetime( )`** 方法

Neptune 會提供 `datetime` 方法，為 Gremlin Groovy 變體中傳送的查詢指定日期和時間。這包括 Gremlin 主控台、使用 HTTP REST API 的文字字串，以及任何其他使用 Groovy 的序列化。

> **Important**
>
> 這「只」適用於以「文字字串」傳送 Gremlin 查詢的方法。如果使用的是 Gremlin 語言變體，您必須使用該語言的原生日期類別與函數。如需詳細資訊，請參閱下一節「the section called "原生日期與時間"」。
> 從 TinkerPop `3.5.2` (在 Neptune 引擎 1.1.1.0 版 中引入) 開始，`datetime` 是 TinkerPop 不可或缺的一部分。

您可以使用 `datetime` 方法來存放和比較日期：

```
g.V('3').property('date',datetime('2001-02-08'))
```

```
g.V().has('date',gt(datetime('2000-01-01')))
```

在 GLV 時間資料使用原生日期和時間

如果您是使用 Gremlin 語言變體 (GLV)，必須使用該程式語言為 Gremlin 時間資料提供的原生日期和時間類別與函數。

官方 TinkerPop Java、Node.js (JavaScript)、Python 或 .NET 程式庫是所有的 Gremlin 語言變體程式庫。

> **Important**
>
> 這「只」適用於「Gremlin 語言變體 (GLV) 程式庫」。如果您是使用將 Gremlin 查詢以文字字串傳送的方法，則必須使用 Neptune 提供的 `datetime()` 方法。這包括 Gremlin 主控台、使用 HTTP REST API 的文字字串，以及任何其他使用 Groovy 的序列化。如需詳細資訊，請參閱上一節the section called "`datetime( )`"。

Python

以下是 Python 範例的一部分，它會為頂點建立一個名為 'date' 且 ID 為 '3' 的屬性。它將值設為使用 Python `datetime.now()` 方法產生的日期。

```
import datetime
```

```
g.V('3').property('date',datetime.datetime.now()).next()
```

如需使用 Python 連線到 Neptune 的完整範例，請參閱 使用 Python 連線至 Neptune 資料庫執行個體。

Node.js (JavaScript)

以下是 JavaScript 範例的一部分，它會為頂點建立一個名為 'date' 且 ID 為 '3' 的屬性。它將值設為使用 Node.js `Date()` 建構函數產生的日期。

```
g.V('3').property('date', new Date()).next()
```

如需使用 Node.js 連線到 Neptune 的完整範例，請參閱 使用 Node.js 連線至 Neptune 資料庫執行個體。

Java

以下是 Java 範例的一部分，它會為頂點建立一個名為 'date' 且 ID 為 '3' 的屬性。它將值設為使用 Java `Date()` 建構函數產生的日期。

```
import java.util.date

g.V('3').property('date', new Date()).next();
```

如需使用 Java 連線到 Neptune 的完整範例，請參閱 使用 Java 連線至 Neptune 資料庫執行個體。

.NET (C#)

以下是 C# 範例的一部分，它會為頂點建立一個名為 'date' 且 ID 為 '3' 的屬性。它將值設為使用 .NET *`DateTime.UtcNow`* 屬性產生的日期。

```
Using System;

g.V('3').property('date', DateTime.UtcNow).next()
```

如需使用 C# 連線到 Neptune 的完整範例，請參閱 使用 .NET 連線至 Neptune 資料庫執行個體。

使用 Gremlin Java 用戶端搭配 Neptune 的最佳實務

請使用最新版本的 Apache TinkerPop Java 客戶端

如果可以的話，請始終使用您正在使用的引擎版本所支持的最新版本的 Apache TinkerPop Gremlin Java 客戶端。較新的版本包含許多錯誤修正，其可以提高用戶端的穩定性、效能和可用性。

如需與各種 Neptune 引擎版本相容的用戶端版本清單，請參閱 阿帕奇 TinkerPop Java 小鬼客戶端。

跨多個執行緒重複使用用戶端物件

跨多個執行緒重複使用相同的用戶端 (或 GraphTraversalSource) 物件。也就是說，在您的應用程式中建立 org.apache.tinkerpop.gremlin.driver.Client 類別的共用執行個體，而不是在每個執行緒中建立。Client 物件是安全執行緒，初始化執行緒的額外負荷很大。

這也適用於 GraphTraversalSource，這會在內部建立 Client 物件。例如，以下程式碼會執行個體化新的 Client 物件：

```
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;

  /////

GraphTraversalSource traversal = traversal()
                                    .withRemote(DriverRemoteConnection.using(cluster));
```

為讀取和寫入端點建立個別的 Gremlin Java 用戶端物件

只在寫入器端點執行寫入，以及只從一個或多個唯讀端點讀取，可以提高效能。

```
Client readerClient = Cluster.build("https://reader-endpoint")
          ...
          .connect()

Client writerClient = Cluster.build("https://writer-endpoint")
          ...
          .connect()
```

將多個僅供讀取複本端點新增至 Gremlin Java 連線集區

建立 Gremlin Java `Cluster` 物件時，可以使用 `.addContactPoint()` 方法將多個僅供讀取複本執行個體新增到連線集區的接觸點。

```
Cluster.Builder readerBuilder = Cluster.build()
          .port(8182)
          .minConnectionPoolSize(…)
          .maxConnectionPoolSize(…)
          ……
          .addContactPoint("reader-endpoint-1")
          .addContactPoint("reader-endpoint-2")
```

關閉用戶端以避免連線限制

完成用戶端後，請務必關閉用戶端，以確保伺服器關閉 WebSocket 連線，並釋放與連線相關聯的所有資源。如果您使用 `Cluster.close( )` 關閉叢集，此程序會自動發生，因為會隨後在內部呼叫 `client.close( )`。

如果用戶端未正確關閉，Neptune 會在 20 到 25 分鐘後終止所有閒置 WebSocket連線。不過，如果您在完成 WebSocket連線時並未明確關閉連線，且即時連線數目達到WebSocket 並行連線限制，則會拒絕其他連線，並顯示 HTTP 429 錯誤碼。此時，您必須重新啟動 Neptune 執行個體才能關閉連線。

呼叫 `cluster.close()` 的建議不適用於 Java AWS Lambda 函數。如需詳細資訊，請參閱 管理 AWS Lambda 函數中的 Gremlin WebSocket 連線。

在容錯移轉之後建立新連線

如果發生容錯移轉，因為叢集 DNS 名稱已解析為 IP 地址，所以 Gremlin 驅動程式可能會繼續連線到舊的寫入器。如果發生這種情況，您可以在容錯移轉後建立新的 `Client` 物件。

將 `maxInProcessPerConnection` 和 `maxSimultaneousUsagePerConnection` 設定為相同值。

`maxInProcessPerConnection`和`maxSimultaneousUsagePerConnection`參數都與您可以在單一連線上提交的最大同時查詢數量有 WebSocket關。在內部，這些參數是相互關聯的，修改一個參數而不修改另一個，可能導致用戶端在嘗試從用戶端連線集區擷取連線時收到逾時。

我們建議保持預設的最小處理中和同時使用量值，並將 `maxInProcessPerConnection` 和 `maxSimultaneousUsagePerConnection` 設定為相同的值。

設定這些參數的值是查詢複雜性和資料模型的函數。在傳回許多資料的查詢使用案例中，每個查詢需要更多連線頻寬，因此，參數應該具有更低的值，而 maxConnectionPoolSize 應該具有更高的值。

相反地，如果查詢傳回較少的資料量，則 maxInProcessPerConnection 和 maxSimultaneousUsagePerConnection 應設定為大於 maxConnectionPoolSize 的值。

將查詢以位元碼形式而非字串形式傳送至伺服器

在以下情況提交查詢時，使用位元碼來取代字串具有優勢：

- 早期擷取無效的查詢語法：使用位元碼變體可讓您在編譯階段偵測無效的查詢語法。如果您使用字串變體，直到查詢送至伺服器之前都無法發現無效的語法，此時會傳回錯誤。
- 避免基於字符串的性能損失：任何基於字符串的查詢提交（無論您使用 WebSockets 還是 HTTP）都會導致分離的頂點，這意味著 Vertex 對象包含 ID，Label 以及與頂點關聯的所有屬性（請參閱元素的屬性）。

  在不需要屬性的情況下，這可能會導致伺服器上不必要的運算。例如，如果客戶想要使用查詢 g.V("hakuna#1") 取得 ID 為「hakuna#1」的頂點。如果查詢以字串提交傳送出去，伺服器需花時間擷取 ID、標籤以及與此頂點關聯的所有屬性。如果查詢以位元碼提交傳送出去，伺服器只需花費擷取頂點 ID 和標籤的時間。

換言之，與其提交這種查詢：

```
 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();
 }
```

不如使用位元碼提交查詢，如下所示：

```
 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();
 }
```

始終完全消耗查詢返回的 ResultSet 或迭代器

用戶端物件應一律完全耗用 `ResultSet` (若是字串提交) 或 `GraphTraversal` 傳回的反覆運算器。如果查詢結果未完全耗用，伺服器保留它們，直到用戶端完成耗用它們為止。

如果您的應用程式只需要一部分結果，您可以使用 `limit(X)` 步驟加上您的查詢，來限制伺服器產生的結果數量。

在批次中大量新增頂點和邊緣

每個對 Neptune 資料庫的查詢都在單一交易的範圍中執行，除非您使用工作階段。這表示如果您需要使用 gremlin 查詢插入許多資料，使用 50-100 的批次大小來同時批次處理，可透過降低建立的負載交易量來改善效能。

舉例來說，新增 5 個頂點到資料庫會如下所示：

```
// 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();
```

在 Java 虛擬機器中停用 DNS 快取

在想要負載平衡多個僅供讀取複本請求的環境中，您需要停用 Java 虛擬機器 (JVM) 中的 DNS 快取並在建立叢集時提供 Neptune 的讀取器端點。停用 JVM DNS 快取可確保每次新連線都會再次解析 DNS，如此一來，就會將請求分配到所有僅供讀取複本。您可以在應用程式的初始化程式碼中使用下列一行執行此動作：

```
java.security.Security.setProperty("networkaddress.cache.ttl", "0");
```

但是，Amazon Gemlin Java 客戶端代碼提供了一個更完整和強大的負載平衡解決方案。GitHubAmazon Java Gemlin 用戶端知道您的叢集拓撲，並將連線和請求公平分配到 Neptune 叢集中的一組執行個體。如需使用該用戶端的 Java Lambda 函數範例，請參閱此部落格文章。

或者，也可以在每個查詢層級設定逾時

Neptune 可讓您使用參數群組選項 `neptune_query_timeout` 來設定查詢的逾時 (請參閱 參數)。不過從 Java 用戶端版本 3.3.7 開始，您也可以使用類似下列的程式碼，覆寫全域逾時：

```
  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();
  }
```

或者，對於字串查詢提交，程式碼將如下所示：

```
  RequestOptions options = RequestOptions.build().timeout(500).create();
```

```
List<Result> result = client.submit("g.V()", options).all().get();
```

> **Note**
>
> 如果您將查詢逾時值設得太高，特別是在無伺服器執行個體上，可能會產生非預期的成本。若沒有合理的逾時設定，您的查詢執行時間可能會比預期的長得多，進而產生您從未預期的成本。這在無伺服器執行個體上尤是如此，因為該執行個體在執行查詢時可能會縱向擴展為大型且昂貴的執行個體類型。
>
> 您可以使用符合您預期之執行階段的查詢逾時值，避免此類非預期的費用，而且只會導致異常的長時間執行逾時。

java.util.concurrent.TimeoutException 疑難排解

Gemlin Java 客戶端在等待其中一個 WebSocket 連接中的插槽變為可用`java.util.concurrent.TimeoutException`時，Gimlin Java 客戶端會拋出一個 Gimlin 請求超時。此逾時持續時間是由 `maxWaitForConnection` 用戶端可設定參數控制。

> **Note**
>
> 因為在用戶端逾時的請求永遠都不會傳送至伺服器，所以它們不會反映在伺服器中擷取的任何指標中，例如 `GremlinRequestsPerSec`。

這種逾時通常是由下列兩種方式之一引起：

- 伺服器實際上已達到容量上限。如果是這種情況，伺服器上的佇列就會填滿，您可以藉由監督「MainRequestQueuePending要求」測量結果來偵 CloudWatch 測。伺服器可以處理的平行查詢數目取決於其執行個體大小。

  如果 `MainRequestQueuePendingRequests` 指標未在伺服器上顯示待定請求的累積，則伺服器可以處理更多請求，而且逾時是由用戶端限流造成的。
- 請求的用戶端限流。通常可以透過變更用戶端組態設定來修正此問題。

  用戶端可以傳送的平行請求數目上限，可以大致估計如下：

```
maxParallelQueries = maxConnectionPoolSize * Max( maxSimultaneousUsagePerConnection,
 maxInProcessPerConnection )
```

傳送至用戶端的數目若超過 maxParallelQueries，會導致 java.util.concurrent.TimeoutException 例外狀況。您通常可採取幾種方式來解決此問題：

- 增加連線逾時持續時間。如果延遲對您的應用程式而言並不重要，請增加用戶端的 maxWaitForConnection 設定。然後，用戶端會等待更長的時間才逾時，因而增加延遲。
- 增加每個連線的請求數上限。這允許使用相同的 WebSocket 連接發送更多請求。透過增加用戶端的 maxSimultaneousUsagePerConnection 和 maxInProcessPerConnection 設定來執行此動作。這些設定通常應該具有相同的值。
- 增加連線集區中的連線數目。透過增加用戶端的 maxConnectionPoolSize 設定來執行此動作。成本會增加資源消耗，因為每個連線都使用記憶體和作業系統檔案描述元，並且在初始化期間需要 SSL 和 WebSocket交握。

使用 openCypher 和 Bolt 的 Neptune 最佳實務

搭配 Neptune 使用 openCypher 查詢語言和 Bolt 通訊協定時，請遵循這些最佳實務。如需在 Neptune 使用 openCypher 的相關資訊，請參閱 使用 openCypher 存取 Neptune 圖形。

偏好查詢中的定向至雙向邊緣

Neptune 執行查詢最佳化時，雙向邊緣會使建立最佳化查詢計劃變得困難。次佳計劃要求引擎執行不必要的工作並導致更差的效能。

因此，盡可能使用定向邊緣而不是雙向邊緣。例如，使用：

```
MATCH p=(:airport {code: 'ANC'})-[:route]->(d) RETURN p)
```

而不是：

```
MATCH p=(:airport {code: 'ANC'})-[:route]-(d) RETURN p)
```

大多數資料模型實際上不需要周遊兩個方向的邊緣，因此查詢可以透過切換到使用定向邊緣來達到大幅的效能改進。

如果您的資料模型確實需要周遊雙向邊緣，則使 MATCH 模式中的第一個節點 (左側) 成為篩選最嚴格的節點。

舉個例子，「為我找到所有往返 ANC 機場的 routes」。編寫此查詢，從 ANC 機場開始，如下所示：

```
MATCH p=(src:airport {code: 'ANC'})-[:route]-(d) RETURN p
```

引擎可以執行最少的工作量來滿足查詢，因為受到最多限制的節點會放置在模式中作為第一個節點 (左側)。然後，引擎可以最佳化查詢。

這比在模式結束時篩選 ANC 機場更好，如下所示：

```
MATCH p=(d)-[:route]-(src:airport {code: 'ANC'}) RETURN p
```

當受到最多限制的節點未放在模式中的第一位時，引擎必須執行額外的工作，因為它無法最佳化查詢，而且必須執行其他查詢才能得出結果。

Neptune 不支援交易中的多個並行查詢

雖然 Bolt 驅動程式本身允許交易中的並行查詢，但 Neptune 不支援交易中的多個查詢同時執行。相反，Neptune 會要求交易中的多個查詢循序執行，並且在啟動下一個查詢之前，完全耗用每個查詢的結果。

以下範例說明如何使用 Bolt 在交易中循序執行多個查詢，以便在下一個查詢開始之前完全耗用每個查詢的結果：

```
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());
    }
  }
}
```

在容錯移轉之後建立新連線

若發生容錯移轉，Bolt 驅動程式可以繼續連線至舊的寫入器執行個體，而不是新的作用中執行個體，因為 DNS 名稱已解析為特定 IP 地址。

若要避免這種情況發生，請在任何容錯移轉之後關閉 `Driver` 物件，然後重新連線該物件。

適用於長期命應用的連線處理

建置長期應用程式 (例如在容器內或在 Amazon EC2 執行個體上執行的應用程式) 時，請將 `Driver` 物件執行個體化一次，然後在應用程式的生命週期內重複使用該物件。`Driver` 物件是安全執行緒，初始化執行緒的額外負荷很大。

連線處理 AWS Lambda

由於螺栓驅動器的連接開銷和管理需求，因此不建議在 AWS Lambda 功能中使用螺栓驅動器。請改用 HTTPS 端點。

完成後關閉驅動程式物件

完成後請務必關閉用戶端，以便伺服器關閉 Bolt 連線，以及釋放與連線相關聯的所有資源。如果您使用 `driver.close()` 關閉驅動程式，則此情況會自動發生。

如果驅動程式未正確關閉，Neptune 會在 20 分鐘後終止所有閒置的 Bolt 連線，或者如果您使用 IAM 驗證，則會在 10 天後終止。

Neptune 支援不超過 1000 個並行 Bolt 連線。如果您在完成後未明確關閉連線，且即時連線數目達到 1000 限制，則任何新的連線嘗試都會失敗。

使用明確的交易模式進行讀取和寫入

使用交易搭配 Neptune 和 Bolt 驅動程式時，最好將讀取和寫入交易的存取模式明確設定為正確的設定。

唯讀交易

對於唯讀交易，如果您在建置工作階段時未傳入適當的存取模式組態，則會使用預設隔離層級，也就是變動查詢隔離。因此，對於唯讀交易來說，將存取模式明確設定為 `read` 很重要。

自動遞交讀取交易範例：

```
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()
}
```

讀取交易範例：

```
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)
    }
  }
);
```

在這兩種情況下，SNAPSHOT 隔離都是使用 Neptune 唯讀交易語義來達成的。

因為僅供讀取複本只接受唯讀查詢，所以提交至僅供讀取複本的任何查詢都會在 SNAPSHOT 隔離語義下執行。

唯讀交易沒有已變更讀取或不可重複讀取。

唯讀交易

對於變動查詢，有三種不同的機制來建立寫入交易，每個方法描述如下：

隱含寫入交易範例：

```
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)
    }
  }
);
```

自動遞交寫入交易範例：

```
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()
}
```

明確寫入交易範例：

```
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();
```

寫入交易的隔離層級

- 作為變動查詢一部分進行的讀取會在 READ COMMITTED 交易隔離下執行。
- 作為變動查詢一部分進行的讀取沒有任何已讀取讀取。
- 在變動查詢中讀取時，會鎖定記錄和記錄範圍。
- 當變動交易讀取某個索引範圍時，強力保證在讀取結束之前，任何並行交易都不會修改此範圍。

變動查詢不是執行緒安全的。

如需衝突，請參閱 使用鎖定等待逾時的衝突解決機制。

變動查詢不會在失敗的情況下自動重試。

例外狀況的重試邏輯

對於允許重試的所有例外狀況，通常最好使用指數退避和重試策略，該策略會在重試之間提供逐漸加長的等待時間，以便更好地處理暫時性問題，例如 ConcurrentModificationException 錯誤。下面顯示指數退避和重試模式的範例：

```
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);
  }
}
```

使用單個 SET 子句一次設置多個屬性

而不是使用多個 SET 子句來設定個別屬性，而是使用對應一次為實體設定多個屬性。

您可以使用：

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n += {property1 : 'value1',
property2 : 'value2',
property3 = 'value3'}
```

而不是：

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n.property1 = 'value1'
SET n.property2 = 'value2'
SET n.property3 = 'value3'
```

SET 子句接受單個屬性或映射。如果在單個實體上更新多個屬性，則使用帶有 map 的單個 SET 子句允許在單個操作中執行更新，而不是多個操作，這些操作可以更有效地執行。

使用 SET 子句一次移除多個屬性

當使用 OpenCypher 語言，刪除用於從實體中刪除屬性。在 Neptune 中，要移除的每個內容都需要個別作業，以增加查詢延遲。您可以改為使用 SET 與地圖將所有屬性值設置為null，這在 Neptune 中相當於刪除屬性。當需要移除單一實體上的多個屬性時，Neptune 將會提升效能。

使用：

```
WITH {prop1: null, prop2: null, prop3: null} as propertiesToRemove
MATCH (n)
SET n += propertiesToRemove
```

而不是：

```
MATCH (n)
REMOVE n.prop1, n.prop2, n.prop3
```

使用參數化查詢

建議在使用 OpenCypher 進行查詢時始終使用參數化查詢。查詢引擎可以利用重複的參數化查詢功能，例如查詢計畫快取，其中重複叫用具有不同參數的相同參數化結構可利用快取的計畫。只有在 100 毫秒內完成且參數類型為「數字」、「布林值」或「STRING」時，才會快取並重複使用針對參數化查詢產生的查詢計畫。

使用：

```
MATCH (n:foo) WHERE id(n) = $id RETURN n
```

使用參數：

```
parameters={"id": "first"}
parameters={"id": "second"}
parameters={"id": "third"}
```

而不是：

```
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
```

在 RELUSE 子句中使用扁平化的映射而不是嵌套

深層巢狀結構可以限制查詢引擎產生最佳查詢計畫的能力。為了部分緩解此問題，下列定義的模式將會針對下列案例建立最佳計劃：

- 場景 1：使用密碼文字列表展開，其中包括數字，字符串和布爾值。
- 場景 2：使用扁平化映射列表展開，其中僅包含密碼文字（數字，字符串，布爾值）作為值。

撰寫包含 RELUSH 子句的查詢時，請使用上述建議來改善效能。

情況 1 示例：

```
UNWIND $ids as x
MATCH(t:ticket {`~id`: x})
```

使用參數：

```
parameters={
  "ids": [1, 2, 3]
}
```

案例 2 的範例是產生要建立或合併的節點清單。請使用下列模式將屬性定義為一組扁平化對應，而不是發出多個陳述式：

```
UNWIND $props as p
CREATE(t:ticket {title: p.title, severity:p.severity})
```

使用參數：

```
parameters={
  "props": [
    {"title": "food poisoning", "severity": "2"},
    {"title": "Simone is in office", "severity": "3"}
  ]
}
```

而不是嵌套的節點對象，如：

```
UNWIND $nodes as n
```

```
CREATE(t:ticket n.properties)
```

使用參數：

```
parameters={
  "nodes": [
    {"id": "ticket1", "properties": {"title": "food poisoning", "severity": "2"}},
    {"id": "ticket2", "properties": {"title": "Simone is in office", "severity": "3"}}
  ]
}
```

在可變長度路徑 (VLP) 運算式的左側放置更嚴格的節點

在變數長度路徑 (VLP) 查詢中，查詢引擎會選擇在運算式的左側或右側啟動周遊，藉此將評估最佳化。該決定基於左側和右側的圖案的基數。基數是指定模式匹配的節點的數量。

- 如果右圖案的基數為 1，則右側將是起點。
- 如果左側和右側的基數為 1，則會在兩側檢查膨脹，並以較小的膨脹從側面開始。擴充是指左側節點和 VLP 運算式右側節點的傳出或傳入邊緣數目。只有在 VLP 關係是單向且提供關係類型時，才會使用此部分最佳化。
- 否則，左側將是起點。

對於 VLP 運算式鏈結，此最佳化只能套用至第一個運算式。其他 VLP 會從左側開始進行評估。舉例來說，讓 (a)、(b) 的基數為一，而 (c) 的基數大於 1。

- `(a)-[*1..]->(c)`: 評估從 (a) 開始。
- `(c)-[*1..]->(a)`: 評估從 (a) 開始。
- `(a)-[*1..]-(c)`: 評估從 (a) 開始。
- `(c)-[*1..]-(a)`: 評估從 (a) 開始。

現在讓 (a) 的進入邊為兩個，(a) 的外出邊為三條，(b) 的進入邊為四邊，(b) 的外出邊為五條。

- `(a)-[*1..]->(b)`: 評估以 (a) 開始，因為 (a) 的外出邊緣小於 (b) 的入料邊緣。
- `(a)<-[*1..]-(b)`: 評估以 (a) 開始，因為 (a) 的進入邊緣小於 (b) 的外出邊緣。

一般而言，請將更嚴格的模式置於 VLP 運算式的左側。

使用細微關係名稱，避免多餘的節點標籤檢查

針對效能進行最佳化時，使用節點模式專用的關係標籤可移除節點上的標籤篩選。考慮一個圖形模型，其中的關係likes僅用於定義兩個person節點之間的關係。我們可以寫下面的查詢來找到這個模式：

```
MATCH (n:person)-[:likes]->(m:person)
RETURN n, m
```

n 和 m 上的person標籤檢查是多餘的，因為我們定義了僅在兩者都屬於類型時才出現的關係person。為了優化性能，我們可以編寫如下查詢：

```
MATCH (n)-[:likes]->(m)
RETURN n, m
```

當屬性是單一節點標籤的專用性質時，也可以套用此模式。假設只有person節點具有屬性email，因此驗證節點標籤匹配person是多餘的。將此查詢寫入為：

```
MATCH (n:person)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

比將此查詢編寫為以下方式效率低：

```
MATCH (n)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

只有在效能很重要且您已在建模過程中進行檢查時，才應採用此模式，以確保這些邊標籤不會重複用於涉及其他節點標籤的模式。如果您稍後在另一個節點標籤 (例如) 上引入email屬性company，則這兩個查詢版本之間的結果會有所不同。

盡可能指定邊標示

建議在陣列中指定邊時盡可能提供邊標籤。請考慮下面的示例查詢，該查詢用於將居住在一個城市的所有人與所有訪問該城市的人聯繫起來。

```
MATCH (person)-->(city {country: "US"})-->(anotherPerson)
RETURN person, anotherPerson
```

如果您的圖形模型將人員連結到使用多個邊緣標籤的城市以外的節點，則不指定結束標籤，Neptune 將需要評估稍後將捨棄的其他路徑。在上面的查詢中，由於沒有給出邊緣標籤，引擎首先執行更多的工作，然後過濾掉值以獲得正確的結果。上述查詢的更好版本可能是：

```
MATCH (person)-[:livesIn]->(city {country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

這不僅有助於評估，而且可以使查詢規劃員創建更好的計劃。您甚至可以將此最佳實踐與冗餘節點標籤檢查結合使用，以刪除城市標籤檢查並將查詢寫為：

```
MATCH (person)-[:livesIn]->({country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

盡可能避免使用 WITH 子句

OpenCypher 中的 WITH 子句充當邊界，其中一切在它執行之前，然後將結果值傳遞給查詢的剩餘部分。當您需要臨時彙總或想要限制結果數目時，需要 WITH 子句，但除此之外，您還應該盡量避免使用 WITH 子句。一般指引是移除這些簡單的 WITH 子句 (不含彙總、排序或限制)，讓查詢規劃工具能夠處理整個查詢，以建立全域最佳化計劃。舉個例子，假設您編寫了一個查詢來返回居住在以下位置的所有人India：

```
MATCH (person)-[:lives_in]->(city)
WITH person, city
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

在上述版本中，WITH 子句限制了模式的位置(city)-[:part_of]->(country {name: 'India'})（這是更嚴格的）。(person)-[:lives_in]->(city)這使得計劃不是最佳的。此查詢的優化將是刪除 WITH 子句，並讓計劃者計算最佳計劃。

```
MATCH (person)-[:lives_in]->(city)
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

儘早在查詢中放置限制性篩選器

在所有情況下，篩選器的早期放置在查詢有助於減少查詢計劃必須考慮的中繼解決方案。這表示執行查詢所需的記憶體和較少的運算資源。

下列範例可協助您瞭解這些影響。假設你寫了一個查詢來返回所有居住的人India。查詢的其中一個版本可能是：

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WITH country, collect(n.firstName + " "  + n.lastName) AS result
WHERE country.name = 'India'
RETURN result
```

上述版本的查詢並不是實現此用例的最佳方式。篩選器稍後country.name = 'India'會出現在查詢模式中。它將首先收集所有人和他們居住的地方，並按國家對其進行分組，然後僅過濾該組country.name = India。只查詢居住在其中的人員，India然後執行收集彙總的最佳方式。

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WHERE country.name = 'India'
RETURN collect(n.firstName + " "  + n.lastName) AS result
```

一般規則是在引入變數之後盡快放置篩選器。

明確檢查屬性是否存在

根據 OpenCypher 語義，當訪問屬性時，它相當於一個可選的聯接，並且必須保留所有行，即使該屬性不存在。如果您根據圖形結構描述知道該實體永遠存在特定屬性，則明確檢查該屬性是否存在，可讓查詢引擎建立最佳計劃並改善效能。

考慮一個圖形模型，其中類型的節點person始終具有屬性name。而不是這樣做：

```
MATCH (n:person)
RETURN n.name
```

使用 IS NOT NULL 檢查明確驗證查詢中的屬性是否存在：

```
MATCH (n:person)
WHERE n.name IS NOT NULL
RETURN n.name
```

不要使用命名路徑（除非它是必需的）

查詢中的命名路徑始終需要支付額外費用，這可能會在更高的延遲和內存使用情況方面增加懲罰。請考處下列查詢：

```
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
```

在上面的查詢中，假設我們只想知道節點的屬性，使用路徑「p」是不必要的。通過將命名路徑指定為變量，使用 DISTINCT 的聚合操作在時間和內存使用方面都將變得昂貴。上述查詢的更優化版本可能是：

```
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
```

避免收集（不同（））

收集（DISTINCT（））每當一個列表將被形成包含不同的值被使用。COLLECT 是一個聚合函數，並且分組是基於其他鍵被投射在同一個語句中完成的。使用 distinct 時，輸入被分成多個塊，其中每個塊表示一個用於減少的組。效能會隨著群組數量的增加而受到影響。在 Neptune 中，在實際收集/形成列表之前執行 DISTINCT 要高得多。這允許直接在整個塊的分組鍵上進行分組。

請考處下列查詢：

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH n, collect(distinct(p.post_id)) as post_list
RETURN n, post_list
```

編寫此查詢的更優化方式是：

```
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
```

擷取所有性質值時，偏好使用屬性函數而非個別性質查找

該properties()函數用於返回包含實體的所有屬性的映射，並且比單獨返回屬性更有效。

假設您的Person節點包含 5 個屬性 firstName lastNameage，dept，，和company，以下查詢將是首選：

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN properties(n) as personDetails
```

而不是使用：

```
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
```

在查詢之外執行靜態計算

建議在客戶端解決靜態計算（簡單的數學/字符串操作）。考慮這個例子，你想找到所有的人一歲或小於作者：

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= ($age + 1)
RETURN m
```

在這$age裡，通過參數注入到查詢中，然後添加到一個固定值。然後將此值與進行比較p.age。相反，一個更好的方法是在客戶端上進行加法，並將計算值作為參數 $ageplusone 傳遞。這有助於查詢引擎創建優化計劃，並避免每個傳入行的靜態計算。遵循這些準則，查詢的效率更高：

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= $ageplusone
```

```
RETURN m
```

使用 GRUSH 而非個別陳述式的 Batch 輸入

每當需要為不同的輸入執行相同的查詢時，而不是每個輸入執行一個查詢，對一批輸入運行查詢會更加高效。

如果要在一組節點上合併，則可以選擇每個輸入運行合併查詢：

```
MERGE (n:Person {`~id`: $id})
SET n.name = $name, n.age = $age, n.employer = $employer
```

使用參數：

```
params = {id: '1', name: 'john', age: 25, employer: 'Amazon'}
```

上面的查詢需要為每個輸入執行。雖然這種方法有效，但可能需要為大量輸入執行許多查詢。在這個案例中，批次處理可能有助於減少伺服器上執行的查詢數目，以及改善整體輸送量。

使用下列模式：

```
UNWIND $persons as person
MERGE (n:Person {`~id`: person.id})
SET n += person
```

使用參數：

```
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'}...]}
```

建議您嘗試不同的批次大小，以確定哪些最適合您的工作負載。

喜歡使用節點/關係的自定義IDS

Neptune 允許使用者在節點和關係上明確指派 ID。ID 在資料集中必須是全域唯一的，且具決定性才有用。確定性 ID 可以像屬性一樣用作查閱或篩選機制；不過，從查詢執行的角度來看，使用 ID 比使用屬性要優化得多。使用自訂 ID 有幾個好處-

- 對於現有實體，屬性可以為 null，但 ID 必須存在。這可讓查詢引擎在執行期間使用最佳化的聯結。

- 執行並行變異查詢時，使用 ID 來存取節點時，同時修改例外狀況 (CME) 的可能性會大幅降低，因為其強制的唯一性，因此使用 ID 的鎖定少於屬性。
- 使用 ID 可避免建立重複資料的機會，因為 Neptune 會強制執行 ID 的唯一性，與屬性不同。

下列查詢範例使用自訂 ID：

> **Note**
>
> 該屬性用`~id`於指定 ID，而只`id`是存儲為任何其他屬性。

```
CREATE (n:Person {`~id`: '1', name: 'alice'})
```

不使用自定義 ID：

```
CREATE (n:Person {id: '1', name: 'alice'})
```

如果使用後一種機制，則沒有唯一性強制執行，您可以稍後執行查詢：

```
CREATE (n:Person {id: '1', name: 'john'})
```

這將創建`id=1`具有命名的第二個節點`john`。在這種情況下，您現在將有兩個節點`id=1`，每個節點都有不同的名稱-（alice 和 John）。

避免在查`~id`詢中進行計算

在查詢中使用自訂 ID 時，請務必在查詢外部執行靜態計算，並在參數中提供這些值。提供靜態值時，引擎能夠更好地最佳化查詢，並避免掃描和篩選這些值。

如果你想在數據庫中現有的節點之間創建邊緣，一個選項可能是：

```
UNWIND $sections as section
MATCH (s:Section {`~id`: 'Sec-' + section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

使用參數：

```
parameters={sections: [{id: '1'}, {id: '2'}]}
```

在上面的查詢中，id部分的正在查詢中計算。由於計算是動態的，因此引擎無法靜態內聯 ID，最終掃描所有節點。然後，引擎會針對必要的節點執行後置篩選。如果資料庫中有許多區段節點，這可能會很昂貴。

實現此目的的更好的方法是在傳遞到數據庫的 id 中Sec-加上前面：

```
UNWIND $sections as section
MATCH (s:Section {`~id`: section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

使用參數：

```
parameters={sections: [{id: 'Sec-1'}, {id: 'Sec-2'}]}
```

使用 SPARQL 的 Neptune 最佳實務

搭配 Neptune 使用 SPARQL 查詢語言時，請遵循這些最佳實務。如需在 Neptune 中使用 SPARQL 的相關資訊，請參閱 使用 SPARQL 存取 Neptune 圖形。

預設查詢所有具名圖形

Amazon Neptune 會將每個三元組與一個具名圖形建立關聯。預設圖形是定義成所有具名圖形的聯集。

若您提交 SPARQL 查詢而未透過 GRAPH 關鍵字或 FROM NAMED 之類的建構式明確指定圖形，Neptune 一律會考慮資料庫執行個體中的所有三元組。例如，以下查詢將從 Neptune SPARQL 端點傳回所有三元組：

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

顯現於多個圖形中的三元組將僅傳回一次。

如需預設圖形規格的相關資訊，請參閱 SPARQL 1.1 查詢語言規格的 RDF 資料集一節。

指定要載入的具名圖表

Amazon Neptune 會將每個三元組與一個具名圖形建立關聯。如果您在載入、插入或更新三元組時未指定具名圖形，Neptune 會使用由 URI 定義的備用具名圖形 (`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`)。

如果您是使用 Neptune 大量載入器，則可以使用 parserConfiguration: namedGraphUri 參數，指定要用於所有三元組 (或第四位置空白的四元組) 的具名圖形。如需 Neptune 載入器 Load 命令語法的相關資訊，請參閱 the section called “載入器命令”。

在您的查詢中選擇 FILTER、FILTER...IN 和 VALUES

有三種基本方式可在 SPARQL 查詢中注入值：FILTER、FILTER...IN 和 VALUES。

例如，假設您想要在單次查詢中，查詢多名人員的好友。使用 FILTER，您可以建構如下查詢：

```
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)}
```

這會傳回將 ?s 繫結至 ex:person1 或 ex:person2，且外寄節點標示為 foaf:knows 之圖形中的所有三元組。

您也可以使用傳回同等結果的 FILTER...IN 建立查詢：

```
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))}
```

您也可以使用在這種情況下也會傳回同等結果的 VALUES 建立查詢：

```
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}}
```

雖然在許多情況下，這些查詢在語意上是相等的，但在某些情況下，兩個 FILTER 變體和 VALUES 變體不同：

- 第一種情況是當您注入重複值時，例如注入相同的人員兩次。在這種情況下，VALUES 查詢會在結果中包含重複項目。您可以將 DISTINCT 新增至 SELECT 子句，明確消除這類重複。但有時候，您可能真的需要在查詢結果中出現重複項目，以注入多餘值。

不過，當相同的值出現多次時，FILTER 和 FILTER...IN 版本只擷取一次值。

- 第二種情況與 VALUES 一律執行完全符合，而 FILTER 可能會套用類型提升，並在某些情況下執行模糊映射有關。

  例如，當您在值子句中包含 "2.0"^^xsd:float 等常值時，VALUES 查詢會尋找與此常值完全符合的項目，包括常值的值和資料類型。

  相反地，FILTER 則會產生與這些數值常值模糊符合的項目。符合項目可能包括值相同、但數值資料類型不同的常值，例如 xsd:double。

  > **Note**
  >
  > 列舉字串常值或 URI 時，FILTER 和 VALUES 的行為間無差異。

FILTER 和 VALUES 之間的差異會影響最佳化和產生的查詢評估策略。除非您的使用案例需要模糊相符，否則建議您使用 VALUES，因為它會避免查看有關類型轉換的特殊案例。因此，VALUES 通常會產生更有效率的查詢，執行速度更快但價格低廉。

Amazon Neptune 限制

區域

Amazon Neptune 在以下 AWS 區域提供：

- 美國東部 (維吉尼亞北部)：us-east-1
- 美國東部 (俄亥俄)：us-east-2
- 美國西部 (加利佛尼亞北部)：us-west-1
- 美國西部 (奧勒岡)：us-west-2
- 加拿大 (中部)：ca-central-1
- 南美洲 (聖保羅)：sa-east-1
- 歐洲 (斯德哥爾摩)：eu-north-1
- 歐洲 (愛爾蘭)：eu-west-1
- 歐洲 (倫敦)：eu-west-2
- 歐洲 (巴黎)：eu-west-3
- 歐洲 (法蘭克福)：eu-central-1
- 中東 (巴林)：me-south-1
- 中東 (阿拉伯聯合大公國)：me-central-1
- 以色列 (特拉維夫)：il-central-1
- 非洲 (開普敦)：af-south-1
- 亞太區域 (香港)：ap-east-1
- 亞太區域 (東京)：ap-northeast-1
- 亞太區域 (首爾)：ap-northeast-2
- 亞太區域 (大阪): ap-northeast-3
- 亞太區域 (新加坡)：ap-southeast-1
- 亞太區域 (雪梨)：ap-southeast-2
- 亞太區域 (孟買)：ap-south-1
- 中國 (北京)：cn-north-1
- 中國 (寧夏)：cn-northwest-1
- AWS GovCloud (美國西部): us-gov-west-1

- AWS GovCloud (美國東部): `us-gov-east-1`

中國區域的差異

與許多 AWS 服務一樣，Amazon Neptune 在中國（北京）和中國（寧夏）的運營方式與其他 AWS 地區略有不同。

例如，當 Neptune ML 使用 Amazon API Gateway 建立其匯出服務時，預設會啟用 IAM 身分驗證。在中國區域，變更該選項的程序與其他地區略有不同。

這些和其他差異會在這裡說明。

儲存叢集磁碟區的大小上限

除了中國以外的所有支援區域，Neptune 叢集磁碟區的大小上限為 128 TB (TiB) GovCloud，限制為 64 TiB。以 版本：1.0.2.2 (2020 年 3 月 9 日) 開頭的所有引擎版本都是如此。請參閱Amazon Neptune 儲存體、可靠性和可用性。

支援的資料庫執行個體大小

Neptune 支援不同 AWS 區域的不同資料庫執行個體類別 若要了解特定區域中支援的類別，請參閱 Amazon Neptune 定價，然後選擇您有興趣的區域。

每個 AWS 帳戶的限制

對於某些管理功能，Amazon Neptune 使用與 Amazon Relational Database Service (Amazon RDS) 共用的操作技術。

每個 AWS 帳戶對於您可以建立的 Amazon 海王星和 Amazon RDS 資源數量，每個區域都有限制。這些資源包括資料庫執行個體和資料庫叢集。

在您到達某項資源的上限時，所發出用來建立該資源的額外呼叫都會伴隨著異常而失敗。

如需 Amazon Neptune 與 Amazon RDS 之間共用的限制清單，請參閱《Amazon RDS 使用者指南》中的 Amazon RDS 中的限制。

連線到 Neptune 需要 VPC

Amazon Neptune 是一種純虛擬私有雲端 (VPC) 服務。

此外，該服務不允許從 VPC 外部來存取執行個體。

Neptune 需要 SSL

從引擎版本 `1.0.4.0` 開始，Amazon Neptune 只允許透過 HTTPS 對任何執行個體或叢集端點進行 Secure Sockets Layer (SSL) 連線。

Neptune 需要 TLS 1.2 版，使用下列強式密碼套件：

- `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`

可用區域與資料庫子網路群組

Amazon Neptune 要求每個叢集都需具備一個資料庫子網路群組，且其在至少兩個支援的可用區域 (AZ) 內擁有子網路。

建議您在不同的可用區域中使用三個或多個子網路。

HTTP 請求承載上限 (150 MB)

Gremlin 與 SPARQL HTTP 請求的總大小必須小於 150 MB。若請求超過此大小，Neptune 會傳回 `HTTP 400: BadRequestException`。

此限制不適用於小鬼連線 WebSockets 。

Gremlin 實作差異

Amazon Neptune Gremlin 實作具有特定的實作詳細資訊，可能會和其他 Gremlin 實作不同。

如需詳細資訊，請參閱 Amazon Neptune 中的 Gremlin 標準合規。

Neptune 不支援字串資料中的空字元

Neptune 不支援字串中的 Null 字元 Gremlin 和 openCypher 的屬性圖資料，以及 RDF/SPARQL 資料也是如此。

從 URI 進行 SPARQL UPDATE LOAD

來自 URI 的 `SPARQL UPDATE LOAD` 僅適用於相同 VPC 中的資源。

這包括與建立 Amazon S3 VPC 端點的叢集位於同一區域的 Amazon S3 URL。

Amazon S3 URL 必須為 HTTPS，且 URL 中必須包含任何身分驗證。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的驗證請求：使用查詢參數。

如需有關建立 VPC 端點的資訊，請參閱 建立 Amazon S3 VPC 端點。

如果您需要從檔案載入資料，建議您使用 Amazon Neptune 載入器 API。如需詳細資訊，請參閱 使用 Amazon Neptune 大量載入器擷取資料。

> **Note**
>
> Amazon Neptune 載入器 API 不具備 ACID。

IAM 身分驗證與存取控制

在 1.2.0.0 版之前的 Neptune 引擎版本中，僅會在資料庫叢集層級支援 IAM 身分驗證和存取控制。不過，從版本 `1.2.0.0` 開始，您可以在 IAM 政策中使用條件金鑰，在更精細的層級控制查詢型存取。如需詳細資訊，請參閱 在 Neptune 資料存取政策陳述式中使用查詢動作 和 Amazon Neptune AWS Identity and Access Management （ IAM ）概述

Amazon Neptune 主控台需要NeptuneReadOnlyAccess許可。您可以限制對 IAM 使用者的存取，方法是撤銷此存取。如需更多資訊，請參閱AWS Amazon Neptune 的受管 (預先定義) 政策

Amazon Neptune 並不支援使用者名稱/密碼類型存取控制。

WebSocket 並發連接和最長連接時間

每個 Neptune 資料庫執行個體的同時 WebSocket 連線數目有限制。達到該限制時，Neptune 會限制任何請求以打開新 WebSocket 連接，以防止耗盡所有分配的堆積內存。

對於 Neptune 和所有無伺服器執行個體支援的所有較大型執行個體類型， WebSocket 連線的並行數目上限為 32K (32,768)。

下表列出較小執行個體類型的最大同時 WebSocket 連線數目：

執行個體類型	最大同時 WebSocket 連線
db.t3.medium	512
db.t4g.medium	512
db.r5.large	2,048
db.r5d.large	2,048
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	2,048
db.r6gd.large	2,048
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	2,048

執行個體類型	最大同時 WebSocket 連線
db.x2gd.large	2,048
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
db.x2g.4xlarge	16,384
db.x2gd.4xlarge	16,384
db.x2iedn.2xlarge	16,384
db.x2iezn.2xlarge	16,384
serverless	32,768
(其他大型執行個體類型)	32,768

> **Note**
>
> 從 Neptune 引擎 1.1.0.0 版開始，Neptune 不再支援 R4 執行個體類型。

當用戶端以正常方式關閉連線時，該關閉會立即反映在開啟的連線數量上。

如果用戶端未關閉連線，則連線可能會在 20 到 25 分鐘的閒置逾時後自動關閉 (閒置逾時是自從用戶端收到最後一則訊息後所經過的時間)。不過，只要未達到閒置逾時，Neptune 就會無限期地保持連線開啟。

啟用 IAM 身份驗證後，如果 WebSocket 連線尚未關閉，則連線一律會在建立 10 天後的幾分鐘內中斷連線。

屬性和標籤的限制

您在一個圖形中可擁有無限個頂點和邊緣或 RDF quad。

任何一個頂點或邊緣也可以擁有無限個屬性或標籤。

個別屬性或標籤的大小上限為 55 MB。在 RDF 術語中，這意味著 RDF quad 的任何資料欄 (S、P、O 或 G) 中的值都不能超過 55 MB。

如果您需要將較大的物件 (例如影像) 與圖形中的頂點或節點建立關聯，您可以在 Amazon S3 中將其儲存為檔案，並使用 Amazon S3 路徑做為屬性或標籤。

影響 Neptune 大量載入器的限制

您不能一次將超過 64 個 Neptune 大量載入工作排入佇列。

Neptune 只會追蹤最近 1,024 個大量載入工作。

Neptune 只會儲存每個工作的最後 10,000 個錯誤詳細資訊。

使用其他 AWS 服務

您可以搭配許多其他 AWS 服務使用 Amazon Neptune：

Neptune 與其他服務的整合

- AWS Glue – AWS Glue 是無伺服器資料整合服務，可協助您對資料執行擷取、轉換和載入 (ETL) 工作。

  Neptune 會提供一個開放原始碼程式庫 (neptune-python-utilities)，其可簡化在 Glue 工作內使用 Python 和 Gremlin 的過程。也支援 Neo4j Spark Connector，用於執行 Scala 和 openCypher Glue 工作。

- Amazon SageMaker – Amazon SageMaker 是功能齊全的機器學習平台，用於建置、訓練和部署高品質的機器學習模型。

  Neptune 透過兩種主要方式與 SageMaker 整合：

  - Neptune 為 Jupyter 筆記本提供了一個開放原始碼 Python 套件，此套件可在 GitHub 上的 Neptune 圖形筆記本專案中找到。此套件包含一組 Jupyter 魔法、教學課程筆記本，以及在互動式編碼環境中提供的程式碼範例，您可以在其中了解圖形技術和 Neptune。Neptune 為 SageMaker 託管的 Jupyter 筆記本提供全受管環境，並自動連結至開放原始碼 Neptune 圖形筆記本專案中的筆記本。
  - Neptune ML 功能可讓您在數小時而不是數週的時間內，在大型圖形上建置並訓練有用的機器學習模型。為了達成這個目標，Neptune ML 會使用由 Amazon SageMaker 和 Deep Graph Library (DGL) 提供的圖形神經網路 (GNN)技術。

- AWS Lambda – AWS Lambda 功能在 Neptune 應用程式中有很多用途。

  如需如何使用 Lambda 函數搭配任何熱門 Gemlin 驅動程式和語言變體的相關資訊，以及以 Java、JavaScript 和 Python 撰寫之 Lambda 函數的特定範例，請參閱 使用 Amazon Neptune 中的 AWS Lambda 函數。

- Amazon Athena – 是一種互動式查詢服務，可讓您在 Amazon Simple Storage Service 和其他聯合資料來源中使用標準 SQL 輕鬆地分析資料。

  Neptune 提供 Athena 連接器，可讓 Athena 與儲存在 Neptune 中的資料進行通訊。

- AWS Database Migration Service (AWS DMS) – AWS Database Migration Service 是一種 AWS Web 服務，您可以使用此服務，將資料從一個資料庫遷移到另一個資料庫。

AWS DMS 可以快速安全地從支援的來源資料庫將資料載入至 Neptune。來源資料庫在遷移期間仍然能夠維持完全正常運作，讓倚賴它的應用程式可以將停機時間縮到最短。

- AWS Backup – AWS Backup 是一項全受管備份服務，可讓您在雲端以及內部部署環境內輕鬆集中管理各 AWS 服務間的資料備份，並予以自動化。

  AWS Backup 可讓您跨受支援 AWS 服務的集中式資料保護政策用於資料庫、儲存體和運算，來建立 Neptune 叢集的自動化定期快照。

- AWS SDK for pandas – The AWS SDK for pandas (以前稱為 AWS Data Wrangler 或 `awswrangler`) 是一項 AWS Professional Service 開放原始碼 Python 計劃，其可將 `pandas` Python 資料分析程式庫的強大功能延伸到 AWS，進而連線 `DataFrames` 和 30 多個 AWS 資料相關服務 (包括 Neptune)。

  除了 SDK 之外，還有一個關於如何搭配 Neptune 使用它的教學課程，以及幾個範例 Neptune 筆記本，即詐騙集團偵測、合成身分偵測和物流分析。

- JDBC 驅動程式 – Neptune JDBC 驅動程式支援 openCypher、Gremlin、SQL-Gremlin 和 SPARQL 查詢。

  JDBC 連線能力可讓您透過商業智慧 (BI) 工具 (例如 Tableau) 輕鬆連線至 Neptune。

Neptune 工具和公用程式

Amazon Neptune 提供許多工具和公用程式，可透過圖形簡化和自動化您使用圖形的工作。其中包括以下項目：

Amazon Neptune 工具

- 適用於 GraphQL 的 Amazon Neptune 公用程式 – 適用於 GraphQL 的 Amazon Neptune 公用程式是開放原始碼 Node.js 命令列工具，可協助您為 Neptune 屬性圖資料庫建立和維護 GraphQL API。它是一種無程式碼方式，可以為 GraphQL 查詢建立 GraphQL 解析程式，而這些查詢具有可變數量的輸入參數並會傳回可變數量的巢狀欄位。

適用於 GraphQL 的 Amazon Neptune 公用程式

適用於 GraphQL 的 Amazon Neptune 公用程式是開放原始碼 Node.js 命令列工具，可協助您為 Neptune 屬性圖資料庫建立和維護 GraphQL API (它尚未使用 RDF 資料)。它是一種無程式碼方式，可以為 GraphQL 查詢建立 GraphQL 解析程式，而這些查詢具有可變數量的輸入參數並會傳回可變數量的巢狀欄位。

它已作為一個開源項目發布，位於 https://github.com/aws/ amazon-neptune-for-graphql。

您可以像這樣使用 NPM 安裝公用程式 (如需詳細資訊，請參閱安裝和設定)：

```
npm i @aws/neptune-for-graphql -g
```

此公用程式可以探索現有 Neptune 屬性圖的圖形結構描述，包括節點，邊緣，屬性和邊緣基數。然後，它會產生 GraphQL 結構描述，其中包含將 GraphQL 類型映射至資料庫節點和邊緣所需的指令，並自動產生解析程式碼。解析程式碼旨在透過僅傳回 GraphQL 查詢所請求的資料來將延遲降至最低。

您也可以從現有的 GraphQL 結構描述和空白的 Neptune 資料庫開始，然後讓公用程式推斷將該 GraphQL 結構描述映射至資料 (要載入至資料庫) 節點和邊緣所需的指令。或者，您可以從 GraphQL 結構描述和您已經建立或修改的指令開始。

此公用程式能夠建立管線所需的所有AWS資源，包括 AWS AppSync API、IAM 角色、資料來源、結構描述和解析程式，以及查詢 Neptune 的 AWS Lambda 函數。

> **Note**
>
> 這裡的命令列範例假設使用 Linux 主控台。如果您使用的是 Windows，請將行尾的反斜線 (「\」) 取代為插入號 (「^」)。

主題

安裝和設定適用於 GraphQL 的 Amazon Neptune 公用程式

如果您將要使用公用程式搭配現有的 Neptune 資料庫，您需要它才能連線到資料庫端點。根據預設，Neptune 資料庫只能從其所在的 VPC 內存取。

因為公用程式是 Node.js 命令列工具，所以您必須安裝 Node.js (第 18 版或更新版本) 才能執行公用程式。若要在與 Neptune 資料庫相同的 VPC 中的 EC2 執行個體上安裝 Node.js，請遵循這裡的指示。要執行公用程式的大小下限執行個體為 t2.micro。在建立執行個體期間，從常見安全群組下拉式功能表中選取 Neptune 資料庫 VPC。

不過，從引擎 1.2.0.0 版開始，您可以為 Neptune 資料庫建立可在 VPC 外部存取的公用端點。如果您已建立公用端點，則可以在本機電腦上安裝 Node.js 和公用程式。若要在 macOS 或 Windows 上安裝 Node.js，請造訪 Node.js 網站以下載安裝程式。

若要在 EC2 執行個體或本機電腦上安裝公用程式本身，請使用 NPM：

```
npm install aws-neptune-for-graphql -g
```

然後，您可以執行公用程式的 help 命令來檢查它是否正確安裝：

```
neptune-for-graphql --help
```

您可能還想要安裝 AWS CLI 來管理 AWS 資源。

掃描現有 Neptune 資料庫中的資料

無論您是否熟悉 GraphQL，下面的命令都是建立 GraphQL API 的最快方法。這假設您已依照安裝一節中所述安裝並設定 GraphQL 的 Neptune 公用程式，以便它可以連線到 Neptune 資料庫的端點。

```
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
```

公用程式會分析資料庫以探索其中節點，邊緣和屬性的結構描述。根據該結構描述，它會透過相關聯的查詢和變動推斷 GraphQL 結構描述。然後它創建一個 AppSync GraphQL API 和使用它所需的AWS資源。這些資源包括一對 IAM 角色，以及一個包含 GraphQL 解析程式碼的 Lambda 函數。

公用程式完成後，您會在 AppSync主控台中找到新的 GraphQL API，並在指令中指派的名稱下方。要對其進行測試，請使用菜單上的「 AppSync 查詢」選項。

如果您在將更多資料新增至資料庫後再次執行相同的命令，則會相應地更新 AppSync API 和 Lambda 程式碼。

若要釋放與命令相關聯的所有資源，請執行：

```
neptune-for-graphql \
  --remove-aws-pipeline-name (your new GraphQL API name from above)
```

從沒有指令的 GraphQL 結構描述開始

您可以從空白的 Neptune 資料庫開始，並使用沒有指令的 GraphQL 結構描述來建立資料並進行查詢。以下命令會自動建立 AWS 資源來執行此操作：

```
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
```

GraphQL 結構描述檔案必須包含 GraphQL 結構描述類型，如以下 TODO 範例所示。公用程式會分析您的結構描述，並根據您的類型建立延伸版本。它會針對圖形資料庫中儲存的節點新增查詢和變動，而且如果你的結構描述具有巢狀類型，它會新增類型 (儲存為資料庫中的邊緣) 之間的關係。

該實用程序創建一個 AppSync GraphQL API，以及所有必需的AWS資源。其中包括一對 IAM 角色，以及一個包含 GraphQL 解析程式碼的 Lambda 函數。命令完成後，您可以找到具有您在 AppSync 控制台中指定名稱的新 GraphQL API。要對其進行測試，請使用 AppSync 菜單中的查詢。

以下範例說明其運作方式。

Todo 範例，從沒有指令的 GraphQL 結構描述開始

在這個範例中，我們從一個沒有指令的 Todo GraphQL 結構描述開始，你可以在 *???samples???* 目錄中找到該結構描述。它包括以下兩種類型：

```
type Todo {
  name: String
  description: String
  priority: Int
  status: String
  comments: [Comment]
}

type Comment {
  content: String
}
```

此命令處理待辦事項模式和空 Neptune 數據庫的端點，以在以下位置創建 GraphQL API：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
```

該實用程序在名為的輸出文件夾中創建一個新文件`TodoExample.source.graphql`，並在中AppSync創建 GraphQL API。公用程式會推斷以下事項：

- 在待辦事項類型中，它@relationship為新 CommentEdge 類型添加。這指示解析器使用稱為圖形數據庫邊緣將待辦事項連接到註釋。 CommentEdge
- 它增加了一個名為幫助 TodoInput 查詢和突變的新輸入。
- 它已針對每個類型新增兩個查詢 (Todo、Comment)：一個用於使用 id 或輸入中列出的任何類型欄位來擷取單一類型，而另一個用於擷取多個值，同時使用該類型的輸入進行篩選。
- 它已針對每種類型新增三個變動：create、update 和 delete。要刪除的類型是使用 id 或該類型的輸入來指定。這些變動會影響 Neptune 資料庫中儲存的資料。
- 它已針對連線新增兩個變動：connect 和 delete。它們會採取 Neptune 所使用的來源和目標頂點的節點 ID 作為輸入，而且連線是資料庫中的邊緣。

解析程式透過名稱辨識查詢和變動，但您可以自訂它們，如下面所示。

以下是 TodoExample.source.graphql 檔案的內容：

```
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
}

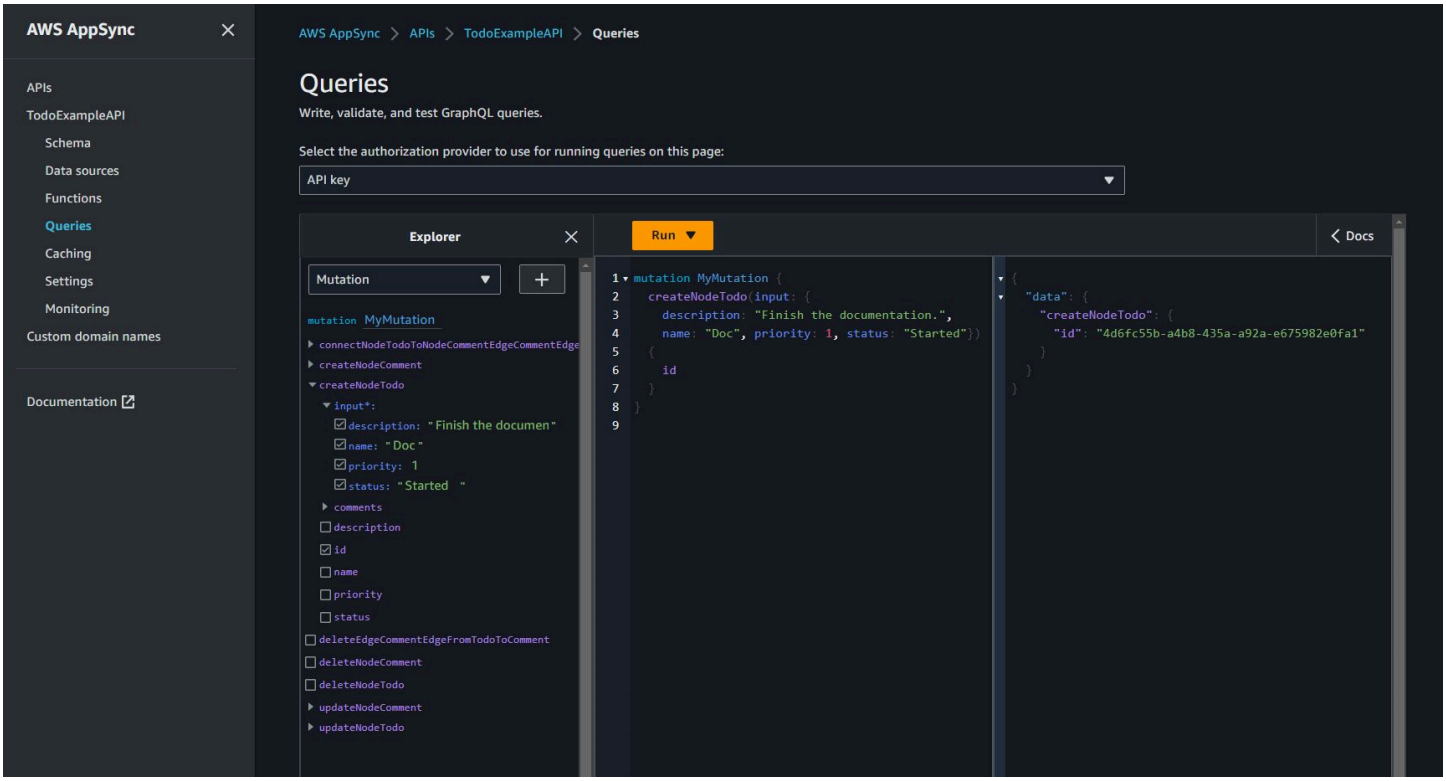
input Options {
  limit: Int
}

type Query {
  getNodeTodo(filter: TodoInput, options: Options): Todo
  getNodeTodos(filter: TodoInput): [Todo]
  getNodeComment(filter: CommentInput, options: Options): Comment
  getNodeComments(filter: CommentInput): [Comment]
}

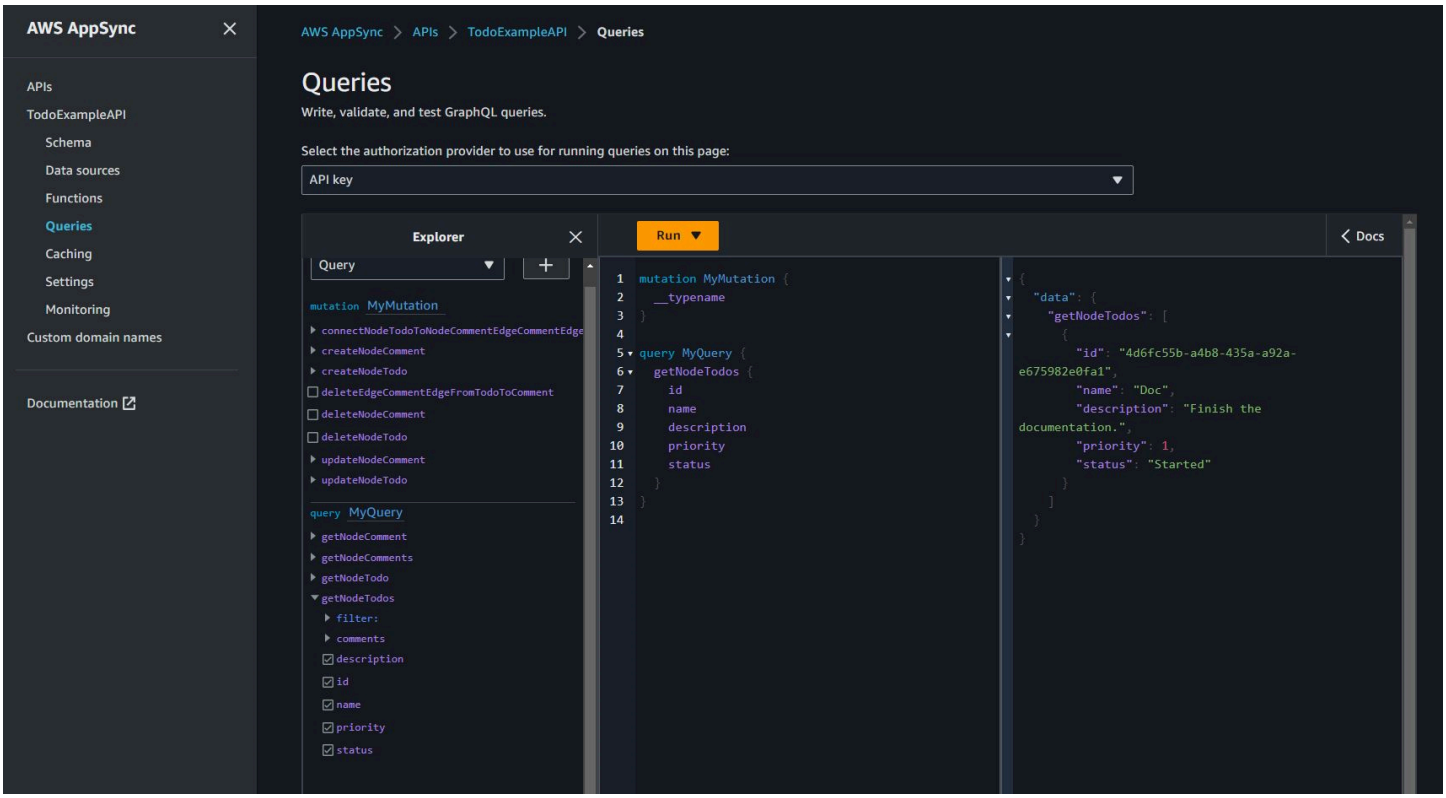
type Mutation {
  createNodeTodo(input: TodoInput!): Todo
  updateNodeTodo(input: TodoInput!): Todo
  deleteNodeTodo(_id: ID!): Boolean
  connectNodeTodoToNodeCommentEdgeCommentEdge(from_id: ID!, to_id: ID!): CommentEdge
  deleteEdgeCommentEdgeFromTodoToComment(from_id: ID!, to_id: ID!): Boolean
  createNodeComment(input: CommentInput!): Comment
  updateNodeComment(input: CommentInput!): Comment
  deleteNodeComment(_id: ID!): Boolean
}

schema {
  query: Query
  mutation: Mutation
}
```

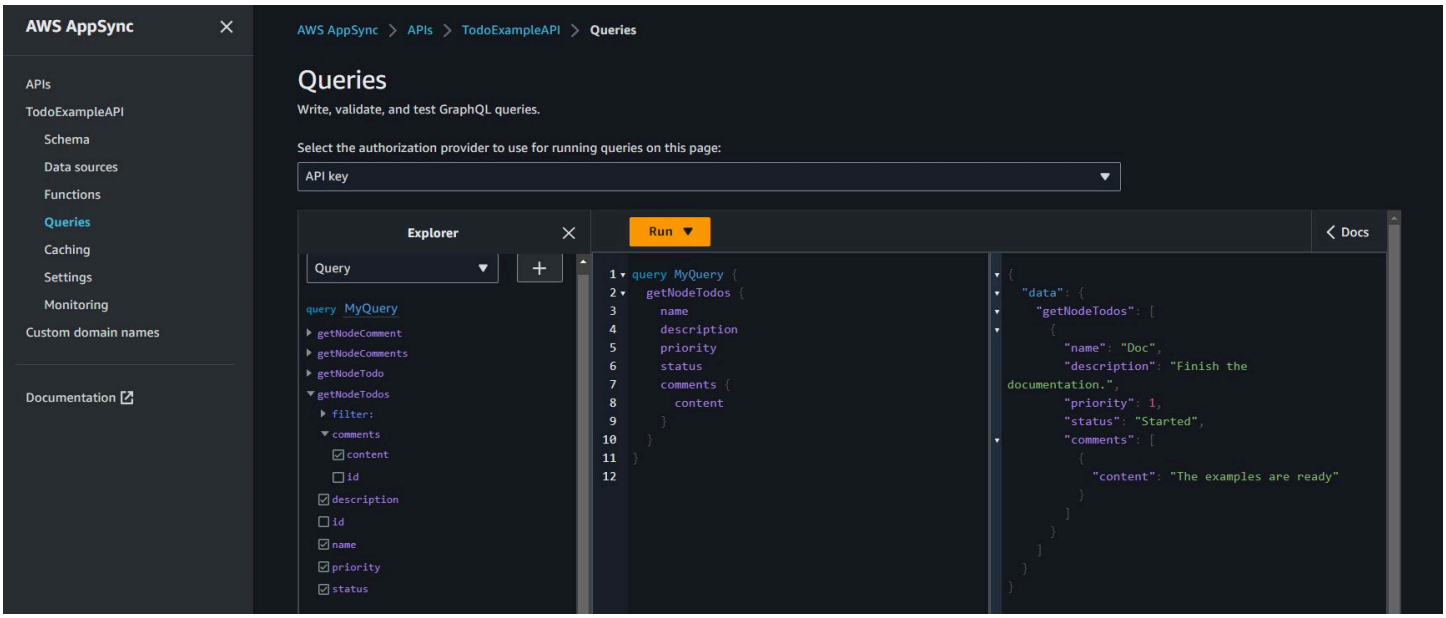
現在，您可以建立和查詢資料。以下是用於測試新 GraphQL API 的 AppSync查詢控制台的快照，TodoExampleAPI在這種情況下命名。在中間視窗中，Explorer 會為您顯示查詢和變動的清單，您可以從中挑選查詢、輸入參數和傳回欄位。此螢幕擷取畫面顯示如何使用 `createNodeTodo` 變動建立 Todo 節點類型：



此螢幕擷取畫面顯示如何使用 `getNodeTodos` 查詢來查詢所有 Todo 節點：



在使用 createNodeComment 建立了註解之後，您可以使用 connectNodeTodoToNodeCommentEdgeCommentEdge 變動，透過指定其 ID 來連線它們。以下是要擷取 Todos 及其附加註解的巢狀查詢：



如果您想要依照使用指令中所述對 TodoExample.source.graphql 檔案進行變更，則可以使用已編輯的結構描述作為輸入，然後再次執行公用程式。公用程式接著會相應地修改 GraphQL API。

使用 GraphQL 結構描述的指令

您可以使用如下的命令，從已有指令的 GraphQL 結構描述開始：

```
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
```

您可以修改公用程式已建立的指令，或將您自己的指令新增至 GraphQL 結構描述。以下是一些使用指令的方法：

執行公用程式，以便它不會產生變動

若要防止公用程式在 GraphQL API 中產生變動，請在 neptune-for-graphql 命令中使用 --output-schema-no-mutations 選項。

`@alias` 指令

`@alias` 指令可以套用至 GraphQL 結構描述類型或欄位。它會在圖形資料庫與 GraphQL 結構描述之間映射不同的名稱。語法是：

```
@alias(property: (property name))
```

在下面範例中，`airport` 是映射到 `Airport` GraphQL 類型的圖形資料庫節點標籤，而 `desc` 是映射到 `description` 欄位的圖形節點屬性 (請參閱航線範例)：

```
type Airport @alias(property: "airport") {
  city: String
  description: String @alias(property: "desc")
}
```

請注意，標準 GraphQL 格式會要求 Pascal 大小寫類型名稱和駝峰式大小寫欄位名稱。

`@relationship` 指令

`@relationship` 指令會將巢狀 GraphQL 類型映射到圖形資料庫邊緣。語法是：

```
@relationship(edgeType: (edge name), direction: (IN or OUT))
```

以下為範例命令：

```
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)
}
```

您可以在 Todo 範例和航線範例中找到 `@relationship` 指令。

`@graphQuery` 和 `@cypher` 指令

您可以定義 OpenCypher 查詢來解析欄位值、新增查詢或新增變動。例如，這會將新的 `outboundRoutesCount` 欄位新增至 `Airport` 類型，以對傳出路由進行計數：

```
type Airport @alias(property: "airport") {
  ...
  outboundRoutesCount: Int @graphQuery(statement: "MATCH (this)-[r:route]->(a) RETURN
 count(r)")
}
```

以下是新查詢和變動的範例

```
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")
}
```

請注意，如果您省略了 RETURN，解析程式會假設關鍵字 this 是傳回範圍。

您也可以使用 Gemlin 查詢來新增查詢或變動：

```
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
}
```

目前，Gemlin 查詢限於傳回純量值或 elementMap() (若為單一節點) 或 elementMap().fold() (若為節點清單) 的查詢。

@id 指令

@id 指令會辨識映射到 id 圖形資料庫實體的欄位。圖形資料庫 (例如 Amazon Neptune) 對於在大量匯入期間指派或自動產生的節點和邊緣，永遠都有唯一的 id。例如：

```
type Airport {
  _id: ID! @id
  city: String
  code: String
}
```

保留類型、查詢和變動名稱

公用程式會自動產生查詢和變動，以建立運作的 GraphQL API。這些名稱的模式由解析程式辨識別並保留。以下是類型 Airport 和連線類型 Route 的範例：

Options 類型是保留的。

```
input Options {
  limit: Int
}
```

filter 和 options 函數參數是保留的。

```
type Query {
  getNodeAirports(filter: AirportInput, options: Options): [Airport]
}
```

查詢名稱的 getNode 字首是保留的，並且變動名稱 (例如 createNode、updateNode、deleteNode、connectNode、deleteNode、updateEdge 和 deleteEdge) 的字首是保留的。

```
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
}
```

將變更套用至 GraphQL 結構描述

您可以修改 GraphQL 來源結構描述，然後再次執行公用程式，從 Neptune 資料庫取得最新的結構描述。每次公用程式在資料庫中發現新的結構描述時，就會產生新的 GraphQL 結構描述。

您也可以手動編輯 GraphQL 來源結構描述，並使用來源結構描述做為輸入，而不是 Neptune 資料庫端點，來重新執行公用程式。

最後，您可以使用以下 JSON 格式將變更放入檔案中：

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

例如：

```
[
  {
    "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": ""
  }
]
```

然後，當您在命令中使用 `--input-schema-changes-file` 參數，在此檔案上執行公用程式時，公用程式會立即套用您的變更。

GraphQL 公用程式的命令列引數

- **`--help, -h`** – 將 GraphQL 公用程式的說明文字傳回至主控台。

- **`--input-schema`** ***`(schema text)`*** – 要用作輸入的 GraphQL 結構描述 (含或不含指令)。

- **`--input-schema-file`** ***`(file URL)`*** – 檔案的 URL，此檔案包含要用作輸入的 GraphQL 結構描述。

- **`--input-schema-changes-file`** ***`(file URL)`*** – 檔案的 URL，此檔案包含您要對 GraphQL 結構描述進行的變更。如果您多次針對 Neptune 資料庫執行公用程式，並且還手動變更 GraphQL 來源結構描述，也許新增自訂查詢，您的手動變更將會失去。若要避免這種情況，請將您的變更放入變更檔案中，並使用此引數傳遞它。

  變更檔案會使用下列 JSON 格式：

  ```
  [
    {
      "type": "(GraphQL type name)",
      "field": "(GraphQL field name)",
      "action": "(remove or add)",
      "value": "(value)"
    }
  ]
  ```

  如需詳細資訊，請參閱 Todo 範例。

- **`--input-graphdb-schema` *(schema text)*** – 您可以透過文字形式表示 graphdb 結構描述以用作輸入，而不是針對 Neptune 資料庫執行公用程式。graphdb 結構描述具有如下的 JSON 格式：

```
{
  "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)*** – 您可以將 graphdb 結構描述儲存在要用作輸入的檔案中，而不是針對 Neptune 資料庫執行公用程式。如需 graphdb 結構描述檔案的 JSON 格式範例，請參閱上述 `--input-graphdb-schema`。

- **`--input-graphdb-schema-neptune-endpoint` *(endpoint URL)*** – 公用程式應從中擷取 graphdb 結構描述的 Neptune 資料庫端點。

- **`--output-schema-file` *`(file name)`*** – GraphQL 結構描述的輸出檔案名稱。如果未指定，則預設值為 `output.schema.graphql`，除非已使用 `--create-update-aws-pipeline-name` 設定管道名稱，在此情況下預設檔案名稱為 *`(pipline name)`*`.schema.graphql`。

- **`--output-source-schema-file` *`(file name)`*** – GraphQL 結構描述 (含指令) 的輸出檔案名稱。如果未指定，則預設值為 `output.source.schema.graphql`，除非已使用 `--create-update-aws-pipeline-name` 設定管道名稱，在此情況下預設名稱為 *`(pipeline name)`*`.source.schema.graphql`。

- **`--output-schema-no-mutations`** – 如果存在此引數，則公用程式不會在 GraphQL API 中產生任何變動，只會產生查詢。

- **`--output-neptune-schema-file` *`(file name)`*** – 公用程式探索的 Neptune graphdb 結構描述的輸出檔案名稱。如果未指定，則預設值為 `output.graphdb.json`，除非已使用 `--create-update-aws-pipeline-name` 設定管道名稱，在此情況下預設檔案名稱為 *`(pipeline name)`*`.graphdb.json`。

- **`--output-js-resolver-file` *`(file name)`*** – 解析程式碼副本的輸出檔案名稱。如果未指定，則預設值為 `output.resolver.graphql.js`，除非已使用 `--create-update-aws-pipeline-name` 設定管道名稱，在此情況下檔案名稱為 *`(pipeline name)`*`.resolver.graphql.js`。

  此檔案會壓縮在程式碼套件中，此程式碼套件會上傳至執行解析程式的 Lambda 函數。

- **`--output-resolver-query-sdk`** – 此引數指定公用程式的 Lambda 函數應該使用 Neptune 資料 SDK 查詢 Neptune，該 SDK 已從 Neptune 引擎版本 1.2.1.0.R5 開始可供使用 (這是預設值)。不過，如果公用程式偵測到較舊的 Neptune 引擎版本，則其會建議改用 HTTPS Lambda 選項，您可以使用 `--output-resolver-query-https` 引數調用該選項。

- **`--output-resolver-query-https`** – 此引數指定公用程式的 Lambda 函數應該使用 Neptune HTTPS API 查詢 Neptune。

- **`--create-update-aws-pipeline`**— 此引數會觸發建立供 GraphQL API 使用的AWS資源，包括 AppSync GraphQL API 和執行解析器的 Lambda。

- **`--create-update-aws-pipeline-name`** ***`(pipeline name)`***— 此引數會設定管線的名稱，例如 Lambda `pipeline-name` 函數的 `pipeline-name` API AppSync 或函數。如果未指定名稱，則 `--create-update-aws-pipeline` 會使用 Neptune 資料庫名稱。
- **`--create-update-aws-pipeline-region`** ***`(AWS region)`*** – 此引數會設定在其中建立 GraphQL API 管道的 AWS 區域。如果未指定，則預設區域為 `us-east-1` 或 Neptune 資料庫所在的區域 (從資料庫端點擷取的區域)。
- **`--create-update-aws-pipeline-neptune-endpoint`** ***`(endpoint URL)`*** – 此引數會設定 Lambda 函數用來查詢資料庫的 Neptune 資料庫端點。如果未設定，則會使用 `--input-graphdb-schema-neptune-endpoint` 所設定的端點。
- **`--remove-aws-pipeline-name`** ***`(pipeline name)`*** – 此引數會移除使用 `--create-update-aws-pipeline` 建立的管道。要移除的資源會列示在名為 *`(pipeline name)`*`.resources.json` 的檔案中。
- **`--output-aws-pipeline-cdk`**— 此引數會觸發 CDK 檔案的建立，該檔案可用來建立 GraphQL API 的AWS資源，包括 Gra AppSync phQL API 和執行解析器的 Lambda 函數。
- **`--output-aws-pipeline-cdk-neptume-endpoint`** ***`(endpoint URL)`*** – 此引數會設定 Lambda 函數用來查詢 Neptune 資料庫的 Neptune 資料庫端點。如果未設定，則會使用 `--input-graphdb-schema-neptune-endpoint` 所設定的端點。
- **`--output-aws-pipeline-cdk-name`** ***`(pipeline name)`***— 此引數設定要使用的 AppSync API 和 Lambda 管線名稱函數的管線名稱名稱。如果未指定，則 `--create-update-aws-pipeline` 會使用 Neptune 資料庫名稱。
- **`--output-aws-pipeline-cdk-region`** ***`(AWS region)`*** – 這會設定在其中建立 GraphQL API 管道的 AWS 區域。如果未指定，則其會預設為 `us-east-1` 或 Neptune 資料庫所在的區域 (從資料庫端點擷取的區域)。
- **`--output-aws-pipeline-cdk-file`** ***`(file name)`*** – 這會設定 CDK 檔案名稱。如果未設定，則預設值為 *`(pipeline name)`*`-cdk.js`。

Neptune 服務錯誤

Amazon Neptune 有不同的兩組錯誤：

- 圖形引擎錯誤，僅發生在 Neptune 資料庫叢集端點。
- 這些錯誤與使用 AWS SDK 和 AWS Command Line Interface (AWS CLI) 建立與修改 Neptune 資源的 API 有關。

主題

圖形引擎錯誤訊息和代碼

Amazon Neptune 端點遇到錯誤時，會傳回 Gremlin 和 SPARQL 的標準錯誤。

也可以從相同的端點傳回 Neptune 的特定錯誤。本節記錄 Neptune 錯誤訊息、代碼和建議的動作。

> **Note**
>
> 這些錯誤僅發生在 Neptune 資料庫叢集端點。若是 AWS SDK 和 AWS CLI，其用來建立和修改 Neptune 資源的 API 具有不同的常見錯誤集。如需這些錯誤的資訊，請參閱the section called “API 錯誤”。

圖形引擎錯誤格式

Neptune 錯誤訊息傳回相關的 HTTP 錯誤代碼和 JSON 格式回應。

```
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."
}
```

圖形引擎查詢錯誤

下表包含錯誤代碼、訊息和 HTTP 狀態。

也會指出是否可以重試請求。一般而言，如果新嘗試可能成功，就可以重試請求。

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 modification conflict. The client should retry the request.
`ConstraintViolationException`	400	Yes	The query engine discovered, during the execution of the

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
			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.
`InvalidNumericDataException`	400	No	Invalid use of numeric data which cannot be represented in 64-bit storage size.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`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.
`MissingParameterException`	400	No	A required parameter for the specified action is not supplied.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`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.
`TooManyRequestsException`	429	Yes	The rate of requests exceeds the maximum throughput. OK to retry.

Neptune Service Error Code	HTTP status	Ok to Retry?	Message
`UnsupportedOperationException`	400	No	The request uses a currently unsupported feature or construct.

IAM 身分驗證錯誤

這些錯誤只發生在有啟用 IAM 身分驗證的叢集。

下表包含錯誤代碼、訊息和 HTTP 狀態。

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 '*region*'.
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 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 `SignedHea`

Neptune Service Error Code	HTTP status	Message
		`der` , 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 ##`*.)
Future Date	403	Signature not yet current: *`20500224T213559Z`* is still later than *`20181108T225925Z`* (*`20181108T225425Z`* + *`5 ##`*.)
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.
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.

Neptune Service Error Code	HTTP status	Message
TooManyRequestsException	429	The rate of requests exceeds the maximum throughput. OK to retry.

資料庫叢集管理 API 錯誤訊息和代碼

這些 Amazon Neptune 錯誤與使用 AWS SDK 和 AWS CLI 建立和修改 Neptune 資源的 API 有關。

下表包含錯誤代碼、訊息和 HTTP 狀態。

Neptune Service Error Code	HTTP status	Message
AccessDeniedException	403	您沒有足夠存取權可執行此動作。
IncompleteSignature	400	請求簽署不符合 AWS 標準。
InternalFailure	500	因為不明的錯誤、例外或故障，處理請求失敗。
InvalidAction	400	請求的動作或操作無效。確認已正確輸入動作。
InvalidClientTokenId	403	提供的 X.509 憑證或 AWS 存取金鑰 ID 不存在於我們的記錄中。
InvalidParameterCombination	400	同時使用了不應搭配使用的參數。
InvalidParameterValue	400	為輸入參數提供了無效或超出範圍的值。
InvalidQueryParameter	400	為輸入參數提供了無效或超出範圍的值。
MalformedQueryString	400	查詢字串包含語法錯誤。

Neptune Service Error Code	HTTP status	Message
MissingAction	400	請求中遺失動作或必要參數。
MissingAuthenticationToken	403	請求必須包含有效 (已註冊) 的 AWS 存取金鑰 ID 或 X.509 憑證。
MissingParameter	400	未提供適用於指定動作的必要參數。
OptInRequired	403	AWS 存取金鑰 ID 需要訂閱服務。
RequestExpired	400	請求之送達時間為超過請求之日期戳記的 15 分鐘後或超過請求過期日期的 15 分鐘後 (例如預先簽章的 URL)，或者請求的日期戳記為未來 15 分鐘後。
ServiceUnavailable	503	由於伺服器暫時故障，請求失敗。
ThrottlingException	500	由於請求調節，因此請求遭到拒絕。
ValidationError	400	輸入不符合 AWS 服務規定的限制。

Neptune 載入器的錯誤和饋送訊息

以下訊息由 Neptune 載入器的 `status` 端點傳回。如需更多詳細資訊，請參閱 Get-Status API。

下表是載入器的饋送代碼和描述。

Error or Feed Code	Description
LOAD_NOT_STARTED	已記錄載入，但未開始。

Error or Feed Code	Description
LOAD_IN_PROGRESS	指出載入正在進行中，並指定目前載入的檔案數目。 當載入器剖析檔案時，它會建立一或多個要平行載入的區塊。因為單一檔案可以產生多個區塊，所以此訊息隨附的檔案計數通常小於大量載入程序所使用的執行緒數目。
LOAD_COMPLETED	載入已完成，無任何錯誤或錯誤在可接受的閾值內。
LOAD_CANCELLED_BY_USER	使用者已取消載入。
LOAD_CANCELLED_DUE_TO_ERRORS	系統因為錯誤已取消載入。
LOAD_UNEXPECTED_ERROR	載入失敗並發生未預期的錯誤。
LOAD_FAILED	由於一或多個錯誤，載入失敗。
LOAD_S3_READ_ERROR	由於間歇性或暫時性的 Amazon S3 連線問題導致饋送失敗。如有任何饋送收到此錯誤，整體載入狀態將設為 LOAD_FAILED。
LOAD_S3_ACCESS_DENIED_ERROR	存取 S3 儲存貯體遭拒。如有任何饋送收到此錯誤，整體載入狀態將設為 LOAD_FAILED。
LOAD_COMMITTED_W_WRITE_CONFLICTS	載入資料遞交時發生未解決的寫入衝突。 將嘗試解決寫入衝突載入在單獨的交易和更新饋送的負載狀態的進度。如果最終饋送狀態為 LOAD_COMMITTED_W_WRITE_CONFLICTS，請嘗試重新繼續載入，它可能會成功而不會發生寫入衝突。寫入衝突通常與錯誤輸入資料無關，但資料中的重複會提高發生寫入衝突的可能性。
LOAD_DATA_DEADLOCK	Load was automatically rolled back due to deadlock.

Error or Feed Code	Description
LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETED	饋送失敗，因為檔案已在載入開始後刪除或更新。
LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED	未執行載入請求，因為其相依性檢查失敗。
LOAD_IN_QUEUE	載入請求已排入佇列，正在等待執行。
LOAD_FAILED_INVALID_REQUEST	載入失敗，因為請求無效 (例如，指定來源/儲存貯體可能不存在，或檔案格式無效)。

Amazon Neptune 的引擎版本

Amazon Neptune 會定期發行引擎更新。

您可以使用執行個體狀態 API 或 Neptune 主控台，來判斷您目前安裝的引擎發行版本。版本號碼旨在說明您執行的是原始主要發行版本或次要版本或修補程式版本。如需版本編號的更多資訊，請參閱引擎版本編號。

如需有關一般更新的詳細資訊，請參閱 叢集維護。

從引擎版本 1.3.0.0 開始，引擎版本將會具有下表所示的結構。次要版本號碼是會針對 `AutoMinorVersionUpgrade` 處理進行評估的版本號碼。

版本	產品版本	主要版本	次要版本	修補程式版本	Status	Released	生命週期結束	升級至：
1.3.2.1	1	3	2	1	作用中	2024-06-20	2025-11-30	N/A
1.3.2.0	1	3	2	0	作用中	2024-06-10	2025-11-30	1.3.2.1
1.3.1.0	1	3	1	0	作用中	2024-03-06	2025-11-30	1.3.2.1
1.3.0.0	1	3	0	0	作用中	2023-11-15	2025-11-30	1.3.2.1

下表列出自 1.0.1.0 以來的所有引擎版本，以及有關版本的資訊。 end-of-life您可以使用此表中的日期來規劃測試和升級週期。

版本	主要版本	次要版本	Status	Released	生命週期結束	升級至：
1.2.1.1	1.2	1.1	作用中	2024-03-11	2025-03-06	1.3.0.0

版本	主要版本	次要版本	Status	Released	生命週期結束	升級至：
1.2.1.0	1.2	1.0	作用中	2023-03-08	2025-03-06	1.3.0.0
1.2.0.2	1.2	0.2	作用中	2022-11-16	2025-03-06	1.3.0.0
1.2.0.1	1.2	0.1	作用中	2022-10-26	2025-03-06	1.3.0.0
1.2.0.0	1.2	0.0	作用中	2022-07-21	2025-03-06	1.3.0.0
1.1.1.0	1.1	1.0	作用中	2022-04-19	2024-10-31	1.2.1.0
1.1.0.0	1.1	0.0	作用中	2021-11-19	2024-10-31	1.1.1.0
1.0.5.1	1.0	5.1	已棄用	2021-10-01	2023-01-30	1.1.0.0
1.0.5.0	1.0	5.0	已棄用	2021-07-27	2023-01-30	1.1.0.0
1.0.4.2	1.0	4.2	已棄用	2021-06-01	2023-01-30	1.1.0.0
1.0.4.1	1.0	4.1	已棄用	2020-12-08	2023-01-30	1.1.0.0
1.0.4.0	1.0	4.0	已棄用	2020-10-12	2023-01-30	1.1.0.0
1.0.3.0	1.0	3.0	已棄用	2020-08-03	2023-01-30	1.1.0.0
1.0.2.2	1.0	2.2	已棄用	2020-03-09	2022-07-29	1.0.3.0
1.0.2.1	1.0	2.1	已棄用	2019-11-22	2022-07-29	1.0.3.0
1.0.2.0	1.0	2.0	已棄用	2019-11-08	2020-05-19	1.0.3.0
1.0.1.2	1.0	1.2	已棄用	2019-10-15	—	—
1.0.1.1	1.0	1.1	已棄用	2019-08-13	—	—
1.0.1.0.*	1.0	1.0.*	已棄用	2019-07-02 及之前	—	—

主要引擎版本 end-of-life 規劃

Neptune 引擎版本幾乎總是在行事曆季度結束時達到其生命週期結束。僅在發生重要安全性或可用性問題時，才會發生例外狀況。

當引擎版本到達其生命週期結束時，您需要將 Neptune 資料庫升級至更新的版本。

一般而言，Neptune 引擎版本繼續可供使用，如下所示：

- 次要引擎版本：次要引擎版本在發行後至少仍能使用 6 個月。
- 主要引擎版本：主要引擎版本在發行後至少仍能使用 12 個月。

在引擎版本到達生命週期結束前至少 3 個月， AWS 會向與您 AWS 帳戶關聯的電子郵件地址發送自動電子郵件通知，並將相同的消息發佈到您的 AWS Health 儀表板。這會讓您有時間規劃和準備升級。

當引擎版本到達其生命週期結束時，您將再也無法使用該版本建立新的叢集或執行個體，自動擴展也無法使用該版本建立執行個體。

實際到達生命週期結束的引擎版本將在維護時段期間自動升級。在引擎版本生命週期結束前 3 個月傳送給您的訊息將包含此自動更新所涉及內容的詳細資料，包括您將自動升級至哪個版本、對資料庫叢集的影響，以及我們建議的動作。

> **Important**
>
> 您負責將資料庫引擎版本保持在最新狀態。 AWS 會促使所有客戶將其資料庫升級至最新的引擎版本，以便得益於最新的安全性、隱私權和可用性防禦措施。如果在超過棄用日期的不受支援引擎或軟體 (「舊版引擎」) 上操作資料庫，您可能會面臨更大的安全性、隱私權和操作風險 (包括停機事件)。
>
> 在任何引擎上操作您的數據庫均受制於管理您使用 AWS 服務的協議。舊版引擎無法正常使用。 AWS 如果認為傳統引擎對服務、其關係企業或任何第三方構成安全性或責任風險，或有傷害風險，則 AWS 不再對 Legacy Engine 提供支援，且 AWS 可能隨時限制對任何 Legacy Engine 的存取或使用任何傳統引擎。 AWS您決定繼續在舊版引擎中執行您的內容，可能會導致您的內容無法使用、損毀或無法復原。在舊版引擎上執行的資料庫受服務水準協議 (SLA) 例外狀況約束。
>
> 在舊版引擎上執行的資料庫和相關軟體包含錯誤、瑕疵和/或有害元件。因此，儘管協議或服務條款有任何相反的規定，仍「按原樣」 AWS 提供傳統引擎。

Amazon Neptune 引擎

截至二零二零年六月二十日，引擎版本 1.3.2.1 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 引擎 1.3.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.3.0.0 之前的引擎版本升級至引擎 1.3.0.0 版或更新版本，則必須使用參數群組系列 neptune1.3 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 neptune1 或 neptune1.2，而這些參數群組不會使用 1.3.0.0 版及更新版本。如需詳細資訊，請參閱Amazon Neptune 參數群組。

此引擎版本中已修正的瑕疵

openCypher 修正

- 針對包含具有SKIP和LIMIT作為參數的內部WITH子句的參數化查詢，在查詢計畫快取功能中偵測到錯誤。SKIP/LIMIT 值未正確參數化，因此，具有不同參數值的相同快取查詢計畫的後續執行仍會傳回與第一次執行相同的結果。這個問題已被修正。

```
# 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
  } ]
}%
```

- 修正當傳遞的參數尚未存在於資料庫中`InternalFailureException`時，參數化變異查詢擲回的錯誤。
- 修復了在查詢資源清理期間遇到競爭條件後，參數化 Bolt 查詢會卡住的錯誤。

1.3.2.1 中的變化從 1.3.2.0 延續

從引擎版本 1.3.2.0 進行的改進

一般改善

- Support TLS 版本 1.3，包括加密套件 TLS 1.3 是一個選項-TLS 1.2 仍然是最低限度。
- 此版本的開放密碼擴展支持是在 Lab_mode 中。我們鼓勵您對其進行測試。

小精靈的改進

- TinkerPop 3.7.x 升級

  - 提供了大量的小精靈語言擴展。
    - 處理字符串，列表和日期的新步驟。
    - 用於指定與步驟基數的新語法。`mergeV()`
    - `union()`現在可以用作開始步驟。
    - 若要進一步了解 3.7.x 中的變更，請參閱TinkerPop 升級文件。
  - 在升級 Java 的客戶端 Gemlin 語言驅動程序時，請注意，序列化程序類沒有進行一些重命名。如果有指定，您將需要更新組態檔案和程式碼中的套件和類別命名。
- `StrictTimeoutValidation`(僅當透過 labmode 透`StrictTimeoutValidation`過包含啟用時`StrictTimeoutValidation=enabled`)：當`StrictTimeoutValidation`參數的值為時`enabled`，指定為請求選項或查詢提示的每個查詢逾時值不能超過參數群組中全域設定的值。在這種情況下，Neptune 將拋出一個`InvalidParameterException`。當值為時，可以在`/status`端點上的回應中確認此設定`disabled`，而在 Neptune 版本 1.3.2.0 和 1.3.2.1 中，此參數的預設值為。`Disabled`

openCypher 改進

- 低延遲查詢和輸送量效能改善：低延遲 OpenCypher 查詢的整體效能改進。新版本還提高了此類查詢的吞吐量。使用參數化查詢時，改進會更顯著。
- 查詢計畫快取的 Support：當查詢提交至 Neptune 時，查詢字串會剖析、最佳化，並轉換成查詢計畫，然後由引擎執行。應用程序通常由具有不同值實例化的常見查詢模式支持。查詢計畫快取可藉由快取查詢計畫，進而避免剖析和最佳化這類重複模式，以減少整體延遲。
- 不同彙總查詢的效能改善。
- 涉及可空變量的連接的性能改進。
- 涉及不等於 id（節點/關係）謂詞的查詢的性能改進。
- 對日期時間功能的擴展支持（僅通過包括`DatetimeMillisecond=enabled`實驗室模式`DatetimeMillisecond`啟用。如需詳細資訊，請參閱 Neptune OpenCypher 實現中的時間支持（Neptune 分析和 Neptune 數據庫 1.3.2.0 及更高版本）。

從引擎版本 1.3.2.0 延續的瑕疵修正

一般改善

- 更新驗證圖形值區的存取權時，Neptune EML 錯誤訊息。

Gremlin 修正

- 已修正 DFE 查詢轉譯中遺失的標籤資訊，適用於非路徑貢獻步驟包含標籤的案例。例如：

```
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')
```

- 修正 DFE 查詢轉譯中的`NullPointerException`錯誤，此錯誤會在兩個 DFE 片段中執行查詢，並將第一個片段最佳化為無法滿足的節點時發生。例如：

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- 修復了`InternalFailureException`當查詢包含 by () 調製器 ValueTraversal 內部並且其輸入為 Map 時，Neptune 可能會拋出一個錯誤。例如：

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

openCypher 修正

- 改善 RESLONE 作業 (例如，將值清單擴充為個別值)，以協助防止記憶體不足 (OOM) 情況。例如：

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- 修復了通過展開注入 id 的多個 MERGE 操作的情況下的自定義 ID 優化。例如：

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- 在規劃具有屬性存取的複雜查詢和具有雙向關係的多個躍點時，記憶體爆炸的問題已修正 例如：

```
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
```

- 固定的聚合查詢與 null 作為組變量。例如：

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

SPARQL 修正

- 修正 SPARQL 剖析器以改善大型查詢的剖析時間，例如 INSERT DATA 包含許多三元組和大型記號。

此版本支援的查詢語言版本

在將資料庫叢集升級至 1.3.2.1 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.7.1`

- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.3.2.1 的升級路徑

您可以從引擎版本 1.2.0.0 或更新版本升級至此版本。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要此 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，Sup AWS port 團隊可以透過社群論壇和進AWS 階 Support 取得。

Amazon Neptune 引擎

截至 2024 年 6 月 10 日，引擎版本 1.3.2.0 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 引擎 1.3.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.3.0.0 之前的引擎版本升級至引擎 1.3.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.3` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1` 或 `neptune1.2`，而這些參數群組不會使用 1.3.0.0 版及更新版本。如需詳細資訊，請參閱Amazon Neptune 參數群組。

> **Warning**
>
> 我們在內部`WITH`子句中使用`skip`或`limit`參數化時，在查詢計劃緩存中檢測到問題。若要避免這個問題，請在提交包含參數化略過和/或 limit 子子子子句的查詢`QUERY:PLANCACHE "disabled"`時新增查詢提示。或者，您可以將值硬式編碼到查詢中。如需更多資訊，請參閱緩解查詢計劃緩存問題。

此引擎版本的改進

一般改善

- Support TLS 版本 1.3，包括加密套件 TLS 1.3 是一個選項-TLS 1.2 仍然是最低限度。

小精靈的改進

- TinkerPop 3.7.x 升級
  - 提供了大量的小精靈語言擴展。
    - 處理字符串，列表和日期的新步驟。
    - 用於指定與步驟基數的新語法。`mergeV()`
    - `union()`現在可以用作開始步驟。
    - 若要進一步了解 3.7.x 中的變更，請參閱TinkerPop 升級文件。

- 在升級 Java 的客戶端 Gemlin 語言驅動程序時，請注意，序列化程序類沒有進行一些重命名。如果有指定，您將需要更新組態檔案和程式碼中的套件和類別命名。

- `StrictTimeoutValidation`(僅當透過 labmode 透`StrictTimeoutValidation`過包含啟用時`StrictTimeoutValidation=enabled`)：當`StrictTimeoutValidation`參數的值為時`enabled`，指定為請求選項或查詢提示的每個查詢逾時值不能超過參數群組中全域設定的值。在這種情況下，Neptune 將拋出一個`InvalidParameterException`。當值為時，可以在`/status`端點上的回應中確認此設定`disabled`，而在 Neptune 版本 1.3.2.0 中，此參數的預設值為。`Disabled`

openCypher 改進

- 低延遲查詢和輸送量效能改善：低延遲 OpenCypher 查詢的整體效能改進。新版本還提高了此類查詢的吞吐量。使用參數化查詢時，改進會更顯著。
- 查詢計畫快取的 Support：當查詢提交至 Neptune 時，查詢字串會剖析、最佳化，並轉換成查詢計畫，然後由引擎執行。應用程序通常由具有不同值實例化的常見查詢模式支持。查詢計畫快取可藉由快取查詢計畫，進而避免剖析和最佳化這類重複模式，以減少整體延遲。
- 不同彙總查詢的效能改善。
- 涉及可空變量的連接的性能改進。
- 涉及不等於 id（節點/關係）謂詞的查詢的性能改進。
- 對日期時間功能的擴展支持（僅通過包括`DatetimeMillisecond=enabled`通過實驗室模式`DatetimeMillisecond`啟用。如需詳細資訊，請參閱 Neptune OpenCypher 實現中的時間支持（Neptune 分析和 Neptune 數據庫 1.3.2.0 及更高版本）。

此引擎版本中已修正的瑕疵

一般改善

- 更新驗證圖形值區的存取權時，Neptune EML 錯誤訊息。

Gremlin 修正

- 已修正 DFE 查詢轉譯中遺失的標籤資訊，適用於非路徑貢獻步驟包含標籤的案例。例如：

```
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')
```

- 修正 DFE 查詢轉譯中的NullPointerException錯誤，此錯誤會在兩個 DFE 片段中執行查詢，並將第一個片段最佳化為無法滿足的節點時發生。例如：

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- 修復了InternalFailureException當查詢包含 by () 調製器 ValueTraversal 內部並且其輸入為 Map 時，Neptune 可能會拋出一個錯誤。例如：

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

openCypher 修正

- 改善 RESLONE 作業 (例如，將值清單擴充為個別值)，以協助防止記憶體不足 (OOM) 情況。例如：

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- 修復了通過展開注入 id 的多個 MERGE 操作的情況下的自定義 ID 優化。例如：

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- 在規劃具有屬性存取的複雜查詢和具有雙向關係的多個躍點時，記憶體爆炸的問題已修正 例如：

```
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
```

- 固定的聚合查詢與 null 作為組變量。例如：

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

SPARQL 修正

- 修正 SPARQL 剖析器以改善大型查詢的剖析時間，例如 INSERT DATA 包含許多三元組和大型記號。

緩解查詢計劃緩存問題

對於版本 1.3.2.0，我們在內部`WITH`子句中使用`skip`或`limit`參數化時檢測到查詢計劃緩存中的問題。例如：

```
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}
```

在這種情況下，來自第一個計劃的 skip 和 limit 的參數值也會套用至後續查詢，從而導致非預期的結果。

缓解

若要避免這個問題，請在提交包含參數化略過和/或 limit 子子子子句的查詢`QUERY:PLANCACHE "disabled"`時新增查詢提示。或者，您可以將值硬式編碼到查詢中。

選項 1：使用查詢提示停用計劃快取：

```
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}
```

選項 2：使用硬編碼值進行跳過和限制：

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip 2 LIMIT 3
RETURN n.name, n.age

parameters={"age": 21}
```

此版本支援的查詢語言版本

在將資料庫叢集升級至 1.3.2.0 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.7.1`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.3.2.0 的升級路徑

您可以從引擎版本 1.2.0.0 或更新版本升級至此版本。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要此 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，Sup AWS port 團隊可以透過社群論壇和進AWS 階 Support 取得。

Amazon Neptune 引擎

截至 2024 年 3 月 06 日，引擎版本 1.3.1.0 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 引擎 1.3.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.3.0.0 之前的引擎版本升級至引擎 1.3.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.3` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1` 或 `neptune1.2`，而這些參數群組不會使用 1.3.0.0 版及更新版本。如需詳細資訊，請參閱Amazon Neptune 參數群組。

此引擎版本的改進

一般改善

- Neptune 改進了配置文件/說明中顯示的警告。
- 從 TLS 交涉期間使用的預設命名群組中移除過時的 NIST EC 曲線。移除的曲線為截面 409k1、段 409r1 及第 571k1 節。

小精靈的改進

- 改善 DFE 統計資料運算，以避免非常高的無伺服器執行個體 NCU。
- 內部的小精靈性能改進。

此引擎版本中已修正的瑕疵

Gremlin 修正

- 小靈 DFE 查詢計劃的其他改進。
- 錯誤修復了一個可選的遍歷小鬼查詢，例如，對於形式 `G.V（）的查詢 .hasLabel（'人'）.group（）.by（id（））.by（__.in（'朋友'）.id（）.fold（））`，沒有人沒有朋友邊緣得到分組。
- 修正了一個包含by調製器內聯合步驟的 Gremlin 查詢會導致在使用 DFE 引擎執行時返回錯誤的錯誤。
- 修正在連線到僅供讀取複本時，無法在 Grimlin 工作階段中執行的唯讀查詢運作的錯誤。
- 針對 Grimlin 的初始 Websocket 連線要求，而稽核記錄中未出現 IAM ARN 的錯誤修正。
- 合併步驟，識別 DFE 的步驟覆蓋範圍。
- 整個 DFE 計劃的特性集優化。

openCypher 修正

- 錯誤修復了 OpenCypher SET 子句中的錯誤，以允許對非變量表達式進行設置（即：匹配（N：測試）集（當 n.prop = 2，然後 n 結束時）。
- 錯誤修復了 OpenCypher 查詢失敗，涉及聚合和排序。
- 改進了包含靜態映射的大列表的展開。
- 錯誤修正使用具有重複值的自訂識別碼的 OpenCypher 合併查詢。

SPARQL 修正

- 修復了有關可選模式中變量作用域的 SPARQL 錯誤。

此版本支援的查詢語言版本

在將資料庫叢集升級至 1.3.1.0 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.5`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.3.1.0 的升級路徑

您可以從引擎版本 1.2.0.0 或更新版本升級至此版本。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 --no-apply-immediately，而不是 --apply-immediately。若要執行主要版本升級，需要此 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，Sup AWS port 團隊可以透過社群論壇和進AWS 階 Support 取得。

Amazon Neptune 引擎 1.3.0.0 版 (2023-11-15)

截至 2023-11-15，引擎 1.3.0.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 引擎 1.3.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.3.0.0 之前的引擎版本升級至引擎 1.3.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.3` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1` 或 `neptune1.2`，而這些參數群組不會使用 1.3.0.0 版及更新版本。如需詳細資訊，請參閱Amazon Neptune 參數群組。

這個引擎版本的新功能

- 已發行 Neptune 資料 API。

Amazon Neptune 資料 API 為超過 40 種 Neptune 的資料操作提供 SDK 支援，包括資料載入、查詢執行、資料查詢和機器學習。其支援所有三種 Neptune 查詢語言(Gremlin、openCypher 和 SPARQL)，並且適用於所有 SDK 語言。它會自動簽署 API 請求，並大幅簡化將 Neptune 整合至您應用程式的操作。

- 新增OpenSearch無伺服器與 Neptune 整合的支援。

這個引擎版本的改善項目

Neptune 引擎更新的改進

Neptune 已變更發布引擎更新的方式，以讓您更緊密控制更新程序。Neptune 現在不會針對非中斷變更發行修補程式，而是發行可使用 AutoMinorVersionUpgrade 執行個體欄位控制的次要版本，以及您可以透過訂閱RDS-EVENT-0156 事件來接收通知的次要版本。

如需有關這些變更的詳細資訊，請參閱維護 Amazon Neptune 資料庫叢集。

傳輸中加密改進

Neptune 不再支援下列密碼套件：

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

Neptune 僅支援使用 TLS 1.2 的下列強式密碼套件：

- 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

Gremlin 改進

- 在 DFE 引擎中新增了對下列 Gremlin 步驟的支援：
  - FoldStep
  - GroupStep

  - GroupCountStep
  - TraversalMapStep
  - UnfoldStep
  - LabelStep
  - PropertyKeyStep
  - PropertyValueStep
  - AndStep
  - OrStep
  - ConstantStep
  - CountGlobalStep
- 最佳化的 Gemlin DFE 查詢計畫，以避免在使用 by() 調變時進行完整的頂點掃描。
- 大幅改善低基數和低延遲查詢的效能。
- 已新增對 TinkerPop 0r篩選述詞的 DFE 支援。
- 針對如下所示的查詢，改進 DFE 對同一鍵上周遊篩選條件的支援：

```
g.withSideEffect("Neptune#useDFE", true)
 .V()
 .has('name', 'marko')
 .and(
   or(
     has('name', eq("marko")),
     has('name', eq("vardas"))
   )
 )
```

- 改進對 fail() 步驟的錯誤處理。

openCypher 改進

- 大幅改善低基數和低延遲查詢的效能。
- 當查詢包含許多節點類型時，改善查詢規劃效能。
- 減少所有 VLP 查詢的延遲。
- 透過移除單一節點模式查詢的備援管道聯結，藉此改善效能。
- 改進包含週期之多躍點模式的查詢效能，如下所示：

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

SPARQL 改進

- 引入新的 SPARQL 運算子：PipelineHashIndexJoin。
- 改善 SPARQL 查詢的 URI 驗證效能。
- 藉由批次解析字典詞彙，改善 SPARQL 全文檢索搜尋查詢的效能。

此引擎版本中修正的缺陷

Gremlin 修正

- 已修正 Gemlin 錯誤，其中針對不是在 DFE 引擎中原生處理的步驟，檢查 Gremlin 查詢狀態端點是否有子周遊中具有述詞的查詢時，將會發生交易洩漏。
- 已修正 Gremlin 錯誤，其中未在 DFE 引擎的 by() 周遊下將 valueMap() 最佳化。
- 已修正 Gremlin 錯誤，其中未將連接至 UnionStep 的步驟標籤各別傳播至其子周遊的最後一個路徑元素。
- 修復了一個小鬼錯誤，其中查詢會失敗，因為它包含太多 TinkerPop 步驟，然後無法清理。
- 已修正 Gremlin 錯誤，其中 NullPointerException 會在 mergeV 和 mergeE 步驟中擲回。
- 已修正 Grimlin 錯誤，其中 order() 將不會正確排序字串輸出，因為這些字串輸出中有一些包含空格字元時。
- 已修正在 DFE 引擎中處理 valueMap 步驟時所發生的 Gemlin 正確性問題。
- 已修正 GroupStep 或 GroupCountStep 嵌套於鍵周遊中時發生的 Gemlin 正確性問題。

openCypher 修正

- 已修正涉及 NULL 字元錯誤處理的 openCypher 錯誤。
- 已修正 openCypher Bolt 交易處理中的錯誤。

SPARQL 修正

- 已修正遞迴函式中的值未正確解析的 SPARQL 錯誤。

- 已修正使用 VALUES 子句注入大量值時，導致效能下降的 SPARQL 錯誤。
- 已修正 SPARQL 錯誤，該錯誤會導致語言標記文字的REGEX 運算子呼叫永遠不會成功。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.3.0.0 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎 1.3.0.0 版的升級途徑

您可以從引擎版本 1.2.0.0 或更新版本升級至此版本。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要此 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，Sup AWS port 團隊可以透過社群論壇和進AWS 階 Support 取得。

Amazon Neptune 引擎

截至 2024 年 3 月 11 日，引擎版本 1.2.1.1 正在普遍部署中。請注意，新版本需要數天才能在每個區域推出。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需詳細資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且`UndoLogsListSize` CloudWatch 指標必須降至零，才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果`UndoLogsListSize` CloudWatch 量度非常大，開啟支援案例可協助您探索其他策略來降低指標。
- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑

可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

一般改善

Neptune 改進了配置文件/說明中顯示的警告。

Gremlin 改進

- 改善 DFE 統計資料運算，以避免非常高的無伺服器執行個體 NCU。
- 內部的小精靈性能改進。

此引擎版本中修正的缺陷

Gremlin 修正

- 錯誤修復了與小鬼 DFE 引擎查詢計劃的順序。
- 修正錯誤與小鬼 out-of-memory 錯誤時，最初報告為. InternalFailureException
- 針對成功的初始 websocket 連線要求，稽核記錄中未顯示 IAM ARN 的錯誤修正。
- 當 TinkerPop 工作階段中的查詢失敗時，即使所有這些都是唯讀且連線至讀取器執行個體，也會啟用工作階段的 Gemlin 查詢的錯誤修正。

openCypher 修正

- 錯誤修復了 OpenCypher SET 子句中的錯誤，以允許對非變量表達式進行設置（即：匹配（N：測試）集（當 n.prop = 2，然後 n 結束時）。
- 錯誤修復了 OpenCypher 查詢失敗，涉及聚合和排序。
- 改進了包含靜態映射的大列表的展開。
- 錯誤修正使用具有重複值的自訂識別碼的 OpenCypher 合併查詢。

SPARQL 修正

- 修正了 SPARQL DFE 查詢規劃工具中的錯誤。

- 與 BIND 和選用關鍵字搭配使用時，SPARQL 的錯誤修正。

此版本支援的查詢語言版本

在將資料庫叢集升級至 1.2.1.1 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.2`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.1.1 的升級路徑

您可以從引擎版本 1.2.0.0 或更新版本升級至此版本。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要此 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
--db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，Sup AWS port 團隊可以透過社群論壇和進AWS 階 Support 取得。

Amazon Neptune 引擎 1.2.1.0 版 (2023 年 3 月 8 日)

截至 2023 年 3 月 8 日，引擎 1.2.1.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此版本的後續修補程式版本

- 版本：1.2.1.0.R2 (2023 年 5 月 2 日)
- 版本：1.2.1.0.R3 (2023 年 6 月 13 日)
- 版本：1.2.1.0.R4 (2023 年 8 日 10 日)
- 版本：1.2.1.0.R5 (2023 年 9 月 2 日)
- 版本：1.2.1.0.R6 (2023 年 9 月 12 日)
- 版本：1.2.1.0.R7 (2023-10-06)

這個引擎版本的新功能

- 已新增對 TinkerPop 3.6.2 的支援，這會新增許多新的 Gremlin 功能，例如新的 `mergeV()`、`mergeE()`、`element()` 和 `fail()` 步驟。`mergeV()` 和 `mergeE()` 步驟特別值得注意，因為它們提供了一個期待已久的宣告式選項，用於執行類似 Upsert 的操作，這應該大大簡化現有的程式碼模式，並使 Gemlin 更容易讀取。3.6.x 版也新增了 regex 述詞、將一個新的過載新增到採取 `Map` 的步驟 `property()`，以及新增了 `by()` 調變行為的主要修訂，進而在使用它的所有步驟中都更加一致。

  如需 3.6 版中變更的相關資訊，以及升級時應考量的事項，請參閱 TinkerPop 變更日誌與升級頁面。

  如果您使用 `fold().coalesce(unfold(), <mutate>)` 進行條件式插入，我們建議您移轉至這裡和這裡所述的新 `mergeV/E()` 語法。Neptune 會對 `Merge` 使用比 `Coalesce` 更窄的鎖定模式，這可以減少並行修改例外狀況 (CME)。

  如需有關此 TinkerPop 版本中可用的新功能的詳細資訊，請參閱 Stephen Mallette 的部落格探索 Amazon Neptune 中 Apache TinkerPop 3.6.x 的新功能。
- 已新增對 R6i 執行個體類型的支援，此執行個體類型是由第三代 Intel Xeon 可擴展處理器提供。這些非常適合記憶體密集型工作負載，與同類 R5 執行個體類型相比，提供每個 vCPU 最多可提升 15% 的運算/價格效能，以及最多可提高 20% 的記憶體頻寬。
- 已同時新增屬性圖和 RDF 圖形的圖形摘要 API 端點，可讓您取得有關圖形的快速摘要報告。

  對於屬形 (PG) 圖，圖形摘要 API 會提供節點和邊緣標籤以及屬性索引鍵的唯讀清單，也會提供節點、邊緣和屬性的計數。對於 RDF 圖形，它會提供類別和述詞索引鍵的清單，以及四元組、主旨和述詞的計數。

新的圖形摘要 API 進行了以下變更：

- 已新增一個 GetGraphSummary 資料平面動作。
- 已新增 `rdf/statistics` 端點來取代現已遭棄用的 `sparql/statistics` 端點。
- 將統計資料狀態回應中的 `summary` 欄位名稱變更為 `signatureInfo`，以免將其與圖形摘要資訊混淆。先前的引擎版本會繼續在 JSON 回應中使用 `summary`。
- 將統計資料狀態回應中 `date` 欄位的精確度從分鐘變更為毫秒。先前的格式為 `2020-05-07T23:13Z` (分鐘精確度)，而新格式為 `2023-01-24T00:47:43.319Z` (毫秒精確度)。兩者都符合 ISO 8601 標準，但此變更可能會破壞現有的程式碼，取決於日期的剖析方式。
- 已在工作台中新增 `%statistics` 行魔法，可讓您擷取 DFE 引擎統計資料。
- 已在工作台中新增 `%summary` 行魔法，可讓您擷取圖形摘要資訊。

- 已新增慢查詢記錄，以記錄需要的執行時間超過指定閾值的查詢。您可以使用兩個新的動態參數 (即 neptune_enable_slow_query_log, 和 neptune_slow_query_log_threshold) 來啟用和控制慢查詢記錄。
- 已新增對兩個動態參數的支援，即新的叢集參數 neptune_enable_slow_query_log 和 neptune_slow_query_log_threshold。對動態參數進行變更時，它會立即生效，無需重新啟動任何執行個體。
- 已新增 Neptune 特定的 OpenCypher removeKeyFromMap() 函數，此函數會從映射中移除指定的索引鍵並傳回產生的新映射。

這個引擎版本的改善項目

- 已將 Gremlin DFE 支援延伸至具有本機範圍的 `limit` 步驟。
- 已在 DFE 引擎中新增對 Gremlin `DedupGlobalStep` 的 `by()` 調變支援。
- 已新增對 Gremlin `SelectStep` 和 `SelectOneStep` 的支援。
- 各種 Gemlin 運算子 (包括 `repeat`、`coalesce`、`store` 和 `aggregate`) 的效能改善和正確性修正。
- 已改善涉及 `MERGE` 和 `OPTIONAL MATCH` 的 openCypher 查詢效能。
- 已改善涉及常值映射清單的 `UNWIND` 的 OpenCypher 查詢效能。
- 已改善對 `id` 具有 `IN` 篩選條件的 OpenCypher 查詢效能。例如：

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- 已新增使用 BASE 陳述式，為 SPARQL 查詢指定基礎 IRI 的功能 (請參閱 查詢和更新的預設基礎 IRI)。
- 已縮短 Gremlin 和 OpenCypher 僅邊緣大量載入的載入處理等待時間。
- 已使大量載入在 Neptune 重新啟動時以非同步方式繼續，以避免在繼續嘗試失敗之前，因 Amazon S3 連線問題所造成的漫長等待時間。
- 已改善 SPARQL DESCRIBE 查詢的處理，這些查詢具有查詢提示設定為 "CBD" (簡潔的界限描述) 的 describeMode 查詢，以及涉及大量空白節點。

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中查詢傳回了字串 "null"，而不是 Bolt 和 SPARQL-JSON 中的 null 值。
- 已修正清單理解中的 OpenCypher 錯誤，此錯誤產生了空值，而不是為清單元素提供的值。
- 已修正位元組值未正確序列化的 OpenCypher 錯誤。
- 已修正 UnionStep 中在下列情況發生的 Gremlin 錯誤：輸入是周遊至子周遊內頂點的邊緣。
- 已修正 Grimlin 錯誤，此錯誤導致了與 UnionStep 相關聯的步驟標籤無法正確傳播到每個子周遊的最後一個步驟。
- 已針對其標籤跟在 repeat 步驟後面的 dedup 步驟修正了 Gremlin 錯誤，其中附加到 dedup 步驟的標籤無法在查詢中進一步使用。
- 已修正 Gremlin 錯誤，其中轉換 union 步驟內的 repeat 步驟失敗，發生內部錯誤。
- 已透過返回 Tinkerpop 修正了 DEF 查詢的 Grimlin 正確性問題，這些查詢具有 limit 做為非聯集步驟的子周遊。表單中像這樣的查詢會受到影響：

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- 已修正 SPARQL 錯誤，其中 SPARQL GRAPH 模式不會考慮 FROM NAMED 子句提供的資料集。
- 已修正 SPARQL 錯誤，其中具有某些 FROM 和/或 FROM NAMED 子句的 SPARQL DESCRIBE 並不總是正確地使用來自預設圖形的資料，而且有時會擲回例外狀況。請參閱關於預設圖形的 SPARQL DESCRIBE 行為。
- 已修正 SPARQL 錯誤，以便在拒絕 null 字元時傳回正確的例外狀況訊息。
- 已修正 SPARQL Explain 錯誤，此錯誤影響了包含 PipelinedHashIndexJoin 運算子的計劃。
- 已修正錯誤，此錯誤已在提交一個傳回常數值的查詢時，導致擲回內部錯誤。
- 已修正死鎖偵測器邏輯偶爾會使引擎沒有回應的問題。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.2.1.0 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.2`
- openCypher 版本：`Neptune-9.0.20190305-1.1`
- SPARQL 版本：`1.1`

引擎 1.2.1.0 版的升級途徑

您可以將任何大於或等於 1.1.0.0 的先前 Neptune 引擎版本手動升級至此版本。

> **Note**
>
> 從引擎 1.2.0.0 版開始，與 `1.2.0.0` 以前的引擎版本搭配使用的所有自訂參數群組和自訂叢集參數群組，現在必須使用參數群組系列 `neptune1.2` 重新建立。舊版已使用參數群組系列 `neptune1`，而這些參數群組將不會使用 `1.2.0.0` 以上的版本。如需更多資訊，請參閱Amazon Neptune 參數群組。

您不會自動升級至此主要版本。

升級至此版本

Amazon Neptune 1.2.1.0 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎更新，版本 1.2.1.0.R7 (2023-10-06)。

截至 2023-10-06，引擎版本 1.2.1.0.R7 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此引擎版本中修正的缺陷

- 已修正在某些情況下，未適當地關閉失敗交易的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.1.0.R7 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.2`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

升級至此版本

Amazon Neptune 1.2.1.0.R7 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎更新，版本 1.2.1.0.R6，於 2023 年 9 月 12 日發行。

截至 2023 年 9 月 12 日，引擎版本 1.2.1.0.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。
>
>   如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
> - 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的新功能

- 已發行 Neptune 資料 API。

  Amazon Neptune 資料 API 為載入資料、執行查詢、取得資料相關資訊，以及執行機器學習操作提供 SDK 支援。它支援 Neptune 中的 Gremlin 和 OpenCypher 查詢語言，並且可在所有 SDK 語言中使用。它會自動簽署 API 請求，並大幅簡化將 Neptune 整合至應用程式的操作。

此引擎版本中修正的缺陷

- 已修正慢查詢日誌啟用時，在高載入下可能導致 CPU 峰值的錯誤。
- 已修正 Gremlin 錯誤，其中新增邊緣及其後面跟著 `inV()` 或 `outV()` 的屬性引發了 `InternalFailureException`。
- 已修正 IAM 角色鏈結在某些情況下造成大量載入器效能降低的數個問題。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.1.0.R6 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.2`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

升級至此版本

Amazon Neptune 1.2.1.0.R6 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎更新，版本 1.2.1.0.R5，於 2023 年 9 月 2 日發行。

截至 2023 年 9 月 2 日，引擎版本 1.2.1.0.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。
>
>   如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
> - 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的新功能

- 已發行 Neptune 資料 API。

  Amazon Neptune 資料 API 為載入資料、執行查詢、取得資料相關資訊，以及執行機器學習操作提供 SDK 支援。它支援 Neptune 中的 Gremlin 和 OpenCypher 查詢語言，並且可在所有 SDK 語言中使用。它會自動簽署 API 請求，並大幅簡化將 Neptune 整合至應用程式的操作。

此引擎版本中修正的缺陷

- 已修正 Gremlin 錯誤，其中新增邊緣及其後面跟著 `inV()` 或 `outV()` 的屬性引發了 `InternalFailureException`。
- 已修正 IAM 角色鏈結在某些情況下造成大量載入器效能降低的數個問題。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.1.0.R5 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.2`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

升級至此版本

Amazon Neptune 1.2.1.0.R5 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎更新 1.2.1.0.R4 (2023 年 8 日 10 日)

截至 2023 年 8 月 10 日，引擎版本 1.2.1.0.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

在某些情況下，此引擎版本中引進的變更可能會導致您觀察到大量載入效能降低。因此，已暫時暫停升級至此版本，直到問題得到解決為止。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 為 Gremlin 新增 GraphSON-1.0 支援。若要使用 GraphSON-1.0，請傳遞其值如下的 `Accept header`：

```
application/vnd.gremlin-v1.0+json;types=false
```

此引擎版本中修正的缺陷

- 已修正 Gemlin 錯誤，其中在對於不是原生處理的步驟，檢查 Gremlin 查詢狀態端點是否有子周遊中具有述詞的查詢時，將會發生交易洩漏。
- 已修正 Bolt 交易處理中的 openCypher 錯誤。
- 已修正儲存層上可能導致當機的並行問題。
- 已修正慢查詢日誌中的錯誤，以確保它們停用時不是作用中。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.1.0.R4 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.5`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.1.0.R4 的升級途徑

升級至此版本

Amazon Neptune 1.2.1.0.R4 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.2.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎更新 1.2.1.0.R3 (2023 年 6 日 13 日)

截至 2023 年 6 月 13 日，引擎版本 1.2.1.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

在某些情況下，此引擎版本中引進的變更可能會導致您觀察到大量載入效能降低。因此，已暫時暫停升級至此版本，直到問題得到解決為止。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

如果 UndoLogsListSize CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的新功能

- 已新增使用 IAM 角色鏈結進行跨帳戶大量載入的支援。

這個引擎版本的改善項目

- 已改善 Gremlin 的 `fail()` 步驟，以區分它從泛型 `InternalFailureException` 產生的例外狀況，並確保提供給它的任何使用者提供的訊息都回傳給呼叫者。
- 已改善 `store`、`aggregate`、`cap`、`limit` 和 `hasLabel` 的 Gemlin 查詢引擎最佳化。
- 已新增對 openCypher 三角函數的支援：
  - `acos()`
  - `asin()`
  - `atan()`
  - `atan2()`
  - `cos()`
  - `cot()`
  - `degrees()`
  - `pi()`
  - `radians()`
  - `sin()`
  - `tan()`
- 已新增對數個 openCypher 彙總函數的支援：
  - `percentileDisc()`
  - `stDev()`

- 已新增對將 `datetime` 轉換為 `epochmillis` 的 openCypher `epochmillis()` 函數的支援。例如：

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

- 已新增對 openCypher 模數 (%) 運算子的支援。
- 已新增對 OpenCypher 靜態偵錯 Explain 工具的支援。
- 已新增對 openCypher `randomUUID()` 函數的支援。
- 已改善 openCypher 效能：
  - 已改善剖析器和查詢規劃器。
  - 已改善 DFE 引擎中的 CPU 使用率。
  - 已改善包含多個更新子句重複使用相同變數的查詢效能。範例如下：

```
MERGE (n {name: 'John'})
  or
MERGE (m {name: 'Jim'})
  or
MERGE (n)-[:knows {since: 2023}]#(m)
```

  - 已針對多躍點查詢模式最佳化查詢計劃，例如：

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

  - 透過參數化查詢改善了清單和映射注入的效能。例如：

```
UNWIND $idList as id MATCH (n {`~id`: id})
RETURN n.name
```

  - 已改善包含 `WITH` 的查詢執行，方法為使其成為適當的屏障。
  - 已最佳化，以避免 `Unfold` 和彙總函數中的值進行備援實體化。
- 已改善 SPARQL 查詢的效能，這些查詢在 `VALUES` 子句中包含大量靜態輸入，例如：

```
SELECT ?n WHERE { VALUES (?name) { ("John") ("Jim") ... many values ... } ?n a ?
n_type . ?n ?name . }
```

- 已改善 SPARQL CBD 查詢效能。

此引擎版本中修正的缺陷

- 已修正 Gemlin 錯誤，其中在查詢規劃階段期間，具有深層巢狀的長時間查詢造成了高 CPU 使用率和查詢逾時。
- 已修正 Gremlin 錯誤，其中在使用 `mergeV` 或 `mergeE` 時可能會擲回無效的 `NullPointerException`。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.1.0.R3 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.2`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.1.0.R3 的升級途徑

升級至此版本

Amazon Neptune 1.2.1.0.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
```

```
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎更新，版本 1.2.1.0.R2，於 2023 年 5 月 2 日發行。

截至 2023 年 5 月 2 日，引擎版本 1.2.1.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。
>
>   如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
> - 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已將 `enableInterContainerTrafficEncryption` 參數新增至所有 Neptune ML API，您可以使用此參數，在訓練或超參數調校工作中啟用和停用容器間流量加密。

- 已新增對 Gremlin `mergeV()` 和 `mergeE()` 的多標籤支援。

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中更新和傳回查詢並未適當地處理 `orderBy`、`limit` 或 `skip`。
- 已修正 OpenCypher 錯誤，此錯誤允許一個請求中包含的參數被另一個同時請求中包含的參數覆寫。
- 已修正 OpenCypher 錯誤，其中慢查詢日誌未包含正確的查詢時間。
- 已修正 Gremlin 錯誤，其中包含 `GroupCountStep` 的查詢作為字串提交時可能發生交易洩漏。
- 已修正慢查詢日誌啟用時 WebSocket 查詢失敗的 Gremlin 錯誤。
- 已修正 Grimlin 錯誤，其中 WebSocket 請求的慢查詢日誌中缺少儲存體計數器偵錯日誌。
- 已修正數個涉及 `mergeV()` 和 `mergeE()` 的 Gremlin 錯誤。
- 已修正 SPARQL 錯誤，其中錯誤估計了具名圖形查詢成本，導致次佳查詢計畫和記憶體不足錯誤。
- 已修正在啟用 IAM 的叢集上影響了 Gremlin 和 OpenCypher 查詢授權的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.1.0.R2 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.6.2`
- 支援的 Gremlin 最新版本：`3.6.2`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.1.0.R2 的升級途徑

升級至此版本

Amazon Neptune 1.2.1.0.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.2.0.2 版 (2022 年 11 月 20 日)

截至 2022 年 11 月 20 日，引擎 1.2.0.2 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此版本的後續修補程式版本

- 版本：1.2.0.2.R2 (2022 年 12 月 15 日)
- 版本：1.2.0.2.R3 (2023 年 3 月 27 日)
- 版本：1.2.0.2.R4 (2023 年 5 月 8 日)
- 版本：1.2.0.2.R5 (2023 年 8 月 16 日)
- 版本：1.2.0.2.R6 (2023 年 9 月 12 日)

這個引擎版本的新功能

- 已在 Neptune ML 中引進 Gemlin 的即時歸納推論。
- 已引進 OpenCypher 延伸模組，支援指定實體的自訂 ID 值，而不是 Neptune 以其他方式產生的 UUID。指派自訂 ID 的能力使得從 Neo4j 遷移至 Neptune 更加容易。

> **Warning**
>
> OpenCypher 規格的這個延伸模組不與舊版相容，因為 `~id` 現在被視為保留的屬性名稱。如果您已在資料和查詢中使用 `~id` 做為屬性，則必須在升級至此版本之前將 `~id` 屬性遷移至新的屬性金鑰。

- 已新增數個新的 SPARQL DESCRIBE 模式以及查詢提示來設定它們。

這個引擎版本的改善項目

- 已改善 openCypher 效能，特別是針對 VLP 查詢。
- 已針對具有非終端限制的 Gemlin 查詢改善了 DFE 效能，例如：

```
g.withSideEffect('Neptune#useDFE',true).V().hasLabel('Student').limit(5).out('takesCourse')
```

此版本支援的查詢語言版本

將資料庫叢集升級至 1.2.0.2 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎 1.2.0.2 版的升級途徑

升級至此版本

Amazon Neptune 1.2.0.2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.2.R6 (2023 年 9 月 12 日)

截至 2023 年 9 月 12 日，引擎版本 1.2.0.2.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。
>
>   如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
> - 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此引擎版本中修正的缺陷

- 已修正 SPARQL 錯誤，其中在呼叫語言標記的常值時，`REGEX` 運算子永遠不會成功。
- 已修正導致大量載入效能下降的問題。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.2.R6 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.5`

- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.2.R6 的升級途徑

如果您執行的是引擎版本 `1.2.0.2`，您的 Neptune 資料庫叢集將在下一個維護時段自動升級至此維護修補程式版本。

升級至此版本

Amazon Neptune 1.2.0.2.R6 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.2.R5 (2023 年 8 月 16 日)

截至 2023 年 8 月 16 日，引擎版本 1.2.0.2.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

在某些情況下，此引擎版本中引進的變更可能會導致您觀察到大量載入效能降低。因此，已暫時暫停升級至此版本，直到問題得到解決為止。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此引擎版本中修正的缺陷

- 已修正 Grimlin 錯誤，其中 `order()` 將不會正確排序字串輸出，因為這些字串輸出中有一些包含空格字元時。

- 已修正 Gemlin 錯誤，其中在對於不是原生處理的步驟，檢查 Gremlin 查詢狀態端點是否有子周遊中具有述詞的查詢時，將會發生交易洩漏。
- 已修正 Bolt 交易處理中的 openCypher 錯誤。
- 已修正儲存層上可能導致當機的並行問題。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.2.R5 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.5`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.2.R5 的升級途徑

如果您執行的是引擎版本 `1.2.0.2`，您的 Neptune 資料庫叢集將在下一個維護時段自動升級至此維護修補程式版本。

升級至此版本

Amazon Neptune 1.2.0.2.R5 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
```

```
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.2.R4 (2023 年 5 月 8 日)

截至 2023 年 5 月 8 日，引擎版本 1.2.0.2.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。
>
>   如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
> - 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此引擎版本中修正的缺陷

- 已修正 SPARQL 錯誤，其中透過 `VALUES` 子句插入大量值可能導致效能降低。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.2.R4 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.6`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.2.R4 的升級途徑

如果您執行的是引擎版本 `1.2.0.2`，您的 Neptune 資料庫叢集將在下一個維護時段自動升級至此維護修補程式版本。

升級至此版本

Amazon Neptune 1.2.0.2.R4 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.2.R3 (2023 年 3 月 27 日)

截至 2023 年 3 月 27 日，引擎版本 1.2.0.2.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。
>
>   如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
> - 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 對於無伺服器資料庫叢集，已將最小容量設定變更為 1.0 NCU，並將最低有效的上限設定變更為 2.5 NCU。請參閱 Neptune Serverless 資料庫叢集中的容量擴展
- 已將 `enableInterContainerTrafficEncryption` 參數新增至所有 Neptune ML API，您可以使用此參數，在訓練或超參數調校工作中啟用和停用容器間流量加密。

此引擎版本中修正的缺陷

- 已修正 Gremlin 錯誤，其中 `option(Predicate)` 未被辨識為有效的 Gremlin 語法。
- 已修正 Gremlin 錯誤，此錯誤導致了查詢若失敗，將不會進行適當的清除，因為它們包含太多的步驟。
- 已透過返回 Tinkerpop 修正了影響 DEF 查詢的 Grimlin 正確性問題，這些查詢具有 `limit` 做為非聯集步驟的子周遊。以下是此類查詢的範例：

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- 已修正以字串形式提交的查詢包含 `GroupCountStep` 時潛在的 Gemlin 交易洩漏。
- 已修正 OpenCypher 錯誤，其中參數值的類型並未在清單或映射清單中正確地推論出來。
- 已修正 OpenCypher 錯誤，其中更新和傳回查詢並未適當地處理 `orderBy`、`limit` 或 `skip`。
- 已修正 OpenCypher 錯誤，此錯誤允許一個請求中包含的參數被另一個同時請求中包含的參數覆寫。
- 已修正 SPARQL 錯誤，其中在 `VALUES` 子句插入大量值可能導致效能降低。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.2.0.2.R3 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.6`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.2.R3 的升級途徑

如果您執行的是引擎版本 `1.2.0.2`，您的 Neptune 資料庫叢集將在下一個維護時段自動升級至此維護修補程式版本。

升級至此版本

Amazon Neptune 1.2.0.2.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
>   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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.2.R2 (2022 年 12 月 15 日)

截至 2022 年 12 月 15 日，引擎版本 1.2.0.2.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

如果 UndoLogsListSize CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已改善涉及 `MERGE` 和 `OPTIONAL MATCH` 的 openCypher 查詢效能。
- 已改善涉及常值映射清單的 `UNWIND` 的 OpenCypher 查詢效能。
- 已改善對 `id` 具有 `IN` 篩選條件的 OpenCypher 查詢效能。例如：

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- 各種 Gemlin 運算子 (包括 `repeat`、`coalesce`、`store` 和 `aggregate`) 的效能改善和正確性修正。

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中查詢傳回了字串 `"null"`，而不是 Bolt 和 SPARQL-JSON 中的 null 值。
- 已修正 Gremlin 錯誤，此錯誤導致了附加至 `UnionStep` 的步驟標籤不會傳播至其子周遊的最後一個路徑元素。
- 已修正 Gremlin 錯誤，此錯誤已導致 `valueMap()` 未在 DFE 引擎的 `by()` 周遊下進行最佳化。
- 已修正 Grimlin 錯誤，其中作為較長 Gemlin 交易一部分執行的讀取查詢不會鎖定資料列。
- 已修正稽核日誌錯誤，此錯誤已導致記錄不必要的資訊，以及某些欄位從日誌中遺失。
- 已修正稽核日誌錯誤，其中未記錄對啟用 IAM 的資料庫叢集發出的 HTTP 請求的 IAM ARN。
- 已修正查詢快取錯誤，以便限制用於寫入快取的增量記憶體。
- 已修正查閱快取錯誤，此錯誤涉及在寫入失敗時為查詢快取設定唯讀模式。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.2.R2 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.2.R2 的升級途徑

如果您執行的是引擎版本 `1.2.0.2`，您的 Neptune 資料庫叢集將在下一個維護時段自動升級至此維護修補程式版本。

升級至此版本

Amazon Neptune 1.2.0.2.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.2.0.1 版 (2022 年 10 月 26 日)

截至 2022 年 10 月 26 日，引擎 1.2.0.1 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。
>
>   如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
> - 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此版本的後續修補程式版本

- 維護版本：1.2.0.1.R2 (2022 年 12 月 13 日)
- 維護版本：1.2.0.1.R3 (2023 年 9 月 27 日)

這個引擎版本的新功能

- 已引進 Amazon Neptune Serverless，這是一種隨需自動擴展組態，可擴增您的資料庫叢集以滿足處理需求的增加，然後在需求減少時再次縮減。

這個引擎版本的改善項目

- 已改善 Gremlin `order-by` 查詢的效能。在 `NeptuneGraphQueryStep` 結尾具有 `order-by` 的 Gemlin 查詢現在會使用較大的區塊大小以取得更好的效能。這不適用於查詢計劃的內部 (非根) 節點上的 `order-by`。
- 已改善 Gremlin 更新查詢的效能。在新增邊緣或屬性時，必須鎖住頂點和邊緣以防刪除。此變更會消除交易內的重複鎖定，進而改善效能。
- 已透過將 dedup 向下推送至原生執行層，改善了在 `repeat()` 子查詢內部使用 `dedup()` 的 Gemlin 查詢效能。
- 已對 IAM 身分驗證錯誤新增了使用者易於了解的錯誤訊息。這些訊息現在會顯示您的 IAM 使用者或角色 ARN、資源 ARN，以及請求的未經授權動作清單。未經授權的動作清單可協助您查看所使用的 IAM 政策中可能遺漏或明確拒絕的內容。

此引擎版本中修正的缺陷

- 已修正 Gremlin 錯誤，其中在升級至 TinkerPop 3.5 之後使用 `PartitionStrategy`，不正確地造成了阻止執行周遊的錯誤，並顯示訊息「PartitionStrategy 不會使用匿名周遊」。
- 已修正涉及 `WherePredicateStep` 轉換的 Gemlin 正確性錯誤，其中 Neptune 的查詢引擎對使用 `where(P.neq('x'))` 和其變化的查詢產生不正確的結果。
- 已修正 `MERGE` 子句中在某些情況下導致了建立重複節點和邊緣的 OpenCypher 錯誤。
- 已修正處理查詢時發生的 SPARQL 錯誤，這些查詢在 `OPTIONAL` 子句內包含 `(NOT) EXISTS`，在某些情況下會遺失查詢結果。
- 已修正大量載入器錯誤，此錯誤在大量插入負載下造成了效能迴歸。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.2.0.1 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`

- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.1 的升級途徑

升級至此版本

Amazon Neptune 1.2.0.1 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.1.R3 (2023 年 9 月 27 日)

截至 2023 年 9 月 27 日，引擎版本 1.2.0.1.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列

neptune1.2 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 neptune1，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。

- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 UndoLogsListSize CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 UndoLogsListSize CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。
- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：request.setResourcePath("/openCypher"));。在其他語言中，/openCypher 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已將 enableInterContainerTrafficEncryption 參數新增至所有 Neptune ML API，您可以使用此參數，在訓練或超參數調校工作中啟用和停用容器間流量加密。
- 對於無伺服器資料庫叢集，已將最小容量設定變更為 1.0 NCU，並將最低有效的上限設定變更為 2.5 NCU。請參閱 Neptune Serverless 資料庫叢集中的容量擴展 *(((#####################))))#*

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中更新和傳回查詢並未適當地處理 orderBy、limit 或 skip。
- 已修正 OpenCypher 錯誤，此錯誤允許一個請求中包含的參數被另一個同時請求中包含的參數覆寫。
- 已修正 Bolt 交易處理中的 openCypher 錯誤。
- 已透過返回 Tinkerpop 修正了 DEF 查詢的 Grimlin 正確性問題，這些查詢具有 limit 做為非聯集步驟的子周遊。例如，對於像這樣的查詢：

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- 已修正 Gremlin 錯誤，其中查詢將會失敗，因為它包含了太多的 TinkerPop 步驟，接著並不會將其清除。
- 已修正 Grimlin 錯誤，其中 `order()` 將不會正確排序字串輸出，因為這些字串輸出中有一些包含空格字元時。
- 已修正 Gremlin 錯誤，其中當以字串和包含的 `GroupCountStep` 形式提交查詢時，可能發生交易洩漏。
- 已修正 Gemlin 錯誤，其中在對於不是原生處理的步驟，檢查 Gremlin 查詢狀態端點是否有子周遊中具有述詞的查詢時，將會發生交易洩漏。
- 已修正 Gremlin 錯誤，其中新增邊緣及其後面跟著 `inV()` 或 `outV()` 的屬性導致了 `InternalFailureException`。
- 已修正儲存層中的並行問題。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.1.R3 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.6`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.1.R3 的升級途徑

如果您執行的是引擎 1.2.0.1 版，您的 Neptune 資料庫叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Amazon Neptune 1.2.0.1.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.1.R2 (2022 年 12 月 13 日)

截至 2022 年 12 月 13 日，引擎版本 1.2.0.1.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

如果 UndoLogsListSize CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已改善涉及常值映射清單上 `UNWIND` 的 OpenCypher 查詢效能。
- 各種 Gemlin 運算子 (包括 `repeat`、`coalesce`、`store` 和 `aggregate`) 的效能改善和正確性修正。

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中查詢傳回了字串 `"null"`，而不是 Bolt 和 SPARQL-JSON 中的 null 值。
- 已修正稽核日誌錯誤，此錯誤已導致記錄不必要的資訊，以及某些欄位從日誌中遺失。
- 已修正查詢快取錯誤，以便限制用於寫入快取的增量記憶體。
- 已修正查閱快取錯誤，此錯誤涉及在寫入失敗時為查詢快取設定唯讀模式。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.1.R2 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.1.R2 的升級途徑

如果您執行的是引擎 1.2.0.1 版，您的 Neptune 資料庫叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Amazon Neptune 1.2.0.1.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.2.0.0 版 (2022 年 7 月 21 日)

截至 2022 年 7 月 21 日，引擎 1.2.0.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列

neptune1.2 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 neptune1，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。

- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 UndoLogsListSize CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 UndoLogsListSize CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：request.setResourcePath("/openCypher"));。在其他語言中，/openCypher 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

此版本的後續修補程式版本

- 版本：1.2.0.0.R2 (2022 年 10 月 14 日)
- 版本：1.2.0.0.R3 (2022 年 12 月 15 日)
- 版本：1.2.0.0.R4 (2023 年 9 月 29 日)

這個引擎版本的新功能

- 已新增對全球資料庫的支援。Neptune 全球資料庫跨越多個 AWS 區域，且由一個區域中的主要資料庫叢集，以及其他區域中最多五個次要資料庫叢集所組成。
- 根據資料平面動作，已新增在 Neptune IAM 政策中支援比之前所提供更精細的存取控制。這是一項重大變更，其中以棄用的 connect 動作為基礎的現有 IAM 政策必須進行調整，才能使用更精細的資料平面動作。請參閱IAM 政策的類型。
- 已改善讀取器執行個體可用性。之前，當寫入器執行個體重新啟動時，Neptune 叢集中的所有讀取器執行個體也會自動重新啟動。從引擎 1.2.0.0 版開始，讀取器執行個體會在寫入器重新啟動之後保

持作用中，進而改善讀取器可用性。您可以個別重新啟動讀取器執行個體，以取得參數群組變更。請參閱在 Amazon Neptune 中重新啟動資料庫執行個體。

- 已新增 neptune_streams_expiry_days 資料庫叢集參數，此參數可讓許您設定串流記錄在刪除之前保留在伺服器上的天數。範圍為 1 至 90，而預設值為 7。

這個引擎版本的改善項目

- 已改善 ByteCode 查詢的 Grimlin 序列化效能。
- Neptune 現在會使用 DFE 引擎處理文字述詞，以提升效能。
- Neptune 現在會使用 DFE 引擎處理 Grimlin `limit()` 步驟，包括非終端和子周遊限制。
- 已變更 Gemlin `union()` 步驟的 DFE 處理方式，以使用其他新功能，這表示參考節點會如預期般顯示在查詢設定檔中。
- 已透過序列化 DFE 內某些昂貴的聯結操作來改善這些操作的效能，最多可提升 5 倍。
- 已對 Gremlin DFE 引擎的 `OrderGlobalStep order(global)` 新增了 `by()` 調變支援。
- 已在 DFE 的解釋詳細資訊中新增了注入靜態值的顯示。
- 已改善剔除重複模式時的效能。
- 已在 Gremlin DFE 引擎中新增了順序保留支援。
- 已改善具有空篩選條件的 Gremlin 查詢效能，例如下列查詢：

```
g.V().hasId(P.within([]))
```

```
g.V().hasId([])
```

- 已改善 SPARQL 查詢使用的數值對 Neptune 太大而無法在內部呈現時的錯誤訊息。
- 已透過在串流停用時減少索引搜尋，改善了捨棄具有相關聯邊緣之頂點的效能。
- 已將 DFE 支援延伸至 `has()` 步驟的其他變體，特別是延伸至 `hasKey()`、`hasLabel()`，以及延伸至 `has()` 內字串/URI 的範圍述詞。這會影響如下的查詢：

```
// 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)"))
```

- 已新增 Neptune 特定的 OpenCypher join() 函數，此函數會將清單中的字串串連成單一字串。
- 已更新 Neptune 受管政策，以包含資料存取許可，以及新全球資料庫 API 的許可。

此引擎版本中修正的缺陷

- 已修正沒有指定 content-type 的 HTTP 請求將會自動失敗的錯誤。
- 已修正查詢最佳化工具中阻止在查詢內使用服務呼叫的 SPARQL 錯誤。
- 已修正 Turtle RDF 剖析器中的 SPARQL 錯誤，其中 Unicode 資料的特定組合導致了失敗。
- 已修正 SPARQL 錯誤，其中 GRAPH 和 SELECT 子句的特定組合產生了不正確的查詢結果。
- 已修正 Grimlin 錯誤，此錯誤對於在聯集步驟中使用任何篩選步驟的查詢造成了正確性問題，如下所示：

```
g.V("1").union(hasLabel("person"), out())
```

- 已修正 Gremlin 錯誤，其中若沒有 count()，both().simplePath() 的 count() 將會導致傳回的結果實際數量增加一倍。
- 已修正 OpenCypher 錯誤，其中向啟用 IAM 身分驗證的叢集發出 Bolt 請求時，伺服器產生了錯誤的簽章不符例外狀況。
- 已修正 OpenCypher 錯誤，其中使用 HTTP 保持連線的查詢，若在失敗的請求後提交，則系統可能會錯誤地關閉它。
- 已修正在提交傳回常數值的查詢時，可能會導致擲回內部錯誤的 OpenCypher 錯誤。
- 已修正解釋詳細資訊中的錯誤，以便 DFE 子查詢 Time(ms) 現在可以正確地將 DFE 子查詢內運算子的 CPU 時間加總。請考慮下列解釋輸出摘錄做為範例：

```
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      #
#########################################################################################
```

下表最後一欄的子查詢時間加起來為 0.36 毫秒 (.04 + .29 + .01 + .02 = .36)。當加上該子查詢 (.36 + .026 = .386) 的協調時間時，您得到的結果會接近上表最後一欄中所記錄的子查詢時間，即 0.38 毫秒。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.2.0.0 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：3.5.2
- 支援的 Gremlin 最新版本：3.5.4
- openCypher 版本：Neptune-9.0.20190305-1.0
- SPARQL 版本：1.1

引擎 1.2.0.0 版的升級途徑

因為這是一個主要引擎版本，所以不會自動升級至此版本。

您只能從引擎 1.1.1.0 版的最新修補程式版本手動升級至版本 1.2.0.0。先前的引擎版本必須先升級至 1.1.1.0 的最新版本，才能升級至 1.2.0.0。

因此，在您嘗試升級至此版本之前，請確認您目前執行的是最新的修補程式版本 1.1.1.0。如果您不是，首先升級至最新的修補程式版本 1.1.1.0。

在升級之前，您還必須使用參數群組系列 neptune1.2，重新建立任何已與舊版搭配使用的自訂資料庫叢集參數群組。如需更多資訊，請參閱Amazon Neptune 參數群組。

如果您先升級至版本 1.1.1.0，然後立即升級至 1.2.0.0，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成 (請參閱 維護 Amazon Neptune 資料庫叢集)。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
```

```
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.0.R4 (2023 年 9 月 29 日)

截至 2023 年 9 月 29 日，引擎版本 1.2.0.0.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Note

如果從 1.2.0.0 以前的引擎版本升級：

- 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
- 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

  您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

  如果 `UndoLogsListSize` CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已將 `enableInterContainerTrafficEncryption` 參數新增至所有 Neptune ML API，您可以使用此參數，在訓練或超參數調校工作中啟用和停用容器間流量加密。
- 對於無伺服器資料庫叢集，已將最小容量設定變更為 1.0 NCU，並將最低有效的上限設定變更為 2.5 NCU。請參閱 Neptune Serverless 資料庫叢集中的容量擴展 *(((#####################))))#*

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中更新和傳回查詢並未適當地處理 `orderBy`、`limit` 或 `skip`。
- 已修正 OpenCypher 錯誤，此錯誤允許一個請求中包含的參數被另一個同時請求中包含的參數覆寫。
- 已修正 Bolt 交易處理中的 openCypher 錯誤。
- 已透過返回 Tinkerpop 修正了 DEF 查詢的 Grimlin 正確性問題，這些查詢具有 `limit` 做為非聯集步驟的子周遊。例如，對於像這樣的查詢：

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- 已修正 Gremlin 錯誤，其中查詢將會失敗，因為它包含了太多的 TinkerPop 步驟，接著並不會將其清除。
- 已修正 Grimlin 錯誤，其中 `order()` 將不會正確排序字串輸出，因為這些字串輸出中有一些包含空格字元時。
- 已修正 Gremlin 錯誤，其中當以字串和包含的 `GroupCountStep` 形式提交查詢時，可能發生交易洩漏。

- 已修正 Gemlin 錯誤，其中在對於不是原生處理的步驟，檢查 Gremlin 查詢狀態端點是否有子周遊中具有述詞的查詢時，將會發生交易洩漏。
- 已修正 Gremlin 錯誤，其中新增邊緣及其後面跟著 `inV()` 或 `outV()` 的屬性導致了 `InternalFailureException`。
- 已修正儲存層中的並行問題。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.0.R4 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.6`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.0.R4 的升級途徑

如果您執行的是引擎版本 `1.2.0.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您只能從引擎 1.1.1.0 版的最新修補程式版本手動升級至版本 `1.2.0.0`。先前的引擎版本必須先升級至 `1.1.1.0` 的最新版本，才能升級至 `1.2.0.0`。

如果您先升級至版本 `1.1.1.0`，然後立即升級至 `1.2.0.0`，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.0.R3 (2022 年 12 月 15 日)

截至 2022 年 12 月 15 日，引擎版本 1.2.0.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的

版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。

您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

如果 UndoLogsListSize CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已改善涉及 `MERGE` 和 `OPTIONAL MATCH` 的 openCypher 查詢效能。
- 已改善涉及常值映射清單上 `UNWIND` 的 OpenCypher 查詢效能。
- 已改善對 `id` 具有 `IN` 篩選條件的 OpenCypher 查詢效能。例如：

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- 各種 Gemlin 運算子 (包括 `repeat`、`coalesce`、`store` 和 `aggregate`) 的效能改善和正確性修正。

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中查詢傳回了字串 `"null"`，而不是 Bolt 和 SPARQL-JSON 中的 null 值。
- 已修正 OpenCypher 錯誤，以便在值為清單或映射清單時能夠正確地解譯參數類型。
- 已修正稽核日誌錯誤，此錯誤已導致記錄不必要的資訊，以及某些欄位從日誌中遺失。
- 已修正稽核日誌錯誤，其中未記錄對啟用 IAM 的資料庫叢集發出的 HTTP 請求的 IAM ARN。
- 已修正查詢快取錯誤，以便限制用於寫入快取的增量記憶體。
- 已修正查閱快取錯誤，此錯誤涉及在寫入失敗時為查詢快取設定唯讀模式。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.0.R3 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.2.0.0.R3 的升級途徑

如果您執行的是引擎版本 `1.2.0.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您只能從引擎 1.1.1.0 版的最新修補程式版本手動升級至版本 `1.2.0.0`。先前的引擎版本必須先升級至 `1.1.1.0` 的最新版本，才能升級至 `1.2.0.0`。

如果您先升級至版本 `1.1.1.0`，然後立即升級至 `1.2.0.0`，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.2.0.0.R2 (2022 年 10 月 14 日)

截至 2022 年 10 月 14 日，引擎版本 1.2.0.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Note**
>
> 如果從 1.2.0.0 以前的引擎版本升級：
>
> - 引擎 1.2.0.0 版引入了新格式，適用於自訂參數群組和自訂叢集參數群組。因此，如果您要從 1.2.0.0 之前的引擎版本升級至引擎 1.2.0.0 版或更新版本，則必須使用參數群組系列 `neptune1.2` 重新建立所有現有的自訂參數群組和自訂叢集參數群組。舊版已使用參數群組系列 `neptune1`，而這些參數群組不會使用 1.2.0.0 版及更新版本。如需更多資訊，請參閱Amazon Neptune 參數群組。
> - 引擎 1.2.0.0 版也為復原日誌引入了新格式。因此，必須清除舊版引擎建立的任何還原日誌，且 `UndoLogsListSize` CloudWatch 指標必須降至零，然後才能開始從 1.2.0.0 之前的版本進行任何升級。如果在嘗試啟動更新時有太多的還原日誌 (200,000 或更多)，則在等待清除還原日誌完成時，升級嘗試可能會逾時。
>
>   您可以升級叢集的寫入器執行個體 (此為進行清除的位置)，以加快清除率。在嘗試升級之前執行此操作可能會在您開始之前減少還原日誌的數目。將寫入器的大小增加至 24XL 執行個體類型，可以將清除率提高到每小時超過一百萬筆記錄。

如果 UndoLogsListSize CloudWatch 指標非常大，開啟支援案例可協助您探索其他策略，以降低該指標。

- 最後，1.2.0.0 版中有重大變更，這會影響使用 Bolt 通訊協定搭配 IAM 身分驗證的舊版程式碼。從 1.2.0.0 版開始，Bolt 需要一個資源路徑進行 IAM 簽署。在 Java 中，設定資源路徑可能如下所示：`request.setResourcePath("/openCypher"));`。在其他語言中，`/openCypher` 可以附加到端點 URI。如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已改善 Gremlin `order-by` 查詢的效能。在 `NeptuneGraphQueryStep` 結尾具有 `order-by` 的 Gemlin 查詢現在會使用較大的區塊大小以取得更好的效能。這不適用於查詢計劃的內部 (非根) 節點上的 `order-by`。
- 已改善 Gremlin 更新查詢的效能。在新增邊緣或屬性時，必須鎖住頂點和邊緣以防刪除。此變更會消除交易內的重複鎖定，進而改善效能。
- 已透過將 dedup 向下推送至原生執行層，改善了在 `repeat()` 子查詢內部使用 `dedup()` 的 Gemlin 查詢效能。
- 已新增 Gremlin `Neptune#cardinalityEstimates` 查詢提示。設定為 `false` 時，這會停用基數估計。
- 已對 IAM 身分驗證錯誤新增了使用者易於了解的錯誤訊息。這些訊息現在會顯示您的 IAM 使用者或角色 ARN、資源 ARN，以及請求的未經授權動作清單。未經授權的動作清單可協助您查看所使用的 IAM 政策中可能遺漏或明確拒絕的內容。

此引擎版本中修正的缺陷

- 已修正涉及 `WherePredicateStep` 轉換的 Gemlin 正確性錯誤，其中 Neptune 的查詢引擎對使用 `where(P.neq('x'))` 和其變化的查詢產生不正確的結果。
- 已修正 Gremlin 錯誤，其中在升級至 TinkerPop 3.5 之後使用 `PartitionStrategy`，不正確地造成了阻止執行周遊的錯誤，並顯示訊息「PartitionStrategy 不會使用匿名周遊」。
- 已修正與最終聯結的 `joinTime` 和 `Project.ASK` 子群組內的統計資料相關的各種 Gemlin 錯誤。
- 已修正 `MERGE` 子句中在某些情況下導致了建立重複節點和邊緣的 OpenCypher 錯誤。
- 已修正交易錯誤，其中即使回復了對應的並行字典插入，工作階段仍可以插入圖形資料並遞交。
- 已修正大量載入器錯誤，此錯誤在大量插入負載下造成了效能迴歸。

- 已修正處理查詢時發生的 SPARQL 錯誤，這些查詢在 OPTIONAL 子句內包含 (NOT) EXISTS，在某些情況下會遺失查詢結果。
- 已修正驅動程式在下列情況下似乎當機的錯誤：評估開始之前，由於逾時而請求遭到取消。如果在請求佇列中的項目發生逾時時，伺服器上的所有查詢處理執行緒都已使用，則可能會進入此狀態。因為來自請求佇列的逾時並未立即傳送訊息，所以在用戶端看來回應仍處於待定狀態。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.2.0.0.R2 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：3.5.2
- 支援的 Gremlin 最新版本：3.5.4
- openCypher 版本：Neptune-9.0.20190305-1.0
- SPARQL 版本：1.1

引擎版本 1.2.0.0.R2 的升級途徑

如果您執行的是引擎版本 1.2.0.0，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您只能從引擎 1.1.1.0 版的最新修補程式版本手動升級至版本 1.2.0.0。先前的引擎版本必須先升級至 1.1.1.0 的最新版本，才能升級至 1.2.0.0。

如果您先升級至版本 1.1.1.0，然後立即升級至 1.2.0.0，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.1.1.0 版 (2022 年 4 月 19 日)

截至 2022 年 4 月 19 日，引擎 1.1.1.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Important**
>
> 從 **`1.1.0.0`** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
> 為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。

在升級開始時，Neptune 會產生一個快照，其名稱由 preupgrade 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

此版本的後續修補程式版本

- 維護版本：1.1.1.0.R2 (2022 年 5 月 16 日)
- 版本：1.1.1.0.R3 (2022 年 6 月 7 日)
- 版本：1.1.1.0.R4 (2022 年 6 月 23 日)
- 版本：1.1.1.0.R5 (2022 年 7 月 21 日)
- 版本：1.1.1.0.R6 (2022 年 9 月 23 日)
- 版本：1.1.1.0.R7 (2023 年 1 月 23 日)

這個引擎版本的新功能

- OpenCypher 查詢語言現在已普遍適用於生產用途。

> **Warning**
>
> 對於使用 OpenCypher 搭配 IAM 身分驗證的程式碼，此版本中有一項重大變更。在 OpenCypher 的 Neptune 預覽版中，IAM 簽章中的主機字串包含了通訊協定，例如 `bolt://`，如下所示：
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> 從此引擎版本開始，必須省略通訊協定：
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> 如需範例，請參閱 使用 Bolt 通訊協定。

- 已新增對 TinkerPop `3.5.2` 的支援。此版本中的變更包括對遠端交易的支援以及對工作階段的位元碼支援 (使用 `g.tx`)，以及將 `datetime()` 函數新增至 Grimlin 語言。

> **Warning**
>
> 在 TinkerPop 3.5.0、3.5.1 和 3.5.2 中引進了數個重大變更，這些變更可能會影響您的 Gremlin 程式碼。例如，像這樣使用由 GraphTraversalSource 產生的周遊做為子項將不再有效：`g.V().union(identity(), g.V())`。
> 現在，請使用像這樣的匿名周遊：`g.V().union(identity(), __.V())`。

- 已新增對 AWS 全域條件金鑰的支援，您可以在 IAM 資料存取政策中使用這些金鑰，控制對 Neptune 資料庫叢集中儲存之資料的存取。
- Neptune DFE 查詢引擎現在通常可用於與 OpenCypher 查詢語言搭配使用的生產，但尚未用於 Gremlin 和 SPARQL 查詢。現在，您可以使用它自己的 neptune_dfe_query_engine 執行個體參數，而不是實驗室模式參數來啟用它。

這個引擎版本的改善項目

- 已新增功能至 OpenCypher，例如參數化查詢支援、參數化查詢的抽象語法樹 (AST) 快取、可變長度路徑 (VLP) 改善，以及新的運算子和子句。如需目前的語言支援層級，請參閱 Amazon Neptune 中的開放密碼規範合規。

- 對於簡單的讀取和寫入工作負載，已對 OpenCypher 進行了顯著的效能改善，與 1.1.0.0 版相比，產生了更高的輸送量。
- 已移除 OpenCypher 處理可變長度路徑的雙向和深度限制。
- 已完成 DFE 引擎中對 Gremlin `within` 和 `without` 述詞的支援，包括它們與其他述詞運算子合併的案例。例如：

```
g.V().has('age', within(12, 15, 18).or(gt(30)))
```

- 當範圍為全域 (亦即，不是 `order(local)`) 時，以及未使用 `by()` 調幅器時，已在 DFE 引擎中延伸對 Gemlin `order` 步驟的支援。例如，此查詢現在將具有 DFE 支援：

```
 g.V().values("age").order()
```

- 已將 `isLastOp` 欄位新增至 Neptune 串流變更日誌回應格式，以指出記錄是其交易中的最後一個操作。
- 已大幅改善稽核記錄的效能，並在啟用稽核記錄時減少了延遲。
- 已將 Gremlin WebSocket 位元碼和 HTTP 查詢轉換為稽核日誌中使用者可讀取的格式。查詢現在可以直接從要在 Neptune 筆記本和其他位置執行的稽核日誌複製。請注意，這項對目前稽核日誌格式的變更會構成重大變更。

此引擎版本中修正的缺陷

- 已修正罕見的 Gemlin 錯誤，其中使用巢狀 `filter()` 和 `count()` 步驟組合時未傳回任何結果，例如在以下查詢中：

```
g.V("1").filter(out("knows")
        .filter(in("knows")
        .hasId("notExists"))
        .count())
```

- 已修正 Gremlin 錯誤，其中在使用 `to()` 或 `from()` 周遊中的彙總步驟中儲存的頂點，搭配 `addE` 步驟時傳回了錯誤。此類查詢範例如下：

```
g.V("id").aggregate("v").out().addE().to(select("v").unfold())
```

- 已修正使用 DFE 引擎時，`not` 步驟在邊緣情況下失敗的錯誤。例如：

```
g.V().not(V())
```

- 已修正 Gremlin 錯誤，其中 `sideEffect` 值無法在 `to()` 和 `from()` 周遊內使用。
- 已修正偶爾導致快速重設以觸發執行個體容錯移轉的錯誤。
- 已修正大量載入器錯誤，其中失敗的交易在開始下一個載入工作之前不會關閉。
- 已修正大量載入器錯誤，其中記憶體不足的情況可能會導致系統當機。
- 已新增重試，來修正大量載入器錯誤，其中載入器未等待足夠長的時間，讓 IAM 憑證在容錯移轉後變成可用。
- 已修正未針對非查詢端點 (例如 `status` 端點) 適當清除內部憑證快取的錯誤。
- 已修正串流錯誤，以確保串流遞交序號正確排序。
- 已修正在啟用 IAM 的叢集上，長時間執行連線不到十天就終止的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.1.1.0 版之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎 1.1.1.0 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。請注意，主要版本引擎 (1.1.0.0) 之前的版本將需要更長的時間才能升級到此版本。

您不會自動升級至此版本。

升級至此版本

> **Important**
>
> 從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

在升級開始時，Neptune 會產生一個快照，其名稱由 preupgrade 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 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
```

針對 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
```

您可以指定 --no-apply-immediately，而不是 --apply-immediately。若要執行主要版本升級，需要 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 preupgrade 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.1.1.0.R7 (2023 年 1 月 23 日)

截至 2023 年 1 月 23 日，引擎版本 1.1.1.0.R7 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

從 **`1.1.0.0`** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`

    - Database cluster major version has been upgraded
  - 每個執行個體事件訊息：
    - Applying off-line patches to DB instance
    - DB instance shutdown
    - Finished applying off-line patches to DB instance
    - DB instance restarted

這個引擎版本的改善項目

- 已改善涉及 MERGE 和 OPTIONAL MATCH 的 openCypher 查詢效能。
- 已改善涉及常值映射清單的 UNWIND 的 OpenCypher 查詢效能。
- 已改善對 id 具有 IN 篩選條件的 OpenCypher 查詢效能。例如：

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- 各種 Gemlin 運算子 (包括 repeat、coalesce、store 和 aggregate) 的效能改善和正確性修正。

此引擎版本中修正的缺陷

- 已修正 OpenCypher 錯誤，其中使用 HTTP 保持連線的請求，若在失敗的請求後提交，則系統可能會錯誤地關閉它。
- 已修正 OpenCypher 錯誤，其中參數類型並不總是正確解譯為清單或對應清單。
- 已修正 OpenCypher 錯誤，其中查詢傳回了字串 "null"，而不是 Bolt 和 SPARQL-JSON 中的 null 值。
- 已修正查詢逾時失敗和記憶體不足錯誤的 OpenCypher 錯誤代碼和錯誤訊息。
- 已修正 Gremlin 錯誤，此錯誤已導致 valueMap() 未在 DFE 引擎的 by() 周遊下進行最佳化。
- 已修正死鎖偵測器邏輯偶爾會使引擎沒有回應的問題。
- 已修正稽核日誌錯誤，此錯誤已導致記錄不必要的資訊，以及某些欄位從日誌中遺失。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.1.0.R7 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.3`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.1.1.0.R7 的升級途徑

如果您執行的是引擎版本 `1.1.1.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Important

從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

Note

如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.1.1.0.R6 (2022 年 9 月 23 日)

截至 2022 年 9 月 23 日，引擎版本 1.1.1.0.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

從 **`1.1.0.0`** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。

在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。

當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。

此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

對於使用 OpenCypher 搭配 IAM 身分驗證的程式碼，此版本中有一項重大變更。到目前為止，IAM 簽章中的主機字串包含通訊協定，例如 `bolt://`，如下所示：

```
"Host":"bolt://(host URL):(port)"
```

從引擎版本 `1.1.1.0` 開始，必須省略通訊協定：

```
"Host":"(host URL):(port)"
```

如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已改善 Gremlin `order-by` 查詢的效能。在 `NeptuneGraphQueryStep` 結尾具有 `order-by` 的 Gemlin 查詢現在會使用較大的區塊大小以取得更好的效能。這不適用於查詢計劃的內部 (非根) 節點上的 `order-by`。
- 已改善 Gremlin 更新查詢的效能。在新增邊緣或屬性時，必須鎖住頂點和邊緣以防刪除。此變更會消除交易內的重複鎖定，進而改善效能。

此引擎版本中修正的缺陷

- 已修正 MERGE 子句中在某些情況下導致了建立重複節點和邊緣的 OpenCypher 錯誤。
- 已修正處理 SPARQL 查詢時發生的錯誤，這些查詢在 OPTIONAL 子句內包含 (NOT) EXISTS，在某些情況下將會遺失查詢結果。
- 已修正在進行大量載入時延遲伺服器重新啟動的錯誤。
- 已修正錯誤，其中 OpenCypher 可變長度模式雙向周遊搭配關係屬性上的篩選條件會導致錯誤。這類可變長度模式的範例為 `(n)-[r*1..2]->(m)`。
- 已修正與快取資料如何傳回用戶端相關的錯誤，在某些情況下這導致了非預期的長延遲。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.1.0.R6 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.1.1.0.R6 的升級途徑

如果您執行的是引擎版本 `1.1.1.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Important

從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。

當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。

此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.1.1.0.R5 (2022 年 7 月 21 日)

截至 2022 年 7 月 21 日，引擎版本 1.1.1.0.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

從 **1.1.0.0** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

對於使用 OpenCypher 搭配 IAM 身分驗證的程式碼，此版本中有一項重大變更。到目前為止，IAM 簽章中的主機字串包含通訊協定，例如 `bolt://`，如下所示：

```
"Host":"bolt://(host URL):(port)"
```

從引擎版本 `1.1.1.0` 開始，必須省略通訊協定：

```
"Host":"(host URL):(port)"
```

如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已進行改善，以支援死鎖檢測。

此引擎版本中修正的缺陷

- 已修正在特定情況下阻止正常關閉資料庫叢集的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.1.0.R5 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.1.1.0.R5 的升級途徑

如果您執行的是引擎版本 `1.1.1.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Important

從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。

當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.1.1.0.R4 (2022 年 6 月 23 日)

截至 2022 年 6 月 23 日，引擎版本 1.1.1.0.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

從 **1.1.0.0** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。

在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。

當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。

此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> 對於使用 OpenCypher 搭配 IAM 身分驗證的程式碼，此版本中有一項重大變更。到目前為止，IAM 簽章中的主機字串包含通訊協定，例如 `bolt://`，如下所示：
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> 從引擎版本 `1.1.1.0` 開始，必須省略通訊協定：
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> 如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已更新 `x2g` 執行個體類型的執行個體組態。
- 已改善頂點掉落的效能。

此引擎版本中修正的缺陷

- 已修正 Gemlin 錯誤，其中解決方案無法針對多次呼叫的查詢維持穩定順序，也無法針對特定種類的 ASK 聯結在多個讀取器之間維持穩定順序。
- 此外，已縮小之前版本中的變更範圍，此變更導致 Gremlin 中特定種類的 ASK 聯結的效能迴歸。
- 已修正 `union()` 步驟中在下列情況發生的 Gremlin 錯誤：對子周遊內的頂點有邊緣輸入和周遊。
- 已修正 Gremlin 設定檔錯誤，其中報告了一些步驟未最佳化，但實際上它們已最佳化。
- 已修正 SPARQL 錯誤，其中 `FILTER` 運算式內使用與 `UNION` 子句構成巢狀的變數獲指派無效的範圍資訊。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.1.0.R4 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`

- SPARQL 版本：1.1

引擎版本 1.1.1.0.R4 的升級途徑

如果您執行的是引擎版本 1.1.1.0，您的叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

> **Important**
>
> 從 **1.1.0.0** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
> 在升級開始時，Neptune 會產生一個快照，其名稱由 preupgrade 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
> 當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。
> 此過程會產生下列事件：
>
> - 每個叢集事件訊息：
>   - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
>   - `Database cluster major version has been upgraded`
> - 每個執行個體事件訊息：
>   - `Applying off-line patches to DB instance`
>   - `DB instance shutdown`
>   - `Finished applying off-line patches to DB instance`
>   - `DB instance restarted`

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.1.1.0.R3 (2022 年 6 月 7 日)

截至 2022 年 6 月 7 日，引擎版本 1.1.1.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

從 **`1.1.0.0`** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`

- DB instance shutdown
- Finished applying off-line patches to DB instance
- DB instance restarted

> **Note**
>
> 對於使用 OpenCypher 搭配 IAM 身分驗證的程式碼，此版本中有一項重大變更。到目前為止，IAM 簽章中的主機字串包含通訊協定，例如 `bolt://`，如下所示：
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> 從引擎版本 `1.1.1.0` 開始，必須省略通訊協定：
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> 如需範例，請參閱 使用 Bolt 通訊協定。

這個引擎版本的改善項目

- 已新增對 Graviton2 提供的 `x2g` 執行個體類型的支援，並針對記憶體密集型工作負載進行最佳化。這些最初只能在四個 AWS 區域中使用：
  - 美國東部 (維吉尼亞北部) (`us-east-1`)
  - 美國東部 (俄亥俄) (`us-east-2`)
  - 美國西部 (奧勒岡) (`us-west-2`)
  - 歐洲 (愛爾蘭) (`eu-west-1`)

  如需詳細資訊，請參閱 Neptune 定價頁面。
- 已改善 Gemlin 步驟的效能，這些步驟涉及多個邊緣或頂點周遊、屬性查詢或標籤查詢。

此引擎版本中修正的缺陷

- 已修正在子周遊內處理 `otherV()` 步驟時發生的 Gremlin 錯誤。
- 已修正 `union` 僅具有篩選步驟做為子項的查詢中的 Gremlin 錯誤。例如：

```
g.V().union(has("name"), out("knows")).out()
```

- 已修正 SPARQL 錯誤，其中 FILTER 運算式內使用與 UNION 子句構成巢狀的變數獲指派無效的範圍資訊。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.1.0.R3 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.1.1.0.R3 的升級途徑

如果您執行的是引擎版本 `1.1.1.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Important

從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。

當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。

此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`

    - Database cluster major version has been upgraded
  - 每個執行個體事件訊息：
    - Applying off-line patches to DB instance
    - DB instance shutdown
    - Finished applying off-line patches to DB instance
    - DB instance restarted

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 維護版本，版本 1.1.1.0.R2 (2022 年 5 月 16 日)

截至 2022 年 5 月 16 日，引擎版本 1.1.1.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Important**
>
> 從 **`1.1.0.0`** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

對於使用 OpenCypher 搭配 IAM 身分驗證的程式碼，此版本中有一項重大變更。到目前為止，IAM 簽章中的主機字串包含通訊協定，例如 `bolt://`，如下所示：

```
"Host":"bolt://(host URL):(port)"
```

從引擎版本 `1.1.1.0` 開始，必須省略通訊協定：

```
"Host":"(host URL):(port)"
```

如需範例，請參閱 使用 Bolt 通訊協定。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.1.0.R2 之前，請確定您的專案與下列查詢語言版本相容：

- 支援的 Gremlin 最早版本：`3.5.2`
- 支援的 Gremlin 最新版本：`3.5.4`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.1.1.0.R2 的升級途徑

如果您執行的是引擎版本 `1.1.1.0`，您的叢集將在下一個維護時段自動升級至此維護修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Important

從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`

```
    • Finished applying off-line patches to DB instance
    • DB instance restarted
```

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.1.0.0 版 (2021 年 11 月 19 日)

截至 2021 年 11 月 19 日，引擎 1.1.0.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

> **Important**
>
> 從 **`1.1.0.0`** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
> 為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。

在升級開始時，Neptune 會產生一個快照，其名稱由 preupgrade 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> 從此引擎版本開始，Neptune 不再支援 R4 執行個體類型。如果您要在資料庫叢集中使用 R4 執行個體，則在升級至此版本之前，必須手動將其取代為不同的執行個體類型。如果您的寫入器執行個體是 R4，請遵循這些指示移動它。

此版本的後續修補程式版本

- 維護版本：1.1.0.0.R2 (2022 年 5 月 16 日)
- 維護版本：1.1.0.0.R3 (2022 年 12 月 23 日)

這個引擎版本的新功能

- 引進了由 AWS Graviton2 處理器提供的一般用途 T4g 和記憶體最佳化的 R6g 資料庫執行個體。與目前同級 x86 型執行個體相比，對於各種工作負載，Graviton2 型執行個體提供明顯更優異的價格/

效能。應用程式可在這些新的執行個體類型上正常運作，而且升級至這些執行個體類型時不需要移植應用程式程式碼。

如需定價和區域可用性的詳細資訊，請參閱 Amazon Neptune 定價頁面。

- 在 Neptune ML 中引進了自訂模型。
- 已在 Neptune ML 中新增對 SPARQL 推論查詢的支援。
- 已為屬性圖資料新增了新的串流端點，即：

```
https://Neptune-DNS:8182/propertygraph/stream
```

此端點的輸出格式 (名為 PG_JSON) 與舊版 gremlin/stream 的 GREMLIN_JSON 格式輸出完全相同。

新的 propertygraph/stream 端點會將 Neptune 串流支援延伸至 OpenCypher，並將 gremlin/stream 端點取代為其相關聯的 GREMLIN_JSON 輸出格式。

這個引擎版本的改善項目

- 已對 Neptune 串流做了改善：
  - 已將 commitTimestamp 欄位新增至 records 物件的 Neptune 串流變更日誌回應格式，為變更日誌串流中的每筆記錄提供時間戳記。
  - 已將 LATEST 值新增至 iteratorType 參數，可讓您從串流擷取最後一個有效的 eventId。請參閱呼叫串流 API。
- 已新增在 Gremlin 節點分類和迴歸查詢中取得推論可信度分數的支援。
- 已在 openCypher 新增對 OPTIONAL MATCH 的支援。
- 已在 openCypher 新增對 MERGE 的支援。
- 已在 openCypher 新增在 WITH 子句中使用 ORDER BY 的支援。
- 已在 OpenCypher 新增對模式理解的支援，以及對超出存在檢查的模式運算式新增延伸支援。
- 已在 OpenCypher 延伸對 DELETE 和 DELETE DETACH 子句的支援，以便它們現在可與其他更新子句搭配使用。
- 已在 openCypher 延伸 CREATE 和 UPDATE 子句與 RETURN 搭配使用的支援。
- 已在 DFE 引擎中新增了對 Gremlin limit、range 和 skip 步驟的支援。
- 已改善在既未請求 explain 也未請求 profile 時 DFE 引擎中的查詢執行。
- 已改善 DFE 引擎中針對 value 運算式的查詢執行。

- 已改善一些鏈結的 Gemlin 條件式插入模式，以避免並行修改例外狀況，並允許鏈結查詢模式，如下所示：
  - 依 ID 進行條件式頂點插入，例如：

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1").property(id,ID))
```

  - 具有多個標籤的條件式頂點插入，例如：

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1::L2").property(id,ID))
```

  - 依 ID 進行條件式邊緣插入，例如：

```
g.E(ID).fold().coalesce(unfold(), V(from).addE(label).to(V(to)).property(id, ID))
```

  - 具有多個標籤的條件式邊緣插入，例如：

```
g.E(ID).fold().coalesce(unfold(),
 g.addE(label).from(V(from)).to(V(to)).property(id, ID))
```

  - 後面跟著查詢的條件式插入，例如：

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID)).project("myvalues").by(valueMap())
```

  - 具有所新增屬性的條件式插入，例如：

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID).property("name","pumba"))
```

此引擎版本中修正的缺陷

- 已在 `T3.medium` 執行個體類型上停用無法支援的統計資料功能。
- 已在 `explain` 中修正 `IN` 函數採取非常數值的 SPARQL 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.1.0.0 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`

- SPARQL 版本：1.1

引擎 1.1.0.0 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

您不會自動升級至此版本。

升級至此版本

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

您可以指定 `--no-apply-immediately`，而不是 `--apply-immediately`。若要執行主要版本升級，需要 allow-major-version-upgrade 參數。此外，請務必包含引擎版本，否則您的引擎可能會升級至不同版本。

如果您的叢集使用自訂叢集參數群組，請務必包含此參數來指定它：

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

同樣地，如果叢集中有任何執行個體使用自訂資料庫參數群組，請務必包含此參數來指定它：

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 維護版本，版本 1.1.0.0.R3 (2022 年 12 月 23 日)

截至 2022 年 12 月 23 日，引擎版本 1.1.0.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

從 **1.1.0.0** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

為了成功完成升級，每個可用區域 (AZ) 中的每個子網路都必須每個 Neptune 執行個體至少有一個可用的 IP 地址。例如，如果子網路 1 中有一個寫入器執行個體和兩個讀取器執行個體，而子網路 2 中有兩個讀取器執行個體，則子網路 1 必須至少有 3 個可用的 IP 地址，而子網路 2 必須至少有 2 個可用的 IP 地址，才能開始升級。

在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。

當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。

此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

這個引擎版本的改善項目

- 各種 Gemlin 運算子 (包括 `repeat`、`coalesce`、`store` 和 `aggregate`) 的效能改善和正確性修正。

此引擎版本中修正的缺陷

- 已修正 CPU 峰值問題。
- 已修正 OpenCypher 錯誤，其中查詢傳回了字串 "`null`"，而不是 Bolt 和 SPARQL-JSON 中的 null 值。
- 已修正稽核日誌錯誤，此錯誤已導致記錄不必要的資訊，以及某些欄位從日誌中遺失。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.0.0.R3 之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.1.0.0.R3 的升級途徑

如果您執行的是引擎版本 `1.1.0.0`，您的叢集將在下一個維護時段自動升級至此維護修補程式版本。

> **Important**
>
> 從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
> 在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
> 當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。
> 此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> 從此引擎版本開始，Neptune 不再支援 R4 執行個體類型。如果您要在資料庫叢集中使用 R4 執行個體，則在升級至此版本之前，必須手動將其取代為不同的執行個體類型。如果您的寫入器執行個體是 R4，請遵循這些指示進行移動。

升級至此版本

Amazon Neptune 1.1.0.0.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
```

```
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 維護版本，版本 1.1.0.0.R2 (2022 年 5 月 16 日)

截至 2022 年 5 月 16 日，引擎版本 1.1.0.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

Important

從 **`1.1.0.0`** 以前的版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。
在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。
當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前將有數分鐘無法使用。升級完成後，您就可以繼續寫入工作負載。
此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

此引擎版本中修正的缺陷

- 已修正未針對非查詢端點 (例如狀態端點) 適當清除內部憑證快取的錯誤。
- 已修正引擎升級後造成複寫延遲增加的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.1.0.0.R2 之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- openCypher 版本：`Neptune-9.0.20190305-1.0`
- SPARQL 版本：`1.1`

引擎版本 1.1.0.0.R2 的升級途徑

如果您執行的是引擎版本 `1.1.0.0`，您的叢集將在下一個維護時段自動升級至此維護修補程式版本。

Important

從 **`1.1.0.0`** 以前的任何版本升級至此引擎版本，也會在資料庫叢集中的所有執行個體上觸發作業系統升級。因為不會處理作業系統升級期間發生的作用中寫入請求，所以在開始升級之前，您必須暫停對要升級之叢集的所有寫入工作負載 (包括大量資料載入)。

在升級開始時，Neptune 會產生一個快照，其名稱由 `preupgrade` 組成，後面接著根據資料庫叢集資訊自動產生的識別符。您不需要為此快照付費，而且如果在升級過程中出現任何問題，您可以使用它來還原資料庫叢集。

當引擎升級本身完成時，新的引擎版本將可在舊版作業系統上短暫使用，但不到 5 分鐘的時間，叢集中的所有執行個體都會同時開始作業系統升級。您的資料庫叢集目前大約 6 分鐘將無法使用。升級完成後，您就可以繼續寫入工作負載。

此過程會產生下列事件：

- 每個叢集事件訊息：
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- 每個執行個體事件訊息：
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> 從此引擎版本開始，Neptune 不再支援 R4 執行個體類型。如果您要在資料庫叢集中使用 R4 執行個體，則在升級至此版本之前，必須手動將其取代為不同的執行個體類型。如果您的寫入器執行個體是 R4，請遵循這些指示進行移動。

升級至此版本

Amazon Neptune 1.1.0.0.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.5.1 版 (2021 年 10 月 1 日)

截至 2021 年 10 月 1 日，引擎 1.0.5.1 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.5.1.R2 (2021 年 10 月 26 日)
- 版本：1.0.5.1.R3 (2022 年 1 月 13 日)
- 維護版本：1.0.5.1.R4 (2022 年 5 月 16 日)

這個引擎版本的新功能

- 已新增用於快取所指定查詢結果的結果快取。
- 已在 Neptune openCypher 中新增了日期/時間支援。
- 已增加對 `List` 和 `Map` 可存取 Neptune openCypher 中元素的支援。

這個引擎版本的改善項目

- 已使 Neptune OpenCypher 端點名稱需區分大小寫。
- 已改善 openCypher Explain。
- 已改善以 `iterate()` 和 `profile()` 步驟終止的 Gremlin 單一 upsert 查詢模式。
- 已改善 Gremlin `keys()` 和 `property()` 函數中的效能。
- 當 Gremlin `dedup()` 步驟與全域範圍搭配使用時，該步驟是在 DFE 中執行。
- DFE 引擎啟用時，下列 Gremlin HAS 述詞是在 DFE 引擎中執行：
  - `EQ`
  - `NEQ`
  - `LT`
  - `LTE`
  - `GT`
  - `GTE`
  - `BETWEEN`
  - `INSIDE`
  - `OUTSIDE`
  - `WITHIN`
  - `AND (connectives)`
  - `OR (connectives)`
- 已改善 LIMIT 查詢效能。
- 已改善 openCypher 一般彙總查詢的效能。

此引擎版本中修正的缺陷

- 已修正允許邊緣連線到另一個邊緣的 Gremlin 錯誤。
- 已修正導致選擇次佳聯結策略的 Gremlin 錯誤。
- 已修正當存在 100 個以上屬性時，導致了節點和關係序列化停滯的 Gemlin 錯誤。
- 已修正為具有大型圖形模式的查詢進行查詢執行規畫時速度變慢的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.5.1 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎 1.0.5.1 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

您不會自動升級至此版本。

升級至此版本

Amazon Neptune 1.0.5.1 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 維護版本，版本 1.0.5.1.R4 (2022 年 5 月 16 日)

截至 2022 年 5 月 16 日，引擎版本 1.0.5.1.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.5.1.R4 之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎版本 1.0.5.1.R4 的升級途徑

如果您執行的是引擎版本 `1.0.5.1`，您的叢集將在下一個維護時段自動升級至此維護修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.5.1.R4 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
```

```
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.5.1.R3 (2022 年 1 月 13 日)

截至 2022 年 1 月 13 日，引擎版本 1.0.5.1.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已修正查詢無法取得其需要的所有資源時，可能導致資源洩漏的錯誤。
- 已修正查詢執行期間由未宣告的記憶體配置所造成的少量記憶體洩漏。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.5.1.R3 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎版本 1.0.5.1.R3 的升級途徑

如果您執行的是引擎版本 `1.0.5.1`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.5.1.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.5.1.R2 (2021 年 10 月 26 日)

截至 2021 年 10 月 26 日，引擎版本 1.0.5.1.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已修正下列錯誤：在可重複讀取隔離下建立舊版圖形元素的同時發生暫時性錯誤，導致了伺服器重新啟動。Neptune 現在反而會傳回錯誤，以便用戶端可以重試。
- 已修正在單一基數更新期間發生暫時性錯誤時，導致了伺服器重新啟動的錯誤。Neptune 現在反而會傳回錯誤，以便用戶端可以重試。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.5.1.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎版本 1.0.5.1.R2 的升級途徑

如果您執行的是引擎版本 `1.0.5.1`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.5.1.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.5.0 版 (2021 年 7 月 27 日)

截至 2021 年 7 月 27 日，引擎 1.0.5.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.5.0.R2 (2021 年 8 月 16 日)
- 版本：1.0.5.0.R3 (2021 年 9 月 15 日)
- 維護版本：1.0.5.0.R5 (2022 年 5 月 16 日)

這個引擎版本的新功能

- 已針對生產發行了與許多新功能搭配使用的 Neptune ML，而且 Neptune ML 不再處於實驗室模式。
- 已在實驗室模式下新增了對 openCypher 查詢語言的初始支援。openCypher 是 Cypher 查詢語言的開放原始碼標準。它的語法在 Cypher 查詢語言參考 (第 9 版) 中加以指定，並由 openCypher 維護。

  如需 Neptune 語言實作的相關資訊，請參閱 使用 openCypher 存取 Neptune 圖形。

也支援對 Bolt 通訊協定的支援，Neptune 用戶端會將此通訊協定用於 OpenCypher 查詢。請參閱使用 Bolt 通訊協定，對 Neptune 進行 openCypher 查詢。

對 OpenCypher 的支援現在會自動啟用，但這取決於目前僅能在實驗室模式中使用的 Neptune DFE 引擎。neptune_lab_mode 資料庫叢集參數中的預設 DFEQueryEngine 設定現在為 DFEQueryEngine=viaQueryHint，這表示引擎已啟用，但僅用於存在 useDFE 查詢提示且其設定為 true 的查詢。如果您透過設定 DFEQueryEngine=disabled 來停用 DFE 引擎，將無法使用 OpenCypher。

- 已新增對 SPARQL 1.1 圖形存放區 HTTP 通訊協定的支援。請參閱在 Amazon Neptune 中使用 SPARQL 1.1 圖形存放區 HTTP 通訊協定 (GSP)。
- 已將 Neptune DFE 引擎 的預設實驗室模式設定設為 viaQueryHint，這表示 DEF 引擎現在預設為啟用，但僅用於存在 true 查詢提示且其設定為 useDFE 的查詢。
- 已新增 Amazon CloudWatch 指標 StatsNumStatementsScanned，用於監控 Neptune DFE 引擎統計資料的計算。請參閱使用StatsNumStatementsScanned CloudWatch 測量結果監督統計資料運算。

這個引擎版本的改善項目

- 已新增對 Apache TinkerPop 3.4.11 的支援。

> **Important**
>
> 已在 TinkerPop 3.4.11 版中進行變更，這可提高查詢處理方式的正確性，但目前有時可能會嚴重影響查詢效能。
> 例如，此類查詢的執行速度可能會明顯變慢：
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```
>
> 由於 TinkerPop 3.4.11 變更，限制步驟之後的頂點現在會以非最佳方式擷取。若要避免這種情況，您可以在 order().by() 之後的任何點新增 barrier() 步驟來修改查詢。例如：
>
> ```
> g.V().hasLabel('airport').
>   order().
> ```

```
  by(out().count(),desc).
limit(10).
barrier().
out()
```

- Neptune DFE 替代查詢引擎現在支援 SPARQL `joinOrder` 查詢提示。
- Neptune 狀態 API 的輸出已擴充並重組，讓您更清晰地了解資料庫叢集設定和功能。

  新輸出具有最上層 `features` 物件，其中包含有關資料庫叢集功能的狀態資訊，也具有最上層 `settings` 物件，其中包含設定資訊。若要檢閱新格式，請參閱 來自執行個體狀態命令的輸出範例。
- 已改善在下列情況處理串流變更日誌的方式：使用伺服器上的最後一個事件 ID 請求 `AFTER_SEQUENCE_NUMBER` 串流，但該事件 ID 已過期。如果請求的事件 ID 是伺服器上最近清除的事件 ID，則伺服器不再擲回過期的事件 ID 錯誤。

此引擎版本中修正的缺陷

- 已修正與數值排序相關的 Gremlin 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.5.0 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎 1.0.5.0 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

您不會自動升級至此版本。

升級至此版本

Amazon Neptune 1.0.5.0 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 維護版本，版本 1.0.5.0.R5 (2022 年 5 月 16 日)

從 2022 年 5 月 16 日起，引擎版本 1.0.5.0.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.5.0.R5 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎版本 1.0.5.0.R5 的升級途徑

如果您執行的是引擎 1.0.5.0 版 ，您的叢集將在下一個維護時段自動升級至此維護修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.5.0.R5 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.5.0.R3 (2021 年 9 月 15 日)

截至 2021 年 9 月 15 日，引擎版本 1.0.5.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已修正在下列任一情況下，導致引擎無回應的錯誤：
  - 大量載入與自動統計資料計算同時發生。
  - 統計資料計算是在已有一個統計資料計算發生的同時手動請求的。
- 已修正死鎖偵測和鎖定擷取中可能導致引擎當機的錯誤。
- 已修正 Gemlin 錯誤，其中引擎在 Gremlin 推論查詢中遇到來自遠端 ML 端點的未知資料時會擲回錯誤。
- 已修正 ML 模型管理 API 中與模型轉換工作和執行個體建議相關的數個錯誤。
- 已修正在產生節點和邊緣 ID 時可能會導致引擎當機的錯誤。
- 已修正為具有大型圖形模式的查詢產生查詢計畫時速度變慢的錯誤。
- 已修正 OpenCypher 錯誤，此錯誤可能導致查詢在擷取具有 100 個以上屬性的節點時停滯。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.5.0.R3 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎版本 1.0.5.0.R3 的升級途徑

如果您執行的是引擎 1.0.5.0 版 ，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.5.0.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.5.0.R2 (2021 年 8 月 16 日)

截至 2021 年 8 月 16 日，引擎版本 1.0.5.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已停用在引擎版本 1.0.5.0中進行的最佳化，此最佳化使 Neptune 查閱快取在引擎對複本重新啟動時仍然存在。複本重新啟動現在會清除查閱快取。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.5.0.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.11`
- SPARQL 版本：`1.1`

引擎版本 1.0.5.0.R2 的升級途徑

如果您執行的是引擎版本 `1.0.5.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.5.0.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.4.2 版 (2021 年 6 月 1 日)

> **Note**
>
> 引擎發行版本 1.0.4.2.R2 是 1.0.4.2 實際發行的第一個版本。

主題

Amazon Neptune 引擎版本 1.0.4.2.R5 (2021 年 8 月 16 日)

截至 2021 年 8 月 16 日，引擎版本 1.0.4.2.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已停用在引擎版本 `1.0.4.2.R4`中進行的最佳化，此最佳化使 Neptune 查閱快取在引擎對複本重新啟動時仍然存在。複本重新啟動現在會清除查閱快取。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.2.R5 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.10`
- SPARQL 版本：`1.1`

引擎版本 1.0.4.2.R5 的升級途徑

如果您執行的是引擎版本 `1.0.4.2`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

Amazon Neptune 引擎版本 1.0.4.2.R4 (2021 年 7 月 23 日)

截至 2021 年 7 月 23 日，引擎版本 1.0.4.2.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

這個引擎版本的改善項目

- 已改善查閱快取的行為，以避免在複本上執行快速重設之後清除備援快取。
- 已改善在下列情況處理串流變更日誌的方式：使用伺服器上的最後一個事件 ID 請求 AFTER_SEQUENCE_NUMBER 串流，但該事件 ID 已過期。如果請求的事件 ID 是伺服器上最近清除的事件 ID，則伺服器不再擲回過期的事件 ID 錯誤。

此引擎版本中修正的缺陷

- 已修正 1.0.4.0.R1 中引入的錯誤，其中查詢不會傳回整個大於 760 個字元的字串值。受此錯誤影響的術語是 RDF 常值和 URI，或 Grimlin ID、金鑰和字串值。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.2.R4 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：3.4.10
- SPARQL 版本：1.1

引擎版本 1.0.4.2.R4 的升級途徑

如果您執行的是引擎版本 1.0.4.2，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

Amazon Neptune 引擎版本 1.0.4.2.R3 (2021 年 6 月 28 日)

截止 2021 年 6 月 28 日，引擎版本 1.0.4.2.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中的已知問題

問題：

如果存在空格，則無法在 Accept 標頭中遵循媒體類型的 SPARQL 錯誤。

例如，具有 `-H "Accept: text/csv; q=1.0, */*; q=0.1"` 的查詢會傳回 JSON 輸出而非 CSV 輸出。

解決方法：

如果您移除標頭中 Accept 子句中的空格，引擎會以正確的請求格式傳回輸出。換句話說，不是 `-H "Accept: text/csv; q=1.0, */*; q=0.1"`，而是使用：

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

此引擎版本中修正的缺陷

- 已修正快速重設之後清除複本上查閱快取的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.2.R3 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.10`
- SPARQL 版本：`1.1`

引擎版本 1.0.4.2.R3 的升級途徑

除非您的資料庫叢集使用一或多個 R5d 執行個體，否則此修補程式版本為選用的。如果您的叢集具有 R5d 執行個體，它會在下一個維護時段自動升級。否則，它不會自動升級至此修補程式版本。

您可以使用 AWS CLI apply-pending-maintenance-action 命令 (ApplyPendingMaintenanceAction API)，手動將版本 `1.0.4.2.R2` 升級至此 `1.0.4.2.R3` 版本。

Amazon Neptune 引擎版本 1.0.4.2.R2 (2021 年 6 月 1 日)

截至 2021 年 6 月 1 日，引擎版本 1.0.4.2.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.4.2.R3 (2021 年 6 月 28 日)

此引擎版本中的已知問題

問題：

如果存在空格，則無法在 Accept 標頭中遵循媒體類型的 SPARQL 錯誤。

例如，具有 `-H "Accept: text/csv; q=1.0, */*; q=0.1"` 的查詢會傳回 JSON 輸出而非 CSV 輸出。

解決方法：

如果您移除標頭中 Accept 子句中的空格，引擎會以正確的請求格式傳回輸出。換句話說，不是 `-H "Accept: text/csv; q=1.0, */*; q=0.1"` ，而是使用：

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

這個引擎版本的新功能

- 已新增 R5d 執行個體類型，其中包含查閱快取，可在涉及大量屬性值或 RDF 常值查詢的使用案例中加快讀取速度。請參閱Neptune 查詢快取可加速讀取查詢。
- 已新增一個新的實驗室模式參數，該參數僅可讓實驗 DFE 引擎使用查詢提示 useDFE 根據每個查詢進行調用。

這個引擎版本的改善項目

- 已新增對 TinkerPop 3.4.10 的支援。
- 已新增在傳送 Gremlin 指令碼請求時使用 `withStrategies( )` 組態步驟的支援。具體而言，SubgraphStrategy、PartitionStrategy、ReadOnlyStrategy、EdgeLabelVerification 和 ReservedKeysVerificationStrategy 全都受到支援。
- 已新增對查詢中間的 `V()` 周遊的最佳化。以前，這樣的周遊未在 Neptune 中進行最佳化。
- 已新增對 RFC 2141 URNs 用作 `baseUri` 和 `namedGraphUri` 參數進行大量載入的支援。

此引擎版本中修正的缺陷

- 已修正剖析器中將不正確的查詢視為有效的 Gremlin 錯誤。
- 已修正 Gremlin 錯誤，其中使用 `cap().unfold()` 將 `aggregate()` 副作用展開至 `valueMap()` 會引發例外狀況。

- 已修正 Gremlin 錯誤，其中在 `addV()` 後面的一些 `property()` 步驟會由於發生「無法轉換為字串」錯誤而失敗。
- 已修正 Gremlin 錯誤，以防止某些條件式插入模式引發並行修改例外狀況。
- 已修正 Gimlin 錯誤，以便查詢請求逾時現在無法超過工作階段逾時。
- 已修正 SPARQL 錯誤，其中當遠端伺服器無法使用時，使用 LOAD 或 UNLOAD 的更新可能會失敗，並出現 HTTP 代碼 500 而非 HTTP 代碼 400。
- 已修正錯誤，其中當使用大於 32 位元帶正負號整數限制 (2,147,483,647) 的 `commitNum` 或 opNum 值時，串流 API 呼叫失敗。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.2.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.10`
- SPARQL 版本：`1.1`

引擎版本 1.0.4.2.R2 的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

您不會自動升級至此版本。

升級至此版本

Amazon Neptune 1.0.4.2.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.4.2.R1 (2021 年 5 月 27 日)

從未部署引擎版本 1.0.4.2.R1。

Amazon Neptune 引擎 1.0.4.1 版 (2020 年 12 月 8 日)

截至 2020 年 12 月 8 日，引擎 1.0.4.1 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.4.1.R1.1 (2021 年 3 月 22 日)
- 版本：1.0.4.1.R2 (2021 年 2 月 24 日)

> **Important**
>
> 版本：1.0.4.0 (2020 年 10 月 12 日) 已要求與 Amazon Neptune 的所有連線都必須使用 TLS 1.2 和 HTTPS。不過，該版本中的一個錯誤已允許 HTTP 連線和/或過期的 TLS 連線，繼續適用於先前設定資料庫叢集參數以防止強制執行 HTTPS 連線的客戶。
> 該錯誤已在修補程式版本 1.0.4.0.R2 和 1.0.4.1.R2 中修正，但該修正在修補程式自動安裝時造成了非預期的連線失敗。基於這個原因，這兩個修補程式都已還原，而且只能手動安裝，讓您有機會更新 TLS 1.2 的設定。
> 必須針對與 Neptune 的所有連線使用 SSL/TLS，這會影響您與 Girmlin 主控台、Gremlin 驅動程式、Gremlin Python、.NET、nodeJs、REST API 的連線，同時也會影響負載平衡器連線。如果您到目前為止一直針對任何或全部這些連線使用 HTTP 或舊版 TLS 版本，則必須在將您的系統更新至最新修補程式之前，更新相關的用戶端和驅動程式，並將您的程式碼變更為僅使用 HTTPS。

這個引擎版本的新功能

- 已引進 Neptune ML 功能，為 Amazon Neptune 帶來強大的機器學習功能。請參閱用於圖形上機器學習的 Amazon Neptune ML。
- 已新增自訂 SPARQL UNLOAD 操作，用於移除從遠端來源擷取的資料。請參閱SPARQL UPDATE UNLOAD。

這個引擎版本的改善項目

- 已最佳化一些 Gemlin 條件式插入模式，以避免並行修改例外狀況。

此引擎版本中修正的缺陷

- 已修正可能會導致使用 `as()` 步驟的特定查詢模式遺失結果的 Gemlin 錯誤。
- 已修正在另一個步驟 (例如 `union()`) 內以巢狀方式使用 `project()` 步驟時可能會導致錯誤的 Gemlin 錯誤。
- 已修正 `project()` 步驟中的 Gremlin 錯誤。
- 已修正字串型周遊中 `none()` 步驟無法運作的 Gremlin 錯誤。
- 已修正字串型周遊中的 Gremlin 錯誤，其中不支援空對應做為 `inject()` 步驟的論點。
- 已修正 DFE 引擎中字串型周遊執行中的 Gremlin 錯誤，其中終端方法 (例如 `toList()`) 無法正常運作。
- 已修正在字串查詢中使用 `iterate()` 步驟時無法關閉交易的 Gemlin 錯誤。
- 已修正在某些情況下可能導致使用 `is(P.gte(0))` 模式的查詢擲回例外狀況的 Gemlin 錯誤。
- 已修正在某些情況下可能導致使用 `order().by(T.id)` 模式的查詢擲回例外狀況的 Gemlin 錯誤。
- 已修正在某些情況下可能導致使用 `addV().aggregate()` 模式的查詢提供不正確結果的 Gemlin 錯誤。
- 已修正在某些情況下可能導致使用 `path()` 步驟 (後面跟著 `project()` 步驟模式) 的查詢擲回例外狀況的 Gemlin 錯誤。
- 已修正 SPARQL 錯誤，其中 SUBSTR 函數發出錯誤信號，而不是傳回空字串。
- 已修正 DFE 引擎中的錯誤，此錯誤可能導致非封鎖查詢計畫中的聯結操作，在未繫結的變數存在時產生不正確的結果。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.4.1 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.8`
- SPARQL 版本：`1.1`

引擎 1.0.4.1 版的升級途徑

如果您執行的是引擎版本 `1.0.4.1`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.4.1 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.4.1.R1.1 (2021 年 3 月 22 日)

截至 2021 年 3 月 22 日，引擎版本 1.0.4.1.R1.1 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已停用對可以新增或附加至現有標籤和屬性的 Gemlin 條件式插入模式進行最佳化。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.1.R1.1 之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.8`
- SPARQL 版本：`1.1`

引擎版本 1.0.4.1.R1.1 的升級途徑

如果您執行的是引擎版本 `1.0.4.1`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Amazon Neptune 1.0.4.1.R1.1 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.4.1.R2 (2021 年 2 月 24 日)

截至 2021 年 2 月 24 日，引擎版本 1.0.4.1.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.4.1.R2.1 (2021 年 3 月 11 日)

這個引擎版本的新功能

- Neptune 現在支援以 bzip2 格式壓縮單一檔案，進行大量載入。請參閱載入資料格式。

此引擎版本中修正的缺陷

- 已修正 版本：1.0.4.0 (2020 年 10 月 12 日) 中的錯誤，該錯誤允許使用 HTTP 或更舊版本的 TLS，而不是 HTTPS 和 TLS 1.2 連線到 Neptune。

  > **Important**
  >
  > 與 Neptune 的所有連線都必須使用 SSL/TLS，可能是重大變更。它會影響您與 Girmlin 主控台、Gremlin 驅動程式、Gremlin Python、.NET、nodeJs、REST API 的連線，同時也會影響負載平衡器連線。如果您到目前為止一直針對任何或全部這些連線使用 HTTP 或舊版 TLS 版本，則必須在安裝此修補程式之前更新相關的用戶端和驅動程式，並將您的程式碼變更為僅使用 HTTPS。

- 已修正在某些情況下，當 `ConcurrentModificationException` 發生時 `InternalFailureException` 被設定為回應代碼的 Girmlin 錯誤。
- 已修正在某些情況下更新邊緣或頂點可能導致暫時性 `InternalFailureException` 的 Gemlin 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.1.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.8`
- SPARQL 版本：`1.1`

引擎版本 1.0.4.1.R2 的升級途徑

如果您執行的是引擎版本 `1.0.4.1`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

升級至此版本

Amazon Neptune 1.0.4.1.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎更新，版本 1.0.4.1.R2.1，於 2021 年 3 月 11 日發行。

截至 2021 年 3 月 11 日，引擎版本 1.0.4.1.R2.1 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已停用對可以新增或附加至現有標籤和屬性的 Gemlin 條件式插入模式進行最佳化。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.1.R2.1 之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.8`
- SPARQL 版本：`1.1`

引擎版本 1.0.4.1.R2.1 的升級途徑

如果您執行的是引擎版本 `1.0.4.1.R2`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.4.1.R2.1 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1.R2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1.R2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.4.0 版 (2020 年 10 月 12 日)

截至 2020 年 10 月 12 日，引擎 1.0.4.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.4.0.R2 (2021 年 2 月 24 日)

這個引擎版本的新功能

- 已為 Gremlin 新增框架級壓縮。

這個引擎版本的改善項目

- Amazon Neptune 現在需要使用 Secure Sockets Layer (SSL) 搭配 TLSv1.2 通訊協定，以便在所有區域中使用這些強式密碼套件進行與 Neptune 的所有連線：
  - `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`

  對於與 Neptune 的 REST 和 WebSocket 連線都是如此，這表示在所有區域中連線至 Neptune 時，您必須使用 HTTPS 而非 HTTP。

  因為在任何地方都不再支援使用 HTTP 或 TLS 1.1 的用戶端連線，所以在升級至此引擎版本之前，請確定您的用戶端和程式碼已更新為使用 TLS 1.2 和 HTTPS。

  > **Important**
  >
  > 與 Neptune 的所有連線都必須使用 SSL/TLS，可能是重大變更。它會影響您與 Girmlin 主控台、Gremlin 驅動程式、Gremlin Python、.NET、nodeJs、REST API 的連線，同時也會影響負載平衡器連線。如果您一直在使用 HTTP 進行任何或所有這些連線，則現在必須更新相關的用戶端和驅動程式，並變更您的程式碼以使用 HTTPS，否則您的連線將失敗。
  > 此版本中的一個錯誤已允許 HTTP 連線和/或過期的 TLS 連線，繼續適用於先前設定資料庫叢集參數以防止強制執行 HTTPS 連線的客戶。該錯誤已在修補程式版本 1.0.4.0.R2 和 1.0.4.1.R2 中修正，但該修正在修補程式自動安裝時造成了非預期的連線失敗。
  > 基於這個原因，這兩個修補程式都已還原，而且只能手動安裝，讓您有機會更新 TLS 1.2 的設定。

- 已將 TinkerPop 升級至 3.4.8 版。這是回溯相容升級。如需最新消息，請參閱 TinkerPop 變更日誌。

- 已改善 Gremlin `properties()` 步驟的效能。
- 已在 Explain 和設定檔報告中新增有關 `BindOp` 和 `MultiplexerOp` 的詳細資訊。
- 已新增資料預先提，以在有快取遺漏時改善效能。
- 已在大量載入器的 `parserConfiguration` 參數中新增 `allowEmptyStrings` 設定，這允許在 CSV 載入中將空字串視為有效屬性值 (請參閱 Neptune 載入器請求參數)。
- 載入器現在允許在多值 CSV 資料行中使用逸出分號。

此引擎版本中修正的缺陷

- 已修正與 `both()` 步驟相關的潛在 Gremlin 記憶體洩漏問題。
- 已修正以下錯誤：由於未正確處理以 '/' 結尾的端點，遺失了請求指標。
- 已修正在實驗室模式下啟用 DFE 引擎時，導致複本落後並在大量載入下重新啟動的錯誤。
- 已修正當大量載入由於記憶體不足的情況而失敗時，無法回報正確錯誤訊息的錯誤。
- 已修正 SPARQL 錯誤，其中字元編碼置於 SPARQL 查詢回應中的 Content-Encoding 標頭中。現在 `charset` 改為置於 Content-Type 標頭中，這可讓 HTTP 用戶端辨識正在自動使用的字元集。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.4.0 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.8`
- SPARQL 版本：`1.1`

引擎 1.0.4.0 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

您不會自動升級至此版本。

升級至此版本

Amazon Neptune 1.0.4.0 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.4.0.R2 (2021 年 2 月 24 日)

截至 2021 年 2 月 24 日，引擎版本 1.0.4.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已修正 版本：1.0.4.0 (2020 年 10 月 12 日) 中的錯誤，該錯誤允許使用 HTTP 或更舊版本的 TLS，而不是 HTTPS 和 TLS 1.2 連線到 Neptune。

  > **Important**
  >
  > 與 Neptune 的所有連線都必須使用 SSL/TLS，可能是重大變更。它會影響您與 Girmlin 主控台、Gremlin 驅動程式、Gremlin Python、.NET、nodeJs、REST API 的連線，同時也會影響負載平衡器連線。如果您到目前為止一直針對任何或全部這些連線使用 HTTP 或舊版 TLS 版本，則必須在安裝此修補程式之前更新相關的用戶端和驅動程式，並將您的程式碼變更為僅使用 HTTPS。

- 已修正 CSV 大量載入中涉及標籤以 # 結尾的錯誤。
- 已修正在某些情況下，當 `ConcurrentModificationException` 發生時 `InternalFailureException` 被設定為回應代碼的 Girmlin 錯誤。

- 已修正在某些情況下更新邊緣或頂點可能導致暫時性 `InternalFailureException` 的 Gemlin 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.4.0.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.8`
- SPARQL 版本：`1.1`

引擎版本 1.0.4.0.R2 的升級途徑

如果您執行的是引擎版本 `1.0.4.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.4.0.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.3.0 版 (2020 年 8 月 3 日)

截至 2020 年 8 月 3 日，引擎 1.0.3.0 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.3.0.R2 (2020 年 10 月 12 日)
- 版本：1.0.3.0.R3 (2021 年 2 月 19 日)

這個引擎版本的新功能

- Neptune 引進了一種新的替代查詢引擎 (DFE)，可以大幅地加快查詢執行速度。請參閱Amazon Neptune 替代查詢引擎（DFE）。
- DFE 依賴有關 Neptune 圖形資料的預先產生統計資料，這些統計資料是透過新的統計資料端點管理的。請參閱DFE 統計資料。
- 您現在可以將新的 `includeQueuedLoads` 參數設定為 FALSE，從載入器 Get-Status API 傳回的載入 ID 清單中排除排入佇列的載入工作。請參閱Neptune 載入器 Get-Status 請求參數。
- Neptune 現在支援 SPARQL 查詢回應的結尾標頭，如果請求在開始傳回回應區塊之後失敗，這些查詢回應可能會包含錯誤代碼和訊息。請參閱用於多部分 SPARQL 回應的選用 HTTP 結尾標頭。
- Neptune 現在也可以讓您針對 Gremlin 查詢啟用區塊回應編碼。與 SPARQL 案例一樣，回應區塊的結尾標頭可能會包含錯誤代碼和訊息，前提是在查詢開始傳回回應區塊之後發生失敗。請參閱使用選用的 HTTP 結尾標頭來啟用多部分 Gremlin 回應。

這個引擎版本的改善項目

- 您現在可以將批次請求的大小提供給 ElasticSearch，以在 Gemlin 中進行全文檢索搜尋。
- 已改善 SPARQL GROUP BY 的記憶體使用量。
- 新增了一個新的 Gemlin 查詢最佳化工具來刪改某些未限制的篩選條件。
- 已將使用 IAM 驗證的 WebSocket 連線可以保持開啟狀態的時間上限從 36 小時增加到 10 天。

此引擎版本中修正的缺陷

- 已修正下列錯誤：如果您在 POST 請求中傳送了未編碼的 URL 參數，Neptune 會傳回 HTTP 狀態碼 500 和 `InternalServerErrorException`。現在，Neptune 會傳回 HTTP 狀態碼 400

和 BadRequestException，並顯示訊息：Failure to process the POST request parameters。

- 已修正未正確報告 WebSocket 連線失敗的 Gremlin 錯誤。
- 已修正涉及 sideEffects 消失的 Gremlin 錯誤。
- 已修正未適當支援全文檢索搜尋 batchsize 參數的 Girmlin 錯誤。
- 已修正 Gremlin 錯誤，為 bothE 上的每個方向單獨處理 toV 和 fromV。
- 已修正 hasLabel 步驟中涉及 Edge pathType 的 Gremlin 錯誤。
- 已修正使用靜態繫結來重新排序聯結無法正常運作的 SPARQL 錯誤。
- 已修正 SPARQL UPDATE LOAD 錯誤，其中未正確報告無法使用的 Amazon S3 儲存貯體。
- 已修正 SPARQL 錯誤，其中未正確報告子查詢中的 SERVICE 節點發生問題。
- 已修正 SPARQL 錯誤，其中包含巢狀 FILTER EXISTS 或 FILTER NOT EXISTS 條件的查詢未適當地進行評估。
- 已修正 SPARQL 錯誤，以在透過產生查詢呼叫 SPARQL 服務端點時，正確處理重複產生的繫結。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.3.0 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：3.4.3
- SPARQL 版本：1.1

引擎 1.0.3.0 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

如果您的叢集已將其 AutoMinorVersionUpgrade 參數設為 True，則在維護時段期間，您的叢集會在此發行日期後兩至三週自動升級至此引擎版本。

升級至此版本

Amazon Neptune 1.0.3.0 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.3.0.R3 (2021 年 2 月 19 日)

截至 2021 年 2 月 19 日，引擎版本 1.0.3.0.R3 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已修正 CSV 大量載入中涉及標籤以 # 結尾的錯誤。
- 已修正可能會導致使用 `as()` 步驟的特定查詢模式遺失結果的 Gemlin 錯誤。
- 已修正在另一個步驟 (例如 `union()`) 內以巢狀方式使用 `project()` 步驟時可能會導致錯誤的 Gemlin 錯誤。
- 已修正當使用類似 `toList()` 的終端方法時，於實驗 DFE 引擎中進行字串周遊執行所發生的 Gremlin 錯誤。
- 已修正在字串查詢中使用 `iterate()` 步驟時無法關閉交易的 Gemlin 錯誤。
- 已修正在特定情況下可能導致使用 `is(P.gte(0))` 模式的查詢擲回例外狀況的 Gemlin 錯誤。
- 已修正在某些情況下，當 `ConcurrentModificationException` 發生時 `InternalFailureException` 被設定為回應代碼的 Girmlin 錯誤。
- 已修正在某些情況下更新邊緣或頂點可能導致暫時性 `InternalFailureException` 的 Gemlin 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.3.0.R3 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：3.4.8
- SPARQL 版本：1.1

引擎版本 1.0.3.0.R3 的升級途徑

如果您執行的是引擎版本 1.0.3.0，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.3.0.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.3.0.R2 (2020 年 10 月 12 日)

截至 2020 年 10 月 12 日，引擎版本 1.0.3.0.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

這個引擎版本的改善項目

- 已改善 Gremlin `properties()` 步驟的效能。

- 已在 Explain 和設定檔報告中新增有關 `BindOp` 和 `MultiplexerOp` 的詳細資訊。
- 對於 SPARQL 查詢回應，已將 `charset` 新增至 Content-Type 標頭，讓 HTTP 用戶端可以辨識自動使用的字元集。

此引擎版本中修正的缺陷

- 已修正未處理 `CancellationException` 的 SPARQL 錯誤。
- 已修正 SPARQL 錯誤，其中包含巢狀選項的查詢無法正常運作。
- 已修正 LOAD 中的 SPARQL 錯誤，其中 `ConcurrentModificationException` 可能導致查詢停止回應。
- 已修正 SPARQL 錯誤，此錯誤會阻止查詢回應使用 gzip 進行壓縮。
- 已修正 `groupBy()` 步驟中的 Gremlin 錯誤。
- 已修正與在 `local()` 步驟內使用 `aggregate()` 步驟相關的 Gremlin 錯誤。
- 已修正與使用 `bothE()` (其後跟著使用彙總值的述詞) 相關的 Gremlin 錯誤。
- 已修正與使用 `bothE()` 步驟搭配 `repeat()` 步驟相關的 Gremlin 錯誤。
- 已修正與 `both()` 步驟相關的潛在 Gremlin 記憶體洩漏問題。
- 已修正以下錯誤：由於未正確處理以 '/' 結尾的端點，遺失了請求指標。
- 已修正即使請求佇列未滿，也可能引發 `ThrottlingException` 的錯誤。
- 已修正當載入由於 `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE` 等原因失敗時，擷取載入狀態的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.3.0.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.3`
- SPARQL 版本：`1.1`

引擎版本 1.0.3.0.R2 的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

如果您的叢集已將其 `AutoMinorVersionUpgrade` 參數設為 `True`，則在維護時段期間，您的叢集會在此發行日期後兩至三週自動升級至此引擎版本。

升級至此版本

Amazon Neptune 1.0.3.0.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.2.2 版 (2020 年 3 月 9 日)

截至 2020 年 3 月 9 日，引擎 1.0.2.2 版已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此版本的後續修補程式版本

- 版本：1.0.2.2.R2 (2020 年 4 月 2 日)
- 版本：1.0.2.2.R3 (2020 年 7 月 22 日)
- 版本：1.0.2.2.R4 (2020 年 7 月 23 日)
- 版本：1.0.2.2.R5 (2020 年 10 月 12 日)
- 版本：1.0.2.2.R6 (2021 年 2 月 19 日)

這個引擎版本的改善項目

- 已將正要復原之交易的相關資訊新增至狀態 API。請參閱執行個體狀態。
- 已將 Apache TinkerPop 的版本升級至 3.4.3。

3.4.3 版可與 Neptune 支援的舊版 (3.4.1) 相容。它確實在行為中引入次要變更：當您嘗試關閉不存在的工作階段時，Gremlin 不再傳回錯誤 (請參閱防止在關閉不存在的工作階段時發生錯誤)。

- 已移除執行 Gremlin 全文搜尋步驟時的效能瓶頸。

此引擎版本中修正的缺陷

- 已修正在處理查詢中的空白圖形模式時發生的 SPARQL 錯誤。
- 已修正在處理 URL 編碼查詢中的未編碼分號時發生的 SPARQL 錯誤。
- 已修正在處理 Union 步驟中的重複頂點時發生的 Gremlin 錯誤。
- 已修正一個 Gremlin 錯誤，此錯誤會導致在 `.repeat()` 內具有 `.simplePath()` 或 `.cyclicPath()` 的查詢傳回不正確的結果。
- 已修正一個 Gremlin 錯誤，此錯誤會導致 `.project()` 在其子周遊未傳回任何解決方案時傳回不正確的結果。
- 已修正一個 Gremlin 錯誤，其中來自讀寫衝突的錯誤引發了 `InternalFailureException` 而不是 `ConcurrentModificationException`。
- 已修正導致 `.group().by(...).by(values("property"))` 失敗的 Gremlin 錯誤。
- 已修正全文搜尋步驟之設定檔輸出中的 Gremlin 錯誤。
- 已修正 Gremlin 工作階段中的資源洩漏。
- 已修正在某些情況下狀態 API 無法報告正確的可訂購版本的錯誤。
- 已修正一個大量載入器錯誤，此錯誤允許使用 Amazon S3 以外之位置的 URL 做為大量載入請求中的來源。
- 已修正詳細載入狀態下的大量載入器錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.2.2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.3`
- SPARQL 版本：`1.1`

引擎 1.0.2.2 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

如果您的叢集已將其 AutoMinorVersionUpgrade 參數設為 True，則在維護時段期間，您的叢集會在此發行日期後兩至三週自動升級至此引擎版本。

升級至此版本

Amazon Neptune 1.0.2.2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.2.R6 (2021 年 2 月 19 日)

截至 2021 年 2 月 19 日，引擎版本 1.0.2.2.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 已修正在某些情況下，當 `ConcurrentModificationException` 發生時 `InternalFailureException` 被設定為回應代碼的 Girmlin 錯誤。
- 已修正在某些情況下更新邊緣或頂點可能導致暫時性 `InternalFailureException` 的 Gemlin 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.2.R6 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：3.4.8
- SPARQL 版本：1.1

引擎版本 1.0.2.2.R6 的升級途徑

如果您執行的是引擎版本 1.0.2.2，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.2.2.R6 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.2.R5 (2020 年 10 月 12 日)

截至 2020 年 10 月 12 日，引擎版本 1.0.2.2.R5 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

這個引擎版本的改善項目

- 已改善 Gremlin `properties()` 步驟的效能。

- 已在 Explain 和設定檔報告中新增有關 `BindOp` 和 `MultiplexerOp` 的詳細資訊。
- 對於 SPARQL 查詢回應，已將 `charset` 新增至 Content-Type 標頭，讓 HTTP 用戶端可以辨識自動使用的字元集。

此引擎版本中修正的缺陷

- 已修正未處理 `CancellationException` 的 SPARQL 錯誤。
- 已修正 SPARQL 錯誤，其中包含巢狀選項的查詢無法正常運作。
- 已修正 LOAD 中的 SPARQL 錯誤，其中 `ConcurrentModificationException` 可能導致查詢停止回應。
- 已修正 SPARQL 錯誤，此錯誤會阻止查詢回應使用 gzip 進行壓縮。
- 已修正 `groupBy()` 步驟中的 Gremlin 錯誤。
- 已修正與在 `local()` 步驟內使用 `aggregate()` 步驟相關的 Gremlin 錯誤。
- 已修正與使用 `bothE()` (其後跟著使用彙總值的述詞) 相關的 Gremlin 錯誤。
- 已修正與使用 `bothE()` 步驟搭配 `repeat()` 步驟相關的 Gremlin 錯誤。
- 已修正與 `both()` 步驟相關的潛在 Gremlin 記憶體洩漏問題。
- 已修正以下錯誤：由於未正確處理以 '/' 結尾的端點，遺失了請求指標。
- 已修正即使請求佇列未滿，也可能引發 `ThrottlingException` 的錯誤。
- 已修正當載入由於 `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE` 等原因失敗時，擷取載入狀態的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.2.R5 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.3`
- SPARQL 版本：`1.1`

引擎版本 1.0.2.2.R5 的升級途徑

如果您執行的是引擎版本 `1.0.2.2`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.2.2.R5 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.2.R4 (2020 年 7 月 23 日)

截至 2020 年 7 月 23 日，引擎版本 1.0.2.2.R4 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

這個引擎版本的改善項目

- 更頻繁地將未使用的記憶體釋放回作業系統，藉此改善了記憶體使用量。
- 也改善了 SPARQL GROUP BY 查詢的記憶體使用量。
- 已將使用 IAM 驗證的 WebSocket 連線可以保持開啟狀態的時間上限從 36 小時增加到 10 天。
- 已新增 `BufferCacheHitRatio` CloudWatch 指標，其在診斷查詢延遲和調校執行個體類型時很有用。請參閱Neptune 指標。

此引擎版本中修正的缺陷

- 已修正關閉閒置或過期 IAM WebSocket 連線時發生的錯誤。Neptune 現在會關閉連線之前傳送一個關閉框架。

- 已修正在評估包含巢狀 FILTER EXISTS 和/或 FILTER NOT EXISTS 條件的查詢時發生的 SPARQL 錯誤。
- 已修正 SPARQL 查詢終止錯誤，該錯誤在特定極端情況下導致了伺服器上的執行緒遭到封鎖。
- 已修正 `hasLabel` 步驟中涉及邊緣 pathType 的 Gremlin 錯誤。
- 已修正 Gremlin 錯誤，為 `bothE` 上的每個方向單獨處理 `toV` 和 `fromV`。
- 已修正涉及 sideEffects 消失的 Gremlin 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.2.R4 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.3`
- SPARQL 版本：`1.1`

引擎版本 1.0.2.2.R4 的升級途徑

如果您執行的是引擎版本 `1.0.2.2`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.2.2.R4 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
```

```
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.2.R3 (2020 年 7 月 22 日)

引擎版本 1.0.2.2.R3 已納入引擎版本 1.0.2.2.R4。

Amazon Neptune 引擎版本 1.0.2.2.R2 (2020 年 4 月 2 日)

截至 2020 年 4 月 2 日，引擎版本 1.0.2.2.R2 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

這個引擎版本的改善項目

- 您現在可以將多達 64 個大量載入的任務排入佇列，而不必等待其中一個完成，再啟動下一個。您也可以在使用 `load` 命令的 `dependencies` 參數順利完成一或多個先前佇列的載入任務時，執行佇列的載入請求。請參閱Neptune 載入器命令。
- 現在可以對全文搜尋輸出進行排序 (請參閱全文檢索搜尋參數)。
- 現在已有用於呼叫 Neptune 串流的資料庫叢集參數，且該功能已移出實驗室模式。請參閱啟用 Neptune 串流。

此引擎版本中修正的缺陷

- 已修正伺服器啟動時的隨機故障，此故障會延遲執行個體的建立。
- 已修正最佳化工具問題，其中，查詢中的 `BIND` 陳述式讓最佳化工具在連結順序規劃中使用非選擇性模式展開。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.2.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.3`
- SPARQL 版本：`1.1`

引擎版本 1.0.2.2.R2 的升級途徑

如果您執行的是引擎版本 `1.0.2.2`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.2.2.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.2.1 版 (2019 年 11 月 22 日)

此版本的後續修補程式版本

- 版本：1.0.2.1.R6 (2020 年 4 月 22 日)
- 版本：1.0.2.1.R5 (2020 年 4 月 22 日) 未部署此修補程式版本。
- 版本：1.0.2.1.R4 (2019 年 12 月 20 日)
- 版本：1.0.2.1.R3 (2019 年 12 月 12 日)
- 版本：1.0.2.1.R2 (2019 年 11 月 25 日)

這個引擎版本的新功能

- 已透過與 Amazon OpenSearch Service 整合，新增全文檢索搜尋功能。請參閱 Neptune 全文檢索搜尋

- 已新增使用實驗室模式的選項，為大量述詞建立第四個索引 (OSGP 索引)。請參閱OSGP 索引。
- 將詳細資訊模式新增至 SPARQL Explain。如需詳細資訊，請參閱使用 SPARQL `explain`和詳細資訊模式輸出。
- 將實驗室模式資訊新增至引擎狀態報告。如需詳細資訊，請參閱 執行個體狀態。
- 資料庫叢集快照現在可以跨 AWS 區域複製。請參閱複製快照。

這個引擎版本的改善項目

- 改進處理大量述詞時的效能。
- 增強查詢最佳化。儘管這對客戶而言，應該是完全透明的，但建議您在升級前，先測試應用程式，以確保應用程式如預期運作。
- 錯誤報告的小規模增強功能。
- 已新增 Gremlin `.project()` 和 `.identity()` 步驟的最佳化。
- 已新增非終結 Gremlin `.union()` 案例的最佳化。
- 已新增 Gremlin `.path().by()` 周遊的原生支援。
- 已新增 Gremlin `.coalesce()` 的原生支援。
- 進一步最佳化大量寫入。
- 我們現在要求 HTTPS 連線至少使用 TLS 版本 1.2 或更高版本，以防止使用過時/不安全的密碼。

此引擎版本中修正的缺陷

- 已修正 Gremlin `addE()` 內部周遊處理錯誤。
- 已修正由 AST 註釋從子系周遊洩漏到父系周遊造成的 Gremlin 錯誤。
- 已修正在 `select()` 之後呼叫 `.otherV()` 時，在 Gremlin 中發生的錯誤。
- 已修正某些 `.hasLabel()` 步驟出現在 `bothE()` 步驟之後會失敗的 Gremlin 錯誤。
- 對 Gremlin .sum() 和 .project() 進行小規模修正。
- 已修正處理 SPARQL 查詢缺少右括號的錯誤。
- 已修正 SPARQL Explain 中的一些小錯誤。
- 已修正處理並行取得載入狀態要求時的錯誤。
- 減少利用 `.project()` 步驟執行一些 Gremlin 周遊所使用的記憶體。
- 已修正 SPARQL 中特殊值的數值比較。請參閱標準合規。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.2.1 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`
- SPARQL 版本：`1.1`

引擎 1.0.2.1 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

您不會自動升級至此版本。

升級至此版本

Amazon Neptune 1.0.2.1 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.1.R6 (2020 年 4 月 22 日)

截至 2020 年 4 月 22 日，引擎版本 1.0.2.1.R6 已普遍部署。請注意，新版本需要數天才能在每個區域推出。

此引擎版本中修正的缺陷

- 修正以下錯誤：ConcurrentModificationConflictException 和 TransactionException 未轉換成 NeptuneGremlinException，導致傳回 InternalFailureException 給客戶。
- 已修正以下錯誤：在伺服器完全準備就緒之前，Neptune 回報其狀態良好。
- 修正以下錯誤：同時插入兩個 value->id 對應時，未按照順序遞交字典和使用者交易。
- 修正載入狀態序列化的錯誤。
- 修正 Gremlin 工作階段錯誤。
- 已修正以下錯誤：伺服器無法啟動時，Neptune 無法擲回例外狀況。
- 已修正以下錯誤：Neptune 無法在關閉頻道前傳送 Web Socket 關閉框架。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.1.R6 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：3.4.1
- SPARQL 版本：1.1

引擎版本 1.0.2.1.R6 的升級途徑

如果您執行的是引擎版本 1.0.2.1，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.2.1.R6 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
```

```
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.1.R5 (2020 年 4 月 22 日)

從未部署引擎版本 1.0.2.1.R5。

Amazon Neptune 引擎版本 1.0.2.1.R4 (2019 年 12 月 20 日)

這個引擎版本的改善項目

- Neptune 現在會始終嘗試先在執行管道中放置任何全文檢索搜尋呼叫。這減少了對 OpenSearch 的呼叫量，因而可大幅提升效能。請參閱全文檢索搜尋查詢執行。
- 如果您嘗試存取不存在的屬性、頂點或邊緣，則 Neptune 現在會引發 `IllegalArgumentException`。先前，Neptune 已在該情況下引發了 `UnsupportedOperationException`。

  例如，如果您嘗試新增一個參考不存在頂點的邊緣，則您現在將引發一個 `IllegalArgumentException`。

此引擎版本中修正的缺陷

- 已修正 Gremlin 錯誤，其中 `project-by` 內的 `union` 周遊不會傳回結果或傳回不正確的結果。
- 已修正導致巢狀 `.project().by()` 步驟傳回不正確結果的 Gremlin 錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.1.R4 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`

- SPARQL 版本：1.1

引擎版本 1.0.2.1.R4 的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

不過，不支援自動更新到此版本。

升級至此版本

Amazon Neptune 1.0.2.1.R4 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.1.R3 (2019 年 12 月 12 日)

此引擎版本中修正的缺陷

- 已修正下列錯誤：即使已在 `neptune_lab_mode` 參數中使用 `ObjectIndex` 值，正確地啟用在 實驗室模式 使用的功能，仍會停用 OSGP 索引。
- 已修正因 `.project().by()` 步驟內部的 `.fold()` 而影響 Gremlin 查詢的錯誤。例如，此錯誤會造成下列查詢傳回不完整的結果：

```
g.V().project("a").by(valueMap().fold())
```

- 已修正大量載入 RDF 資料的效能瓶頸。
- 已修正在啟用串流時造成複本當機，且複本在主版之前重新啟動的錯誤。
- 已修正在未重新啟動書執行個體的情況下，未挑選在執行個體上輪換的 SSL 憑證的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.1.R3 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`
- SPARQL 版本：`1.1`

引擎版本 1.0.2.1.R3 的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

不過，不支援自動更新到此版本。

升級至此版本

Amazon Neptune 1.0.2.1.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
```

```
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.1.R2 (2019 年 11 月 25 日)

此引擎版本中修正的缺陷

- 已修正透過非循環配置資源依序周遊和非 `path()` 依序周遊影響所有 `project().by()` 查詢的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.1.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`
- SPARQL 版本：`1.1`

引擎版本 1.0.2.1.R2 的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

不過，不支援自動更新到此版本。

升級至此版本

Amazon Neptune 1.0.2.1.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.2.0 版 (2019 年 11 月 8 日)

重要：此引擎版本現在已棄用

從 2020 年 5 月 19 日開始，將不會建立任何使用此引擎版本的新執行個體。

此引擎版本現已由 1.0.2.1 版取代，其中包含此版本中的所有錯誤修正及其他功能，例如全文檢索搜尋整合、OSGP 索引支援，以及跨 AWS 區域的資料庫快照叢集複製。

從 2020 年 6 月 1 日開始，Neptune 將在下一個維護時段期間，自動將執行此引擎版本的任何叢集升級為 1.0.2.1 版的最新修補程式。您可以在此之前手動升級，如此處所述。

如果您對升級有任何問題，請透過 AWS Support 或 AWS 開發人員論壇與我們聯絡。

此版本的後續修補程式版本

- 版本：1.0.2.0.R3 (2020 年 5 月 5 日)
- 版本：1.0.2.0.R2 (2019 年 11 月 21 日)

這個引擎版本的新功能

除了維護更新以外，此版本還增加了新功能，一次支援多個引擎版本 (請參閱 維護 Amazon Neptune 資料庫叢集)。

因此，引擎版本的編號已變更 (請參閱 引擎版本 1.3.0.0 之前的版本編號)。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.2.0 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`

- SPARQL 版本：1.1

引擎 1.0.2.0 版的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

您不會自動升級至此版本。

升級至此版本

Amazon Neptune 1.0.2.0 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.0.R3 (2020 年 5 月 5 日)

重要：此引擎版本現在已棄用

從 2020 年 5 月 19 日開始，將不會建立任何使用此引擎版本的新執行個體。

此引擎版本現已由 1.0.2.1 版取代，其中包含此版本中的所有錯誤修正及其他功能，例如全文檢索搜尋整合、OSGP 索引支援，以及跨 AWS 區域的資料庫快照叢集複製。

從 2020 年 6 月 1 日開始，Neptune 將在下一個維護時段期間，自動將執行此引擎版本的任何叢集升級為 1.0.2.1 版的最新修補程式。您可以在此之前手動升級，如此處所述。

如果您對升級有任何問題，請透過 AWS Support 或 AWS 開發人員論壇與我們聯絡。

此引擎版本中修正的缺陷

- 修正以下錯誤：`ConcurrentModificationConflictException` 和 `TransactionException` 被回報為一般 `InternalFailureException`。
- 修正運作狀態檢查中導致伺服器在啟動期間頻繁重新啟動的錯誤。
- 修正以下錯誤：資料因為在某些情況下未按照順序遞交而無法顯示在複本上。
- 已修正載入狀態序列化的以下錯誤：因為缺乏 Amazon S3 存取許可而導致載入失敗。
- 已修正 Gremlin 工作階段中的資源洩漏。
- 已修正運作狀態檢查的以下錯誤：在啟動管理 IAM 驗證的元件時隱藏狀況不良狀態。
- 已修正以下錯誤：Neptune 無法在關閉頻道前傳送 WebSocket 關閉框架。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.0.R3 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`
- SPARQL 版本：`1.1`

引擎版本 1.0.2.0.R3 的升級途徑

如果您執行的是引擎版本 `1.0.2.0`，您的叢集將在下一個維護時段自動升級至此修補程式版本。

您可以將任何較早的 Neptune 引擎版本手動升級為此版本。

升級至此版本

Amazon Neptune 1.0.2.0.R3 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：

```
  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.
```

如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎版本 1.0.2.0.R2 (2019 年 11 月 21 日)

重要：此引擎版本現在已棄用

從 2020 年 5 月 19 日開始，將不會建立任何使用此引擎版本的新執行個體。

此引擎版本現已由 1.0.2.1 版取代，其中包含此版本中的所有錯誤修正及其他功能，例如全文檢索搜尋整合、OSGP 索引支援，以及跨 AWS 區域的資料庫快照叢集複製。

從 2020 年 6 月 1 日開始，Neptune 將在下一個維護時段期間，自動將執行此引擎版本的任何叢集升級為 1.0.2.1 版的最新修補程式。您可以在此之前手動升級，如此處所述。

如果您對升級有任何問題，請透過 AWS Support 或 AWS 開發人員論壇與我們聯絡。

此引擎版本中修正的缺陷

- 為伺服器上的中途分頁改善快取策略，以便在伺服器進入記憶體不足的狀態時，`FreeableMemory` 能夠更快復原。
- 已修正伺服器上處理許多並行載入狀態及/或啟動載入要求時，可能導致競爭條件和當機的錯誤。

此版本支援的查詢語言版本

將資料庫叢集升級至版本 1.0.2.0.R2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`
- SPARQL 版本：`1.1`

引擎版本 1.0.2.0.R2 的升級途徑

您可以將任何先前的 Neptune 引擎版本手動升級為此版本。

不過，不支援自動更新到此版本。

升級至此版本

Amazon Neptune 1.0.2.0.R2 現已正式推出。

如果資料庫叢集執行的引擎版本具有升級至此版本的途徑，則有資格立即升級。您可以使用主控台上的資料庫叢集操作或使用 SDK 來升級任何有資格的叢集。以下 CLI 命令將立即升級有資格的叢集：

對於 Linux、OS X 或 Unix：

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

針對 Windows：

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，所有這些執行個體都需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集。

一律在升級之前執行測試

發佈新的主要或次要 Neptune 引擎版本時，請一律先在其上測試 Neptune 應用程式，然後再升級至其中。即使是次要升級，也可能會引入將影響程式碼的新功能或行為。

首先，請將目前版本中的版本備註頁面與目標版本的版本備註頁面進行比較，以查看查詢語言版本中是否將有變更，或有其他重大變更。

在升級生產資料庫叢集之前測試新版本的最佳方式是複製您的生產叢集，以便複製執行新的引擎版本。然後，您可以在複製上執行查詢，而不會影響生產資料庫叢集。

升級前一律建立手動快照

在執行升級之前，強烈建議您一律建立資料庫叢集的手動快照。具有自動快照僅會提供短期保護，而手動快照仍然可用，直到您明確將其刪除為止。

在某些情況下，Neptune 會為您建立手動快照，作為升級程序的一部分，但您不應該依賴此快照，而且在任何情況下都應該建立自己的手動快照。

確定不需要將資料庫叢集還原為升級前狀態時，您可以明確刪除您自己建立的手動快照，以及 Neptune 可能已建立的手動快照。如果 Neptune 建立手動快照集，它會具有開頭為 `preupgrade` 的名稱，後面跟著資料庫叢集的名稱、來源引擎版本、目標引擎版本和日期。

> **Note**
>
> 如果您嘗試在待定動作進行中時進行升級，可能會遇到如下錯誤：
>
> ```
> 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.
> ```
>
> 如果遇到此錯誤，請等候待定動作完成，或立即觸發維護時段，讓先前的升級完成。

如需有關升級引擎版本的詳細資訊，請參閱 維護 Amazon Neptune 資料庫叢集。如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

Amazon Neptune 引擎 1.0.1.2 版 (2020 年 6 月 10 日)

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

這個引擎版本的改善項目

- 如果您嘗試存取不存在的屬性、頂點或邊緣，則 Neptune 現在會引發 `IllegalArgumentException`。先前，Neptune 已在該情況下引發了 `UnsupportedOperationException`。

例如，如果您嘗試新增一個參考不存在頂點的邊緣，則您現在將引發一個 `IllegalArgumentException`。

此引擎版本中修正的缺陷

- 修正以下錯誤：同時插入兩個 `value->id` 對應時，未按照順序遞交字典和使用者交易。
- 修正載入狀態序列化的錯誤。
- 已修正伺服器啟動時的隨機故障，此故障會延遲執行個體的建立。
- 已修正游標流失的問題。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.1.2 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：`3.4.1`
- SPARQL 版本：`1.1`

Amazon Neptune 引擎 1.0.1.1 版 (2020 年 6 月 26 日)

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

此引擎版本中修正的缺陷

- 修正了同時插入時提交的失序錯誤。
- 修正載入狀態序列化的錯誤。
- 已修正伺服器啟動時的隨機故障，此故障會延遲執行個體的建立。
- 已修正記憶體流失。

此版本支援的查詢語言版本

將資料庫叢集升級至 1.0.1.1 版之前，請確定您的專案與下列查詢語言版本相容：

- Gremlin 版本：3.3.2
- SPARQL 版本：1.1

Amazon Neptune 引擎 1.0.1.0 版 (2019 年 7 月 2 日)

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

Amazon Neptune 引擎更新 2019 年 10 月 31 日

版本：

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

此引擎版本中修正的缺陷

- 修正了當用戶端使用 `traversal().withRemote(...)` (亦即使用 GLV 位元碼) 連接到 Neptune 時，`tree()` 步驟回應序列化中的 Gremlin 錯誤。

  此版本解決了使用 `traversal().withRemote(...)` 連線至 Neptune 的用戶端接收到對包含 `tree()` 步驟之 Gremlin 查詢的無效回應的問題。
- 修正了 `DELETE WHERE LIMIT` 查詢中的 SPARQL 錯誤，其中查詢終止程序會因競爭條件而停止回應，造成查詢逾時。

Amazon Neptune 引擎更新 2019 年 10 月 15 日

版本：1.0.1.0.200463.0

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

這個引擎版本的新功能

- 已新增 Gremlin Explain/描述檔功能 (請參閱 使用 Gremlin `explain` 分析 Neptune 查詢執行)。

- 已新增 支援 Gremlin 指令碼型工作階段 以在單一交易中啟用多個 Gremlin 周遊的執行。
- 已在 Neptune 中新增對 SPARQL 聯合查詢延伸模組的支援 (請參閱 SPARQL 1.1 聯合查詢和 Neptune 中使用 SERVICE 延伸模組的 SPARQL 聯合查詢)。
- 已新增一項功能，讓您可透過 HTTP URL 參數或透過 SPARQL `queryId` 查詢提示 (請參閱 將自訂 ID 注入至 Neptune Gremlin 或 SPARQL 查詢)，將自己的 `queryId` 注入至 Gremlin 或 SPARQL 查詢。
- 已將 實驗室模式 功能新增至 Neptune，讓您可以試用尚未準備好用於生產環境但即將推出的功能。
- 已新增即將推出的 Neptune 串流 功能，可將對資料庫進行的每項變更可靠地記錄到持續一週的串流中。此功能只能在實驗室模式中使用。
- 更新並行交易的正式語意 (請參閱Neptune 中的交易語意)。此功能針對並行提供產業標準保證。

  根據預設，會啟用這些交易語意。在某些情況下，此功能可能會變更目前的載入行為，並降低載入效能。您可以在參數值中包含 `ReadWriteConflictDetection=disabled`，以使用資料庫叢集 `neptune_lab_mode` 參數來還原到先前語意。

這個引擎版本的改善項目

- 已改善 執行個體狀態 API，方法為回報引擎使用哪個 TinkerPop 版本和哪個 SPARQL 版本。
- 已提升 Gremlin 子圖形運算子的效能。
- 已提升 Gremlin 回應序列化的效能。
- 已提升 Gremlin Union 步驟的效能。
- 已縮短簡單 SPARQL 查詢的延遲。

此引擎版本中修正的缺陷

- 已修正 Gremlin 錯誤，其中未正確地以內部失敗形式傳回逾時。
- 已修正 SPARQL 錯誤，其中局部變數集的 ORDER BY 已導致內部伺服器錯誤。

Amazon Neptune 引擎更新 2019 年 9 月 19 日

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

版本：1.0.1.0.200457.0

Amazon Neptune 1.0.1.0.200457.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200457.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令來升級 DB 叢集：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

此引擎版本中修正的缺陷

- 已修正先前引擎版本 (1.0.1.0.200369.0) 中引進的 Gremlin 正確性問題，方法為移除造成此問題之接續述詞處理的效能改善。
- 已修正 SPARQL 錯誤，此錯誤會導致搭配 `DISTINCT` 的查詢，以及包裝為 `OPTIONAL` 的單一模式產生 `InternalServerError`。

Amazon Neptune 引擎更新 2019 年 8 月 13 日

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

這個引擎版本的新功能

- 已將 `OVERSUBSCRIBE` 選項新增至 Neptune 載入器命令 的 `parallelism` 參數，這會導致 Neptune 大量載入器使用所有可用的執行緒和資源。

這個引擎版本的改善項目

- 改善包含簡單邏輯 OR 運算式的 SPARQL 篩選條件的效能。

- 改善處理接續述詞方面的 Gremlin 效能。

此引擎版本中修正的缺陷

- 已修正防止從 `xsd:date` 減去 `xsd:duration` 的 SPARQL 錯誤。
- 修正導致存在 `UNION` 時靜態內嵌的結果不完整的 SPARQL 錯誤。
- 修正查詢取消中的 SPARQL 錯誤。
- 修正類型提升期間導致溢位的 Gremlin 錯誤。
- 修正 `addE().from().to()` 步驟中處理頂點元素時的 Gremlin 錯誤。
- 修正涉及單一基數插入中 NaN 雙精位數和浮點數處理的 Gremlin 錯誤 (2019-07-26 於引擎版本 1.0.1.0.200366.0 中發行)。
- 修正涉及以屬性為基礎的搜尋中產生查詢計畫的錯誤。

Amazon Neptune 引擎更新 2019 年 7 月 26 日

版本：1.0.1.0.200366.0

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

這個引擎版本的新功能

- 已升級至 TinkerPop 3.4.1 (請參閱 TinkerPop 升級資訊以及 TinkerPop 3.4.1 變更日誌)。

  對於 Neptune 客戶，這些變更可提供新的功能和改善，例如：

  - `GraphBinary` 現在以序列化格式提供。
  - 已修正造成 TinkerPop Java 驅動程式中記憶體洩漏的保持有效錯誤，因此不再需要解決方法。

  不過，在少數情況下，它們可能會影響 Neptune 中現有的 Gremlin 程式碼。例如：

  - `valueMap()` 現在會傳回 `Map<Object,Object>` 而非 `Map<String,Object>`。
  - 已修正 `within()` 步驟的不一致行為，使其能與其他步驟一致運作。先前，類型必須相符，比較作業才能進行。現在，可以準確比較不同類型的數字。例如，`33` 現在會被比較為等於 `33L`，而之前並未這麼做。
  - 已修正 `ReducingBarrierStep` 中的錯誤，使得如果沒有可用於輸出的元素，現在不會傳回任何值。

- `select()` 範圍的順序已變更 (順序現在為 `maps`、`side-effects`、`paths`)。這會變更會將 `side-effects` 和 `select` 與具有相同金鑰名稱的 `side-effects` 的 `select` 結合的罕見查詢結果。
- `bulkSet()` 現在是 GraphSON 通訊協定的一部分。以 `toBulkSet()` 結尾的查詢不適用於較舊的用戶端。
- `Submit()` 步驟的一個參數化已從 3.4 用戶端移除。

TinkerPop 3.4 中引進的許多其他變更不會影響目前 Neptune 的行為。例如，Gremlin `io()` 已新增為 `Traversal` 的步驟，並且現在已在 `Graph` 中棄用，但從未在 Neptune 中啟用。

- 將單一基數頂點屬性支援新增到 Gremlin 的大量載入器，用於載入屬性圖資料。
- 新增可在大量載入器中覆寫單一基數屬性之現有值的選向。
- 新增擷取 Gremlin 查詢的狀態，以及取消 Gremlin 查詢的功能。
- 新增 SPARQL 查詢逾時的查詢提示。
- 新增在狀態 API 查看執行個體角色的能力 (請參閱 執行個體狀態)。
- 新增對資料庫複製的支援 (請參閱 Neptune 中的資料庫複製)。

這個引擎版本的改善項目

- 改進 SPARQL 查詢說明，從 FROM 子句顯示圖形變數。
- 改進篩選、等於篩選、VALUES 子句和範圍計數的 SPARQL 效能。
- 改進 Gremlin 步驟排序的效能。
- 改進 Gremlin `.repeat.dedup` 周遊的效能。
- 改善 Gremlin `valueMap()` 和 `path().by()` 周遊的效能。

此引擎版本中修正的缺陷

- 修正 SPARQL 屬性路徑的多個問題，包括具名圖形的操作。
- 修正造成記憶體問題的 SPARQL CONSTRUCT 查詢問題。
- 修正 RDF Turtle 剖析器與本機名稱的問題。
- 修正更正顯示給使用者之錯誤訊息的問題。
- 修正 Gremlin `repeat()...drop()` 周遊的問題。
- 修正 Gremlin `drop()` 步驟的問題。
- 修正 Gremlin 標籤篩選條件的問題。

- 修正 Gremlin 查詢逾時的問題。

Amazon Neptune 引擎更新 2019 年 7 月 2 日

重要：此引擎版本現在已棄用

從 2021 年 4 月 27 日開始，將不會建立任何使用此引擎版本的新執行個體。

此引擎版本中修正的缺陷

- 修正會導致具有屬性名稱和值的特定模式不會最佳化的錯誤。

早期 Neptune 引擎版本

主題

Amazon Neptune 引擎更新 2019 年 6 月 12 日

版本：1.0.1.0.200310.0

Amazon Neptune 1.0.1.0.200310.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200310.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
```

```
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200310.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 修正同時插入和捨棄邊緣可能導致多個邊緣具有產生相同 ID 的錯誤。
- 其他次要修正和改進。

Amazon Neptune 引擎更新 2019 年 5 月 1 日

版本：1.0.1.0.200296.0

Amazon Neptune 1.0.1.0.200296.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200296.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200296.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 已將新 `explain` 功能新增至 Neptune SPARQL 查詢，以協助您視覺化查詢計劃，並在必要時採取步驟來進行最佳化。如需相關資訊，請參閱SPARQL `explain`。
- 以各種方式改善 SPARQL 效能和報告。
- 以各種方式改善 Gremlin 效能和行為。
- 改善長時間執行的 `drop( )` 查詢逾時。
- 改善 `otherV( )` 查詢的效能。
- 已將兩個欄位新增至在您查詢資料庫叢集或執行個體的 Neptune 健康狀態時所傳回的資訊，也就是引擎版本號碼和叢集或執行個體的啟動時間。請參閱 執行個體狀態。
- Neptune 載入器 `Get-Status` API 現在會傳回 `startTime` 欄位，記錄載入工作啟動的時間。
- 載入器命令現在接受選用的 `parallelism` 參數，讓您可以限制載入器使用的執行緒數量。

Amazon Neptune 引擎更新 2019 年 1 月 21 日

版本：1.0.1.0.200267.0

Amazon Neptune 1.0.1.0.200267.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200267.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
```

```
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200267.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- Neptune 會等待更長的時間 (在指定的查詢逾時內)，讓任何衝突得以解決。這可減少用戶端需要處理之並行修改例外狀況的數目 (請參閱查詢錯誤)。
- 已修改 Gremlin 基數強制執行有時會造成引擎重新啟動的問題。
- 已提升 `emit.times` 重複查詢的 Gremlin 效能。
- 已修正 Gremlin 問題，其中 `repeat.until` 允許應該透過其進行篩選的 `.emit` 解決方案。
- 已改善 Gremlin 中的錯誤處理能力。

Amazon Neptune 引擎更新 2018 年 11 月 19 日

版本：1.0.1.0.200264.0

Amazon Neptune 1.0.1.0.200264.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200264.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
```

```
--resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200264.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 新增了對 the section called “查詢提示” 的支援。
- 改進 IAM 身分驗證的錯誤訊息。如需更多詳細資訊，請參閱 the section called “IAM 錯誤”。
- 提升包含大量述詞的 SPARQL 查詢效能。
- 提升 SPARQL 屬性路徑的效能。
- 提升於 `addV()`、`addE()`、`property()` 步驟所用的 Gremlin 條件式變動 (例如 `fold().coalesce(unfold(), …)` 樣式) 效能。
- 提升 Gremlin 的 `by()` 和 `sack()` 調變效能。
- 提升 Gremlin 的 `group()` 和 `groupCount()` 步驟的效能。
- 提升 Gremlin 的 `store()`、`sideEffect()`、`cap().unfold()` 步驟的效能。
- 改進對 Gremlin 單一基數屬性限制的支援。
  - 改進標示為單一基數屬性的邊緣屬性和頂點屬性的單一基數強制性。
  - 引進錯誤：如果在 Neptune 載入工作中為現有邊緣屬性指定額外的屬性值，會發生錯誤。

Amazon Neptune 引擎更新 2018 年 11 月 8 日

版本:1.0.1.0.200258.0

Amazon Neptune 1.0.1.0.200258.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200258.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200258.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 新增了對 SPARQL 查詢提示 的支援。
- 提升 SPARQL FILTER (NOT) Exists 查詢的效能。
- 提升 SPARQL DESCRIBE 查詢效能。
- 提升 Gremlin 中 repeat until 模式的效能。
- 提升在 Gremlin 中新增邊緣的效能。
- 修正 SPARQL Update DELETE 查詢在某些情況下可能會失敗的問題。
- 修正 Gremlin WebSocket 伺服器處理逾時的問題。

Amazon Neptune 引擎更新 2018 年 10 月 29 日

版本:1.0.1.0.200255.0

Amazon Neptune 1.0.1.0.200255.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200255.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200255.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 在稽核日誌中新增 IAM 身分驗證資訊。
- 新增對臨時登入資料的支援 (使用 IAM 角色和執行個體描述檔)。
- 在許可被撤銷或是 IAM 使用者或角色被刪除時，為 IAM 身分驗證新增 WebSocket 連線終止。
- 每一執行個體的 WebSocket 連線上限數為 60,000 個。
- 提升較小執行個體類型的大量載入效能。
- 提升 Gremlin 中包含 `and()`、`or()`、`not()`、`drop()` 運算子的查詢效能。
- NTriples 解析器現在會拒絕無效的 URI，例如包含空格的 URI。

Amazon Neptune 引擎更新 2018 年 9 月 6 日

版本：1.0.1.0.200237.0

Amazon Neptune 1.0.1.0.200237.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200237.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200237.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 已修正部分 SPARQL `COUNT(DISTINCT)` 查詢失敗的問題。
- 已修正具有 `DISTINCT` 子句的 `COUNT`、`SUM` 和 `MIN` 查詢可能會用盡記憶體的問題。
- 已修正 `BLOB` 類型的資料可能導致 Neptune 載入器任務失敗的問題。
- 已修正重複插入可能導致交易失敗的問題。
- 已修正無法將 `DROP ALL` 查詢取消的問題。
- 已修正 Gremlin 用戶端可能間歇性停止回應的問題。
- 已針對承載大於 150M 的情況將所有錯誤碼更新為 `HTTP 400`。
- 已改進單次三元組模式 `COUNT()` 查詢的效能與準確性。
- 已改進具有 `BIND` 子句的 SPARQL `UNION` 查詢的效能。

Amazon Neptune 引擎更新 2018 年 7 月 24 日

版本：1.0.1.0.200236.0

Amazon Neptune 1.0.1.0.200236.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200236.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200236.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 已更新 `xsd:string` 資料類型的 SPARQL 序列化。xsd:string 不再隨附於 JSON 序列化，目前已與其他輸出格式一致。
- 已修正 `xsd:double`/`xsd:float` 無限的處理問題。`-INF`、`NaN` 和 `INF` 的值現已可在所有 SPARQL 資料載入器格式、SPARQL 1.1 更新及 SPARQL 1.1 查詢中正確加以辨識和處理。
- 包含空字串值的 Gremlin 查詢非預期失敗問題已獲修正。
- 在空圖表上的 Gremlin `aggregate()`和 `cap()` 非預期失敗問題已獲解決。
- 基數規格無效時 (例如 `.property(set,id,'10')`和 `.property(single,id,'10')`)，Gremlin 回傳不正確錯誤回應的問題已獲解決。
- 無效的 Gremlin 語法回傳為 InternalFailureException 的問題已獲解決。
- 將錯誤訊息中 `TimeLimitExceeededException`的拼法修正為 `TimeLimitExceededException`。

- 變更 SPARQL 和 GREMLIN 終端節點，在無指令碼時以一致方式回應。
- 釐清並行請求過多的錯誤訊息。

Amazon Neptune 引擎更新 2018 年 6 月 22 日

版本：1.0.1.0.200233.0

Amazon Neptune 1.0.1.0.200233.0 正式推出。在該區域的引擎更新完成之後，所有新的 Neptune 資料庫叢集 (包括從快照還原的資料庫叢集) 將會在 Neptune 1.0.1.0.200233.0 中建立。

您可以在主控台上使用資料庫叢集操作或使用軟體開發套件，立即將現有叢集升級至此版本。您可以使用以下 CLI 命令，立即將資料庫叢集升級至此版本：

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

在系統維護時段期間，Neptune 資料庫叢集會自動升級至引擎版本 1.0.1.0.200233.0。套用更新的時間取決於資料庫叢集的區域和維護時段設定，以及更新類型。

> **Note**
>
> 執行個體維護時段無法進行引擎更新。

更新會同時套用到資料庫叢集中的所有執行個體。更新時，資料庫叢集中的所有執行個體需要重新啟動資料庫，因此您會經歷 20 至 30 秒的停機時間，之後就可以繼續使用資料庫叢集或叢集。您可以在 Neptune 主控台上檢視或變更您的維護時段設定。

如果您有任何問題或疑慮，請透過社群論壇及 AWS Premium Support 聯繫 AWS Support 團隊。

改善項目

- 修正快速連續發出大量大批載入請求會導致錯誤的問題。
- 修正查詢失敗並出現 InternalServerError 的資料相關問題。以下範例顯示受到影響的查詢類型。

```
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")
```

- 修正在長時間執行的查詢逾時之後，Gremlin Java 用戶端無法使用相同 WebSocket 連線連接到伺服器的問題。
- 修正以下問題：包含在透過 HTTP 的 Gremlin 查詢或透過 WebSocket 連線的字串查詢中的逸出序列無法正確處理。

Amazon Neptune API 的使用簡介

Amazon Neptune 管理 API 會提供 SDK 支援，用於建立、管理和刪除 Neptune 資料庫叢集和執行個體，而 Neptune 資料 API 則會提供 SDK 支援，用於將資料載入圖形、執行查詢、取得圖形中資料的相關資訊，以及執行機器學習操作。這些 API 在所有 SDK 語言中皆可使用。透過自動簽署 API 請求，它們大幅簡化了將 Neptune 整合至應用程式的操作。

本頁面提供如何使用這些 API 的詳細資訊。

與其 Neptune 資料 API SDK 對應項目不同名稱的 IAM 動作

當您在已啟用 IAM 身分驗證的叢集上呼叫 Neptune API 方法時，您必須將 IAM 政策附加到進行呼叫的使用者或角色，此政策為您想要進行的動作提供許可。您可以使用對應的 IAM 動作，在政策中設定這些許可。您也可以使用 IAM 條件金鑰限制可以採取的動作。

大多數 IAM 動作與它們對應的 API 方法具有相同的名稱，但是資料 API 中的某些方法具有不同的名稱，因為有些名稱是由多個方法共用。下表列出資料方法及其對應的 IAM 動作：

資料 API 操作名稱	IAM 對應關係
CancelGremlinQuery (cancel_gremlin_query)	動作：neptune-db: **CancelQuery**
CancelLoaderJob (cancel_loader_job)	動作：neptune-db:CancelLoaderJob
CancelMLDataProcessingJob (cancel_ml_data_processing_job)	動作：neptune-db:CancelMLDataProcessingJob
CancelMLModelTrainingJob (cancel_ml_model_training_job)	動作：neptune-db:CancelMLModelTrainingJob
CancelOpenCypherQuery (cancel_open_cypher_query)	動作：neptune-db: **CancelQuery**

資料 API 操作名稱	IAM 對應關係	
CreateMLEndpoint (create_ml_endpoint)	動作：neptune-db:CreateMLEndpoint	
DeleteMLEndpoint (delete_ml_endpoint)	動作：neptune-db:DeleteMLEndpoint	
DeletePropertygraphStatistics (delete_propertygraph_statistics)	動作：neptune-db: **DeleteStatistics**	
DeleteSparqlStatistics (delete_sparql_statistics)	動作：neptune-db: **DeleteStatistics**	
ExecuteFastReset execute_fast_reset()	動作：neptune-db: **ResetDatabase**	
ExecuteGremlinExplainQuery (execute_gremlin_explain_query)	動作: • neptune-db: **ReadDataViaQuery** • neptune-db: **WriteDataViaQuery** • neptune-db: **DeleteDataViaQuery** 條件金鑰：neptune-db:QueryLanguage:Gremlin	
ExecuteGremlinProfileQuery (execute_gremlin_profile_query)	動作：neptune-db: **ReadDataViaQuery** 條件金鑰：neptune-db:QueryLanguage:Gremlin	

資料 API 操作名稱	IAM 對應關係	
ExecuteGremlinQuery (execute_gremlin_query)	動作: • neptune-db: **ReadDataViaQuery** • neptune-db: **WriteDataViaQuery** • neptune-db: **DeleteDataViaQuery** 條件金鑰：neptune-db:QueryLanguage:Gremlin	
ExecuteOpenCypherExplainQuery (execute_open_cypher_explain_query)	動作：neptune-db: **ReadDataViaQuery** 條件金鑰：neptune-db:QueryLanguage:OpenCypher	
ExecuteOpenCypherQuery (execute_open_cypher_query)	動作: • neptune-db: **ReadDataViaQuery** • neptune-db: **WriteDataViaQuery** • neptune-db: **DeleteDataViaQuery** 條件金鑰：neptune-db:QueryLanguage:OpenCypher	

資料 API 操作名稱	IAM 對應關係	
GetEngineStatus (get_engine_status)	動作：neptune-db:GetEngineStatus	
GetGremlinQueryStatus (get_gremlin_query_status)	動作：neptune-db: **:GetQueryStatus** 條件金鑰：neptune-db:QueryLanguage:Gremlin	
GetLoaderJobStatus (get_loader_job_status)	動作：neptune-db:GetLoaderJobStatus	
GetMLDataProcessingJob (get_ml_data_processing_job)	動作：neptune-db: **GetMLDataProcessingJobStatus**	
GetMLEndpoint (get_ml_endpoint)	動作：neptune-db: **GetMLEndpointStatus**	
GetMLModelTrainingJob (get_ml_model_training_job)	動作：neptune-db: **GetMLModelTrainingJobStatus**	
GetMLModelTransformJob (get_ml_model_transform_job)	動作：neptune-db: **GetMLModelTransformJobStatus**	
GetOpenCypherQueryStatus (get_open_cypher_query_status)	動作：neptune-db: **:GetQueryStatus** 條件金鑰：neptune-db:QueryLanguage:OpenCypher	

資料 API 操作名稱	IAM 對應關係	
GetPropertygraphStatistics (get_propertygraph_statistics)	動作：neptune-db: **GetStatisticsStatus**	
GetPropertygraphStream (get_propertygraph_stream)	動作：neptune-db: **GetStreamRecords** 條件金鑰： • neptune-db:QueryLanguage:Gremlin • neptune-db:QueryLanguage:OpenCypher	
GetPropertygraphSummary (get_propertygraph_summary)	動作：neptune-db: **GetGraphSummary**	
GetRDFGraphSummary (get_rdf_graph_summary)	動作：neptune-db: **GetGraphSummary**	
GetSparqlStatistics (get_sparql_statistics)	動作：neptune-db: **GetStatisticsStatus**	
GetSparqlStream (get_sparql_stream)	動作：neptune-db: **:GetStreamRecords** 條件金鑰：neptune-db:QueryLanguage:Sparql	
ListGremlinQueries (list_gremlin_queries)	動作：neptune-db: **:GetQueryStatus** 條件金鑰：neptune-db:QueryLanguage:Gremlin	

資料 API 操作名稱	IAM 對應關係	
ListMLEndpoints (list_ml_endpoints)	動作：neptune-db:ListMLEndpoints	
ListMLModelTrainingJobs (list_ml_model_training_jobs)	動作：neptune-db:ListMLModelTrainingJobs	
ListMLModelTransformJobs (list_ml_model_transform_jobs)	動作：neptune-db:ListMLModelTransformJobs	
ListOpenCypherQueries (list_open_cypher_queries)	動作：neptune-db: **:GetQueryStatus** 條件金鑰：neptune-db:QueryLanguage:OpenCypher	
ManagePropertygraphStatistics (manage_propertygraph_statistics)	動作：neptune-db: **ManageStatistics**	
ManageSparqlStatistics (manage_sparql_statistics)	動作：neptune-db: **ManageStatistics**	
StartLoaderJob (start_loader_job)	動作：neptune-db:StartLoaderJob	
StartMLModelDataProcessingJob (start_ml_data_processing_job)	動作：neptune-db:StartMLModelDataProcessingJob	
StartMLModelTrainingJob (start_ml_model_training_job)	動作：neptune-db:StartMLModelTrainingJob	

資料 API 操作名稱	IAM 對應關係	
StartMLModelTransformJob (start_ml_model_transform_job)	動作：neptune-db:StartMLModelTransformJob	

Amazon Neptune 管理 API 參考

本章記載您可以用來管理和維護您 Neptune 資料庫叢集的 Neptune API 操作。

Neptune 會在複寫拓撲中連線的資料庫伺服器叢集上運作。因此，管理 Neptune 通常會涉及將變更部署到多部伺服器，並確保所有 Neptune 複本都和主伺服器保持同步。

由於 Neptune 會隨著您的資料增加而透明地擴展基礎儲存體，因此管理 Neptune 只需要進行相對較少的磁碟儲存體管理。同樣地，因為 Neptune 會自動執行連續備份，因此 Neptune 叢集不需要為執行備份進行大量的規劃或停機時間。

內容

- ModifyGlobalCluster (動作)
- DescribeGlobalClusters (動作)
- FailoverGlobalCluster (動作)
- RemoveFromGlobalCluster (動作)
- 結構：
- GlobalCluster (結構)
- GlobalClusterMember (結構)

- Neptune 執行個體 API
  - CreateDBInstance (動作)
  - DeleteDBInstance (動作)
  - ModifyDBInstance (動作)
  - RebootDBInstance (動作)
  - DescribeDBInstances (動作)
  - DescribeOrderableDBInstanceOptions (動作)
  - DescribeValidDBInstanceModifications (動作)
  - 結構：
  - DBInstance (結構)
  - DBInstanceStatusInfo (結構)
  - OrderableDBInstanceOption (結構)
  - PendingModifiedValues (結構)
  - ValidStorageOptions (結構)
  - ValidDBInstanceModificationsMessage (結構)
- Neptune 參數 API
  - CopyDBParameterGroup (動作)
  - CopyDBClusterParameterGroup (動作)
  - CreateDBParameterGroup (動作)
  - CreateDBClusterParameterGroup (動作)
  - DeleteDBParameterGroup (動作)
  - DeleteDBClusterParameterGroup (動作)
  - ModifyDBParameterGroup (動作)

- ModifyDBClusterParameterGroup (動作)
- ResetDBParameterGroup (動作)
- ResetDBClusterParameterGroup (動作)
- DescribeDBParameters (動作)
- DescribeDBParameterGroups (動作)
- DescribeDBClusterParameters (動作)
- DescribeDBClusterParameterGroups (動作)
- DescribeEngineDefaultParameters (動作)
- DescribeEngineDefaultClusterParameters (動作)
- 結構：
- Parameter (結構)
- DBParameterGroup (結構)
- DBClusterParameterGroup (結構)
- DBParameterGroupStatus (結構)

- Neptune 子網路 API
  - CreateDBSubnetGroup (動作)
  - DeleteDBSubnetGroup (動作)
  - ModifyDBSubnetGroup (動作)
  - DescribeDBSubnetGroups (動作)
  - 結構：
  - Subnet (結構)
  - DBSubnetGroup (結構)
- Neptune 快照 API
  - CreateDBClusterSnapshot (動作)
  - DeleteDBClusterSnapshot (動作)
  - CopyDBClusterSnapshot (動作)
  - ModifyDBClusterSnapshotAttribute (動作)
  - RestoreDBClusterFromSnapshot (動作)
  - RestoreDBClusterToPointInTime (動作)
  - DescribeDBClusterSnapshots (動作)

- DescribeDBClusterSnapshotAttributes (動作)
- 結構：
- DBClusterSnapshot (結構)
- DBClusterSnapshotAttribute (結構)
- DBClusterSnapshotAttributesResult (結構)

- Neptune 事件 API
  - CreateEventSubscription (動作)
  - DeleteEventSubscription (動作)
  - ModifyEventSubscription (動作)
  - DescribeEventSubscriptions (動作)
  - AddSourceIdentifierToSubscription (動作)
  - RemoveSourceIdentifierFromSubscription (動作)
  - DescribeEvents (動作)
  - DescribeEventCategories (動作)
  - 結構：
  - Event (結構)
  - EventCategoriesMap (結構)
  - EventSubscription (結構)
- 其他 Neptune API
  - AddTagsToResource (動作)
  - ListTagsForResource (動作)
  - RemoveTagsFromResource (動作)
  - ApplyPendingMaintenanceAction (動作)
  - DescribePendingMaintenanceActions (動作)
  - DescribeDBEngineVersions (動作)
  - 結構：
  - DBEngineVersion (結構)
  - EngineDefaults (結構)
  - PendingMaintenanceAction (結構)
  - ResourcePendingMaintenanceActions (結構)

- DBLogFileNotFoundFault (結構)
- DBParameterGroupAlreadyExistsFault (結構)
- DBParameterGroupNotFoundFault (結構)
- DBParameterGroupQuotaExceededFault (結構)
- DBSecurityGroupAlreadyExistsFault (結構)
- DBSecurityGroupNotFoundFault (結構)
- DBSecurityGroupNotSupportedFault (結構)
- DBSecurityGroupQuotaExceededFault (結構)
- DBSnapshotAlreadyExistsFault (結構)
- DBSnapshotNotFoundFault (結構)
- DBSubnetGroupAlreadyExistsFault (結構)
- DBSubnetGroupDoesNotCoverEnoughAZs (結構)
- DBSubnetGroupNotAllowedFault (結構)
- DBSubnetGroupNotFoundFault (結構)
- DBSubnetGroupQuotaExceededFault (結構)
- DBSubnetQuotaExceededFault (結構)
- DBUpgradeDependencyFailureFault (結構)
- DomainNotFoundFault (結構)
- EventSubscriptionQuotaExceededFault (結構)
- GlobalClusterAlreadyExistsFault (結構)
- GlobalClusterNotFoundFault (結構)
- GlobalClusterQuotaExceededFault (結構)
- InstanceQuotaExceededFault (結構)
- InsufficientDBClusterCapacityFault (結構)
- InsufficientDBInstanceCapacityFault (結構)
- InsufficientStorageClusterCapacityFault (結構)
- InvalidDBClusterEndpointStateFault (結構)
- InvalidDBClusterSnapshotStateFault (結構)
- InvalidDBClusterStateFault (結構)
- InvalidDBInstanceStateFault (結構)

- InvalidDBParameterGroupStateFault (結構)
- InvalidDBSecurityGroupStateFault (結構)
- InvalidDBSnapshotStateFault (結構)
- InvalidDBSubnetGroupFault (結構)
- InvalidDBSubnetGroupStateFault (結構)
- InvalidDBSubnetStateFault (結構)
- InvalidEventSubscriptionStateFault (結構)
- InvalidGlobalClusterStateFault (結構)
- InvalidOptionGroupStateFault (結構)
- InvalidRestoreFault (結構)
- InvalidSubnet (結構)
- InvalidVPCNetworkStateFault (結構)
- KMSKeyNotAccessibleFault (結構)
- OptionGroupNotFoundFault (結構)
- PointInTimeRestoreNotEnabledFault (結構)
- ProvisionedIopsNotAvailableInAZFault (結構)
- ResourceNotFoundFault (結構)
- SNSInvalidTopicFault (結構)
- SNSNoAuthorizationFault (結構)
- SNSTopicArnNotFoundFault (結構)
- SharedSnapshotQuotaExceededFault (結構)
- SnapshotQuotaExceededFault (結構)
- SourceNotFoundFault (結構)
- StorageQuotaExceededFault (結構)
- StorageTypeNotSupportedFault (結構)
- SubnetAlreadyInUse (結構)
- SubscriptionAlreadyExistFault (結構)
- SubscriptionCategoryNotFoundFault (結構)
- SubscriptionNotFoundFault (結構)

Neptune 資料庫叢集 API

動作:

- CreateDBCluster (動作)
- DeleteDBCluster (動作)
- ModifyDBCluster (動作)
- StartDBCluster (動作)
- StopDBCluster (動作)
- AddRoleToDBCluster (動作)
- RemoveRoleFromDBCluster (動作)
- FailoverDBCluster (動作)
- PromoteReadReplicaDBCluster (動作)
- DescribeDBClusters (動作)

結構：

- DBCluster (結構)
- DBClusterMember (結構)
- DBClusterRole (結構)
- CloudwatchLogsExportConfiguration (結構)
- PendingCloudwatchLogsExports (結構)
- ClusterPendingModifiedValues (結構)

CreateDBCluster (動作)

> 此 API 的 AWS CLI 名稱是：`create-db-cluster`。

建立新的 Amazon Neptune 資料庫叢集。

您可以使用 `ReplicationSourceIdentifier` 參數來建立資料庫叢集，做為另一個資料庫叢集或 Amazon Neptune 資料庫執行個體的僅供讀取複本。

請注意，當您直接使用 CreateDBCluster 建立新叢集時，預設會停用刪除保護 (當您在主控台中建立新的生產叢集時，預設會啟用刪除保護)。如果資料庫叢集的 DeletionProtection 欄位設定為 false，您只能刪除資料庫叢集。

請求

- AvailabilityZones (在 CLI 中：--availability-zones) – 字串，類型為：string (UTF-8 編碼的字串)。

  EC2 可用區域的清單，在這些可用區域中可以建立資料庫叢集中的執行個體。

- BackupRetentionPeriod (在 CLI 中：--backup-retention-period) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  自動備份保留的天數。您必須指定一個值 (最小值為 1)。

  預設：1

  約束:

  - 該值必須介於 1 到 35 之間

- CopyTagsToSnapshot (在 CLI 中:--copy-tags-to-snapshot) – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  如果設定為 *true*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- DatabaseName (在 CLI 中：--database-name) – 字串，類型為：string (UTF-8 編碼的字串)。

  您資料庫的名稱最長可達 64 個英數字元。若您沒有提供名稱，Amazon Neptune 將不會在您建立的資料庫叢集中建立資料庫。

- DBClusterIdentifier (在 CLI 中：--db-cluster-identifier) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集識別碼 此參數是以小寫字母字串的形式儲存。

  約束:

  - 必須包含 1 到 63 個字母、數字或連字號。
  - 第一個字元必須是字母。
  - 不能以連字號結尾或連續包含兩個連字號。

  範例：my-cluster1

- DBClusterParameterGroupName (在 CLI 中：`--db-cluster-parameter-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要與此資料庫叢集建立關聯的資料庫叢集參數群組名稱。若省略此引數，則會使用預設。

  約束:

  - 若有提供，則必須符合現有 DBClusterParameterGroup 的名稱。
- DBSubnetGroupName (在 CLI 中：`--db-subnet-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要與此資料庫叢集建立關聯的資料庫子網路群組。

  限制條件：必須符合現有 DBSubnetGroup 的名稱。絕不能為預設值。

  範例：`mySubnetgroup`
- DeletionProtection (在 CLI 中:`--deletion-protection`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護的值。啟用刪除保護時無法刪除資料庫。根據預設，刪除保護是啟用的。
- EnableCloudwatchLogsExports (在 CLI 中：`--enable-cloudwatch-logs-exports`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集應該匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來發佈稽核日誌) 和 `slowquery` (用來發佈慢查詢日誌)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。
- EnableIAMDatabaseAuthentication (在 CLI 中:`--enable-iam-database-authentication`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則會針對整個資料庫叢集啟用 Amazon Identity and Access Management (IAM) 身分驗證 (這無法在執行個體層級設定)。

  預設：`false`。
- Engine (在 CLI 中：`--engine`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要用於此資料庫叢集的資料庫引擎名稱。

  有效值:`neptune`
- EngineVersion (在 CLI 中：`--engine-version`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

要用於新資料庫叢集的資料庫引擎版本號碼。

範例：1.2.1.0

- GlobalClusterIdentifier (在 CLI 中：--global-cluster-identifier) – GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  應將此新資料庫叢集新增至其中之 Neptune 全球資料庫的 ID。

- KmsKeyId (在 CLI 中：--kms-key-id) – 字串，類型為：string (UTF-8 編碼的字串)。

  已加密資料庫叢集的 Amazon KMS 金鑰識別符。

  KMS 金鑰識別符是 KMS 加密金鑰的 Amazon Resource Name (ARN)。如果您使用擁有用來加密新資料庫叢集 KMS 加密金鑰的相同 Amazon 帳戶建立資料庫叢集，您可以改為使用 KMS 金鑰別名，而非 KMS 加密金鑰的 ARN。

  若加密金鑰並未在 KmsKeyId 中指定：

  - 若 ReplicationSourceIdentifier 可識別加密來源，則 Amazon Neptune 會使用用於加密來源的加密金鑰。否則，Amazon Neptune 將使用您的預設加密金鑰。
  - 若 StorageEncrypted 參數為 true 且並未指定 ReplicationSourceIdentifier，則 Amazon Neptune 會使用您的預設加密金鑰。

  Amazon KMS 會為您的 Amazon 帳戶建立預設加密金鑰。您的 Amazon 帳戶在每個 Amazon 區域各有不同的預設加密金鑰。

  如果您在另一個 Amazon 區域內建立加密資料庫叢集的僅供讀取複本，您必須將 KmsKeyId 設為在目標 Amazon 區域中有效的 KMS 金鑰 ID。此金鑰會用於加密該 Amazon 區域內的僅供讀取複本。

- Port (在 CLI 中：--port) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  資料庫叢集中執行個體接受連線的連接埠號碼。

  預設：8182

- PreferredBackupWindow (在 CLI 中：--preferred-backup-window) – 字串，類型為：string (UTF-8 編碼的字串)。

  每日的時間範圍，若使用 BackupRetentionPeriod 參數啟用自動化備份，則會在此期間建立自動化備份。

預設是從每個 Amazon 區域 8 小時的時段內隨機選取的 30 分鐘時段。若要查看可用的時段，請參閱《Amazon Neptune 使用者指南》中的 Neptune 維護時段。

約束:

- 格式必須為 hh24:mi-hh24:mi。
- 必須以國際標準時間 (UTC) 表示。
- 不得和慣用的維護時段衝突。
- 必須至少 30 分鐘。

- PreferredMaintenanceWindow (在 CLI 中：--preferred-maintenance-window) – 字串，類型為：string (UTF-8 編碼的字串)。

  每週可能進行系統維護的時段，以國際標準時間 (UTC) 表示。

  格式：ddd:hh24:mi-ddd:hh24:mi

  預設是從各 Amazon 區域 8 小時時段中隨機選取的 30 分鐘時段，並隨機發生在一週內的某一天。若要查看可用的時段，請參閱《Amazon Neptune 使用者指南》中的 Neptune 維護時段。

  有效日：星期一、星期二、星期三、星期四、星期五、星期六、星期日。

  限制條件：必須至少是 30 分鐘的時段。

- PreSignedUrl (在 CLI 中：--pre-signed-url) – 字串，類型為：string (UTF-8 編碼的字串)。

  目前不支援此參數。

- ReplicationSourceIdentifier (在 CLI 中：--replication-source-identifier) – 字串，類型為：string (UTF-8 編碼的字串)。

  如果此資料庫叢集已做為僅供讀取複本建立，則為來源資料庫執行個體或資料庫叢集的 Amazon Resource Name (ARN)。

- ServerlessV2ScalingConfiguration (在 CLI 中：--serverless-v2-scaling-configuration) – ServerlessV2ScalingConfiguration 物件。

  包含 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- StorageEncrypted (在 CLI 中:--storage-encrypted) – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

指定是否要加密資料庫叢集。

- StorageType (在 CLI 中：`--storage-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新 DB 叢集的儲存類型。

  有效值：

  - **`standard`** – (預設值) 針對具有中等至少量 I/O 使用率的應用程式設定具成本效益的資料庫儲存體。設定為 `standard`時，回應中不會傳回儲存類型。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。
- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給新資料庫叢集的標籤。
- VpcSecurityGroupIds (在 CLI 中：`--vpc-security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與此資料庫叢集關聯的 EC2 VPC 安全群組清單。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。
- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。
- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。

- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：string (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：boolean (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 iopt1 儲存類型。

- KmsKeyId – 字串，類型為：string (UTF-8 編碼的字串)。

  若 StorageEncrypted 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 ModifyDBCluster 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：string (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：string (UTF-8 編碼的字串)。

指定每日的時間範圍，若有啟用自動化備份 (由 BackupRetentionPeriod 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：string (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

  若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：string (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：string (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

- **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
- **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

  僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。
- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterAlreadyExistsFault
- InsufficientStorageClusterCapacityFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBClusterStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- InvalidDBInstanceStateFault
- DBClusterParameterGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterNotFoundFault
- DBInstanceNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DeleteDBCluster (動作)

此 API 的 AWS CLI 名稱是：`delete-db-cluster`。

DeleteDBCluster 動作會刪除先前佈建的資料庫叢集。當您刪除資料庫叢集時，所有該資料庫叢集的自動化備份都會遭到刪除，並且無法復原。指定資料庫叢集的手動資料庫叢集快照則不會刪除。

請注意，如果刪除保護已經啟用，資料庫叢集將無法刪除。如果要刪除，您必須先將其 `DeletionProtection` 欄位設定為 `False`。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  欲刪除資料庫叢集的資料庫叢集識別符。此參數沒有大小寫之分。

  約束:

  - 必須符合現有的 DBClusterIdentifier。
- FinalDBSnapshotIdentifier (在 CLI 中：`--final-db-snapshot-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  當 `SkipFinalSnapshot` 設為 `false` 時，所建立新資料庫叢集快照的資料庫叢集快照識別符。

  > **Note**
  >
  > 指定此參數，同時又將 `SkipFinalShapshot` 參數設為 true 時，便會導致錯誤。

  約束:

  - 必須為 1 到 255 個字母、數字或連字號。
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號
- SkipFinalSnapshot (在 CLI 中:`--skip-final-snapshot`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  判斷最終資料庫叢集快照是否建立於資料庫叢集刪除之前。若指定 `true`，則不會建立資料庫叢集快照。若指定 `false`，則資料庫叢集快照會在刪除資料庫叢集前建立。

> **Note**
>
> 若 SkipFinalSnapshot 是 false，則您必須指定 FinalDBSnapshotIdentifier 參數。

預設：false

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  AllocatedStorage 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。
- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。
- AutomaticRestartTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。
- AvailabilityZones – 字串，類型為：string (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。
- BacktrackConsumedChangeRecords - LongOptional，類型為：long (帶正負號的 64 位元整數)。

  Neptune 不提供支援。
- BacktrackWindow - LongOptional，類型為：long (帶正負號的 64 位元整數)。

  Neptune 不提供支援。
- BackupRetentionPeriod – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 `iopt1` 儲存類型。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 `ModifyDBCluster` 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

  - **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- DBClusterSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

ModifyDBCluster (動作)

此 API 的 AWS CLI 名稱是：`modify-db-cluster`。

修改資料庫叢集的設定。您可以透過在請求中指定這些參數及新的值，來變更一或多個資料庫組態參數。

請求

- AllowMajorVersionUpgrade (在 CLI 中:`--allow-major-version-upgrade`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  一值，指示是否允許不同主要版本之間升級。

  限制條件：在提供 `EngineVersion` 參數，使用與資料庫叢集目前版本不同的主要版本時，您必須設定 allow-major-version-upgrade 旗標。
- ApplyImmediately (在 CLI 中:`--apply-immediately`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定此請求中的修改及任何擱置中的請求是否要以非同步方式盡快套用的值，無論資料庫叢集的 `PreferredMaintenanceWindow` 設定為何。若將此參數設為 `false`，對資料庫叢集的變更會在下次維護時段期間才會套用。

  `ApplyImmediately` 參數只會影響 `NewDBClusterIdentifier` 值。如果您將 `ApplyImmediately` 參數值設定為 false，則對 `NewDBClusterIdentifier` 值的變更會在下一個維護時段期間套用。所有其他的變更都會立即套用，無論 `ApplyImmediately` 參數的值為何。

  預設：`false`
- BackupRetentionPeriod (在 CLI 中：`--backup-retention-period`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

自動備份保留的天數。您必須指定一個值 (最小值為 1)。

預設：1

約束:

- 該值必須介於 1 到 35 之間

- CloudwatchLogsExportConfiguration (在 CLI 中：`--cloudwatch-logs-export-configuration`) – CloudwatchLogsExportConfiguration 物件。

  要啟用的日誌類型的組態設定，適用於匯出到特定資料庫叢集的 CloudWatch Logs。請參閱使用 CLI 將 Neptune 稽核日誌發佈至 CloudWatch Logs。

- CopyTagsToSnapshot (在 CLI 中:`--copy-tags-to-snapshot`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  欲修改叢集的資料庫叢集識別符。此參數不區分大小寫。

  約束:

  - 必須符合現有 DBCluster 的識別碼。

- DBClusterParameterGroupName (在 CLI 中：`--db-cluster-parameter-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集欲使用的資料庫叢集參數群組名稱。

- DBInstanceParameterGroupName (在 CLI 中：`--db-instance-parameter-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要套用至所有資料庫叢集執行個體的資料庫參數群組名稱。

> **Note**
>
> 當您使用 `DBInstanceParameterGroupName` 套用參數群組時，參數變更不會在下一個維護時段期間套用，而是會立即套用。

預設：現有名稱設定

約束:

- 資料庫參數群組必須位於與目標資料庫叢集版本相同的資料庫參數群組系列中。
- `DBInstanceParameterGroupName` 參數只在與 `AllowMajorVersionUpgrade` 參數結合時才有效。

- DeletionProtection (在 CLI 中:`--deletion-protection`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護的值。啟用刪除保護時無法刪除資料庫。根據預設，刪除保護是停用的。

- EnableIAMDatabaseAuthentication (在 CLI 中:`--enable-iam-database-authentication`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  設為 True 以啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的對應，否則為 false。

  預設：`false`

- EngineVersion (在 CLI 中：`--engine-version`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  您要升級的資料庫引擎版本號碼。變更此參數會造成中斷。變更會在下一次維護時段期間套用，除非 `ApplyImmediately` 參數已設為 true。

  如需有效引擎版本的清單，請參閱 Amazon Neptune 的引擎版本，或呼叫 the section called “DescribeDBEngineVersions”。

- NewDBClusterIdentifier (在 CLI 中：`--new-db-cluster-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  重新命名資料庫叢集時的新資料庫叢集識別符。此值會以小寫字母字串的形式儲存。

  約束:

  - 必須包含 1 到 63 個字母、數字或連字號
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號

  範例：`my-cluster2`

- Port (在 CLI 中：`--port`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  資料庫叢集接受連線連接埠號碼。

  限制條件：值必須為 `1150-65535`

  預設值：與原始資料庫叢集相同的連接埠。

- PreferredBackupWindow (在 CLI 中：`--preferred-backup-window`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  每日的時間範圍，若使用 `BackupRetentionPeriod` 參數啟用自動化備份，則會在此期間建立自動化備份。

  預設是從每個 Amazon 區域 8 小時的時段內隨機選取的 30 分鐘時段。

  約束:

  - 格式必須為 `hh24:mi-hh24:mi`。
  - 必須以國際標準時間 (UTC) 表示。
  - 不得和慣用的維護時段衝突。
  - 必須至少 30 分鐘。

- PreferredMaintenanceWindow (在 CLI 中：`--preferred-maintenance-window`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  每週可能進行系統維護的時段，以國際標準時間 (UTC) 表示。

  格式：`ddd:hh24:mi-ddd:hh24:mi`

  預設是從各 Amazon 區域 8 小時時段中隨機選取的 30 分鐘時段，並隨機發生在一週內的某一天。

  有效日：星期一、星期二、星期三、星期四、星期五、星期六、星期日。

  限制條件：必須至少是 30 分鐘的時段。

- ServerlessV2ScalingConfiguration (在 CLI 中：`--serverless-v2-scaling-configuration`) – ServerlessV2ScalingConfiguration 物件。

  包含 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- StorageType (在 CLI 中：`--storage-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

和資料庫叢集相關聯的儲存類型。

有效值：

- **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式設定具成本效益的資料庫儲存體。
- **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

  僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

- VpcSecurityGroupIds (在 CLI 中：`--vpc-security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集將歸屬的 VPC 安全群組清單。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。

- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。

- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。

- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

Neptune 不提供支援。

- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 `iopt1` 儲存類型。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 `ModifyDBCluster` 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

  - **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- InvalidVPCNetworkStateFault
- InvalidDBSubnetGroupStateFault
- InvalidSubnet
- DBClusterParameterGroupNotFoundFault
- InvalidDBSecurityGroupStateFault
- InvalidDBInstanceStateFault
- DBClusterAlreadyExistsFault
- StorageTypeNotSupportedFault

StartDBCluster (動作)

此 API 的 AWS CLI 名稱是：`start-db-cluster`。

透過使用 Amazon 主控台、Amazon CLI stop-db-cluster 命令或 StopDBCluster API，來啟動停止的 Amazon Neptune 資料庫叢集。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要啟動的 Neptune 資料庫叢集的資料庫叢集識別符。此參數是以小寫字母字串的形式儲存。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。

- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。

- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。

- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：audit (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：string (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：string (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：string (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：boolean (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 iopt1 儲存類型。

- KmsKeyId – 字串，類型為：string (UTF-8 編碼的字串)。

  若 StorageEncrypted 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 ModifyDBCluster 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：string (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：string (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 BackupRetentionPeriod 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：string (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

  若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：string (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

  - **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

StopDBCluster (動作)

此 API 的 AWS CLI 名稱是：`stop-db-cluster`。

停止 Amazon Neptune 資料庫叢集。當您停止資料庫叢集時，Neptune 會留資料庫叢集的中繼資料，包括端點和資料庫參數群組。

Neptune 也會保留交易日誌，以便您視需要進行時間點還原。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

要停止的 Neptune 資料庫叢集的資料庫叢集識別符。此參數是以小寫字母字串的形式儲存。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。
- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。
- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。
- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。
- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。
- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。
- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。
- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。
- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。
- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。
- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。
- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。
- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。
- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。
- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。
- DBClusterParameterGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。
- DbClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。
- DBSubnetGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。
- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。
- EarliestBacktrackTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 `iopt1` 儲存類型。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 `ModifyDBCluster` 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

  若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

  - **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

AddRoleToDBCluster (動作)

此 API 的 AWS CLI 名稱是：`add-role-to-db-cluster`。

將 Identity and Access Management (IAM) 角色與 Neptune 資料庫叢集建立關聯。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要與 IAM 角色建立關聯的資料庫叢集名稱。
- FeatureName (在 CLI 中：`--feature-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色與其相關聯之 Neptune 資料庫叢集的功能名稱。如需支援的功能名稱清單，請參閱 the section called “DBEngineVersion”。
- RoleArn (在 CLI 中：`--role-arn`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要與 Neptune 資料庫叢集建立關聯的 IAM 角色 Amazon Resource Name (ARN)，例如 `arn:aws:iam::123456789012:role/NeptuneAccessRole`。

回應

- 無回應參數。

錯誤

- DBClusterNotFoundFault
- DBClusterRoleAlreadyExistsFault
- InvalidDBClusterStateFault
- DBClusterRoleQuotaExceededFault

RemoveRoleFromDBCluster (動作)

此 API 的 AWS CLI 名稱是：`remove-role-from-db-cluster`。

從資料庫叢集解除關聯一個 Identity and Access Management (IAM) 角色。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

要從 IAM 角色解除關聯的資料庫叢集名稱。

- FeatureName (在 CLI 中：`--feature-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色要與其取消關聯之資料庫叢集的功能名稱。如需支援的功能名稱清單，請參閱 the section called “DescribeDBEngineVersions”。
- RoleArn (在 CLI 中：`--role-arn`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要從資料庫叢集解除關聯的 IAM 角色 Amazon Resource Name (ARN)，例如 `arn:aws:iam::123456789012:role/NeptuneAccessRole`。

回應

- 無回應參數。

錯誤

- DBClusterNotFoundFault
- DBClusterRoleNotFoundFault
- InvalidDBClusterStateFault

FailoverDBCluster (動作)

此 API 的 AWS CLI 名稱是：`failover-db-cluster`。

強制資料庫叢集進行容錯移轉。

資料庫叢集的容錯移轉會將資料庫叢集中的其中一個僅供讀取複本 (唯讀執行個體) 提升為主要執行個體 (叢集寫入器)。

Amazon Neptune 會在主要執行個體失敗時，自動容錯移轉至僅供讀取複本 (若有的話)。當您想要模擬主要執行個體故障以進行測試時，可以強制容錯移轉。由於資料庫叢集中的每個執行個體都有自己的端點地址，當容錯移轉完成時，您將需要清除和重新建立任何使用這些端點地址的現有連線。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

要強制其進行容錯移轉的資料庫叢集識別符。此參數不區分大小寫。

約束:

- 必須符合現有 DBCluster 的識別碼。

- TargetDBInstanceIdentifier (在 CLI 中：`--target-db-instance-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要提升為主要執行個體的執行個體名稱。

  您必須為資料庫叢集中的僅供讀取複本指定執行個體識別符。例如：`mydbcluster-replica1`。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。

- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。

- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。

- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 `iopt1` 儲存類型。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 `ModifyDBCluster` 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

  - **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBInstanceStateFault

PromoteReadReplicaDBCluster (動作)

此 API 的 AWS CLI 名稱是：`promote-read-replica-db-cluster`。

不支援。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。
- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。
- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

資料庫叢集將自動重新啟動的時間。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。

- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 `iopt1` 儲存類型。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 `ModifyDBCluster` 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

指定每日的時間範圍，若有啟用自動化備份 (由 BackupRetentionPeriod 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：string (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

  若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：string (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：string (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

- **standard** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
- **iopt1** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

  僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。
- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterNotFoundFault
- InvalidDBClusterStateFault

DescribeDBClusters (動作)

此 API 的 AWS CLI 名稱是：`describe-db-clusters`。

傳回所佈建之資料庫叢集的相關資訊，並支援分頁。

> **Note**
>
> 此操作也可以傳回 Amazon RDS 叢集和 Amazon DocDB 叢集的資訊。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  使用者提供的資料庫叢集識別符。若指定此參數，則只會傳回特定資料庫叢集的資訊。此參數沒有大小寫之分。

  約束:
  - 若有提供此項目，則必須符合現有的 DBClusterIdentifier。
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  篩選條件，指定要描述的一或多個資料庫叢集。

支援的篩選條件：

- db-cluster-id - 接受資料庫叢集識別符及資料庫叢集 Amazon Resource Name (ARN)。結果清單只會包含由這些 ARN 識別的資料庫叢集相關資訊。
- engine -接受引擎名稱 (例如 neptune)，並將結果清單限制為由該引擎建立的資料庫叢集。

例如，若要從 Amazon CLI 調用此 API 並進行篩選，以便只傳回 Neptune 資料庫叢集，您可以使用下列命令：

Example

```
aws neptune describe-db-clusters \
            --filters  Name=engine,Values=neptune
```

- Marker (在 CLI 中：--marker) – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個 the section called “DescribeDBClusters” 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。

- MaxRecords (在 CLI 中：--max-records) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 MaxRecords 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

回應

- DBClusters – 一個 DBCluster 物件陣列。

  包含使用者的資料庫叢集清單。

- Marker – 字串，類型為：string (UTF-8 編碼的字串)。

  可用於後續 DescribeDBClusters 請求的分頁字符。

錯誤

- DBClusterNotFoundFault

結構：

DBCluster (結構)

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

欄位

- AllocatedStorage – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。
- AssociatedRoles - 這是 DBClusterRole 物件的陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。
- AutomaticRestartTime – 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。
- AvailabilityZones - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。
- BacktrackConsumedChangeRecords - 這是 LongOptional 整數，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。
- BacktrackWindow - 這是 LongOptional 整數，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。
- BackupRetentionPeriod – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – 這是 BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – 這是 BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers - 這是 DBClusterMember 物件的陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – 這是 BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：`audit` (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier - 這是 GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 `iopt1` 儲存類型。

- KmsKeyId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues - 這是一個 ClusterPendingModifiedValues 物件。

  此資料類型會在 `ModifyDBCluster` 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration - 這是一個 ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

  - **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

- VpcSecurityGroups - 這是 VpcSecurityGroupMembership 物件的陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

`DBCluster` 會用來做為以下項目的回應元素：

- CreateDBCluster
- DeleteDBCluster
- FailoverDBCluster
- ModifyDBCluster
- PromoteReadReplicaDBCluster
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- StartDBCluster
- StopDBCluster

DBClusterMember (結構)

包含做為資料庫叢集一部分的執行個體相關資訊。

欄位

- DBClusterParameterGroupStatus - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集成員的資料庫叢集參數群組狀態。
- DBInstanceIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集成員的執行個體識別符。
- IsClusterWriter - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  若叢集成員是資料庫叢集的主要執行個體，則該值為 `true`，否則為 `false`。
- PromotionTier – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  此值指定在現有主要執行個體故障後，僅供讀取複本提升為主要執行個體的順序。

DBClusterRole (結構)

描述與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色。

欄位

- FeatureName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

與 Amazon Identity and Access Management (IAM) 角色關聯的功能名稱。如需支援的功能名稱清單，請參閱 the section called “DescribeDBEngineVersions”。

- RoleArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  與資料庫叢集相關聯的 IAM 角色 Amazon Resource Name (ARN)。

- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  描述 IAM 角色和資料庫叢集之間關聯的狀態。Status 屬性會傳回下列其中一個值：

  - `ACTIVE` - 與資料庫叢集相關聯的 IAM 角色 ARN，可用於代您存取其他 Amazon 服務。
  - `PENDING` - IAM 角色 ARN 正在和資料庫叢集建立關聯。
  - `INVALID` - 與資料庫叢集相關聯的 IAM 角色 ARN，但資料庫叢集無法取得 IAM 角色來代您存取其他 Amazon 服務。

CloudwatchLogsExportConfiguration (結構)

要啟用的日誌類型的組態設定，適用於匯出到特定資料庫執行個體或資料庫叢集的 CloudWatch Logs。

`EnableLogTypes` 和 `DisableLogTypes` 陣列決定要將哪些日誌匯出 (或不匯出) 到 CloudWatch Logs。

有效的日誌類型為：`audit` (用來發佈稽核日誌) 和 `slowquery` (用來發佈慢查詢日誌)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

欄位

- DisableLogTypes - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  要停用的日誌類型清單。

- EnableLogTypes - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  要啟用的日誌類型清單。

PendingCloudwatchLogsExports (結構)

組態仍待處理的日誌類型清單。換言之，這些日誌類型正在進行啟用或停用中。

有效的日誌類型為：audit (用來發佈稽核日誌) 和 slowquery (用來發佈慢查詢日誌)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

欄位

- LogTypesToDisable - 這是字串，類型為：string (UTF-8 編碼的字串)。

  正在進行啟用的日誌類型。日誌之後，這些日誌類型會匯出到 CloudWatch Logs。
- LogTypesToEnable - 這是字串，類型為：string (UTF-8 編碼的字串)。

  正在進行停用的日誌類型。停用日誌之後，這些日誌類型不會匯出到 CloudWatch Logs。

ClusterPendingModifiedValues (結構)

此資料類型會在 ModifyDBCluster 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

欄位

- AllocatedStorage – 這是 IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  為資料庫引擎所配置的儲存體大小，以 GiB 為單位。對於 Neptune，AllocatedStorage 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。
- BackupRetentionPeriod – 這是 IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  自動資料庫快照保留的天數。
- DBClusterIdentifier - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的 DBClusterIdentifier 值。
- EngineVersion - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫引擎版本。
- IAMDatabaseAuthenticationEnabled – 這是 BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  一值，指示是否啟用將 AWS Identity and Access Management (IAM) 帳戶對應到資料庫帳戶。
- Iops – 這是 IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  佈建 IOPS (每秒 I/O 操作) 值。此設定僅適用於多可用區域資料庫叢集。

- PendingCloudwatchLogsExports - 這是一個 PendingCloudwatchLogsExports 物件。

  此 `PendingCloudwatchLogsExports` 結構會指定 CloudWatch Logs 啟用和停用的待定變更。

- StorageType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的儲存類型的待處理變更。 有效值：

  - **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式設定具成本效益的資料庫儲存體。
  - **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

    僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。

Neptune 全球資料庫 API

動作:

- CreateGlobalCluster (動作)
- DeleteGlobalCluster (動作)
- ModifyGlobalCluster (動作)
- DescribeGlobalClusters (動作)
- FailoverGlobalCluster (動作)
- RemoveFromGlobalCluster (動作)

結構：

- GlobalCluster (結構)
- GlobalClusterMember (結構)

CreateGlobalCluster (動作)

此 API 的 AWS CLI 名稱是：`create-global-cluster`。

建立分散在多個 Amazon 區域的 Neptune 全球資料庫。全球資料庫包含具有讀寫功能的單一主要叢集，以及唯讀次要叢集，可透過 Neptune 儲存子系統執行的高速複寫接收來自主要叢集的資料。

您可以建立最初空白的全球資料庫，然後將主要叢集和次要叢集新增至其中，也可以在建立操作期間指定現有的 Neptune 叢集，使其成為全球資料庫的主要叢集。

請求

- DatabaseName (在 CLI 中：`--database-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新全球資料庫的名稱 (最多可有 64 個英數字元)。
- DeletionProtection (在 CLI 中:`--deletion-protection`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  新的全域資料庫的刪除保護設定。啟用刪除保護時無法刪除全域資料庫。
- Engine (在 CLI 中：`--engine`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要在全球資料庫中使用的資料庫引擎名稱。

  有效值：`neptune`
- EngineVersion (在 CLI 中：`--engine-version`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫要使用的 Neptune 引擎版本。

  有效值為：`1.2.0.0` 或以上。
- GlobalClusterIdentifier (在 CLI 中：`--global-cluster-identifier`) – 必要：GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  新全域資料庫叢集的叢集識別碼。
- SourceDBClusterIdentifier (在 CLI 中：`--source-db-cluster-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  (選用) 現有 Neptune 資料庫叢集的 Amazon Resource Name (ARN)，此資料庫叢集要用作新全球資料庫的主要叢集。
- StorageEncrypted (在 CLI 中:`--storage-encrypted`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  新全球資料庫叢集的儲存加密設定。

回應

包含 Amazon Neptune 全球資料庫的詳細資訊。

此資料類型會用作 the section called “CreateGlobalCluster”、the section called “DescribeGlobalClusters”、the section called “ModifyGlobalCluster”、the section called “DeleteGlobalCluster”、the section called “FailoverGlobalCluster” 和 the section called “RemoveFromGlobalCluster” 的回應元素。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的刪除保護設定。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 資料庫引擎 (`"neptune"`)。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 引擎版本。

- GlobalClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的 Amazon Resource Name (ARN)。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- GlobalClusterMembers – 一個 GlobalClusterMember 物件陣列。

  屬於全球資料庫之所有資料庫叢集的叢集 ARN 和執行個體 ARN 清單。

- GlobalClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的不可變識別符，在所有區域內都是唯一的。此識別碼可在每次存取資料庫叢集的 KMS 金鑰時，於 CloudTrail 日誌項目中找到。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此全球資料庫的目前狀態。

- StorageEncrypted – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的儲存加密設定。

錯誤

- GlobalClusterAlreadyExistsFault
- GlobalClusterQuotaExceededFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault

DeleteGlobalCluster (動作)

此 API 的 AWS CLI 名稱是：`delete-global-cluster`。

刪除全球資料庫。必須先分離或刪除主要叢集和所有次要叢集。

請求

- GlobalClusterIdentifier (在 CLI 中：`--global-cluster-identifier`) – 必要：GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  要刪除之全球資料庫叢集的叢集識別符。

回應

包含 Amazon Neptune 全球資料庫的詳細資訊。

此資料類型會用作 the section called “CreateGlobalCluster”、the section called “DescribeGlobalClusters”、the section called “ModifyGlobalCluster”、the section called “DeleteGlobalCluster”、the section called “FailoverGlobalCluster” 和 the section called “RemoveFromGlobalCluster” 的回應元素。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的刪除保護設定。
- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 資料庫引擎 (`"neptune"`)。
- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 引擎版本。

- GlobalClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的 Amazon Resource Name (ARN)。
- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。
- GlobalClusterMembers – 一個 GlobalClusterMember 物件陣列。

  屬於全球資料庫之所有資料庫叢集的叢集 ARN 和執行個體 ARN 清單。
- GlobalClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的不可變識別符，在所有區域內都是唯一的。此識別碼可在每次存取資料庫叢集的 KMS 金鑰時，於 CloudTrail 日誌項目中找到。
- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此全球資料庫的目前狀態。
- StorageEncrypted – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的儲存加密設定。

錯誤

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

ModifyGlobalCluster (動作)

此 API 的 AWS CLI 名稱是：`modify-global-cluster`。

修改 Amazon Neptune 全球叢集的設定。您可以透過在請求中指定這些參數及其新值，來變更一或多個資料庫組態參數。

請求

- AllowMajorVersionUpgrade (在 CLI 中:`--allow-major-version-upgrade`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示是否允許升級主要版本的值。

限制條件：如果您為主要版本與資料庫執行個體目前版本不同的 EngineVersion 參數指定一值，則必須允許主要版本升級。

如果您升級全球資料庫的主要版本，叢集和資料庫執行個體參數群組會設定為新版本的預設參數群組，因此您必須在完成升級之後套用任何自訂參數群組。

- DeletionProtection (在 CLI 中:--deletion-protection) – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  指示全球資料庫是否已啟用刪除保護。啟用刪除保護時無法刪除全球資料庫。

- EngineVersion (在 CLI 中：--engine-version) – 字串，類型為：string (UTF-8 編碼的字串)。

  您要升級的資料庫引擎版本號碼。變更此參數將會造成中斷。變更會在下一次維護時段期間套用，除非已啟用 ApplyImmediately。

  若要列出所有可用的 Neptune 引擎版本，請使用以下命令：

  Example

  ```
  aws neptune describe-db-engine-versions \
            --engine neptune \
            --query '*[]|[?SupportsGlobalDatabases == 'true'].[EngineVersion]'
  ```

- GlobalClusterIdentifier (在 CLI 中：--global-cluster-identifier) – 必要：GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  要修改之全球叢集的資料庫叢集識別符。此參數不區分大小寫。

  限制條件：必須符合現有全球資料庫叢集的識別符。

- NewGlobalClusterIdentifier (在 CLI 中：--new-global-cluster-identifier) – GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  要指派給全球資料庫的新叢集識別符。此值會以小寫字母字串的形式儲存。

  約束:

  - 必須包含 1 到 63 個字母、數字或連字號。
  - 第一個字元必須是字母。

- 不能以一個連字號結尾或是連續包含兩個連字號

範例：`my-cluster2`

回應

包含 Amazon Neptune 全球資料庫的詳細資訊。

此資料類型會用作 the section called “CreateGlobalCluster”、the section called “DescribeGlobalClusters”、the section called “ModifyGlobalCluster”、the section called “DeleteGlobalCluster”、the section called “FailoverGlobalCluster” 和 the section called “RemoveFromGlobalCluster” 的回應元素。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的刪除保護設定。
- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 資料庫引擎 (`"neptune"`)。
- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 引擎版本。
- GlobalClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的 Amazon Resource Name (ARN)。
- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。
- GlobalClusterMembers – 一個 GlobalClusterMember 物件陣列。

  屬於全球資料庫之所有資料庫叢集的叢集 ARN 和執行個體 ARN 清單。
- GlobalClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的不可變識別符，在所有區域內都是唯一的。此識別碼可在每次存取資料庫叢集的 KMS 金鑰時，於 CloudTrail 日誌項目中找到。
- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此全球資料庫的目前狀態。

- StorageEncrypted – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  全球資料庫的儲存加密設定。

錯誤

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DescribeGlobalClusters (動作)

此 API 的 AWS CLI 名稱是：describe-global-clusters。

傳回 Neptune 全球資料庫叢集的相關資訊。此 API 支援分頁。

請求

- GlobalClusterIdentifier (在 CLI 中：--global-cluster-identifier) – GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  使用者提供的資料庫叢集識別符。如果指定此參數，則只會傳回所指定資料庫叢集的相關資訊。此參數不區分大小寫。

  限制條件：如果提供，必須符合現有的資料庫叢集的識別符。
- Marker (在 CLI 中：--marker) – 字串，類型為：string (UTF-8 編碼的字串)。

  (選用) 上次呼叫 DescribeGlobalClusters 所傳回的分頁權杖。如果指定此參數，則回應只會包含標記以外的記錄，直到 MaxRecords 指定的數字為止。
- MaxRecords (在 CLI 中：--max-records) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。如果存在的記錄比 MaxRecords 值指定的更多，則分頁標記權杖會包含在回應中，您可以用來擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

回應

- GlobalClusters – 一個 GlobalCluster 物件陣列。

  此請求傳回的全球叢集和執行個體清單。
- Marker – 字串，類型為：string (UTF-8 編碼的字串)。

  分頁權杖。如果在回應中傳回此參數，則有更多記錄可用，這些記錄可以透過一個或多個對 DescribeGlobalClusters 的額外呼叫來擷取 。

錯誤

- GlobalClusterNotFoundFault

FailoverGlobalCluster (動作)

此 API 的 AWS CLI 名稱是：failover-global-cluster。

啟動 Neptune 全球資料庫的容錯移轉程序。

Neptune 全球資料庫的容錯移轉會將其中一個次要唯讀資料庫叢集提升為主要資料庫叢集，並將主要資料庫叢集降級為次要 (唯讀) 資料庫叢集。換句話說，會切換目前主要資料庫叢集和所選目標次要資料庫叢集的角色。選取的次要資料庫叢集會採用 Neptune 全球資料庫的完整讀取/寫入功能。

> **Note**
>
> 此動作僅適用於 Neptune 全球資料庫。此動作旨在用於狀態良好的 Neptune DB 叢集，搭配狀態良好的 Neptune 全球資料庫，且沒有全區域中斷，以測試災難復原案例或重新設定全球資料庫拓撲。

請求

- GlobalClusterIdentifier (在 CLI 中：--global-cluster-identifier) – 必要：GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  應容錯移轉的 Neptune 全球資料庫的識別符。識別符是在建立 Neptune 全球資料庫時，由使用者指派的唯一金鑰。換句話說，它是您要容錯移轉的全球資料庫名稱。

限制條件：必須符合現有 Neptune 全球資料庫的識別符。

- TargetDbClusterIdentifier (在 CLI 中：`--target-db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  次要 Neptune 資料庫叢集的 Amazon Resource Name (ARN)，您想要將此資料庫叢集提升為全球資料庫的主要資料庫叢集。

回應

包含 Amazon Neptune 全球資料庫的詳細資訊。

此資料類型會用作 the section called “CreateGlobalCluster”、the section called “DescribeGlobalClusters”、the section called “ModifyGlobalCluster”、the section called “DeleteGlobalCluster”、the section called “FailoverGlobalCluster” 和 the section called “RemoveFromGlobalCluster” 的回應元素。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的刪除保護設定。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 資料庫引擎 (`"neptune"`)。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 引擎版本。

- GlobalClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的 Amazon Resource Name (ARN)。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- GlobalClusterMembers – 一個 GlobalClusterMember 物件陣列。

  屬於全球資料庫之所有資料庫叢集的叢集 ARN 和執行個體 ARN 清單。

- GlobalClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

全球資料庫的不可變識別符，在所有區域內都是唯一的。此識別碼可在每次存取資料庫叢集的 KMS 金鑰時，於 CloudTrail 日誌項目中找到。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此全球資料庫的目前狀態。

- StorageEncrypted – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的儲存加密設定。

錯誤

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault

RemoveFromGlobalCluster (動作)

> 此 API 的 AWS CLI 名稱是：`remove-from-global-cluster`。

從 Neptune 全球資料庫分離 Neptune 資料庫叢集。次要叢集會變成具有讀寫功能而非唯讀功能的一般獨立叢集，且不再從主要叢集接收資料。

請求

- DbClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  識別要從 Neptune 全球資料庫叢集中分離的 Amazon Resource Name (ARN)。

- GlobalClusterIdentifier (在 CLI 中：`--global-cluster-identifier`) – 必要：GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  要從中分離所指定 Neptune 資料庫叢集的 Neptune 全球資料庫的識別符。

回應

包含 Amazon Neptune 全球資料庫的詳細資訊。

此資料類型會用作 the section called “CreateGlobalCluster”、the section called “DescribeGlobalClusters”、the section called “ModifyGlobalCluster”、the section called “DeleteGlobalCluster”、the section called “FailoverGlobalCluster” 和 the section called “RemoveFromGlobalCluster” 的回應元素。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的刪除保護設定。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 資料庫引擎 (`"neptune"`)。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 引擎版本。

- GlobalClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的 Amazon Resource Name (ARN)。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- GlobalClusterMembers – 一個 GlobalClusterMember 物件陣列。

  屬於全球資料庫之所有資料庫叢集的叢集 ARN 和執行個體 ARN 清單。

- GlobalClusterResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的不可變識別符，在所有區域內都是唯一的。此識別碼可在每次存取資料庫叢集的 KMS 金鑰時，於 CloudTrail 日誌項目中找到。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此全球資料庫的目前狀態。

- StorageEncrypted – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的儲存加密設定。

錯誤

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- DBClusterNotFoundFault

結構：

GlobalCluster (結構)

包含 Amazon Neptune 全球資料庫的詳細資訊。

此資料類型會用作 the section called “CreateGlobalCluster”、the section called “DescribeGlobalClusters”、the section called “ModifyGlobalCluster”、the section called “DeleteGlobalCluster”、the section called “FailoverGlobalCluster” 和 the section called “RemoveFromGlobalCluster” 的回應元素。

欄位

- DeletionProtection – 這是 BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的刪除保護設定。
- Engine - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 資料庫引擎 (`"neptune"`)。
- EngineVersion - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫所使用的 Neptune 引擎版本。
- GlobalClusterArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的 Amazon Resource Name (ARN)。
- GlobalClusterIdentifier - 這是 GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。
- GlobalClusterMembers - 這是 GlobalClusterMember 物件的陣列。

  屬於全球資料庫之所有資料庫叢集的叢集 ARN 和執行個體 ARN 清單。

- GlobalClusterResourceId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  全球資料庫的不可變識別符，在所有區域內都是唯一的。此識別碼可在每次存取資料庫叢集的 KMS 金鑰時，於 CloudTrail 日誌項目中找到。

- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此全球資料庫的目前狀態。

- StorageEncrypted – 這是 BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  全球資料庫的儲存加密設定。

`GlobalCluster` 會用來做為以下項目的回應元素：

- CreateGlobalCluster
- ModifyGlobalCluster
- DeleteGlobalCluster
- RemoveFromGlobalCluster
- FailoverGlobalCluster

GlobalClusterMember (結構)

一種資料結構，其中包含與 Neptune 全球資料庫相關聯之任何主要和次要叢集的相關資訊。

欄位

- DBClusterArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  每個 Neptune 叢集的 Amazon Resource Name (ARN)。

- IsWriter - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定 Neptune 叢集是否為與其相關聯之 Neptune 全球資料庫的主要叢集 (亦即，具有讀寫功能)。

- Readers - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  與 Neptune 全球資料庫相關聯的每個唯讀次要叢集的 Amazon Resource Name (ARN)。

Neptune 執行個體 API

動作:

- CreateDBInstance (動作)
- DeleteDBInstance (動作)
- ModifyDBInstance (動作)
- RebootDBInstance (動作)
- DescribeDBInstances (動作)
- DescribeOrderableDBInstanceOptions (動作)
- DescribeValidDBInstanceModifications (動作)

結構：

- DBInstance (結構)
- DBInstanceStatusInfo (結構)
- OrderableDBInstanceOption (結構)
- PendingModifiedValues (結構)
- ValidStorageOptions (結構)
- ValidDBInstanceModificationsMessage (結構)

CreateDBInstance (動作)

此 API 的 AWS CLI 名稱是：`create-db-instance`。

建立新的資料庫執行個體。

請求

- AutoMinorVersionUpgrade (在 CLI 中:`--auto-minor-version-upgrade`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指出會在維護時段期間將次要引擎升級自動套用至資料庫執行個體。

  預設：`true`

- AvailabilityZone (在 CLI 中：`--availability-zone`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  在其中建立資料庫執行個體的 EC2 可用區域。

  預設值：端點 Amazon 區域中系統隨機選擇的可用區域。

  範例：`us-east-1d`

  限制條件：若 MultiAZ 參數設為 `true`，則無法指定 AvailabilityZone 參數。指定的可用區域和目前端點必須位於同一 Amazon 區域。

- BackupRetentionPeriod (在 CLI 中：`--backup-retention-period`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  自動備份保留的天數。

  不適用。自動化備份的保留期會由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called "CreateDBCluster"。

  預設：1

  約束:

  - 該值必須介於 0 到 35 之間
  - 若資料庫執行個體是僅供讀取複本的來源，則無法設為 0

- CopyTagsToSnapshot (在 CLI 中:`--copy-tags-to-snapshot`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  若為 True，則會從資料庫執行個體將所有標籤複製到資料庫執行個體的快照，否則為 False。預設值為 false。

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  執行個體所屬資料庫叢集的識別碼。

  如需建立資料庫叢集的詳細資訊，請參閱the section called "CreateDBCluster"。

  類型：字串

- DBInstanceClass (在 CLI 中：`--db-instance-class`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫執行個體的運算與記憶體容量，例如 db.m4.large。並非所有 Amazon 區域皆提供所有資料庫執行個體類別。

- DBInstanceIdentifier (在 CLI 中：--db-instance-identifier) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  資料庫執行個體識別符。此參數是以小寫字母字串的形式儲存。

  約束:

  - 必須包含 1 到 63 個字母、數字或連字號。
  - 第一個字元必須是字母。
  - 不能以連字號結尾或連續包含兩個連字號。

  範例：mydbinstance

- DBName (在 CLI 中：--db-name) – 字串，類型為：string (UTF-8 編碼的字串)。

  不支援。

- DBParameterGroupName (在 CLI 中：--db-parameter-group-name) – 字串，類型為：string (UTF-8 編碼的字串)。

  要與此資料庫執行個體建立關聯的資料庫參數群組名稱。若忽略此引數，則會使用指定引擎的預設 DBParameterGroup。

  約束:

  - 必須為 1 到 255 個字母、數字或連字號。
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號

- DBSecurityGroups (在 CLI 中：--db-security-groups) – 字串，類型為：string (UTF-8 編碼的字串)。

  與此資料庫執行個體關聯的資料庫安全群組清單。

  預設值：資料庫引擎的預設資料庫安全群組。

- DBSubnetGroupName (在 CLI 中：--db-subnet-group-name) – 字串，類型為：string (UTF-8 編碼的字串)。

  要與此資料庫執行個體建立關聯的資料庫子網路群組。

如果沒有資料庫子網路群組，則它是非VPC DB執行個體。

- DeletionProtection (在 CLI 中:`--deletion-protection`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫執行個體是否啟用刪除保護的值。啟用刪除保護時無法刪除資料庫。根據預設，刪除保護是停用的。請參閱刪除資料庫執行個體。

  即使刪除保護已在上層資料庫叢集中啟用，資料庫叢集中的資料庫執行個體仍可刪除。

- Domain (在 CLI 中：`--domain`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定要在其中建立執行個體的 Active Directory 網域。

- DomainIAMRoleName (在 CLI 中：`--domain-iam-role-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定對 Directory Service 發出 API 呼叫時要使用的 IAM 角色名稱。

- EnableCloudwatchLogsExports (在 CLI 中：`--enable-cloudwatch-logs-exports`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  需要啟用，以匯出至 CloudWatch Logs 的日誌類型清單。

- EnableIAMDatabaseAuthentication (在 CLI 中:`--enable-iam-database-authentication`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  Neptune 不提供支援 (已忽略)。

- Engine (在 CLI 中：`--engine`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要為此執行個體使用的資料庫引擎名稱。

  有效值:`neptune`

- EngineVersion (在 CLI 中：`--engine-version`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要使用的資料庫引擎版本號碼。目前，設定此參數沒有任何作用。

- Iops (在 CLI 中：`--iops`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  初始配置給資料庫執行個體的佈建IOPS數量 (每秒輸入/輸出作業)。

- KmsKeyId (在 CLI 中：`--kms-key-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  加密資料庫執行個體的 Amazon KMS 金鑰識別符。

KMS 金鑰識別符是 KMS 加密金鑰的 Amazon Resource Name (ARN)。如果您用來建立資料庫執行個體的 Amazon 帳戶，與擁有用於加密新資料庫執行個體 KMS 加密金鑰的 Amazon 帳戶相同，則您可以使用 KMS 金鑰別名，而非 KM 加密金鑰的 ARN。

不適用。KMS 金鑰識別碼會由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called "CreateDBCluster"。

若 `StorageEncrypted` 參數為 True，並且您沒有為 `KmsKeyId` 參數指定值，則 Amazon Neptune 會使用您的預設加密金鑰。Amazon KMS 會為您的 Amazon 帳戶建立預設加密金鑰。您的 Amazon 帳戶在每個 Amazon 區域各有不同的預設加密金鑰。

- LicenseModel (在 CLI 中：`--license-model`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的授權模型資訊。

  有效值：`license-included` | `bring-your-own-license` | `general-public-license`

- MonitoringInterval (在 CLI 中：`--monitoring-interval`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  為資料庫執行個體收集增強型監控指標點之間的間隔 (秒)。若要停用收集增強型監控指標，請指定 0。預設值為 0。

  若指定 `MonitoringRoleArn`，您必須也將 `MonitoringInterval` 設為 0 之外的值。

  有效值:`0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (在 CLI 中：`--monitoring-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，允許 Neptune 將增強型監控指標傳送到 Amazon CloudWatch Logs。例如：`arn:aws:iam:123456789012:role/emaccess`。

  若 `MonitoringInterval` 設為 0 之外的值，則您必須提供 `MonitoringRoleArn` 值。

- MultiAZ (在 CLI 中:`--multi-az`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫執行個體是否為異地同步備份部署。若 MultiAZ 參數設為 True，您便無法設定 AvailabilityZone 參數。

- Port (在 CLI 中：`--port`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  資料庫接受連線的連接埠號碼。

不適用。連接埠會由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called "CreateDBCluster"。

預設：8182

類型：整數

- PreferredBackupWindow (在 CLI 中：`--preferred-backup-window`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  每日時間範圍，自動化備份會在這段期間建立。

  不適用。建立自動化備份的每日時間範圍會由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called "CreateDBCluster"。
- PreferredMaintenanceWindow (在 CLI 中：`--preferred-maintenance-window`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  每週可能進行系統維護的時間範圍，以國際標準時間 (UTC) 表示。

  格式：`ddd:hh24:mi-ddd:hh24:mi`

  預設是從各 Amazon 區域 8 小時時段中隨機選取的 30 分鐘時段，並隨機發生在一週內的某一天。

  有效日：星期一、星期二、星期三、星期四、星期五、星期六、星期日。

  限制條件：必須至少是 30 分鐘的時段。
- PromotionTier (在 CLI 中：`--promotion-tier`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定在現有主要執行個體失敗後，將僅供讀取複本提升為主要執行個體順序的值。

  預設：1

  有效值：0 到 15
- PubliclyAccessible (在 CLI 中:`--publicly-accessible`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  此標記不應繼續使用。
- StorageEncrypted (在 CLI 中:`--storage-encrypted`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

指定是否加密資料庫執行個體。

不適用。資料庫執行個體的加密會由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called “CreateDBCluster”。

預設：false

- StorageType (在 CLI 中：`--storage-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給新執行個體的標籤。

- TdeCredentialArn (在 CLI 中：`--tde-credential-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  來自金鑰存放區的 ARN，執行個體會與此 ARN 建立關聯以進行 TDE 加密。

- TdeCredentialPassword (在 CLI 中：`--tde-credential-password`) – SensitiveString，類型為：`string` (UTF-8 編碼的字串)。

  在金鑰存放區指定 ARN 的密碼，用於存取裝置。

- Timezone (在 CLI 中：`--timezone`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的時區。

- VpcSecurityGroupIds (在 CLI 中：`--vpc-security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與此資料庫執行個體關聯的 EC2 VPC 安全群組清單。

  不適用。資料庫叢集會管理 EC2 VPC 安全群組的關聯清單。如需更多詳細資訊，請參閱 the section called “CreateDBCluster”。

  預設值：資料庫子網路群組 VPC 的預設 EC2 VPC 安全群組。

回應

包含 Amazon Neptune 資料庫執行個體的詳細資訊。

此資料類型在 the section called “DescribeDBInstances” 動作中會用來作為回應元素。

- AutoMinorVersionUpgrade – 布林值，類型為：boolean (布林值 (true 或 false))。

  指示會自動套用次要版本的修補程式。

- AvailabilityZone – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫執行個體所在的可用區域名稱。

- BackupRetentionPeriod – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- CACertificateIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  此資料庫執行個體的憑證授權機構憑證識別碼。

- CopyTagsToSnapshot – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定是否要從資料庫執行個體將標籤複製到資料庫執行個體的快照。

- DBClusterIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  若資料庫執行個體是資料庫叢集的成員，則包含資料庫執行個體所屬資料庫叢集的名稱。

- DBInstanceArn – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon Resource Name (ARN)。

- DBInstanceClass – 字串，類型為：string (UTF-8 編碼的字串)。

  包含資料庫執行個體之運算和記憶體容量類別的名稱。

- DBInstanceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  包含使用者提供的資料庫識別碼。此識別碼是可識別資料庫執行個體的唯一金鑰。

- DbInstancePort – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫執行個體接聽的連接埠。若資料庫執行個體是資料庫叢集的一部分，這可以是與資料庫叢集連接埠不同的連接埠。

- DBInstanceStatus – 字串，類型為：string (UTF-8 編碼的字串)。

  指定此資料庫的目前狀態。

- DbiResourceId – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon 區域唯一、固定識別符。每當存取資料庫執行個體的 Amazon KMS 金鑰時，就會在 Amazon CloudTrail 日誌項目中找到此識別符。

- DBName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫名稱。

- DBParameterGroups – 一個 DBParameterGroupStatus 物件陣列。

  提供要套用到此資料庫執行個體的資料庫參數群組清單。

- DBSecurityGroups – 一個 DBSecurityGroupMembership 物件陣列。

  提供資料庫安全群組元素的清單，元素中只包含 `DBSecurityGroup.Name` 和 `DBSecurityGroup.Status` 子元素。

- DBSubnetGroup – DBSubnetGroup 物件。

  指定與資料庫執行個體相關聯子網路群組上的資訊，包含名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否已啟用刪除保護。如果刪除保護已經啟用，執行個體將無法刪除。請參閱刪除資料庫執行個體。

- DomainMemberships – 一個 DomainMembership 物件陣列。

  不支援

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體已設為匯出至 CloudWatch Logs 的日誌類型清單。

- Endpoint – 端點 物件。

  指定連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供用於此資料庫執行個體的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- EnhancedMonitoringResourceArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Amazon CloudWatch Logs 日誌串流的 Amazon Resource Name (ARN)，該串流會接收資料庫執行個體的增強型監控指標。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

若 Amazon Identity and Access Management (IAM) 身分驗證已啟用，則為 True，否則為 False。

- InstanceCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供建立資料庫執行個體的日期和時間。

- Iops – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定佈建 IOPS (每秒 I/O 操作) 值。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- LicenseModel – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的授權模型資訊。

- MonitoringInterval – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  為資料庫執行個體收集增強型監控指標點之間的間隔 (秒)。

- MonitoringRoleArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，允許 Neptune 將增強型監控指標傳送到 Amazon CloudWatch Logs。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫執行個體是否為異地同步備份部署。

- PendingModifiedValues – PendingModifiedValues 物件。

  指定對資料庫執行個體所進行的變更仍在擱置中。此元素只有在變更擱置中時才會包含在其中。特定的變更會由子元素識別。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- PromotionTier – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  此值指定在現有主要執行個體故障後，僅供讀取複本提升為主要執行個體的順序。

- PubliclyAccessible – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  此標記不應繼續使用。

- ReadReplicaDBClusterIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個資料庫叢集的識別符，這些叢集是此資料庫執行個體的僅供讀取複本。

- ReadReplicaDBInstanceIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個與此資料庫執行個體相關聯僅供讀取複本的識別符。

- ReadReplicaSourceDBInstanceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含來源資料庫執行個體的識別符 (若此資料庫執行個體是僅供讀取複本)。

- SecondaryAvailabilityZone – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若存在，指定具備多個可用區支援之資料庫執行個體的次要可用區名稱。

- StatusInfos – 一個 DBInstanceStatusInfo 物件陣列。

  僅供讀取複本的狀態。若執行個體並非僅供讀取複本，此處即為空白。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。

- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與此資料庫執行個體相關聯的儲存類型。

- TdeCredentialArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  來自金鑰存放區的 ARN，執行個體會與此 ARN 建立關聯以進行 TDE 加密。

- Timezone – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫執行個體所屬 VPC 安全群組元素的清單。

錯誤

- DBInstanceAlreadyExistsFault
- InsufficientDBInstanceCapacityFault
- DBParameterGroupNotFoundFault
- DBSecurityGroupNotFoundFault
- InstanceQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidDBClusterStateFault
- InvalidSubnet
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBClusterNotFoundFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- KMSKeyNotAccessibleFault
- DomainNotFoundFault

DeleteDBInstance (動作)

此 API 的 AWS CLI 名稱是：`delete-db-instance`。

DeleteDBInstance 動作會刪除先前佈建的資料庫執行個體。刪除資料庫執行個體時，系統會一併刪除該執行個體所有的自動化備份，且無法復原。要由 `DeleteDBInstance` 刪除的資料庫執行個體手動資料庫快照則不會遭到刪除。

若您請求最終資料庫快照，則 Amazon Neptune 資料庫執行個體的狀態將為 `deleting`，並會持續到建立資料庫快照為止。API 動作 `DescribeDBInstance` 會用於監控此操作的狀態。動作一旦提交，便無法取消或回復。

請注意，當資料庫執行個體處於故障狀態，且其狀態為 failed、incompatible-restore 或 incompatible-network 時，您只能在 SkipFinalSnapshot 參數設為 true 時刪除它。

若資料庫執行個體是資料庫叢集中的唯一執行個體，或其已啟用刪除保護，則您將無法刪除該執行個體。

請求

- DBInstanceIdentifier (在 CLI 中：--db-instance-identifier) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  要刪除之資料庫執行個體的識別符。此參數沒有大小寫之分。

  約束:

  - 必須符合現有資料庫執行個體的名稱。
- FinalDBSnapshotIdentifier (在 CLI 中：--final-db-snapshot-identifier) – 字串，類型為：string (UTF-8 編碼的字串)。

  當 SkipFinalSnapshot 設為 false 時，所建立新 DBSnapshot 的 DBSnapshotIdentifier。

  > **Note**
  > 指定此參數，同時又將 SkipFinalSnapshot 參數設為 true 時，便會導致錯誤。

  約束:

  - 必須為 1 到 255 個字母或數字。
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號
  - 無法在刪除僅供讀取複本時指定。
- SkipFinalSnapshot (在 CLI 中:--skip-final-snapshot) – 布林值，類型為：boolean (布林值 (true 或 false))。

  判斷最終資料庫快照是否建立於資料庫執行個體刪除之前。若指定 true，則不會建立任何 DBSnapshot。若指定 false，則會在刪除資料庫執行個體前建立資料庫快照。

  請注意，當資料庫執行個體處於故障狀態，且其狀態為 'failed'、'incompatible-restore' 或 'incompatible-network' 時，它便只能在 SkipFinalSnapshot 參數設為 "true" 時刪除。

在刪除僅供讀取複本時指定 `true`。

> **Note**
>
> 若 SkipFinalSnapshot 為 `false`，則必須指定 FinalDBSnapshotIdentifier 參數。

預設：`false`

回應

包含 Amazon Neptune 資料庫執行個體的詳細資訊。

此資料類型在 the section called “DescribeDBInstances” 動作中會用來作為回應元素。

- AutoMinorVersionUpgrade – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示會自動套用次要版本的修補程式。
- AvailabilityZone – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫執行個體所在的可用區域名稱。
- BackupRetentionPeriod – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。
- CACertificateIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的憑證授權機構憑證識別碼。
- CopyTagsToSnapshot – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要從資料庫執行個體將標籤複製到資料庫執行個體的快照。
- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若資料庫執行個體是資料庫叢集的成員，則包含資料庫執行個體所屬資料庫叢集的名稱。
- DBInstanceArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon Resource Name (ARN)。
- DBInstanceClass – 字串，類型為：`string` (UTF-8 編碼的字串)。

包含資料庫執行個體之運算和記憶體容量類別的名稱。

- DBInstanceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫識別碼。此識別碼是可識別資料庫執行個體的唯一金鑰。

- DbInstancePort – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫執行個體接聽的連接埠。若資料庫執行個體是資料庫叢集的一部分，這可以是與資料庫叢集連接埠不同的連接埠。

- DBInstanceStatus – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫的目前狀態。

- DbiResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon 區域唯一、固定識別符。每當存取資料庫執行個體的 Amazon KMS 金鑰時，就會在 Amazon CloudTrail 日誌項目中找到此識別符。

- DBName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫名稱。

- DBParameterGroups – 一個 DBParameterGroupStatus 物件陣列。

  提供要套用到此資料庫執行個體的資料庫參數群組清單。

- DBSecurityGroups – 一個 DBSecurityGroupMembership 物件陣列。

  提供資料庫安全群組元素的清單，元素中只包含 `DBSecurityGroup.Name` 和 `DBSecurityGroup.Status` 子元素。

- DBSubnetGroup – DBSubnetGroup 物件。

  指定與資料庫執行個體相關聯子網路群組上的資訊，包含名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否已啟用刪除保護。如果刪除保護已經啟用，執行個體將無法刪除。請參閱刪除資料庫執行個體。

- DomainMemberships – 一個 DomainMembership 物件陣列。

  不支援

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

此資料庫執行個體已設為匯出至 CloudWatch Logs 的日誌類型清單。

- Endpoint – 端點 物件。

  指定連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供用於此資料庫執行個體的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- EnhancedMonitoringResourceArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Amazon CloudWatch Logs 日誌串流的 Amazon Resource Name (ARN)，該串流會接收資料庫執行個體的增強型監控指標。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若 Amazon Identity and Access Management (IAM) 身分驗證已啟用，則為 True，否則為 False。

- InstanceCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供建立資料庫執行個體的日期和時間。

- Iops – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定佈建 IOPS (每秒 I/O 操作) 值。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- LicenseModel – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的授權模型資訊。

- MonitoringInterval – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  為資料庫執行個體收集增強型監控指標點之間的間隔 (秒)。

- MonitoringRoleArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，允許 Neptune 將增強型監控指標傳送到 Amazon CloudWatch Logs。
- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫執行個體是否為異地同步備份部署。
- PendingModifiedValues – PendingModifiedValues 物件。

  指定對資料庫執行個體所進行的變更仍在擱置中。此元素只有在變更擱置中時才會包含在其中。特定的變更會由子元素識別。
- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。
- PreferredMaintenanceWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。
- PromotionTier – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  此值指定在現有主要執行個體故障後，僅供讀取複本提升為主要執行個體的順序。
- PubliclyAccessible – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  此標記不應繼續使用。
- ReadReplicaDBClusterIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個資料庫叢集的識別符，這些叢集是此資料庫執行個體的僅供讀取複本。
- ReadReplicaDBInstanceIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個與此資料庫執行個體相關聯僅供讀取複本的識別符。
- ReadReplicaSourceDBInstanceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含來源資料庫執行個體的識別符 (若此資料庫執行個體是僅供讀取複本)。
- SecondaryAvailabilityZone – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若存在，指定具備多個可用區支援之資料庫執行個體的次要可用區名稱。
- StatusInfos – 一個 DBInstanceStatusInfo 物件陣列。

  僅供讀取複本的狀態。若執行個體並非僅供讀取複本，此處即為空白。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。
- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與此資料庫執行個體相關聯的儲存類型。
- TdeCredentialArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  來自金鑰存放區的 ARN，執行個體會與此 ARN 建立關聯以進行 TDE 加密。
- Timezone – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援。
- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫執行個體所屬 VPC 安全群組元素的清單。

錯誤

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault
- DBSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- InvalidDBClusterStateFault

ModifyDBInstance (動作)

此 API 的 AWS CLI 名稱是：`modify-db-instance`。

修改資料庫執行個體的設定。您可以透過在請求中指定這些參數及新的值，來變更一或多個資料庫組態參數。若要了解您可以對資料庫執行個體進行的修改，請在呼叫 the section called “ModifyDBInstance” 前呼叫 the section called “DescribeValidDBInstanceModifications”。

請求

- AllowMajorVersionUpgrade (在 CLI 中:`--allow-major-version-upgrade`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示是否允許主要版本升級。變更此參數不會導致中斷，且變更會以非同步方式盡快套用。

- ApplyImmediately (在 CLI 中:`--apply-immediately`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定此請求中的修改及任何擱置中的修改是否要以非同步方式盡快套用，無論資料庫執行個體的 `PreferredMaintenanceWindow` 設定為何。

  若將此參數設為 `false`，則對資料庫執行個體的變更會在下一個維護時段期間套用。某些參數變更會造成中斷，且會在下一次呼叫 the section called “RebootDBInstance”，或是下一次故障重新開機時套用。

  預設：`false`

- AutoMinorVersionUpgrade (在 CLI 中:`--auto-minor-version-upgrade`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指出會在維護時段期間將次要版本升級自動套用至資料庫執行個體。除以下狀況外，變更此參數不會導致中斷，且變更會以非同步方式盡快套用。若在維護時段期間將此參數設為 `true`，且有可用的新次要版本，並且 Neptune 已為該引擎版本啟用自動修補時，便會導致中斷。

- BackupRetentionPeriod (在 CLI 中：`--backup-retention-period`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  不適用。自動化備份的保留期會由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called “ModifyDBCluster”。

  預設值：使用現有設定

- CACertificateIdentifier (在 CLI 中：`--ca-certificate-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指出需要與執行個體建立關聯的憑證。

- CloudwatchLogsExportConfiguration (在 CLI 中：`--cloudwatch-logs-export-configuration`) – CloudwatchLogsExportConfiguration 物件。

  要啟用的日誌類型的組態設定，適用於匯出到特定資料庫執行個體或資料庫叢集的 CloudWatch Logs。

- CopyTagsToSnapshot (在 CLI 中:`--copy-tags-to-snapshot`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  若為 True，則會從資料庫執行個體將所有標籤複製到資料庫執行個體的快照，否則為 False。預設值為 false。

- DBInstanceClass (在 CLI 中：`--db-instance-class`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新的資料庫執行個體運算與記憶體容量，例如 `db.m4.large`。並非所有 Amazon 區域皆提供所有資料庫執行個體類別。

  若您修改資料庫執行個體類別，則會在變更時發生中斷。變更會在下一個維護時段期間套用，除非針對此請求將 `ApplyImmediately` 指定為 `true`。

  預設值：使用現有設定

- DBInstanceIdentifier (在 CLI 中：`--db-instance-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體識別符。此值會以小寫字母字串的形式儲存。

  約束:

  - 必須符合現有 DBInstance 的識別符。

- DBParameterGroupName (在 CLI 中：`--db-parameter-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要套用到資料庫執行個體的資料庫參數群組名稱。變更此設定不會導致中斷。參數群組名稱本身會立即變更，但在您重新啟動實例而不需容錯移轉的情況下，才會套用實際參數變更。資料庫執行個體「不會」自動重新開機，且參數變更也「不會」在下一個維護時段期間套用。

  預設值：使用現有設定

  限制條件：資料庫參數群組必須位於和此資料庫執行個體相同的資料庫參數群組系列中。

- DBPortNumber (在 CLI 中：`--db-port-number`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  資料庫接受連線的連接埠號碼。

  `DBPortNumber` 參數的值不得和任何在資料庫執行個體選項群組中為選項指定的連接埠值相符。

  當您變更 `DBPortNumber` 的值時，您的資料庫會重新啟動，無論 `ApplyImmediately` 參數的值為何。

  預設：8182

- DBSecurityGroups (在 CLI 中：`--db-security-groups`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要在此資料庫執行個體上授權的資料庫安全群組清單。變更此設定不會導致中斷，且變更會以非同步方式盡快套用。

  約束:

  - 若有提供，則必須符合現有的 DBSecurityGroups。
- DBSubnetGroupName (在 CLI 中：`--db-subnet-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的新資料庫子網路群組。您可以使用此參數將資料庫執行個體移動至不同的 VPC。

  變更子網路群組會在變更期間造成中斷。變更會在下一個維護時段期間套用，除非您針對 `ApplyImmediately` 參數指定 `true`。

  限制條件：若有提供，則必須符合現有 DBSubnetGroup 的名稱。

  範例：`mySubnetGroup`
- DeletionProtection (在 CLI 中:`--deletion-protection`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫執行個體是否啟用刪除保護的值。啟用刪除保護時無法刪除資料庫。根據預設，刪除保護是停用的。請參閱刪除資料庫執行個體。
- Domain (在 CLI 中：`--domain`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援。
- DomainIAMRoleName (在 CLI 中：`--domain-iam-role-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援
- EnableIAMDatabaseAuthentication (在 CLI 中:`--enable-iam-database-authentication`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  設為 True 以啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的對應，否則為 false。

  您可以為以下資料庫引擎啟用 IAM 資料庫身分驗證

不適用。將 Amazon IAM 帳戶對應至資料庫帳戶由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called “ModifyDBCluster”。

預設：false

- EngineVersion (在 CLI 中：--engine-version) – 字串，類型為：string (UTF-8 編碼的字串)。

  要升級到的目標資料庫引擎版本號碼。目前，設定此參數沒有任何作用。若要將資料庫引擎升級至最新版本，請使用 the section called “ApplyPendingMaintenanceAction” API。

- Iops (在 CLI 中：--iops) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  執行個體的新佈建 IOPS (每秒 I/O 操作數) 值。

  變更此設定不會導致中斷，且變更會在下一個維護時段期間套用，除非針對此請求將 ApplyImmediately 參數設為 true。

  預設值：使用現有設定

- MonitoringInterval (在 CLI 中：--monitoring-interval) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  為資料庫執行個體收集增強型監控指標點之間的間隔 (秒)。若要停用收集增強型監控指標，請指定 0。預設值為 0。

  若指定 MonitoringRoleArn，您必須也將 MonitoringInterval 設為 0 之外的值。

  有效值:0, 1, 5, 10, 15, 30, 60

- MonitoringRoleArn (在 CLI 中：--monitoring-role-arn) – 字串，類型為：string (UTF-8 編碼的字串)。

  IAM 角色的 ARN，允許 Neptune 將增強型監控指標傳送到 Amazon CloudWatch Logs。例如：arn:aws:iam:123456789012:role/emaccess。

  若 MonitoringInterval 設為 0 之外的值，則您必須提供 MonitoringRoleArn 值。

- MultiAZ (在 CLI 中:--multi-az) – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  指定資料庫執行個體是否為異地同步備份部署。變更此參數不會導致中斷，且變更會在下一個維護時段期間套用，除非針對此請求將 ApplyImmediately 參數設為 true。

- NewDBInstanceIdentifier (在 CLI 中：--new-db-instance-identifier) – 字串，類型為：string (UTF-8 編碼的字串)。

重新命名資料庫執行個體時，資料庫執行個體的新資料庫執行個體識別符。當您變更資料庫執行個體識別符時，若您將 Apply Immediately 設為 True，則執行個體會立即重新開機。若您將 Apply Immediately 設為 False，則會在下一個維護時段期間重新開機。此值會以小寫字母字串的形式儲存。

約束:

- 必須包含 1 到 63 個字母、數字或連字號。
- 第一個字元必須是字母。
- 不能以連字號結尾或連續包含兩個連字號。

範例：mydbinstance

- PreferredBackupWindow (在 CLI 中：--preferred-backup-window) – 字串，類型為：string (UTF-8 編碼的字串)。

  若已啟用自動化備份，則為每日時間範圍，自動化備份會在此期間建立。

  不適用。建立自動化備份的每日時間範圍會由資料庫叢集管理。如需更多詳細資訊，請參閱 the section called “ModifyDBCluster”。

  約束:

  - 格式必須為 hh24:mi-hh24:mi
  - 必須以國際標準時間 (UTC) 表示
  - 不得和慣用的維護時段衝突
  - 必須至少 30 分鐘
- PreferredMaintenanceWindow (在 CLI 中：--preferred-maintenance-window) – 字串，類型為：string (UTF-8 編碼的字串)。

  每週可能進行系統維護的時間範圍 (UTC)，在此期間可能會發生中斷。除以下狀況外，變更此參數不會導致中斷，且變更會以非同步方式盡快套用。若有任何擱置中會造成重新開機的動作，且維護時段已變更成包含目前的時間，則變更此參數便會使資料庫執行個體重新開機。若將此時段移動到目前時間，則在目前時間和時段結束的時間之間必須至少間隔 30 分鐘，以確保套用擱置中的變更。

  預設值：使用現有設定

  格式：ddd:hh24:mi-ddd:hh24:mi

  有效日：星期一 | 星期二 | 星期三 | 星期四 | 星期五 | 星期六 | 星期日

限制條件：必須至少 30 分鐘

- PromotionTier (在 CLI 中：`--promotion-tier`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  此值指定在現有主要執行個體故障後，僅供讀取複本提升為主要執行個體的順序。

  預設：1

  有效值：0 到 15

- PubliclyAccessible (在 CLI 中:`--publicly-accessible`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  此標記不應繼續使用。

- StorageType (在 CLI 中：`--storage-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

- TdeCredentialArn (在 CLI 中：`--tde-credential-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  來自金鑰存放區的 ARN，執行個體會與此 ARN 建立關聯以進行 TDE 加密。

- TdeCredentialPassword (在 CLI 中：`--tde-credential-password`) – SensitiveString，類型為：`string` (UTF-8 編碼的字串)。

  在金鑰存放區指定 ARN 的密碼，用於存取裝置。

- VpcSecurityGroupIds (在 CLI 中：`--vpc-security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要在此資料庫執行個體上授權的 EC2 VPC 安全群組清單。此變更會以非同步的方式盡快套用。

  不適用。資料庫叢集會管理 EC2 VPC 安全群組的關聯清單。如需更多詳細資訊，請參閱 the section called “ModifyDBCluster”。

  約束:

  - 若有提供，則必須符合現有的 VpcSecurityGroupIds。

回應

包含 Amazon Neptune 資料庫執行個體的詳細資訊。

此資料類型在 the section called “DescribeDBInstances” 動作中會用來作為回應元素。

- AutoMinorVersionUpgrade – 布林值，類型為：boolean (布林值 (true 或 false))。

  指示會自動套用次要版本的修補程式。

- AvailabilityZone – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫執行個體所在的可用區域名稱。

- BackupRetentionPeriod – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- CACertificateIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  此資料庫執行個體的憑證授權機構憑證識別碼。

- CopyTagsToSnapshot – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定是否要從資料庫執行個體將標籤複製到資料庫執行個體的快照。

- DBClusterIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  若資料庫執行個體是資料庫叢集的成員，則包含資料庫執行個體所屬資料庫叢集的名稱。

- DBInstanceArn – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon Resource Name (ARN)。

- DBInstanceClass – 字串，類型為：string (UTF-8 編碼的字串)。

  包含資料庫執行個體之運算和記憶體容量類別的名稱。

- DBInstanceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  包含使用者提供的資料庫識別碼。此識別碼是可識別資料庫執行個體的唯一金鑰。

- DbInstancePort – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫執行個體接聽的連接埠。若資料庫執行個體是資料庫叢集的一部分，這可以是與資料庫叢集連接埠不同的連接埠。

- DBInstanceStatus – 字串，類型為：string (UTF-8 編碼的字串)。

  指定此資料庫的目前狀態。

- DbiResourceId – 字串，類型為：string (UTF-8 編碼的字串)。

資料庫執行個體的 Amazon 區域唯一、固定識別符。每當存取資料庫執行個體的 Amazon KMS 金鑰時，就會在 Amazon CloudTrail 日誌項目中找到此識別符。

- DBName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫名稱。

- DBParameterGroups – 一個 DBParameterGroupStatus 物件陣列。

  提供要套用到此資料庫執行個體的資料庫參數群組清單。

- DBSecurityGroups – 一個 DBSecurityGroupMembership 物件陣列。

  提供資料庫安全群組元素的清單，元素中只包含 `DBSecurityGroup.Name` 和 `DBSecurityGroup.Status` 子元素。

- DBSubnetGroup – DBSubnetGroup 物件。

  指定與資料庫執行個體相關聯子網路群組上的資訊，包含名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否已啟用刪除保護。如果刪除保護已經啟用，執行個體將無法刪除。請參閱刪除資料庫執行個體。

- DomainMemberships – 一個 DomainMembership 物件陣列。

  不支援

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體已設為匯出至 CloudWatch Logs 的日誌類型清單。

- Endpoint – 端點 物件。

  指定連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供用於此資料庫執行個體的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- EnhancedMonitoringResourceArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

Amazon CloudWatch Logs 日誌串流的 Amazon Resource Name (ARN)，該串流會接收資料庫執行個體的增強型監控指標。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若 Amazon Identity and Access Management (IAM) 身分驗證已啟用，則為 True，否則為 False。

- InstanceCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供建立資料庫執行個體的日期和時間。

- Iops – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定佈建 IOPS (每秒 I/O 操作) 值。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- LicenseModel – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的授權模型資訊。

- MonitoringInterval – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  為資料庫執行個體收集增強型監控指標點之間的間隔 (秒)。

- MonitoringRoleArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，允許 Neptune 將增強型監控指標傳送到 Amazon CloudWatch Logs。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫執行個體是否為異地同步備份部署。

- PendingModifiedValues – PendingModifiedValues 物件。

  指定對資料庫執行個體所進行的變更仍在擱置中。此元素只有在變更擱置中時才會包含在其中。特定的變更會由子元素識別。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

指定每日的時間範圍，若有啟用自動化備份 (由 BackupRetentionPeriod 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：string (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- PromotionTier – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  此值指定在現有主要執行個體故障後，僅供讀取複本提升為主要執行個體的順序。

- PubliclyAccessible – 布林值，類型為：boolean (布林值 (true 或 false))。

  此標記不應繼續使用。

- ReadReplicaDBClusterIdentifiers – 字串，類型為：string (UTF-8 編碼的字串)。

  包含一或多個資料庫叢集的識別符，這些叢集是此資料庫執行個體的僅供讀取複本。

- ReadReplicaDBInstanceIdentifiers – 字串，類型為：string (UTF-8 編碼的字串)。

  包含一或多個與此資料庫執行個體相關聯僅供讀取複本的識別符。

- ReadReplicaSourceDBInstanceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  包含來源資料庫執行個體的識別符 (若此資料庫執行個體是僅供讀取複本)。

- SecondaryAvailabilityZone – 字串，類型為：string (UTF-8 編碼的字串)。

  若存在，指定具備多個可用區支援之資料庫執行個體的次要可用區名稱。

- StatusInfos – 一個 DBInstanceStatusInfo 物件陣列。

  僅供讀取複本的狀態。若執行個體並非僅供讀取複本，此處即為空白。

- StorageEncrypted – 布林值，類型為：boolean (布林值 (true 或 false))。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。

- StorageType – 字串，類型為：string (UTF-8 編碼的字串)。

  指定與此資料庫執行個體相關聯的儲存類型。

- TdeCredentialArn – 字串，類型為：string (UTF-8 編碼的字串)。

  來自金鑰存放區的 ARN，執行個體會與此 ARN 建立關聯以進行 TDE 加密。

- Timezone – 字串，類型為：string (UTF-8 編碼的字串)。

不支援。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫執行個體所屬 VPC 安全群組元素的清單。

錯誤

- InvalidDBInstanceStateFault
- InvalidDBSecurityGroupStateFault
- DBInstanceAlreadyExistsFault
- DBInstanceNotFoundFault
- DBSecurityGroupNotFoundFault
- DBParameterGroupNotFoundFault
- InsufficientDBInstanceCapacityFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBUpgradeDependencyFailureFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- CertificateNotFoundFault
- DomainNotFoundFault

RebootDBInstance (動作)

此 API 的 AWS CLI 名稱是：`reboot-db-instance`。

您可能需要重新啟動資料庫執行個體，這通常是為了維護的原因。例如，如果您進行某些修改，或變更與資料庫執行個體相關聯的資料庫參數群組，則必須重新啟動執行個體，變更才會生效。

重新啟動資料庫執行個體，將重新啟動資料庫引擎服務。重新啟動資料庫執行個體會暫時中斷，在此期間，資料庫執行個體狀態設定為 rebooting (重新啟動中)。

請求

- DBInstanceIdentifier (在 CLI 中：`--db-instance-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體識別符。此參數是以小寫字母字串的形式儲存。

  約束:

  - 必須符合現有 DBInstance 的識別符。
- ForceFailover (在 CLI 中:`--force-failover`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  若為 `true`，則會透過多個可用區容錯移轉進行重新開機。

  限制條件：若執行個體並未針對多個可用區進行設定，您便無法指定 `true`。

回應

包含 Amazon Neptune 資料庫執行個體的詳細資訊。

此資料類型在 the section called “DescribeDBInstances” 動作中會用來作為回應元素。

- AutoMinorVersionUpgrade – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示會自動套用次要版本的修補程式。
- AvailabilityZone – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫執行個體所在的可用區域名稱。
- BackupRetentionPeriod – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。
- CACertificateIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的憑證授權機構憑證識別碼。
- CopyTagsToSnapshot – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是否要從資料庫執行個體將標籤複製到資料庫執行個體的快照。
- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若資料庫執行個體是資料庫叢集的成員，則包含資料庫執行個體所屬資料庫叢集的名稱。

- DBInstanceArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon Resource Name (ARN)。
- DBInstanceClass – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含資料庫執行個體之運算和記憶體容量類別的名稱。
- DBInstanceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫識別碼。此識別碼是可識別資料庫執行個體的唯一金鑰。
- DbInstancePort – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫執行個體接聽的連接埠。若資料庫執行個體是資料庫叢集的一部分，這可以是與資料庫叢集連接埠不同的連接埠。
- DBInstanceStatus – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫的目前狀態。
- DbiResourceId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon 區域唯一、固定識別符。每當存取資料庫執行個體的 Amazon KMS 金鑰時，就會在 Amazon CloudTrail 日誌項目中找到此識別符。
- DBName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫名稱。
- DBParameterGroups – 一個 DBParameterGroupStatus 物件陣列。

  提供要套用到此資料庫執行個體的資料庫參數群組清單。
- DBSecurityGroups – 一個 DBSecurityGroupMembership 物件陣列。

  提供資料庫安全群組元素的清單，元素中只包含 `DBSecurityGroup.Name` 和 `DBSecurityGroup.Status` 子元素。
- DBSubnetGroup – DBSubnetGroup 物件。

  指定與資料庫執行個體相關聯子網路群組上的資訊，包含名稱、描述，以及子網路群組中的子網路。
- DeletionProtection – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否已啟用刪除保護。如果刪除保護已經啟用，執行個體將無法刪除。請參閱刪除資料庫執行個體。
- DomainMemberships – 一個 DomainMembership 物件陣列。

不支援

- EnabledCloudwatchLogsExports – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體已設為匯出至 CloudWatch Logs 的日誌類型清單。

- Endpoint – 端點 物件。

  指定連線端點。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供用於此資料庫執行個體的資料庫引擎名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- EnhancedMonitoringResourceArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Amazon CloudWatch Logs 日誌串流的 Amazon Resource Name (ARN)，該串流會接收資料庫執行個體的增強型監控指標。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若 Amazon Identity and Access Management (IAM) 身分驗證已啟用，則為 True，否則為 False。

- InstanceCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供建立資料庫執行個體的日期和時間。

- Iops – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定佈建 IOPS (每秒 I/O 操作) 值。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- LicenseModel – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的授權模型資訊。

- MonitoringInterval – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  為資料庫執行個體收集增強型監控指標點之間的間隔 (秒)。

- MonitoringRoleArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，允許 Neptune 將增強型監控指標傳送到 Amazon CloudWatch Logs。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫執行個體是否為異地同步備份部署。

- PendingModifiedValues – PendingModifiedValues 物件。

  指定對資料庫執行個體所進行的變更仍在擱置中。此元素只有在變更擱置中時才會包含在其中。特定的變更會由子元素識別。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- PromotionTier – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  此值指定在現有主要執行個體故障後，僅供讀取複本提升為主要執行個體的順序。

- PubliclyAccessible – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  此標記不應繼續使用。

- ReadReplicaDBClusterIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個資料庫叢集的識別符，這些叢集是此資料庫執行個體的僅供讀取複本。

- ReadReplicaDBInstanceIdentifiers – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個與此資料庫執行個體相關聯僅供讀取複本的識別符。

- ReadReplicaSourceDBInstanceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含來源資料庫執行個體的識別符 (若此資料庫執行個體是僅供讀取複本)。

- SecondaryAvailabilityZone – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若存在，指定具備多個可用區支援之資料庫執行個體的次要可用區名稱。

- StatusInfos – 一個 DBInstanceStatusInfo 物件陣列。

  僅供讀取複本的狀態。若執行個體並非僅供讀取複本，此處即為空白。

- StorageEncrypted – 布林值，類型為：boolean (布林值 (true 或 false))。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。

- StorageType – 字串，類型為：string (UTF-8 編碼的字串)。

  指定與此資料庫執行個體相關聯的儲存類型。

- TdeCredentialArn – 字串，類型為：string (UTF-8 編碼的字串)。

  來自金鑰存放區的 ARN，執行個體會與此 ARN 建立關聯以進行 TDE 加密。

- Timezone – 字串，類型為：string (UTF-8 編碼的字串)。

  不支援。

- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫執行個體所屬 VPC 安全群組元素的清單。

錯誤

- InvalidDBInstanceStateFault
- DBInstanceNotFoundFault

DescribeDBInstances (動作)

此 API 的 AWS CLI 名稱是：describe-db-instances。

傳回所佈建之執行個體的相關資訊，並支援分頁。

> **Note**
>
> 此操作也可以傳回 Amazon RDS 執行個體和 Amazon DocDB 執行個體的資訊。

請求

- DBInstanceIdentifier (在 CLI 中：`--db-instance-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  使用者提供的執行個體識別符。若指定此參數，則只會傳回特定資料庫執行個體的資訊。此參數沒有大小寫之分。

  約束:

  - 若有提供，則必須符合現有 DBInstance 的識別符。
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  篩選條件，指定要描述的一或多個資料庫執行個體。

  支援的篩選條件：

  - `db-cluster-id` - 接受資料庫叢集識別符及資料庫叢集 Amazon Resource Name (ARN)。結果清單只會包含與透過這些 ARN 識別資料庫叢集相關聯的資料庫執行個體資訊。
  - `engine` -接受引擎名稱 (例如 `neptune`)，並將結果清單限制為由該引擎建立的資料庫執行個體。

  例如，若要從 Amazon CLI 調用此 API 並進行篩選，以便只傳回 Neptune 資料庫執行個體，您可以使用下列命令：

  Example

  ```
  aws neptune describe-db-instances \
              --filters  Name=engine,Values=neptune
  ```
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 `DescribeDBInstances` 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

回應

- DBInstances – 一個 DBInstance 物件陣列。

  the section called “DBInstance” 執行個體的清單。
- Marker – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。

錯誤

- DBInstanceNotFoundFault

DescribeOrderableDBInstanceOptions (動作)

此 API 的 AWS CLI 名稱是：describe-orderable-db-instance-options。

傳回指定引擎的可排序資料庫執行個體選項清單。

請求

- DBInstanceClass (在 CLI 中：--db-instance-class) – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫執行個體類別篩選條件值。指定此參數，以僅顯示與指定資料庫執行個體類別相符的可用供應項目。
- Engine (在 CLI 中：--engine) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  要為其擷取資料庫執行個體選項的引擎名稱。
- EngineVersion (在 CLI 中：--engine-version) – 字串，類型為：string (UTF-8 編碼的字串)。

  引擎版本篩選條件值。指定此參數，以僅顯示與指定引擎版本相符的可用供應項目。
- Filters (在 CLI 中：--filters) – 篩選條件 物件的陣列。

  目前不支援此參數。
- LicenseModel (在 CLI 中：--license-model) – 字串，類型為：string (UTF-8 編碼的字串)。

  授權模型篩選條件值。指定此參數，以僅顯示與指定授權模型相符的可用供應項目。

- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 DescribeOrderableDBInstanceOptions 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。
- Vpc (在 CLI 中:`--vpc`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  VPC 篩選條件值。指定此參數，以僅顯示可用的 VPC 或非 VPC 供應項目。

回應

- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 OrderableDBInstanceOptions 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- OrderableDBInstanceOptions – 一個 OrderableDBInstanceOption 物件陣列。

  the section called “OrderableDBInstanceOption” 結構，包含資料庫執行個體可排序選項的資訊。

DescribeValidDBInstanceModifications (動作)

此 API 的 AWS CLI 名稱是：`describe-valid-db-instance-modifications`。

您可以呼叫 the section called “DescribeValidDBInstanceModifications” 來了解您可以對資料庫執行個體進行的修改。您可以在呼叫 the section called “ModifyDBInstance” 時使用此資訊。

請求

- DBInstanceIdentifier (在 CLI 中：`--db-instance-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  客戶識別符或您資料庫執行個體的 ARN。

回應

您可以對資料庫執行個體進行的有效修改相關資訊。包含成功呼叫 the section called “DescribeValidDBInstanceModifications” 動作的結果。您可以在呼叫 the section called “ModifyDBInstance” 時使用此資訊。

- Storage – 一個 ValidStorageOptions 物件陣列。

  您資料庫執行個體的有效儲存體選項。

錯誤

- DBInstanceNotFoundFault
- InvalidDBInstanceStateFault

結構：

DBInstance (結構)

包含 Amazon Neptune 資料庫執行個體的詳細資訊。

此資料類型在 the section called “DescribeDBInstances” 動作中會用來作為回應元素。

欄位

- AutoMinorVersionUpgrade - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示會自動套用次要版本的修補程式。
- AvailabilityZone - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫執行個體所在的可用區域名稱。
- BackupRetentionPeriod - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。
- CACertificateIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的憑證授權機構憑證識別碼。
- CopyTagsToSnapshot - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

指定是否要從資料庫執行個體將標籤複製到資料庫執行個體的快照。

- DBClusterIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  若資料庫執行個體是資料庫叢集的成員，則包含資料庫執行個體所屬資料庫叢集的名稱。

- DBInstanceArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon Resource Name (ARN)。

- DBInstanceClass - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含資料庫執行個體之運算和記憶體容量類別的名稱。

- DBInstanceIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫識別碼。此識別碼是可識別資料庫執行個體的唯一金鑰。

- DbInstancePort - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫執行個體接聽的連接埠。若資料庫執行個體是資料庫叢集的一部分，這可以是與資料庫叢集連接埠不同的連接埠。

- DBInstanceStatus - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫的目前狀態。

- DbiResourceId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的 Amazon 區域唯一、固定識別符。每當存取資料庫執行個體的 Amazon KMS 金鑰時，就會在 Amazon CloudTrail 日誌項目中找到此識別符。

- DBName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫名稱。

- DBParameterGroups - 這是 DBParameterGroupStatus 物件的陣列。

  提供要套用到此資料庫執行個體的資料庫參數群組清單。

- DBSecurityGroups - 這是 DBSecurityGroupMembership 物件的陣列。

  提供資料庫安全群組元素的清單，元素中只包含 `DBSecurityGroup.Name` 和 `DBSecurityGroup.Status` 子元素。

- DBSubnetGroup - 這是一個 DBSubnetGroup 物件。

  指定與資料庫執行個體相關聯子網路群組上的資訊，包含名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – 這是 BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  指出資料庫執行個體是否已啟用刪除保護。如果刪除保護已經啟用，執行個體將無法刪除。請參閱刪除資料庫執行個體。
- DomainMemberships - 這是 DomainMembership 物件的陣列。

  不支援
- EnabledCloudwatchLogsExports - 這是字串，類型為：string (UTF-8 編碼的字串)。

  此資料庫執行個體已設為匯出至 CloudWatch Logs 的日誌類型清單。
- Endpoint - 這是一個 端點 物件。

  指定連線端點。
- Engine - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供用於此資料庫執行個體的資料庫引擎名稱。
- EngineVersion - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指示資料庫引擎版本。
- EnhancedMonitoringResourceArn - 這是字串，類型為：string (UTF-8 編碼的字串)。

  Amazon CloudWatch Logs 日誌串流的 Amazon Resource Name (ARN)，該串流會接收資料庫執行個體的增強型監控指標。
- IAMDatabaseAuthenticationEnabled - 這是布林值，類型為：boolean (布林值 (true 或 false))。

  若 Amazon Identity and Access Management (IAM) 身分驗證已啟用，則為 True，否則為 False。
- InstanceCreateTime – 這是 TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供建立資料庫執行個體的日期和時間。
- Iops – 這是 IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  指定佈建 IOPS (每秒 I/O 操作) 值。
- KmsKeyId - 這是字串，類型為：string (UTF-8 編碼的字串)。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。
- LatestRestorableTime – 這是 TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

指定可透過 point-in-time 還原還原資料庫的最新時間。

- LicenseModel - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料庫執行個體的授權模型資訊。

- MonitoringInterval – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  為資料庫執行個體收集增強型監控指標點之間的間隔 (秒)。

- MonitoringRoleArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，允許 Neptune 將增強型監控指標傳送到 Amazon CloudWatch Logs。

- MultiAZ - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫執行個體是否為異地同步備份部署。

- PendingModifiedValues - 這是一個 PendingModifiedValues 物件。

  指定對資料庫執行個體所進行的變更仍在擱置中。此元素只有在變更擱置中時才會包含在其中。特定的變更會由子元素識別。

- PreferredBackupWindow - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每日的時間範圍，若有啟用自動化備份 (由 `BackupRetentionPeriod` 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- PromotionTier – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  此值指定在現有主要執行個體故障後，僅供讀取複本提升為主要執行個體的順序。

- PubliclyAccessible - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  此標記不應繼續使用。

- ReadReplicaDBClusterIdentifiers - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個資料庫叢集的識別符，這些叢集是此資料庫執行個體的僅供讀取複本。

- ReadReplicaDBInstanceIdentifiers - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含一或多個與此資料庫執行個體相關聯僅供讀取複本的識別符。

- ReadReplicaSourceDBInstanceIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

包含來源資料庫執行個體的識別符 (若此資料庫執行個體是僅供讀取複本)。

- SecondaryAvailabilityZone - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  若存在，指定具備多個可用區支援之資料庫執行個體的次要可用區名稱。
- StatusInfos - 這是 DBInstanceStatusInfo 物件的陣列。

  僅供讀取複本的狀態。若執行個體並非僅供讀取複本，此處即為空白。
- StorageEncrypted - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  不支援：資料庫執行個體的加密會由資料庫叢集管理。
- StorageType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定與此資料庫執行個體相關聯的儲存類型。
- TdeCredentialArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  來自金鑰存放區的 ARN，執行個體會與此 ARN 建立關聯以進行 TDE 加密。
- Timezone - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援。
- VpcSecurityGroups - 這是 VpcSecurityGroupMembership 物件的陣列。

  提供資料庫執行個體所屬 VPC 安全群組元素的清單。

`DBInstance` 會用來做為以下項目的回應元素：

- CreateDBInstance
- DeleteDBInstance
- ModifyDBInstance
- RebootDBInstance

DBInstanceStatusInfo (結構)

提供資料庫執行個體的狀態資訊清單。

欄位

- Message - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

若執行個體發生錯誤，則為錯誤的詳細資訊。若執行個體並未處於錯誤狀態，則此值為空白。

- Normal - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  布林值，若執行個體正常運作，則為 True，若執行個體處於錯誤狀態，則為 False。

- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的狀態。針對僅供讀取複本的 StatusType，其值可以是正在複寫、錯誤、已停止或已終止。

- StatusType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此值目前是「讀取複寫」。

OrderableDBInstanceOption (結構)

包含資料庫執行個體的可用選項清單。

此資料類型在 the section called “DescribeOrderableDBInstanceOptions” 動作中會用來作為回應元素。

欄位

- AvailabilityZones - 這是 AvailabilityZone 物件的陣列。

  資料庫執行個體的可用區域清單。

- DBInstanceClass - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的資料庫執行個體類別。

- Engine - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的引擎類型。

- EngineVersion - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的引擎版本。

- LicenseModel - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體的授權模型。

- MaxIopsPerDbInstance – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

資料庫執行個體的總佈建 IOPS 上限。

- MaxIopsPerGib - 這是 DoubleOptional，類型為：`double` (雙精度 IEEE 754 浮點數)。

  資料庫執行個體的每 GiB 佈建 IOPS 上限。

- MaxStorageSize – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  資料庫執行個體的儲存體大小上限。

- MinIopsPerDbInstance – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  資料庫執行個體的總佈建 IOPS 下限。

- MinIopsPerGib - 這是 DoubleOptional，類型為：`double` (雙精度 IEEE 754 浮點數)。

  資料庫執行個體的每 GiB 佈建 IOPS 下限。

- MinStorageSize – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  資料庫執行個體的儲存體大小下限。

- MultiAZCapable - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否具備多個可用區功能。

- ReadReplicaCapable - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否可以擁有僅供讀取複本。

- StorageType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

- SupportsEnhancedMonitoring - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否支援增強型監控，其間隔介於 1 到 60 秒。

- SupportsGlobalDatabases - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  一值，指示您是否可以使用 Neptune 全球資料庫，搭配其他資料庫引擎屬性的特定組合。

- SupportsIAMDatabaseAuthentication - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否支援 IAM 資料庫身分驗證。

- SupportsIops - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫執行個體是否支援佈建 IOPS。

- SupportsStorageEncryption - 這是布林值，類型為：boolean (布林值 (true 或 false))。

  指出資料庫執行個體是否支援加密儲存體。
- Vpc - 這是布林值，類型為：boolean (布林值 (true 或 false))。

  指出資料庫執行個體是否位於 VPC 中。

PendingModifiedValues (結構)

此資料類型在 the section called “ModifyDBInstance” 動作中會用來作為回應元素。

欄位

- AllocatedStorage – 這是 IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  包含要套用或目前正在套用之資料庫執行個體的新 AllocatedStorage 大小。
- BackupRetentionPeriod – 這是 IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  指定保留自動化備份的待處理天數。
- CACertificateIdentifier - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫執行個體 CA 憑證的識別符。
- DBInstanceClass - 這是字串，類型為：string (UTF-8 編碼的字串)。

  包含要套用或目前正在套用之資料庫執行個體的新 DBInstanceClass。
- DBInstanceIdentifier - 這是字串，類型為：string (UTF-8 編碼的字串)。

  包含要套用或目前正在套用之資料庫執行個體的新 DBInstanceIdentifier。
- DBSubnetGroupName - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫執行個體的新資料庫子網路群組。
- EngineVersion - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指示資料庫引擎版本。
- Iops – 這是 IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  指定要套用或目前正在套用之資料庫執行個體的新佈建 IOPS 值。
- MultiAZ – 這是 BooleanOptional，類型為：boolean (布林值 (true 或 false))。

指出單一可用區資料庫執行個體要變更為異地同步備份部署。

- PendingCloudwatchLogsExports - 這是一個 PendingCloudwatchLogsExports 物件。

  此 `PendingCloudwatchLogsExports` 結構會指定 CloudWatch Logs 啟用和停用的待定變更。

- Port – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫執行個體的待處理連接埠。

- StorageType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

ValidStorageOptions (結構)

不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

欄位

- IopsToStorageRatio - 這是 DoubleRange 物件的陣列。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

- ProvisionedIops - 這是 範圍 物件的陣列。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

- StorageSize - 這是 範圍 物件的陣列。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

- StorageType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  不適用。在 Neptune 中，儲存類型是在資料庫叢集層級進行管理。

ValidDBInstanceModificationsMessage (結構)

您可以對資料庫執行個體進行的有效修改相關資訊。包含成功呼叫 the section called "DescribeValidDBInstanceModifications" 動作的結果。您可以在呼叫 the section called "ModifyDBInstance" 時使用此資訊。

欄位

- Storage - 這是 ValidStorageOptions 物件的陣列。

  您資料庫執行個體的有效儲存體選項。

`ValidDBInstanceModificationsMessage` 會用來做為以下項目的回應元素：

- DescribeValidDBInstanceModifications

Neptune 參數 API

動作:

- CopyDBParameterGroup (動作)
- CopyDBClusterParameterGroup (動作)
- CreateDBParameterGroup (動作)
- CreateDBClusterParameterGroup (動作)
- DeleteDBParameterGroup (動作)
- DeleteDBClusterParameterGroup (動作)
- ModifyDBParameterGroup (動作)
- ModifyDBClusterParameterGroup (動作)
- ResetDBParameterGroup (動作)
- ResetDBClusterParameterGroup (動作)
- DescribeDBParameters (動作)
- DescribeDBParameterGroups (動作)
- DescribeDBClusterParameters (動作)
- DescribeDBClusterParameterGroups (動作)
- DescribeEngineDefaultParameters (動作)
- DescribeEngineDefaultClusterParameters (動作)

結構：

- Parameter (結構)

- DBParameterGroup (結構)
- DBClusterParameterGroup (結構)
- DBParameterGroupStatus (結構)

CopyDBParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`copy-db-parameter-group`。

複製指定的資料庫參數群組。

請求

- SourceDBParameterGroupIdentifier (在 CLI 中：`--source-db-parameter-group-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  來源資料庫參數群組的識別符或 ARN。如需建立 ARN 的資訊，請參閱建構 Amazon Resource Name (ARN)。

  約束:

  - 必須指定有效的資料庫參數群組。
  - 必須指定有效的資料庫參數群組識別符 (例如 `my-db-param-group`)，或有效的 ARN。
- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給所複製資料庫參數群組的標籤。
- TargetDBParameterGroupDescription (在 CLI 中：`--target-db-parameter-group-description`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  所複製資料庫參數群組的描述。
- TargetDBParameterGroupIdentifier (在 CLI 中：`--target-db-parameter-group-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  所複製資料庫參數群組的識別符。

  約束:

  - 不得為無效、空白或留空。
  - 必須包含 1 到 255 個字母、數字或連字號。
  - 第一個字元必須是字母。

- 不能以連字號結尾或連續包含兩個連字號。

範例：`my-db-parameter-group`

回應

包含 Amazon Neptune 資料庫參數群組的詳細資訊。

此資料類型在 the section called “DescribeDBParameterGroups” 動作中會用來作為回應元素。

- DBParameterGroupArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組的 Amazon Resource Name (ARN)。
- DBParameterGroupFamily – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫參數群組相容的資料庫參數群組系列名稱。
- DBParameterGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫參數群組的名稱。
- Description – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫參數群組的客戶指定描述。

錯誤

- DBParameterGroupNotFoundFault
- DBParameterGroupAlreadyExistsFault
- DBParameterGroupQuotaExceededFault

CopyDBClusterParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`copy-db-cluster-parameter-group`。

複製指定的資料庫叢集參數群組。

請求

- SourceDBClusterParameterGroupIdentifier (在 CLI 中：`--source-db-cluster-parameter-group-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

來源資料庫叢集參數群組的識別符或 Amazon Resource Name (ARN)。如需建立 ARN 的資訊，請參閱建構 Amazon Resource Name (ARN)。

約束:

- 必須指定有效的資料庫叢集參數群組。
- 若來源資料庫叢集參數群組位於和複本相同的 Amazon 區域，請指定有效的資料庫參數群組識別符 (例如 `my-db-cluster-param-group`)，或有效的 ARN。
- 若來源資料庫參數群組位於和複本不同的 Amazon 區域，請指定有效的資料庫叢集參數群組 ARN (例如 `arn:aws:rds:us-east-1:123456789012:cluster-pg:custom-cluster-group1`)。

- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給所複製資料庫叢集參數群組的標籤。

- TargetDBClusterParameterGroupDescription (在 CLI 中：`--target-db-cluster-parameter-group-description`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  所複製資料庫叢集參數群組的描述。

- TargetDBClusterParameterGroupIdentifier (在 CLI 中：`--target-db-cluster-parameter-group-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  所複製資料庫叢集參數群組的識別符。

  約束:

  - 不可為 null、空白或留空
  - 必須包含 1 到 255 個字母、數字或連字號
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號

  範例：`my-cluster-param-group1`

回應

包含 Amazon Neptune 資料庫叢集參數群組的詳細資訊。

此資料類型在 the section called “DescribeDBClusterParameterGroups” 動作中會用來作為回應元素。

- DBClusterParameterGroupArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫叢集參數群組的 Amazon Resource Name (ARN)。

- DBClusterParameterGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫叢集參數群組的名稱。

- DBParameterGroupFamily – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫叢集參數群組相容的資料庫參數群組系列名稱。

- Description – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供該客戶指定之此資料庫叢集參數群組的描述。

錯誤

- DBParameterGroupNotFoundFault
- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`create-db-parameter-group`。

建立新的資料庫參數群組。

最初會為資料庫執行個體所使用的資料庫引擎，使用預設參數建立資料庫參數群組。若要為任何參數提供自訂值，您必須在使用 ModifyDBParameterGroup 建立它後修改群組。在您建立資料庫參數群組後，您需要使用 ModifyDBInstance 將它與您的資料庫執行個體建立關聯。當您將新的資料庫參數群組與執行中的資料庫執行個體建立關聯時，您需要在不使用容錯移轉的情況下重新開機資料庫執行個體，才能讓新的資料庫參數群組和相關聯的設定產生效果。

> **Important**
>
> 建立資料庫參數群組之後，您至少應等待 5 分鐘，再使用該資料庫參數群組當作預設參數群組，建立第一個資料庫執行個體。這可讓 Amazon Neptune 在將參數群組用來做為新資料庫執行個體的預設前，完成建立動作。這對建立資料庫執行個體預設資料庫時的關鍵參數尤其重要，像是 `character_set_database` 參數所定義之預設資料庫的字元集。您可以使用

Amazon Neptune 主控台的「參數群組」選項或 DescribeDBParameters 命令來驗證您的資料庫參數群組已建立或修改完成。

請求

- DBParameterGroupFamily (在 CLI 中：`--db-parameter-group-family`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組系列名稱。資料庫參數群組可和一個，並且只能和一個資料庫參數群組系列建立關聯，並且只能套用到執行與該資料庫參數群組系列相容資料庫引擎和引擎版本的資料庫執行個體。
- DBParameterGroupName (在 CLI 中：`--db-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組的名稱。

  約束:
  - 必須為 1 到 255 個字母、數字或連字號。
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號

  > **Note**
  > 此值會以小寫字母字串的形式儲存。
- Description (在 CLI 中：`--description`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組的描述。
- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給新資料庫參數群組的標籤。

回應

包含 Amazon Neptune 資料庫參數群組的詳細資訊。

此資料類型在 the section called “DescribeDBParameterGroups” 動作中會用來作為回應元素。

- DBParameterGroupArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫參數群組的 Amazon Resource Name (ARN)。

- DBParameterGroupFamily – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫參數群組相容的資料庫參數群組系列名稱。

- DBParameterGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫參數群組的名稱。

- Description – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫參數群組的客戶指定描述。

錯誤

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

CreateDBClusterParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`create-db-cluster-parameter-group`。

建立新的資料庫叢集參數群組。

資料庫叢集參數群組中的參數會套用到資料庫叢集中的所有執行個體。

最初會為資料庫叢集中執行個體所使用的資料庫引擎，使用預設參數建立資料庫叢集參數群組。若要為任何參數提供自訂值，您必須在使用 the section called “ModifyDBClusterParameterGroup” 建立它後修改群組。在您建立資料庫叢集參數群組後，您需要使用 the section called “ModifyDBCluster” 將它與您的資料庫叢集建立關聯。當您將新的資料庫叢集參數群組與執行中的資料庫叢集建立關聯時，您需要在不使用容錯移轉的情況下重新開機資料庫叢集中的資料庫執行個體，才能讓新的資料庫叢集參數群組和相關聯的設定產生效果。

> **Important**
>
> 建立資料庫叢集參數群組後，您應等待至少 5 分鐘，然後再建立第一個使用該資料庫叢集參數群組做為預設參數群組的資料庫叢集。這可讓 Amazon Neptune 在將資料庫叢集參數群組用來做為新資料庫叢集的預設前，完成建立動作。這對建立資料庫叢集預設資料庫時的關鍵參數尤其重要，像是 `character_set_database` 參數所定義的預設資料庫

字元集。您可以使用 Amazon Neptune 主控台的「參數群組」選項或 the section called "DescribeDBClusterParameters" 命令，來驗證是否已建立或修改您的資料庫叢集參數群組。

請求

- DBClusterParameterGroupName (在 CLI 中：`--db-cluster-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集參數群組的名稱。

  約束:

  - 必須符合現有 DBClusterParameterGroup 的名稱。

  > **Note**
  > 此值會以小寫字母字串的形式儲存。

- DBParameterGroupFamily (在 CLI 中：`--db-parameter-group-family`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集參數群組的系列名稱。資料庫叢集參數群組可和一個，並且只能和一個資料庫叢集參數群組系列建立關聯，並且只能套用到執行與該資料庫叢集參數群組系列相容資料庫引擎和引擎版本的資料庫叢集。

- Description (在 CLI 中：`--description`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集參數群組的描述。

- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給新資料庫叢集參數群組的標籤。

回應

包含 Amazon Neptune 資料庫叢集參數群組的詳細資訊。

此資料類型在 the section called "DescribeDBClusterParameterGroups" 動作中會用來作為回應元素。

- DBClusterParameterGroupArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集參數群組的 Amazon Resource Name (ARN)。

- DBClusterParameterGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫叢集參數群組的名稱。

- DBParameterGroupFamily – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫叢集參數群組相容的資料庫參數群組系列名稱。

- Description – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供該客戶指定之此資料庫叢集參數群組的描述。

錯誤

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

DeleteDBParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`delete-db-parameter-group`。

刪除指定的 DBParameterGroup。要刪除的 DBParameterGroup 不可和任何資料庫執行個體建立關聯。

請求

- DBParameterGroupName (在 CLI 中：`--db-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組的名稱。

  約束:

  - 必須是現有資料庫參數群組的名稱
  - 您無法刪除預設資料庫參數群組
  - 不可和任何資料庫執行個體建立關聯

回應

- 無回應參數。

錯誤

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DeleteDBClusterParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`delete-db-cluster-parameter-group`。

刪除指定的資料庫叢集參數群組。要刪除的資料庫叢集參數群組不可和任何資料庫叢集建立關聯。

請求

- DBClusterParameterGroupName (在 CLI 中：`--db-cluster-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集參數群組的名稱。

  約束:
  - 必須是現有資料庫叢集參數群組的名稱。
  - 您無法刪除預設資料庫叢集參數群組。
  - 不可和任何資料庫叢集建立關聯

回應

- 無回應參數。

錯誤

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ModifyDBParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`modify-db-parameter-group`。

修改資料庫參數群組的參數。若要修改超過一個參數，請提交下列項目的清單：`ParameterName`、`ParameterValue` 和 `ApplyMethod`。單一請求中最多可修改 20 個參數。

> **Note**
>
> 動態參數的變更會立即套用。對靜態參數進行的變更，需要在不使用容錯移轉至與參數群組關聯資料庫執行個體的情況下重新開機，才能生效。

> **Important**
>
> 修改資料庫參數群組之後，您應該等待至少5分鐘，然後再建立第一個使用該資料庫參數群組做為預設參數群組的資料庫執行個體。這可讓 Amazon Neptune 在將參數群組用來做為新資料庫執行個體的預設前，完成修改動作。這對建立資料庫執行個體預設資料庫時的關鍵參數尤其重要，像是 `character_set_database` 參數所定義之預設資料庫的字元集。您可以使用 Amazon Neptune 主控台的「參數群組」選項或 DescribeDBParameters 命令來驗證您的資料庫參數群組已建立或修改完成。

請求

- DBParameterGroupName (在 CLI 中：`--db-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組的名稱。

  約束:

  - 若有提供，則必須符合現有 DBParameterGroup 的名稱。
- Parameters (在 CLI 中：`--parameters`) – 必要：參數 物件的陣列。

  參數名稱、值，以及參數更新套用方法的陣列。必須至少提供一個參數名稱、值及套用方法；後續引數都是選用的。單一請求中最多可修改 20 個參數。

  有效值 (針對套用方法)：`immediate | pending-reboot`

  > **Note**
  >
  > 您只能在搭配動態參數時使用 immediate 值。您可以針對動態和靜態參數使用 pending-reboot 值，並且變更會在您不使用容錯移轉重新開機資料庫執行個體時套用。

回應

- DBParameterGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫參數群組的名稱。

錯誤

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ModifyDBClusterParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`modify-db-cluster-parameter-group`。

修改資料庫叢集參數群組的參數。若要修改超過一個參數，請提交下列項目的清單：`ParameterName`、`ParameterValue` 和 `ApplyMethod`。單一請求中最多可修改 20 個參數。

> **Note**
>
> 動態參數的變更會立即套用。對靜態參數進行的變更，需要在不使用容錯移轉至與參數群組關聯資料庫叢集的情況下重新開機，才能生效。

> **Important**
>
> 建立資料庫叢集參數群組後，您應等待至少 5 分鐘，然後再建立第一個使用該資料庫叢集參數群組做為預設參數群組的資料庫叢集。這可讓 Amazon Neptune 在將參數群組用來做為新資料庫叢集的預設前，完成建立動作。這對建立資料庫叢集預設資料庫時的關鍵參數尤其重要，像是 `character_set_database` 參數所定義的預設資料庫字元集。您可以使用 Amazon Neptune 主控台的「參數群組」選項或 the section called “DescribeDBClusterParameters” 命令來驗證您的資料庫叢集參數群組已建立或修改完成。

請求

- DBClusterParameterGroupName (在 CLI 中：`--db-cluster-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

欲修改資料庫叢集參數群組的名稱。

- Parameters (在 CLI 中：`--parameters`) – 必要：參數 物件的陣列。

  欲修改資料庫叢集參數群組中參數的清單。

回應

- DBClusterParameterGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集參數群組的名稱。

  約束:

  - 必須為 1 到 255 個字母或數字。
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號

> **Note**
>
> 此值會以小寫字母字串的形式儲存。

錯誤

- DBParameterGroupNotFoundFault
- InvalidDBParameterGroupStateFault

ResetDBParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`reset-db-parameter-group`。

將資料庫參數群組的參數修改為引擎/系統的預設值。若要重設特定參數，請提供下列項目的清單：`ParameterName` 和 `ApplyMethod`。若要重設整個資料庫參數群組，請指定 `DBParameterGroup` 名稱和 `ResetAllParameters` 參數。重設整個群組時，動態參數會立即更新，靜態參數則會設為 `pending-reboot`，並在下一次資料庫執行個體重新啟動或 `RebootDBInstance` 請求時生效。

請求

- DBParameterGroupName (在 CLI 中：`--db-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組的名稱。

  約束:

  - 必須符合現有 DBParameterGroup 的名稱。
- Parameters (在 CLI 中：`--parameters`) – 參數 物件的陣列。

  若要重設整個資料庫參數群組，請指定 `DBParameterGroup` 名稱和 `ResetAllParameters` 參數。若要重設特定參數，請提供下列項目的清單：`ParameterName` 和 `ApplyMethod`。單一請求中最多可修改 20 個參數。

  有效值 (針對套用方法)：`pending-reboot`
- ResetAllParameters (在 CLI 中:`--reset-all-parameters`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定是 (`true`) 否 (`false`) 要將資料庫參數群組中的所有參數重設為預設值。

  預設：`true`

回應

- DBParameterGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫參數群組的名稱。

錯誤

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

ResetDBClusterParameterGroup (動作)

此 API 的 AWS CLI 名稱是：`reset-db-cluster-parameter-group`。

將資料庫叢集參數群組的參數修改為預設值。若要重設特定參數，請提交下列項目的清單：ParameterName 和 ApplyMethod。若要重設整個資料庫叢集參數群組，請指定 DBClusterParameterGroupName 名稱和 ResetAllParameters 參數。

重設整個群組時，動態參數會立即更新，靜態參數則會設為 pending-reboot，並在下一次資料庫執行個體重新啟動或 the section called “RebootDBInstance” 請求時生效。您必須為您欲套用更新後靜態參數資料庫叢集中的每個資料庫執行個體呼叫 the section called “RebootDBInstance”。

請求

- DBClusterParameterGroupName (在 CLI 中：--db-cluster-parameter-group-name) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  要重設的資料庫叢集參數群組名稱。
- Parameters (在 CLI 中：--parameters) – 參數 物件的陣列。

  資料庫叢集參數群組中的參數名稱清單，會將此清單上的參數重設為其預設值。若將 ResetAllParameters 參數設為 true，您便無法使用此參數。
- ResetAllParameters (在 CLI 中:--reset-all-parameters) – 布林值，類型為：boolean (布林值 (true 或 false))。

  會在將資料庫叢集參數群組中所有參數重設為其預設值時設為 true 的值，否則為 false。若有為 Parameters 參數指定參數名稱清單，您便無法使用此參數。

回應

- DBClusterParameterGroupName – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集參數群組的名稱。

  約束:
  - 必須為 1 到 255 個字母或數字。
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號

> **Note**
>
> 此值會以小寫字母字串的形式儲存。

錯誤

- InvalidDBParameterGroupStateFault
- DBParameterGroupNotFoundFault

DescribeDBParameters (動作)

此 API 的 AWS CLI 名稱是：`describe-db-parameters`。

傳回特定資料庫參數群組的詳細參數清單。

請求

- DBParameterGroupName (在 CLI 中：`--db-parameter-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回其詳細資訊的特定資料庫參數群組名稱。

  約束:

  - 若有提供，則必須符合現有 DBParameterGroup 的名稱。
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 `DescribeDBParameters` 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。
- Source (在 CLI 中：`--source`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回的參數類型。

預設：傳回所有參數類型。

有效值: `user | system | engine-default`

回應

- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- Parameters – 一個 參數 物件陣列。

  the section called “參數” 值的清單。

錯誤

- DBParameterGroupNotFoundFault

DescribeDBParameterGroups (動作)

此 API 的 AWS CLI 名稱是：`describe-db-parameter-groups`。

傳回 `DBParameterGroup` 描述的清單。若有指定 `DBParameterGroupName`，則清單只會包含指定資料庫參數群組的描述。

請求

- DBParameterGroupName (在 CLI 中：`--db-parameter-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回其詳細資訊的特定資料庫參數群組名稱。

  約束:
  - 若有提供，則必須符合現有 DBClusterParameterGroup 的名稱。
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

前一個 DescribeDBParameterGroups 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。

- MaxRecords (在 CLI 中：--max-records) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 MaxRecords 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

回應

- DBParameterGroups – 一個 DBParameterGroup 物件陣列。

  the section called “DBParameterGroup” 執行個體的清單。

- Marker – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。

錯誤

- DBParameterGroupNotFoundFault

DescribeDBClusterParameters (動作)

此 API 的 AWS CLI 名稱是：describe-db-cluster-parameters。

傳回特定資料庫叢集參數群組的詳細參數清單。

請求

- DBClusterParameterGroupName (在 CLI 中：--db-cluster-parameter-group-name) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  要傳回其參數詳細資訊的特定資料庫叢集參數群組名稱。

約束:

- 若有提供，則必須符合現有 DBClusterParameterGroup 的名稱。

- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。

- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 `DescribeDBClusterParameters` 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

- Source (在 CLI 中：`--source`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指出只傳回特定來源參數的值。參數來源可以是 `engine`、`service` 或 `customer`。

回應

- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 DescribeDBClusterParameters 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

- Parameters – 一個 參數 物件陣列。

  提供資料庫叢集參數群組的參數清單。

錯誤

- DBParameterGroupNotFoundFault

DescribeDBClusterParameterGroups (動作)

此 API 的 AWS CLI 名稱是：`describe-db-cluster-parameter-groups`。

傳回 `DBClusterParameterGroup` 描述的清單。若有指定 `DBClusterParameterGroupName` 參數，則清單只會包含指定資料庫叢集參數群組的描述。

請求

- DBClusterParameterGroupName (在 CLI 中：`--db-cluster-parameter-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回其詳細資訊的特定資料庫叢集參數群組名稱。

  約束:

  - 若有提供，則必須符合現有 DBClusterParameterGroup 的名稱。
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 `DescribeDBClusterParameterGroups` 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

回應

- DBClusterParameterGroups – 一個 DBClusterParameterGroup 物件陣列。

  資料庫叢集參數群組的清單。
- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

前一個 DescribeDBClusterParameterGroups 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。

錯誤

- DBParameterGroupNotFoundFault

DescribeEngineDefaultParameters (動作)

此 API 的 AWS CLI 名稱是：describe-engine-default-parameters。

傳回指定資料庫引擎的預設引擎和系統參數資訊。

請求

- DBParameterGroupFamily (在 CLI 中：--db-parameter-group-family) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  資料庫參數群組系列的名稱。
- Filters (在 CLI 中：--filters) – 篩選條件 物件的陣列。

  目前不支援。
- Marker (在 CLI 中：--marker) – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個 DescribeEngineDefaultParameters 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。
- MaxRecords (在 CLI 中：--max-records) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 MaxRecords 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

回應

包含成功呼叫 the section called “DescribeEngineDefaultParameters” 動作的結果。

- DBParameterGroupFamily – 字串，類型為：string (UTF-8 編碼的字串)。

  指定要套用引擎預設參數的資料庫參數群組系列名稱。
- Marker – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個 EngineDefaults 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。
- Parameters – 一個 參數 物件陣列。

  包含引擎預設參數的清單。

DescribeEngineDefaultClusterParameters (動作)

此 API 的 AWS CLI 名稱是：describe-engine-default-cluster-parameters。

傳回叢集資料庫引擎的預設引擎和系統參數資訊。

請求

- DBParameterGroupFamily (在 CLI 中：--db-parameter-group-family) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  要傳回其引擎參數資訊的資料庫叢集參數群組系列名稱。
- Filters (在 CLI 中：--filters) – 篩選條件 物件的陣列。

  目前不支援此參數。
- Marker (在 CLI 中：--marker) – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個 DescribeEngineDefaultClusterParameters 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。
- MaxRecords (在 CLI 中：--max-records) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 MaxRecords 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

限制條件：最小 20，最大 100。

回應

包含成功呼叫 the section called “DescribeEngineDefaultParameters” 動作的結果。

- DBParameterGroupFamily – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定要套用引擎預設參數的資料庫參數群組系列名稱。
- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 EngineDefaults 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- Parameters – 一個 參數 物件陣列。

  包含引擎預設參數的清單。

結構：

Parameter (結構)

指定參數。

欄位

- AllowedValues - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定參數值的有效範圍。
- ApplyMethod - 這是 ApplyMethod，類型為：`string` (UTF-8 編碼字串)。

  指出何時套用參數更新。
- ApplyType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定引擎限定的參數類型。
- DataType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定參數的有效資料類型。
- Description - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

提供參數的描述。

- IsModifiable - 這是布林值，類型為：boolean (布林值 (true 或 false))。

  指出是 (true) 否 (false) 可以修改參數。有些參數具有安全或操作上的隱含式，防止他們遭到變更。

- MinimumEngineVersion - 這是字串，類型為：string (UTF-8 編碼的字串)。

  可套用參數的最早引擎版本。

- ParameterName - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指定參數的名稱。

- ParameterValue - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指定參數的值。

- Source - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指出參數值的來源。

DBParameterGroup (結構)

包含 Amazon Neptune 資料庫參數群組的詳細資訊。

此資料類型在 the section called “DescribeDBParameterGroups” 動作中會用來作為回應元素。

欄位

- DBParameterGroupArn - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫參數群組的 Amazon Resource Name (ARN)。

- DBParameterGroupFamily - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫參數群組相容的資料庫參數群組系列名稱。

- DBParameterGroupName - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供資料庫參數群組的名稱。

- Description - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫參數群組的客戶指定描述。

DBParameterGroup 會用來做為以下項目的回應元素：

- CopyDBParameterGroup
- CreateDBParameterGroup

DBClusterParameterGroup (結構)

包含 Amazon Neptune 資料庫叢集參數群組的詳細資訊。

此資料類型在 the section called “DescribeDBClusterParameterGroups” 動作中會用來作為回應元素。

欄位

- DBClusterParameterGroupArn - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集參數群組的 Amazon Resource Name (ARN)。
- DBClusterParameterGroupName - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供資料庫叢集參數群組的名稱。
- DBParameterGroupFamily - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫叢集參數群組相容的資料庫參數群組系列名稱。
- Description - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供該客戶指定之此資料庫叢集參數群組的描述。

DBClusterParameterGroup 會用來做為以下項目的回應元素：

- CopyDBClusterParameterGroup
- CreateDBClusterParameterGroup

DBParameterGroupStatus (結構)

資料庫參數群組的狀態。

此資料類型在下列動作中會用來做為回應元素：

- the section called “CreateDBInstance”

- the section called “DeleteDBInstance”
- the section called “ModifyDBInstance”
- the section called “RebootDBInstance”

欄位

- DBParameterGroupName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫參數群組的名稱。
- ParameterApplyStatus - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  參數更新的狀態。

Neptune 子網路 API

動作:

- CreateDBSubnetGroup (動作)
- DeleteDBSubnetGroup (動作)
- ModifyDBSubnetGroup (動作)
- DescribeDBSubnetGroups (動作)

結構：

- Subnet (結構)
- DBSubnetGroup (結構)

CreateDBSubnetGroup (動作)

> 此 API 的 AWS CLI 名稱是：`create-db-subnet-group`。

建立新的資料庫子網路群組。資料庫子網路群組必須在 Amazon 區域中至少兩個可用區內包含一個子網路。

請求

- DBSubnetGroupDescription (在 CLI 中：`--db-subnet-group-description`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的描述。

- DBSubnetGroupName (在 CLI 中：`--db-subnet-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的名稱。此值會以小寫字母字串的形式儲存。

  限制條件：包含的內容絕不能超過 255 個字母、數字、句號、底線、空格或連字號。絕不能為預設值。

  範例：`mySubnetgroup`

- SubnetIds (在 CLI 中：`--subnet-ids`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的 EC2 子網路 ID。

- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給新資料庫子網路群組的標籤。

回應

包含 Amazon Neptune 子網路群組的詳細資訊。

此資料類型在 the section called “DescribeDBSubnetGroups” 動作中會用來作為回應元素。

- DBSubnetGroupArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的 Amazon Resource Name (ARN)。

- DBSubnetGroupDescription – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的描述。

- DBSubnetGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的名稱。

- SubnetGroupStatus – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的狀態。

- Subnets – 一個 子網 物件陣列。

包含 the section called “子網” 元素的清單。

- VpcId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的 VpcId。

錯誤

- DBSubnetGroupAlreadyExistsFault
- DBSubnetGroupQuotaExceededFault
- DBSubnetQuotaExceededFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DeleteDBSubnetGroup (動作)

此 API 的 AWS CLI 名稱是：`delete-db-subnet-group`。

刪除資料庫子網路群組。

> **Note**
>
> 指定的資料庫子網路群組不能和任何資料庫執行個體建立關聯。

請求

- DBSubnetGroupName (在 CLI 中：`--db-subnet-group-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要刪除的資料庫子網路群組名稱。

  > **Note**
  >
  > 您無法刪除預設子網路群組。

  約束:

限制條件：必須符合現有 DBSubnetGroup 的名稱。絕不能為預設值。

範例：mySubnetgroup

回應

- 無回應參數。

錯誤

- InvalidDBSubnetGroupStateFault
- InvalidDBSubnetStateFault
- DBSubnetGroupNotFoundFault

ModifyDBSubnetGroup (動作)

此 API 的 AWS CLI 名稱是：modify-db-subnet-group。

修改現有的資料庫子網路群組。資料庫子網路群組必須在 Amazon 區域中至少兩個可用區內包含一個子網路。

請求

- DBSubnetGroupDescription (在 CLI 中：--db-subnet-group-description) – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫子網路群組的描述。
- DBSubnetGroupName (在 CLI 中：--db-subnet-group-name) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  資料庫子網路群組的名稱。此值會以小寫字母字串的形式儲存。您無法修改預設子網路群組。

  限制條件：必須符合現有 DBSubnetGroup 的名稱。絕不能為預設值。

  範例：mySubnetgroup
- SubnetIds (在 CLI 中：--subnet-ids) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  資料庫子網路群組的 EC2 子網路 ID。

回應

包含 Amazon Neptune 子網路群組的詳細資訊。

此資料類型在 the section called “DescribeDBSubnetGroups” 動作中會用來作為回應元素。

- DBSubnetGroupArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的 Amazon Resource Name (ARN)。
- DBSubnetGroupDescription – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的描述。
- DBSubnetGroupName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的名稱。
- SubnetGroupStatus – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的狀態。
- Subnets – 一個 子網 物件陣列。

  包含 the section called “子網” 元素的清單。
- VpcId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的 VpcId。

錯誤

- DBSubnetGroupNotFoundFault
- DBSubnetQuotaExceededFault
- SubnetAlreadyInUse
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DescribeDBSubnetGroups (動作)

此 API 的 AWS CLI 名稱是：`describe-db-subnet-groups`。

傳回 DBSubnetGroup 描述的清單。若指定 DBSubnetGroupName，則清單只會包含指定 DBSubnetGroup 的描述。

如需 CIDR 範圍的概觀，請前往 Wikipedia Tutorial。

請求

- DBSubnetGroupName (在 CLI 中：`--db-subnet-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回其詳細資訊的資料庫子網路群組名稱。

- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。

- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 DescribeDBSubnetGroups 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

回應

- DBSubnetGroups – 一個 DBSubnetGroup 物件陣列。

  the section called “DBSubnetGroup” 執行個體的清單。

- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

錯誤

- DBSubnetGroupNotFoundFault

結構：

Subnet (結構)

指定子網路。

此資料類型在 the section called “DescribeDBSubnetGroups” 動作中會用來作為回應元素。

欄位

- SubnetAvailabilityZone - 這是一個 AvailabilityZone 物件。

  指定子網路所在的 EC2 可用區域。
- SubnetIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定子網路的識別符。
- SubnetStatus - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定子網路的狀態。

DBSubnetGroup (結構)

包含 Amazon Neptune 子網路群組的詳細資訊。

此資料類型在 the section called “DescribeDBSubnetGroups” 動作中會用來作為回應元素。

欄位

- DBSubnetGroupArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫子網路群組的 Amazon Resource Name (ARN)。
- DBSubnetGroupDescription - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的描述。
- DBSubnetGroupName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

資料庫子網路群組的名稱。

- SubnetGroupStatus - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的狀態。

- Subnets - 這是 子網 物件的陣列。

  包含 the section called “子網” 元素的清單。

- VpcId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫子網路群組的 VpcId。

`DBSubnetGroup` 會用來做為以下項目的回應元素：

- CreateDBSubnetGroup
- ModifyDBSubnetGroup

Neptune 快照 API

動作:

- CreateDBClusterSnapshot (動作)
- DeleteDBClusterSnapshot (動作)
- CopyDBClusterSnapshot (動作)
- ModifyDBClusterSnapshotAttribute (動作)
- RestoreDBClusterFromSnapshot (動作)
- RestoreDBClusterToPointInTime (動作)
- DescribeDBClusterSnapshots (動作)
- DescribeDBClusterSnapshotAttributes (動作)

結構：

- DBClusterSnapshot (結構)
- DBClusterSnapshotAttribute (結構)
- DBClusterSnapshotAttributesResult (結構)

CreateDBClusterSnapshot (動作)

此 API 的 AWS CLI 名稱是：`create-db-cluster-snapshot`。

建立資料庫叢集的快照。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要建立快照之資料庫叢集的識別符。此參數不區分大小寫。

  約束:

  - 必須符合現有 DBCluster 的識別碼。

  範例：`my-cluster1`
- DBClusterSnapshotIdentifier (在 CLI 中：`--db-cluster-snapshot-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集快照的識別符。此參數是以小寫字母字串的形式儲存。

  約束:

  - 必須包含 1 到 63 個字母、數字或連字號。
  - 第一個字元必須是字母。
  - 不能以連字號結尾或連續包含兩個連字號。

  範例：`my-cluster1-snapshot1`
- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給資料庫叢集快照的標籤。

回應

包含 Amazon Neptune 資料庫叢集快照的詳細資訊

此資料類型在 the section called “DescribeDBClusterSnapshots” 動作中會用來作為回應元素。

- AllocatedStorage – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

以 GiB 為單位指定分配的儲存體大小。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供 EC2 可用區域清單，您可以在此還原資料庫叢集快照中的執行個體。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集識別符，此即建立此資料庫叢集快照的資料庫叢集。

- DBClusterSnapshotArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集快照的 Amazon Resource Name (ARN)。

- DBClusterSnapshotIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集快照的識別碼。必須符合現有快照的識別碼。

  使用 `DBClusterSnapshotIdentifier` 還原資料庫叢集之後，您必須為資料庫叢集未來的任何更新指定相同的 `DBClusterSnapshotIdentifier`。當您為更新指定此屬性時，資料庫叢集不會再次從快照還原，且資料庫中的資料不會變更。

  不過，如果您未指定 `DBClusterSnapshotIdentifier`，則會建立空的資料庫叢集，並刪除原始資料庫叢集。如果您指定的屬性與先前的快照還原屬性不同，則會從 `DBClusterSnapshotIdentifier` 指定的快照還原資料庫叢集，並刪除原始資料庫叢集。

- Engine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫引擎的名稱。

- EngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的資料庫引擎版本。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集快照的 Amazon KMS 金鑰識別符。

- LicenseModel – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的授權模式資訊。
- PercentProgress – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定已傳輸的估計資料百分比。
- Port – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫叢集在快照時接聽的連接埠。
- SnapshotCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供快照拍攝的時間，使用供國際標準時間 (UTC)。
- SnapshotType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫叢集快照的類型。
- SourceDBClusterSnapshotArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  如果資料庫叢集快照複製自來源資料庫叢集快照，則為來源資料庫叢集快照的 Amazon Resource Name (ARN)，否則為空值。
- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集快照的狀態。
- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集快照是否加密。
- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與資料庫叢集快照相關聯的儲存類型。
- VpcId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供與資料庫叢集快照關聯的 VPC ID。

錯誤

- DBClusterSnapshotAlreadyExistsFault
- InvalidDBClusterStateFault
- DBClusterNotFoundFault

- SnapshotQuotaExceededFault
- InvalidDBClusterSnapshotStateFault

DeleteDBClusterSnapshot (動作)

此 API 的 AWS CLI 名稱是：`delete-db-cluster-snapshot`。

刪除資料庫叢集快照。如果正在複製快照，即會終止複製操作。

> **Note**
>
> 資料庫叢集快照必須為 `available` 狀態才能刪除。

請求

- DBClusterSnapshotIdentifier (在 CLI 中：`--db-cluster-snapshot-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要刪除之資料庫叢集快照的識別符。

  限制條件：必須是 `available` 狀態的現有資料庫叢集快照名稱。

回應

包含 Amazon Neptune 資料庫叢集快照的詳細資訊

此資料類型在 the section called “DescribeDBClusterSnapshots” 動作中會用來作為回應元素。

- AllocatedStorage – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  以 GiB 為單位指定分配的儲存體大小。
- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供 EC2 可用區域清單，您可以在此還原資料庫叢集快照中的執行個體。
- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- DBClusterIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集識別符，此即建立此資料庫叢集快照的資料庫叢集。
- DBClusterSnapshotArn – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集快照的 Amazon Resource Name (ARN)。
- DBClusterSnapshotIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫叢集快照的識別碼。必須符合現有快照的識別碼。

  使用 DBClusterSnapshotIdentifier 還原資料庫叢集之後，您必須為資料庫叢集未來的任何更新指定相同的 DBClusterSnapshotIdentifier。當您為更新指定此屬性時，資料庫叢集不會再次從快照還原，且資料庫中的資料不會變更。

  不過，如果您未指定 DBClusterSnapshotIdentifier，則會建立空的資料庫叢集，並刪除原始資料庫叢集。如果您指定的屬性與先前的快照還原屬性不同，則會從 DBClusterSnapshotIdentifier 指定的快照還原資料庫叢集，並刪除原始資料庫叢集。
- Engine – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫引擎的名稱。
- EngineVersion – 字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的資料庫引擎版本。
- IAMDatabaseAuthenticationEnabled – 布林值，類型為：boolean (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。
- KmsKeyId – 字串，類型為：string (UTF-8 編碼的字串)。

  若 StorageEncrypted 為 true，則為加密資料庫叢集快照的 Amazon KMS 金鑰識別符。
- LicenseModel – 字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的授權模式資訊。
- PercentProgress – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定已傳輸的估計資料百分比。
- Port – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫叢集在快照時接聽的連接埠。

- SnapshotCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供快照拍攝的時間，使用供國際標準時間 (UTC)。
- SnapshotType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供資料庫叢集快照的類型。
- SourceDBClusterSnapshotArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  如果資料庫叢集快照複製自來源資料庫叢集快照，則為來源資料庫叢集快照的 Amazon Resource Name (ARN)，否則為空值。
- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定此資料庫叢集快照的狀態。
- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集快照是否加密。
- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與資料庫叢集快照相關聯的儲存類型。
- VpcId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供與資料庫叢集快照關聯的 VPC ID。

錯誤

- InvalidDBClusterSnapshotStateFault
- DBClusterSnapshotNotFoundFault

CopyDBClusterSnapshot (動作)

此 API 的 AWS CLI 名稱是：`copy-db-cluster-snapshot`。

複製資料庫叢集的快照。

若要從共享的手動資料庫叢集快照複製資料庫叢集快照，`SourceDBClusterSnapshotIdentifier` 必須是共享資料庫叢集快照的 Amazon Resource Name (ARN)。

請求

- CopyTags (在 CLI 中:`--copy-tags`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  True，則將來源資料庫叢集快照的所有標籤複製到目標資料庫叢集快照，否則為 false。預設值為 false。
- KmsKeyId (在 CLI 中：`--kms-key-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  已加密資料庫叢集快照的 Amazon KMS 金鑰 ID。KMS 金鑰 ID 是 KMS 加密金鑰的 Amazon Resource Name (ARN)、KMS 金鑰識別符或 KMS 金鑰別名。

  如果您從 Amazon 帳戶複製加密的資料庫叢集快照，您可以指定 `KmsKeyId` 的值，以新的 KMS 加密金鑰來加密副本。如果不指定 `KmsKeyId` 的值，則會使用與來源資料庫叢集快照相同的 KMS 金鑰，加密資料庫叢集快照的副本。

  如果您複製另一個 Amazon 帳戶所共用的加密資料庫叢集快照，則必須指定 `KmsKeyId` 的值。

  KMS 加密金鑰專用於其建立時所在的 Amazon 區域，您無法將一個 Amazon 區域的加密金鑰用在另一個 Amazon 區域。

  複製時，您不能加密未加密的資料庫叢集快照。如果您嘗試複製未加密的資料庫叢集快照並指定 KmsKeyId 參數的值，會傳回錯誤。
- PreSignedUrl (在 CLI 中：`--pre-signed-url`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  目前不支援。
- SourceDBClusterSnapshotIdentifier (在 CLI 中：`--source-db-cluster-snapshot-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要複製之資料庫叢集快照的識別符。此參數不區分大小寫。

  約束:
  - 必須指定「可用」狀態下的有效系統快照。
  - 指定有效的資料庫快照識別符。

  範例：`my-cluster-snapshot1`
- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給新資料庫叢集快照副本的標籤。
- TargetDBClusterSnapshotIdentifier (在 CLI 中：`--target-db-cluster-snapshot-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

要從來源資料庫叢集快照建立之新資料庫叢集快照的識別符。此參數不區分大小寫。

約束:

- 必須包含 1 到 63 個字母、數字或連字號。
- 第一個字元必須是字母。
- 不能以連字號結尾或連續包含兩個連字號。

範例：`my-cluster-snapshot2`

回應

包含 Amazon Neptune 資料庫叢集快照的詳細資訊

此資料類型在 the section called “DescribeDBClusterSnapshots” 動作中會用來作為回應元素。

- AllocatedStorage – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  以 GiB 為單位指定分配的儲存體大小。
- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供 EC2 可用區域清單，您可以在此還原資料庫叢集快照中的執行個體。
- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。
- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集識別符，此即建立此資料庫叢集快照的資料庫叢集。
- DBClusterSnapshotArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集快照的 Amazon Resource Name (ARN)。
- DBClusterSnapshotIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集快照的識別碼。必須符合現有快照的識別碼。

  使用 `DBClusterSnapshotIdentifier` 還原資料庫叢集之後，您必須為資料庫叢集未來的任何更新指定相同的 `DBClusterSnapshotIdentifier`。當您為更新指定此屬性時，資料庫叢集不會再次從快照還原，且資料庫中的資料不會變更。

不過，如果您未指定 DBClusterSnapshotIdentifier，則會建立空的資料庫叢集，並刪除原始資料庫叢集。如果您指定的屬性與先前的快照還原屬性不同，則會從 DBClusterSnapshotIdentifier 指定的快照還原資料庫叢集，並刪除原始資料庫叢集。

- Engine – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫引擎的名稱。

- EngineVersion – 字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的資料庫引擎版本。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：boolean (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- KmsKeyId – 字串，類型為：string (UTF-8 編碼的字串)。

  若 StorageEncrypted 為 true，則為加密資料庫叢集快照的 Amazon KMS 金鑰識別符。

- LicenseModel – 字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的授權模式資訊。

- PercentProgress – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定已傳輸的估計資料百分比。

- Port – 整數，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫叢集在快照時接聽的連接埠。

- SnapshotCreateTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供快照拍攝的時間，使用供國際標準時間 (UTC)。

- SnapshotType – 字串，類型為：string (UTF-8 編碼的字串)。

  提供資料庫叢集快照的類型。

- SourceDBClusterSnapshotArn – 字串，類型為：string (UTF-8 編碼的字串)。

  如果資料庫叢集快照複製自來源資料庫叢集快照，則為來源資料庫叢集快照的 Amazon Resource Name (ARN)，否則為空值。

- Status – 字串，類型為：string (UTF-8 編碼的字串)。

指定此資料庫叢集快照的狀態。

- StorageEncrypted – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集快照是否加密。

- StorageType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與資料庫叢集快照相關聯的儲存類型。

- VpcId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供與資料庫叢集快照關聯的 VPC ID。

錯誤

- DBClusterSnapshotAlreadyExistsFault
- DBClusterSnapshotNotFoundFault
- InvalidDBClusterStateFault
- InvalidDBClusterSnapshotStateFault
- SnapshotQuotaExceededFault
- KMSKeyNotAccessibleFault

ModifyDBClusterSnapshotAttribute (動作)

此 API 的 AWS CLI 名稱是：`modify-db-cluster-snapshot-attribute`。

在手動資料庫叢集快照中新增或移除屬性和值。

若要與其他 Amazon 帳戶共用手動資料庫叢集快照，請指定 `restore` 做為 `AttributeName`，並使用 `ValuesToAdd` 參數新增 Amazon 帳戶 ID 清單，這些帳戶有權還原手動資料庫叢集快照。使用值 `all` 讓手動資料庫叢集快照成為公有的，這表示所有 Amazon 帳戶都可以複製或還原它。如有不想讓所有 Amazon 帳戶取得的私人資訊，請不要新增任何包含此資訊之手動資料庫叢集快照的 `all` 值。如果手動資料庫叢集快照已加密，它可以共用，但只能透過 `ValuesToAdd` 參數指定授權的 Amazon 帳戶 ID 清單。本例中無法使用 `all` 做為該參數的值。

若要檢視哪些 Amazon 帳戶有權複製或還原手動資料庫叢集快照，或手動資料庫叢集快照為公有或私有，請使用 the section called “DescribeDBClusterSnapshotAttributes” API 動作。

請求

- AttributeName (在 CLI 中：`--attribute-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要修改之資料庫叢集快照屬性的名稱。

  若要管理其他 Amazon 帳戶的授權，以複製或還原手動資料庫叢集快照，請將此值設定為 `restore`。
- DBClusterSnapshotIdentifier (在 CLI 中：`--db-cluster-snapshot-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要修改屬性之資料庫叢集快照的識別符。
- ValuesToAdd (在 CLI 中：`--values-to-add`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集快照屬性清單，新增 `AttributeName` 指定的屬性。

  若要授權其他 Amazon 帳戶以複製或還原手動資料庫叢集快照，請設定此清單包含一或多個 Amazon 帳戶 ID，或設為 `all` 讓任何 Amazon 帳戶都能還原手動資料庫叢集快照。如有不想讓所有 Amazon 帳戶取得的私人資訊，請不要新增任何包含此資訊之手動資料庫叢集快照的 `all` 值。
- ValuesToRemove (在 CLI 中：`--values-to-remove`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集快照屬性清單，移除 `AttributeName` 指定的屬性。

  若要移除其他 Amazon 帳戶的授權，使無法複製或還原手動資料庫叢集快照，請設定此清單包含一或多個 Amazon 帳戶識別符，或設為 `all` 移除任何 Amazon 帳戶的授權，使不能複製或還原資料庫叢集快照。如果指定 `all`，則帳戶 ID 明確新增至 `restore` 屬性的 Amazon 帳戶，仍然可以複製或還原手動資料庫叢集快照。

回應

包含成功呼叫 the section called “DescribeDBClusterSnapshotAttributes” API 動作的結果。

手動資料庫叢集快照屬性是用於授權其他 Amazon 帳戶以複製或還原手動資料庫叢集快照。如需詳細資訊，請參閱 the section called “ModifyDBClusterSnapshotAttribute” API 動作。

- DBClusterSnapshotAttributes – 一個 DBClusterSnapshotAttribute 物件陣列。

  手動資料庫叢集快照的屬性和值清單。
- DBClusterSnapshotIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

屬性套用目標之手動資料庫叢集快照的識別符。

錯誤

- DBClusterSnapshotNotFoundFault
- InvalidDBClusterSnapshotStateFault
- SharedSnapshotQuotaExceededFault

RestoreDBClusterFromSnapshot (動作)

此 API 的 AWS CLI 名稱是：`restore-db-cluster-from-snapshot`。

從資料庫快照或資料庫叢集快照建立新的資料庫叢集。

如已指定資料庫快照，則會從具有預設組態和預設安全群組的來源資料庫快照建立目標資料庫叢集。

如已指定資料庫叢集快照，則會從與原始來源資料庫叢集有相同組態之來源資料庫叢集還原點建立目標資料庫叢集，但新的資料庫叢集使用預設安全群組建立時除外。

請求

- AvailabilityZones (在 CLI 中：`--availability-zones`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供 EC2 可用區域清單，已還原資料庫叢集中的執行個體可在這些可用區域中建立。
- CopyTagsToSnapshot (在 CLI 中:`--copy-tags-to-snapshot`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之已還原資料庫叢集的任何快照。
- DatabaseName (在 CLI 中：`--database-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  不支援。
- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要從資料庫快照或資料庫叢集快照建立之資料庫叢集的名稱。此參數沒有大小寫之分。

  約束:

- 必須包含 1 到 63 個字母、數字或連字號
- 第一個字元必須是字母
- 不能以連字號結尾或連續包含兩個連字號

範例：`my-snapshot-id`

- DBClusterParameterGroupName (在 CLI 中：`--db-cluster-parameter-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要與新資料庫叢集建立關聯的資料庫叢集參數群組名稱。

  約束:

  - 若有提供，則必須符合現有 DBClusterParameterGroup 的名稱。
- DBSubnetGroupName (在 CLI 中：`--db-subnet-group-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要用於新資料庫叢集的資料庫子網路群組名稱。

  限制條件：若有提供，則必須符合現有 DBSubnetGroup 的名稱。

  範例：`mySubnetgroup`
- DeletionProtection (在 CLI 中:`--deletion-protection`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護的值。啟用刪除保護時無法刪除資料庫。根據預設，刪除保護是停用的。
- EnableCloudwatchLogsExports (在 CLI 中：`--enable-cloudwatch-logs-exports`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  還原的資料庫叢集將匯出到 Amazon CloudWatch Logs 的日誌清單。
- EnableIAMDatabaseAuthentication (在 CLI 中:`--enable-iam-database-authentication`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  設為 True 以啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的對應，否則為 false。

  預設：`false`
- Engine (在 CLI 中：`--engine`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

要用於新資料庫叢集的資料庫引擎。

預設值：與來源相同

限制：必須與來源引擎相容

- EngineVersion (在 CLI 中：`--engine-version`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要用於新資料庫叢集的資料庫引擎版本。

- KmsKeyId (在 CLI 中：`--kms-key-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  從資料庫快照或資料庫叢集快照還原加密資料庫叢集時要使用的 Amazon KMS 金鑰識別符。

  KMS 金鑰識別符是 KMS 加密金鑰的 Amazon Resource Name (ARN)。如果正在還原的資料庫叢集，與擁有用於加密新資料庫叢集之 KMS 加密金鑰有相同的 Amazon 帳戶，則您可使用 KMS 金鑰別名，而非 KMS 加密金鑰的 ARN。

  如不指定 `KmsKeyId` 參數值，會發生以下情況：

  - 如已加密 `SnapshotIdentifier` 中的資料庫快照或資料庫叢集快照，則會使用用來加密資料庫快照或資料庫叢集快照的 KMS 金鑰，加密還原的資料庫叢集。
  - 如未加密 `SnapshotIdentifier` 中的資料庫快照或資料庫叢集快照，則不加密還原的資料庫叢集。

- Port (在 CLI 中：`--port`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  新資料庫叢集接受連線的連接埠號碼。

  限制條件：值必須為 `1150-65535`

  預設值：與原始資料庫叢集相同的連接埠。

- ServerlessV2ScalingConfiguration (在 CLI 中：`--serverless-v2-scaling-configuration`) – ServerlessV2ScalingConfiguration 物件。

  包含 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- SnapshotIdentifier (在 CLI 中：`--snapshot-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  還原來源的資料庫快照或資料庫叢集快照識別碼。

您可以使用名稱或 Amazon Resource Name (ARN) 指定資料庫叢集快照。不過，您只能使用 ARN 指定資料庫快照。

約束:

- 必須符合現有快照的識別碼。

- StorageType (在 CLI 中：`--storage-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定要和資料庫叢集建立關聯的儲存類型。

  有效值：`standard`、`iopt1`

  預設：`standard`

- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要指派給已還原資料庫叢集的標籤。

- VpcSecurityGroupIds (在 CLI 中：`--vpc-security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新資料庫叢集將歸屬之 VPC 安全群組清單。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。

- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。

- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。

- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：string (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：string (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：audit (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：string (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：string (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：`string` (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：`[A-Za-z][0-9A-Za-z-:._]*`。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 `iopt1` 儲存類型。

- KmsKeyId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若 `StorageEncrypted` 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 `ModifyDBCluster` 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：`string` (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：`string` (UTF-8 編碼的字串)。

指定每日的時間範圍，若有啟用自動化備份 (由 BackupRetentionPeriod 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：string (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

  若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：string (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：string (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

- **`standard`** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
- **`iopt1`** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

  僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。
- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterAlreadyExistsFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterSnapshotNotFoundFault
- InsufficientDBClusterCapacityFault
- InsufficientStorageClusterCapacityFault
- InvalidDBSnapshotStateFault
- InvalidDBClusterSnapshotStateFault
- StorageQuotaExceededFault
- InvalidVPCNetworkStateFault
- InvalidRestoreFault
- DBSubnetGroupNotFoundFault
- InvalidSubnet
- OptionGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterParameterGroupNotFoundFault

RestoreDBClusterToPointInTime (動作)

此 API 的 AWS CLI 名稱是：`restore-db-cluster-to-point-in-time`。

將資料庫叢集還原到任意時間點。使用者可以還原到 LatestRestorableTime 之前的任意時間點，最多 BackupRetentionPeriod 天。目標資料庫叢集使用與原始資料庫叢集有相同組態之來源資料庫叢集建立，但新的資料庫叢集使用預設資料庫安全群組建立時除外。

> **Note**
>
> 這個動作只會還原資料庫叢集，不會還原該資料庫叢集的資料庫執行個體。您必須呼叫 the section called “CreateDBInstance” 動作，為還原的資料庫叢集建立資料庫執行個體，並且在 DBClusterIdentifier 中指定所還原資料庫叢集的識別符。只有在 RestoreDBClusterToPointInTime 動作完成後，且資料庫叢集為可用時，您才能建立資料庫執行個體。

請求

- DBClusterIdentifier (在 CLI 中：--db-cluster-identifier) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  要建立之新資料庫叢集的名稱。

  約束:

  - 必須包含 1 到 63 個字母、數字或連字號
  - 第一個字元必須是字母
  - 不能以連字號結尾或連續包含兩個連字號
- DBClusterParameterGroupName (在 CLI 中：--db-cluster-parameter-group-name) – 字串，類型為：string (UTF-8 編碼的字串)。

  要與新資料庫叢集建立關聯的資料庫叢集參數群組名稱。

  約束:

  - 若有提供，則必須符合現有 DBClusterParameterGroup 的名稱。
- DBSubnetGroupName (在 CLI 中：--db-subnet-group-name) – 字串，類型為：string (UTF-8 編碼的字串)。

  要用於新資料庫叢集的資料庫子網路群組名稱。

  限制條件：若有提供，則必須符合現有 DBSubnetGroup 的名稱。

範例：mySubnetgroup

- DeletionProtection (在 CLI 中:--deletion-protection) – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護的值。啟用刪除保護時無法刪除資料庫。根據預設，刪除保護是停用的。

- EnableCloudwatchLogsExports (在 CLI 中：--enable-cloudwatch-logs-exports) – 字串，類型為：string (UTF-8 編碼的字串)。

  還原的資料庫叢集將匯出到 CloudWatch Logs 的日誌清單。

- EnableIAMDatabaseAuthentication (在 CLI 中:--enable-iam-database-authentication) – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  設為 True 以啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的對應，否則為 false。

  預設：false

- KmsKeyId (在 CLI 中：--kms-key-id) – 字串，類型為：string (UTF-8 編碼的字串)。

  從加密資料庫叢集還原加密資料庫叢集時要使用的 Amazon KMS 金鑰識別符。

  KMS 金鑰識別符是 KMS 加密金鑰的 Amazon Resource Name (ARN)。如果正在還原的資料庫叢集，與擁有用於加密新資料庫叢集之 KMS 加密金鑰有相同的 Amazon 帳戶，則您可使用 KMS 金鑰別名，而非 KMS 加密金鑰的 ARN。

  您可還原至新的資料庫叢集，並使用和加密來源資料庫叢集不同的 KMS 金鑰，加密新的資料庫叢集。新資料庫叢集使用 KmsKeyId 參數識別的 KMS 金鑰加密。

  如不指定 KmsKeyId 參數值，會發生以下情況：

  - 如已加密資料庫叢集，則已還原的資料庫叢集會使用加密來源資料庫叢集的 KMS 金鑰加密。
  - 如未加密資料庫叢集，則不會加密已還原的資料庫叢集。

  如果 DBClusterIdentifier 是指未加密的資料庫叢集，則拒絕還原請求。

- Port (在 CLI 中：--port) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  新資料庫叢集接受連線的連接埠號碼。

  限制條件：值必須為 1150-65535

預設值：與原始資料庫叢集相同的連接埠。

- RestoreToTime (在 CLI 中：`--restore-to-time`) – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  將資料庫叢集還原至此日期和時間。

  有效值：值必須為國際標準時間 (UTC) 格式的時間

  約束:

  - 必須在資料庫執行個體最近一次可還原時間之前。
  - 如未提供 `UseLatestRestorableTime` 參數，則必須指定。
  - 如果 `UseLatestRestorableTime` 參數為 true，則無法指定
  - 如果 `RestoreType` 參數為 `copy-on-write`，則無法指定

  範例：`2015-03-07T23:45:00Z`

- RestoreType (在 CLI 中：`--restore-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要執行的還原類型。您可以指定下列其中一個值：

  - `full-copy` - 新資料庫叢集將還原為來源資料庫叢集的完整複本。
  - `copy-on-write` - 新資料庫叢集將還原為來源資料庫叢集的複製品。

  如果您不指定 `RestoreType` 值，則新資料庫叢集會還原為來源資料庫叢集的完整複本。

- ServerlessV2ScalingConfiguration (在 CLI 中：`--serverless-v2-scaling-configuration`) – ServerlessV2ScalingConfiguration 物件。

  包含 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- SourceDBClusterIdentifier (在 CLI 中：`--source-db-cluster-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  還原來源之來源資料庫叢集的識別符。

  約束:

  - 必須符合現有 DBCluster 的識別碼。

- StorageType (在 CLI 中：`--storage-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

指定要和資料庫叢集建立關聯的儲存類型。

有效值：`standard`、`iopt1`

預設：`standard`

- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要套用到已還原資料庫叢集的標籤。

- UseLatestRestorableTime (在 CLI 中:`--use-latest-restorable-time`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  值設定為 `true` 可將資料庫叢集還原到最近可還原的備份時間，否則為 `false`。

  預設：`false`

  限制條件：如果提供 `RestoreToTime` 參數，則無法指定。

- VpcSecurityGroupIds (在 CLI 中：`--vpc-security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新資料庫叢集所屬之 VPC 安全群組清單。

回應

包含 Amazon Neptune 資料庫叢集的詳細資訊。

此資料類型會在 the section called “DescribeDBClusters” 動作中用作回應元素。

- AllocatedStorage – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  `AllocatedStorage` 一律會傳回 1，因為 Neptune 資料庫叢集儲存體大小並非固定，而是會視需要自動調整。

- AssociatedRoles – 一個 DBClusterRole 物件陣列。

  提供與資料庫叢集相關聯的 Amazon Identity and Access Management (IAM) 角色清單。與資料庫叢集相關聯的 IAM 角色，會授予資料庫叢集代您存取其他 Amazon 服務的許可。

- AutomaticRestartTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  資料庫叢集將自動重新啟動的時間。

- AvailabilityZones – 字串，類型為：`string` (UTF-8 編碼的字串)。

  提供可在其中建立資料庫叢集內執行個體的 EC2 可用區域清單。

- BacktrackConsumedChangeRecords - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BacktrackWindow - LongOptional，類型為：`long` (帶正負號的 64 位元整數)。

  Neptune 不提供支援。

- BackupRetentionPeriod – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  指定保留自動資料庫快照的天數。

- Capacity – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  Neptune 不提供支援。

- CloneGroupId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  識別與資料庫叢集建立關聯的複製群組。

- ClusterCreateTime – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。

- CopyTagsToSnapshot – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 *`true`*，則會將標籤複製到所建立之資料庫叢集的任何快照。

- CrossAccountClone – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，則可跨帳戶複製資料庫叢集。

- DatabaseName – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含此資料庫叢集初始資料庫的名稱，該資料庫是在建立時提供的 (若建立資料庫叢集時有指定的話)。在資料庫叢集使用期間，會傳回此名稱。

- DBClusterArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon Resource Name (ARN)。

- DBClusterIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  包含使用者提供的資料庫叢集識別碼。此識別碼為可識別資料庫叢集的唯一金鑰。

- DBClusterMembers – 一個 DBClusterMember 物件陣列。

  提供組成資料庫叢集的執行個體清單。

- DBClusterParameterGroup – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集參數群組名稱。

- DbClusterResourceId – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的 Amazon 區域唯一、固定識別符。此識別符可在每次存取資料庫叢集的 Amazon KMS 金鑰時，於 Amazon CloudTrail 日誌項目中找到。

- DBSubnetGroup – 字串，類型為：string (UTF-8 編碼的字串)。

  指定與資料庫叢集建立關聯子網路群組的資訊，包括名稱、描述，以及子網路群組中的子網路。

- DeletionProtection – BooleanOptional，類型為：boolean (布林值 (true 或 false))。

  指示資料庫叢集是否已啟用刪除保護。啟用刪除保護時無法刪除資料庫。

- EarliestBacktrackTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  Neptune 不提供支援。

- EarliestRestorableTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最早時間。

- EnabledCloudwatchLogsExports – 字串，類型為：string (UTF-8 編碼的字串)。

  此資料庫叢集已設為匯出至 CloudWatch Logs 的日誌類型清單。有效的日誌類型為：audit (用來將稽核日誌發佈至 CloudWatch) 和 slowquery (用來將慢查詢日誌發佈至 CloudWatch)。請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。

- Endpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫叢集主要執行個體的連線端點。

- Engine – 字串，類型為：string (UTF-8 編碼的字串)。

  提供要用於此資料庫叢集的資料庫引擎名稱。

- EngineVersion – 字串，類型為：string (UTF-8 編碼的字串)。

  指示資料庫引擎版本。

- GlobalClusterIdentifier – GlobalClusterIdentifier，類型為：string (UTF-8 編碼的字串)，不小於 1 或大於 255 ?st?s，符合此規則運算式：[A-Za-z][0-9A-Za-z-:._]*。

  包含使用者提供的全球資料庫叢集識別符。此識別符為可識別全球資料庫的唯一金鑰。

- HostedZoneId – 字串，類型為：string (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。

- IAMDatabaseAuthenticationEnabled – 布林值，類型為：boolean (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- IOOptimizedNextAllowedModificationTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  下次您可以修改資料庫叢集以使用 iopt1 儲存類型。

- KmsKeyId – 字串，類型為：string (UTF-8 編碼的字串)。

  若 StorageEncrypted 為 true，則為加密資料庫叢集的 Amazon KMS 金鑰識別碼。

- LatestRestorableTime – TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定可透過 point-in-time 還原還原資料庫的最新時間。

- MultiAZ – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定資料庫叢集是否在多個可用區域中擁有執行個體。

- PendingModifiedValues – ClusterPendingModifiedValues 物件。

  此資料類型會在 ModifyDBCluster 操作中用作回應元素，並包含將在下一個維護時段期間套用的變更。

- PercentProgress – 字串，類型為：string (UTF-8 編碼的字串)。

  指定操作的進度 (百分比)。

- Port – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

- PreferredBackupWindow – 字串，類型為：string (UTF-8 編碼的字串)。

指定每日的時間範圍，若有啟用自動化備份 (由 BackupRetentionPeriod 決定)，則自動化備份會在此期間建立。

- PreferredMaintenanceWindow – 字串，類型為：string (UTF-8 編碼的字串)。

  指定每週可能進行系統維護的時段，以世界協調時間 (UTC) 表示。

- ReaderEndpoint – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集的讀取器端點。資料庫叢集的讀取器端點，用來針對資料庫叢集中可用僅供讀取複本的連線進行負載平衡。當用戶端向讀取器端點請求新連線時，Neptune 會在資料庫叢集中的僅供讀取複本間分散連線請求。此功能有助於在您資料庫叢集中的多個僅供讀取複本間平衡您的讀取工作負載。

  若發生容錯移轉，且您連線的僅供讀取複本提升為主要執行個體，則會中斷您的連線。若要繼續將您的讀取工作負載傳送到叢集內的其他僅供讀取複本，您可以接著重新連線到讀取器端點。

- ReadReplicaIdentifiers – 字串，類型為：string (UTF-8 編碼的字串)。

  包含與此資料庫叢集相關聯的一或多個僅供讀取複本識別符。

- ReplicationSourceIdentifier – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ReplicationType – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune 不提供支援。

- ServerlessV2ScalingConfiguration – ServerlessV2ScalingConfigurationInfo 物件。

  顯示 Neptune Serverless 資料庫叢集的擴展組態。

  如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

- Status – 字串，類型為：string (UTF-8 編碼的字串)。

  指定此資料庫叢集的目前狀態。

- StorageEncrypted – 布林值，類型為：boolean (布林值 (true 或 false))。

  指定是否要加密資料庫叢集。

- StorageType – 字串，類型為：string (UTF-8 編碼的字串)。

  資料庫叢集使用的儲存類型。

  有效值：

- **standard** – (預設值) 針對具有中至低 I/O 使用量的應用程式提供具成本效益的資料庫儲存體。
- **iopt1** – 啟用 I/O 優化儲存，旨在符合 I/O 密集型圖形工作負載的需求，而這些工作負載則需要可預測的定價、低 I/O 延遲和一致的 I/O 輸送量。

  僅引擎版本 1.3.0.0 開始提供 Neptune I/O 優化儲存。
- VpcSecurityGroups – 一個 VpcSecurityGroupMembership 物件陣列。

  提供資料庫叢集歸屬的 VPC 安全群組清單。

錯誤

- DBClusterAlreadyExistsFault
- DBClusterNotFoundFault
- DBClusterQuotaExceededFault
- DBClusterSnapshotNotFoundFault
- DBSubnetGroupNotFoundFault
- InsufficientDBClusterCapacityFault
- InsufficientStorageClusterCapacityFault
- InvalidDBClusterSnapshotStateFault
- InvalidDBClusterStateFault
- InvalidDBSnapshotStateFault
- InvalidRestoreFault
- InvalidSubnet
- InvalidVPCNetworkStateFault
- KMSKeyNotAccessibleFault
- OptionGroupNotFoundFault
- StorageQuotaExceededFault
- DBClusterParameterGroupNotFoundFault

DescribeDBClusterSnapshots (動作)

此 API 的 AWS CLI 名稱是：`describe-db-cluster-snapshots`。

傳回資料庫叢集快照的相關資訊。此 API 動作支援分頁。

請求

- DBClusterIdentifier (在 CLI 中：`--db-cluster-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  擷取資料庫叢集快照清單的資料庫叢集 ID。此參數無法與 `DBClusterSnapshotIdentifier` 參數一起使用。此參數不區分大小寫。

  約束:

  - 如果提供，必須符合現有 DBCluster 的識別符。
- DBClusterSnapshotIdentifier (在 CLI 中：`--db-cluster-snapshot-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  描述特定的資料庫叢集快照識別符。此參數無法與 `DBClusterIdentifier` 參數一起使用。此值會以小寫字母字串的形式儲存。

  約束:

  - 如果提供，必須符合現有 DBClusterSnapshot 的識別符。
  - 如果這是用於自動快照的識別符，也必須指定 `SnapshotType` 參數。
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- IncludePublic (在 CLI 中:`--include-public`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  True 則包含可由任何 Amazon 帳戶複製或還原的公有手動資料庫叢集快照，否則為 false。預設值為 `false`。預設值為 false。

  您可以使用 the section called “ModifyDBClusterSnapshotAttribute” API 動作，以公有狀態共享手動的資料庫叢集快照。
- IncludeShared (在 CLI 中:`--include-shared`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  True 則包含來自其他 Amazon 帳戶的共用手動資料庫叢集快照，且此 Amazon 帳戶具有複製或還原的許可，否則為 false。預設值為 `false`。

  您可以使用 the section called “ModifyDBClusterSnapshotAttribute” API 動作，授予 Amazon 帳戶還原來自其他 Amazon 帳戶的手動資料庫叢集快照的許可。

- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 `DescribeDBClusterSnapshots` 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

- SnapshotType (在 CLI 中：`--snapshot-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回的資料庫叢集快照類型。您可以指定下列其中一個值：

  - `automated` - 傳回 Amazon Neptune 針對我的 Amazon 帳戶自動建立的所有資料庫叢集快照。
  - `manual` - 傳回我的 Amazon 帳戶建立的所有資料庫叢集快照。
  - `shared` - 傳回與我的 Amazon 帳戶共用的所有手動資料庫叢集快照。
  - `public` - 傳回已標記為公有的所有資料庫叢集快照。

  如不指定 `SnapshotType` 值，則會傳回自動和手動的資料庫叢集快照。您可以將 `IncludeShared` 參數設成 `true`，納入具有這些結果的共享資料庫叢集快照。您可以將 `IncludePublic` 參數設成 `true`，納入具有這些結果的公有資料庫叢集快照。

  `IncludeShared` 和 `IncludePublic` 參數不適用於值為 `SnapshotType` 的 `manual` 或 `automated`。當 `SnapshotType` 設為 `shared` 時，不適用 `IncludePublic` 參數。當 `SnapshotType` 設為 `public` 時，不適用 `IncludeShared` 參數。

回應

- DBClusterSnapshots – 一個 DBClusterSnapshot 物件陣列。

  為使用者提供資料庫叢集快照的清單。

- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 the section called “DescribeDBClusterSnapshots” 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

錯誤

- DBClusterSnapshotNotFoundFault

DescribeDBClusterSnapshotAttributes (動作)

此 API 的 AWS CLI 名稱是：`describe-db-cluster-snapshot-attributes`。

傳回手動資料庫叢集快照之資料庫叢集快照屬性名稱和值的清單。

與其他 Amazon 帳戶共用快照時，`DescribeDBClusterSnapshotAttributes` 會傳回 `restore` 屬性和 Amazon 帳戶的 ID 清單，此 Amazon 帳戶有權複製或還原手動資料庫叢集快照。如果 `all` 包含在 `restore` 屬性的值清單中，則手動資料庫叢集快照為公有，且所有 Amazon 帳戶都可以複製或還原它。

若要新增或移除 Amazon 帳戶複製或還原手動資料庫叢集快照的許可，或使手動資料庫叢集快照成為公有或私有的，請使用 the section called “ModifyDBClusterSnapshotAttribute” API 動作。

請求

- DBClusterSnapshotIdentifier (在 CLI 中：`--db-cluster-snapshot-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要描述屬性之資料庫叢集快照的識別符。

回應

包含成功呼叫 the section called “DescribeDBClusterSnapshotAttributes” API 動作的結果。

手動資料庫叢集快照屬性是用於授權其他 Amazon 帳戶以複製或還原手動資料庫叢集快照。如需詳細資訊，請參閱 the section called “ModifyDBClusterSnapshotAttribute” API 動作。

- DBClusterSnapshotAttributes – 一個 DBClusterSnapshotAttribute 物件陣列。

  手動資料庫叢集快照的屬性和值清單。

- DBClusterSnapshotIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  屬性套用目標之手動資料庫叢集快照的識別符。

錯誤

- DBClusterSnapshotNotFoundFault

結構：

DBClusterSnapshot (結構)

包含 Amazon Neptune 資料庫叢集快照的詳細資訊

此資料類型在 the section called “DescribeDBClusterSnapshots” 動作中會用來作為回應元素。

欄位

- AllocatedStorage - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  以 GiB 為單位指定分配的儲存體大小。
- AvailabilityZones - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供 EC2 可用區域清單，您可以在此還原資料庫叢集快照中的執行個體。
- ClusterCreateTime - 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定建立資料庫叢集的時間，以國際標準時間 (UTC) 表示。
- DBClusterIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集的資料庫叢集識別符，此即建立此資料庫叢集快照的資料庫叢集。
- DBClusterSnapshotArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫叢集快照的 Amazon Resource Name (ARN)。
- DBClusterSnapshotIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫叢集快照的識別碼。必須符合現有快照的識別碼。

  使用 `DBClusterSnapshotIdentifier` 還原資料庫叢集之後，您必須為資料庫叢集未來的任何更新指定相同的 `DBClusterSnapshotIdentifier`。當您為更新指定此屬性時，資料庫叢集不會再次從快照還原，且資料庫中的資料不會變更。

不過，如果您未指定 DBClusterSnapshotIdentifier，則會建立空的資料庫叢集，並刪除原始資料庫叢集。如果您指定的屬性與先前的快照還原屬性不同，則會從 DBClusterSnapshotIdentifier 指定的快照還原資料庫叢集，並刪除原始資料庫叢集。

- Engine - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指定資料庫引擎的名稱。

- EngineVersion - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的資料庫引擎版本。

- IAMDatabaseAuthenticationEnabled - 這是布林值，類型為：boolean (布林值 (true 或 false))。

  若有啟用 Amazon Identity and Access Management (IAM) 帳戶對資料庫帳戶的映射，則為 true，否則為 false。

- KmsKeyId - 這是字串，類型為：string (UTF-8 編碼的字串)。

  若 StorageEncrypted 為 true，則為加密資料庫叢集快照的 Amazon KMS 金鑰識別符。

- LicenseModel - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供此資料庫叢集快照的授權模式資訊。

- PercentProgress - 這是整數，類型為：integer (帶正負號的 32 位元整數)。

  指定已傳輸的估計資料百分比。

- Port - 這是整數，類型為：integer (帶正負號的 32 位元整數)。

  指定資料庫叢集在快照時接聽的連接埠。

- SnapshotCreateTime - 這是 TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  提供快照拍攝的時間，使用供國際標準時間 (UTC)。

- SnapshotType - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供資料庫叢集快照的類型。

- SourceDBClusterSnapshotArn - 這是字串，類型為：string (UTF-8 編碼的字串)。

  如果資料庫叢集快照複製自來源資料庫叢集快照，則為來源資料庫叢集快照的 Amazon Resource Name (ARN)，否則為空值。

- Status - 這是字串，類型為：string (UTF-8 編碼的字串)。

指定此資料庫叢集快照的狀態。

- StorageEncrypted - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指定資料庫叢集快照是否加密。

- StorageType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  與資料庫叢集快照相關聯的儲存類型。

- VpcId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供與資料庫叢集快照關聯的 VPC ID。

`DBClusterSnapshot` 會用來做為以下項目的回應元素：

- CreateDBClusterSnapshot
- CopyDBClusterSnapshot
- DeleteDBClusterSnapshot

DBClusterSnapshotAttribute (結構)

包含手動資料庫叢集快照屬性的名稱和值。

手動資料庫叢集快照屬性是用於授權其他 Amazon 帳戶以還原手動資料庫叢集快照。如需詳細資訊，請參閱 the section called “ModifyDBClusterSnapshotAttribute” API 動作。

欄位

- AttributeName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  手動資料庫叢集快照屬性的名稱。

  名為 `restore` 的屬性是指具有複製或還原手動資料庫叢集快照許可的 Amazon 帳戶清單。如需詳細資訊，請參閱 the section called “ModifyDBClusterSnapshotAttribute” API 動作。

- AttributeValues - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  手動資料庫叢集快照屬性的值。

如果 `AttributeName` 欄位設為 `restore`，則此元素會傳回 Amazon 帳戶 ID 的清單，這些帳戶有權複製或還原手動資料庫叢集快照。如果清單中有 `all` 值，則手動資料庫叢集快照為公有，且可供任何 Amazon 帳戶複製或還原。

DBClusterSnapshotAttributesResult (結構)

包含成功呼叫 the section called “DescribeDBClusterSnapshotAttributes” API 動作的結果。

手動資料庫叢集快照屬性是用於授權其他 Amazon 帳戶以複製或還原手動資料庫叢集快照。如需詳細資訊，請參閱 the section called “ModifyDBClusterSnapshotAttribute” API 動作。

欄位

- DBClusterSnapshotAttributes - 這是 DBClusterSnapshotAttribute 物件的陣列。

  手動資料庫叢集快照的屬性和值清單。
- DBClusterSnapshotIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  屬性套用目標之手動資料庫叢集快照的識別符。

`DBClusterSnapshotAttributesResult` 會用來做為以下項目的回應元素：

- DescribeDBClusterSnapshotAttributes
- ModifyDBClusterSnapshotAttribute

Neptune 事件 API

動作:

- CreateEventSubscription (動作)
- DeleteEventSubscription (動作)
- ModifyEventSubscription (動作)
- DescribeEventSubscriptions (動作)
- AddSourceIdentifierToSubscription (動作)
- RemoveSourceIdentifierFromSubscription (動作)
- DescribeEvents (動作)

- DescribeEventCategories (動作)

結構：

- Event (結構)
- EventCategoriesMap (結構)
- EventSubscription (結構)

CreateEventSubscription (動作)

此 API 的 AWS CLI 名稱是：`create-event-subscription`。

建立事件通知訂閱。這個動作需要 Neptune 主控台、SNS 主控台或 SNS API 建立的主題 ARN (Amazon Resource Name)。若要使用 SNS 取得 ARN，您必須在 Amazon SNS 中建立一個主題並訂閱該主題。SNS 主控台中會顯示 ARN。

您可以指定想要接到通知的來源類型 (SourceType)、提供觸發事件的 Neptune 來源清單 (SourceIds)，以及提供想要接到通知的事件類別 (EventCategories) 清單。例如，您可以指定 SourceType = db-instance；SourceIds = mydbinstance1、mydbinstance2；以及 EventCategories = Availability、Backup。

如果您同時指定 SourceType 和 SourceIds，例如 SourceType = db-instance 和 SourceIdentifier = myDBInstance1，則會收到指定來源所有 db-instance 事件的通知。如果您指定 SourceType 但未指定 SourceIdentifier，則會收到所有 Neptune 來源之該來源類型事件的通知。如果您不指定 SourceType 也不指定 SourceIdentifier，則會屬於您客戶帳戶之所有 Neptune 來源產生的事件通知。

請求

- Enabled (在 CLI 中:`--enabled`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  布林值；設為 true 啟用訂閱，設為 false 建立訂閱，但不啟動訂閱。
- EventCategories (在 CLI 中：`--event-categories`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  您想要訂閱的 SourceType 事件類別清單。您可以使用 DescribeEventCategories 動作查看指定 SourceType 的類別清單。
- SnsTopicArn (在 CLI 中：`--sns-topic-arn`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

針對事件通知所建立 SNS 主題的 Amazon Resource Name (ARN)。此 ARN 是當您建立並訂閱主題時由 Amazon SNS 所建立。

- SourceIds (在 CLI 中：`--source-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  傳回事件的事件來源識別碼清單。如未指定，回應中會包含所有來源。識別碼必須以字母開頭，只能包含 ASCII 字母、數字和連字號，而且不得以連字號結尾，或連續包含兩個連字號。

  約束:

  - 如果提供 SourceIds，也必須提供 SourceType。
  - 如果來源類型是資料庫執行個體，則必須提供 `DBInstanceIdentifier`。
  - 如果來源類型是資料庫安全群組，則必須提供 `DBSecurityGroupName`。
  - 如果來源類型是資料庫參數群組，則必須提供 `DBParameterGroupName`。
  - 如果來源類型是資料庫快照，則必須提供 `DBSnapshotIdentifier`。
- SourceType (在 CLI 中：`--source-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  產生事件的來源類型。例如，如果您希望接到資料庫執行個體所產生的事件通知，請將此參數設為 db-instance。如未指定此值，則會傳回所有事件。

  有效值：`db-instance` | `db-cluster` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster-snapshot`
- SubscriptionName (在 CLI 中：`--subscription-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  訂閱的名稱。

  限制條件：名稱必須少於 255 個字元。
- Tags (在 CLI 中：`--tags`) – Tag 物件的陣列。

  要套用到新事件訂閱的標籤。

回應

包含成功呼叫 the section called “DescribeEventSubscriptions” 動作的結果。

- CustomerAwsId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與事件通知訂閱相關聯的 Amazon 客戶帳戶。

- CustSubscriptionId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫事件通知訂閱 ID。
- Enabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  布林值指出訂閱是否啟用。True 指出訂閱已啟用。
- EventCategoriesList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱事件類別的清單。
- EventSubscriptionArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件訂閱的 Amazon Resource Name (ARN)。
- SnsTopicArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的主題 ARN。
- SourceIdsList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱來源 ID 的清單。
- SourceType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的來源類型。
- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的狀態。

  約束:

  可以是下列其中之一：建立 | 修改 | 刪除 | 作用中 | 無許可 | 主題不存在

  狀態「無許可」表示 Neptune 不再具有發佈 SNS 主題的許可。狀態「主題不存在」表示主題在訂閱建立後已遭刪除。
- SubscriptionCreationTime – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱建立的時間。

錯誤

- EventSubscriptionQuotaExceededFault
- SubscriptionAlreadyExistFault

- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault
- SourceNotFoundFault

DeleteEventSubscription (動作)

此 API 的 AWS CLI 名稱是：`delete-event-subscription`。

刪除事件通知訂閱。

請求

- SubscriptionName (在 CLI 中：`--subscription-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  您要刪除的事件通知訂閱名稱。

回應

包含成功呼叫 the section called “DescribeEventSubscriptions” 動作的結果。

- CustomerAwsId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與事件通知訂閱相關聯的 Amazon 客戶帳戶。
- CustSubscriptionId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫事件通知訂閱 ID。
- Enabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  布林值指出訂閱是否啟用。True 指出訂閱已啟用。
- EventCategoriesList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱事件類別的清單。
- EventSubscriptionArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件訂閱的 Amazon Resource Name (ARN)。

- SnsTopicArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的主題 ARN。
- SourceIdsList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱來源 ID 的清單。
- SourceType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的來源類型。
- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的狀態。

  約束:

  可以是下列其中之一：建立 | 修改 | 刪除 | 作用中 | 無許可 | 主題不存在

  狀態「無許可」表示 Neptune 不再具有發佈 SNS 主題的許可。狀態「主題不存在」表示主題在訂閱建立後已遭刪除。
- SubscriptionCreationTime – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱建立的時間。

錯誤

- SubscriptionNotFoundFault
- InvalidEventSubscriptionStateFault

ModifyEventSubscription (動作)

> 此 API 的 AWS CLI 名稱是：`modify-event-subscription`。

修改現有的事件通知訂閱。請注意，您無法使用此呼叫修改來源識別符；若要變更訂閱的來源識別符，請使用 the section called “AddSourceIdentifierToSubscription” 和 the section called “RemoveSourceIdentifierFromSubscription” 呼叫。

您可以使用 DescribeEventCategories 動作查看指定 SourceType 的事件類別清單。

請求

- Enabled (在 CLI 中:`--enabled`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  布林值；設為 true 啟用訂閱。
- EventCategories (在 CLI 中：`--event-categories`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  您想要訂閱的 SourceType 事件類別清單。您可以使用 DescribeEventCategories 動作查看指定 SourceType 的類別清單。
- SnsTopicArn (在 CLI 中：`--sns-topic-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  針對事件通知所建立 SNS 主題的 Amazon Resource Name (ARN)。此 ARN 是當您建立並訂閱主題時由 Amazon SNS 所建立。
- SourceType (在 CLI 中：`--source-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  產生事件的來源類型。例如，如果您希望接到資料庫執行個體所產生的事件通知，請將此參數設為 db-instance。如未指定此值，則會傳回所有事件。

  有效值：db-instance | db-parameter-group | db-security-group | db-snapshot
- SubscriptionName (在 CLI 中：`--subscription-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的名稱。

回應

包含成功呼叫 the section called “DescribeEventSubscriptions” 動作的結果。

- CustomerAwsId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與事件通知訂閱相關聯的 Amazon 客戶帳戶。
- CustSubscriptionId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫事件通知訂閱 ID。
- Enabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  布林值指出訂閱是否啟用。True 指出訂閱已啟用。
- EventCategoriesList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱事件類別的清單。

- EventSubscriptionArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件訂閱的 Amazon Resource Name (ARN)。

- SnsTopicArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的主題 ARN。

- SourceIdsList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱來源 ID 的清單。

- SourceType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的來源類型。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的狀態。

  約束:

  可以是下列其中之一：建立 | 修改 | 刪除 | 作用中 | 無許可 | 主題不存在

  狀態「無許可」表示 Neptune 不再具有發佈 SNS 主題的許可。狀態「主題不存在」表示主題在訂閱建立後已遭刪除。

- SubscriptionCreationTime – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱建立的時間。

錯誤

- EventSubscriptionQuotaExceededFault
- SubscriptionNotFoundFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault

DescribeEventSubscriptions (動作)

此 API 的 AWS CLI 名稱是：`describe-event-subscriptions`。

列出客戶帳戶的所有訂閱描述。訂閱描述包含 SubscriptionName、SNSTopicARN、CustomerID、SourceType、SourceID、CreationTime 和 Status。

如果您指定 SubscriptionName，請列出該訂閱的描述。

請求

- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 DescribeOrderableDBInstanceOptions 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。
- SubscriptionName (在 CLI 中：`--subscription-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  您要描述的事件通知訂閱名稱。

回應

- EventSubscriptionsList – 一個 EventSubscription 物件陣列。

  如 EventSubscriptions 資料類型的清單。
- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

前一個 DescribeOrderableDBInstanceOptions 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。

錯誤

- SubscriptionNotFoundFault

AddSourceIdentifierToSubscription (動作)

此 API 的 AWS CLI 名稱是：add-source-identifier-to-subscription。

將來源識別符新增至現有的事件通知訂閱。

請求

- SourceIdentifier (在 CLI 中：--source-identifier) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  要新增之事件來源的識別符。

  約束:

  - 如果來源類型是資料庫執行個體，則必須提供 DBInstanceIdentifier。
  - 如果來源類型是資料庫安全群組，則必須提供 DBSecurityGroupName。
  - 如果來源類型是資料庫參數群組，則必須提供 DBParameterGroupName。
  - 如果來源類型是資料庫快照，則必須提供 DBSnapshotIdentifier。
- SubscriptionName (在 CLI 中：--subscription-name) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  您想要新增來源識別符之事件通知訂閱的名稱。

回應

包含成功呼叫 the section called “DescribeEventSubscriptions” 動作的結果。

- CustomerAwsId – 字串，類型為：string (UTF-8 編碼的字串)。

  與事件通知訂閱相關聯的 Amazon 客戶帳戶。

- CustSubscriptionId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫事件通知訂閱 ID。
- Enabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  布林值指出訂閱是否啟用。True 指出訂閱已啟用。
- EventCategoriesList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱事件類別的清單。
- EventSubscriptionArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件訂閱的 Amazon Resource Name (ARN)。
- SnsTopicArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的主題 ARN。
- SourceIdsList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱來源 ID 的清單。
- SourceType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的來源類型。
- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的狀態。

  約束:

  可以是下列其中之一：建立 | 修改 | 刪除 | 作用中 | 無許可 | 主題不存在

  狀態「無許可」表示 Neptune 不再具有發佈 SNS 主題的許可。狀態「主題不存在」表示主題在訂閱建立後已遭刪除。
- SubscriptionCreationTime – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱建立的時間。

錯誤

- SubscriptionNotFoundFault
- SourceNotFoundFault

RemoveSourceIdentifierFromSubscription (動作)

此 API 的 AWS CLI 名稱是：`remove-source-identifier-from-subscription`。

從現有的事件通知訂閱移除來源識別符。

請求

- SourceIdentifier (在 CLI 中：`--source-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  訂閱要移除的來源識別符，例如資料庫執行個體的資料庫執行個體識別符或安全群組的名稱。
- SubscriptionName (在 CLI 中：`--subscription-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  您想要移除來源識別符之事件通知訂閱的名稱。

回應

包含成功呼叫 the section called “DescribeEventSubscriptions” 動作的結果。

- CustomerAwsId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  與事件通知訂閱相關聯的 Amazon 客戶帳戶。
- CustSubscriptionId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫事件通知訂閱 ID。
- Enabled – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  布林值指出訂閱是否啟用。True 指出訂閱已啟用。
- EventCategoriesList – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱事件類別的清單。
- EventSubscriptionArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件訂閱的 Amazon Resource Name (ARN)。
- SnsTopicArn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的主題 ARN。
- SourceIdsList – 字串，類型為：`string` (UTF-8 編碼的字串)。

事件通知訂閱來源 ID 的清單。

- SourceType – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的來源類型。

- Status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的狀態。

  約束:

  可以是下列其中之一：建立 | 修改 | 刪除 | 作用中 | 無許可 | 主題不存在

  狀態「無許可」表示 Neptune 不再具有發佈 SNS 主題的許可。狀態「主題不存在」表示主題在訂閱建立後已遭刪除。

- SubscriptionCreationTime – 字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱建立的時間。

錯誤

- SubscriptionNotFoundFault
- SourceNotFoundFault

DescribeEvents (動作)

此 API 的 AWS CLI 名稱是：`describe-events`。

傳回過去 14 天與資料庫執行個體、資料庫安全群組、資料庫快照和資料庫參數群組有關的事件。提供名稱做為參數可取得特定資料庫執行個體、資料庫安全群組、資料庫快照或資料庫參數群組的專屬事件。根據預設，會傳回過去一小時的事件。

請求

- Duration (在 CLI 中：`--duration`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  擷取事件的分鐘數。

  預設：60

- EndTime (在 CLI 中：`--end-time`) – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  擷取事件的時間間隔終點，以 ISO 8601 格式指定。如需 ISO 8601 的詳細資訊，請前往 ISO8601 Wikipedia 頁面。

  範例：2009-07-08T18:00Z
- EventCategories (在 CLI 中：`--event-categories`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  觸發事件通知訂閱通知的事件類別清單。
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 DescribeEvents 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 `MaxRecords` 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。
- SourceIdentifier (在 CLI 中：`--source-identifier`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回其事件的事件來源識別碼。如未指定，回應中會包含所有來源。

  約束:
  - 如果提供 SourceIdentifier，也必須提供 SourceType。
  - 如果來源類型是 `DBInstance`，則必須提供 `DBInstanceIdentifier`。
  - 如果來源類型是 `DBSecurityGroup`，則必須提供 `DBSecurityGroupName`。
  - 如果來源類型是 `DBParameterGroup`，則必須提供 `DBParameterGroupName`。
  - 如果來源類型是 `DBSnapshot`，則必須提供 `DBSnapshotIdentifier`。

  - 不能以連字號結尾或連續包含兩個連字號。
- SourceType (在 CLI 中：`--source-type`) – SourceType，類型為：`string` (UTF-8 編碼的字串)。

  事件擷取來源的事件來源。如未指定任何值，則會傳回所有事件。
- StartTime (在 CLI 中：`--start-time`) – TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  擷取事件的時間間隔起點，以 ISO 8601 格式指定。如需 ISO 8601 的詳細資訊，請前往 ISO8601 Wikipedia 頁面。

  範例：2009-07-08T18:00Z

回應

- Events – 一個 事件 物件陣列。

  the section called “事件” 執行個體的清單。
- Marker – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 Events 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

DescribeEventCategories (動作)

此 API 的 AWS CLI 名稱是：`describe-event-categories`。

顯示所有事件來源類型或特定來源類型 (如果指定) 的類別清單。

請求

- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- SourceType (在 CLI 中：`--source-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  產生事件的來源類型。

  有效值：db-instance | db-parameter-group | db-security-group | db-snapshot

回應

- EventCategoriesMapList – 一個 EventCategoriesMap 物件陣列。

  EventCategoriesMap 資料類型的清單。

結構：

Event (結構)

此資料類型在 the section called “DescribeEvents” 動作中會用來作為回應元素。

欄位

- Date - 這是 TStamp，類型為：`timestamp` (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  指定事件的日期和時間。
- EventCategories - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定事件類別。
- Message - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供此事件的文字。
- SourceArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件的 Amazon Resource Name (ARN)。
- SourceIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  提供事件來源的識別符。
- SourceType - 這是 SourceType，類型為：`string` (UTF-8 編碼的字串)。

  指定此事件的來源類型。

EventCategoriesMap (結構)

包含成功呼叫 the section called “DescribeEventCategories” 動作的結果。

欄位

- EventCategories - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定的來源類型事件類別
- SourceType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  傳回類別所屬之來源類型

EventSubscription (結構)

包含成功呼叫 the section called “DescribeEventSubscriptions” 動作的結果。

欄位

- CustomerAwsId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  與事件通知訂閱相關聯的 Amazon 客戶帳戶。
- CustSubscriptionId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫事件通知訂閱 ID。
- Enabled - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  布林值指出訂閱是否啟用。True 指出訂閱已啟用。
- EventCategoriesList - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱事件類別的清單。
- EventSubscriptionArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件訂閱的 Amazon Resource Name (ARN)。
- SnsTopicArn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的主題 ARN。
- SourceIdsList - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱來源 ID 的清單。
- SourceType - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的來源類型。

- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱的狀態。

  約束:

  可以是下列其中之一：建立 | 修改 | 刪除 | 作用中 | 無許可 | 主題不存在

  狀態「無許可」表示 Neptune 不再具有發佈 SNS 主題的許可。狀態「主題不存在」表示主題在訂閱建立後已遭刪除。
- SubscriptionCreationTime - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  事件通知訂閱建立的時間。

`EventSubscription` 會用來做為以下項目的回應元素：

- CreateEventSubscription
- ModifyEventSubscription
- AddSourceIdentifierToSubscription
- RemoveSourceIdentifierFromSubscription
- DeleteEventSubscription

其他 Neptune API

動作:

- AddTagsToResource (動作)
- ListTagsForResource (動作)
- RemoveTagsFromResource (動作)
- ApplyPendingMaintenanceAction (動作)
- DescribePendingMaintenanceActions (動作)
- DescribeDBEngineVersions (動作)

結構：

- DBEngineVersion (結構)

- EngineDefaults (結構)
- PendingMaintenanceAction (結構)
- ResourcePendingMaintenanceActions (結構)
- UpgradeTarget (結構)
- Tag (結構)

AddTagsToResource (動作)

此 API 的 AWS CLI 名稱是：`add-tags-to-resource`。

將中繼資料標籤新增到 Amazon Neptune 資源。這些標籤也可以搭配成本分配報告，用來追蹤與 Amazon Neptune 資源相關聯的成本，或用在 Amazon Neptune 之 IAM 政策的條件陳述式中。

請求

- ResourceName (在 CLI 中：`--resource-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要新增標籤的 Amazon Neptune 資源。這個值是 Amazon Resource Name (ARN)。如需建立 ARN 的資訊，請參閱建構 Amazon Resource Name (ARN)。
- Tags (在 CLI 中：`--tags`) – 必要：Tag 物件的陣列。

  要指派給 Amazon Neptune 資源的標籤。

回應

- 無回應參數。

錯誤

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ListTagsForResource (動作)

此 API 的 AWS CLI 名稱是：`list-tags-for-resource`。

列出 Amazon Neptune 資源的所有標籤。

請求

- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援此參數。
- ResourceName (在 CLI 中：`--resource-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  具有要列出之標籤的 Amazon Neptune 資源。這個值是 Amazon Resource Name (ARN)。如需建立 ARN 的資訊，請參閱建構 Amazon Resource Name (ARN)。

回應

- TagList – 一個 Tag 物件陣列。

  ListTagsForResource 操作傳回的標籤清單。

錯誤

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

RemoveTagsFromResource (動作)

此 API 的 AWS CLI 名稱是：`remove-tags-from-resource`。

移除 Amazon Neptune 資源中的中繼資料標籤。

請求

- ResourceName (在 CLI 中：`--resource-name`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要移除標籤的 Amazon Neptune 資源。這個值是 Amazon Resource Name (ARN)。如需建立 ARN 的資訊，請參閱建構 Amazon Resource Name (ARN)。
- TagKeys (在 CLI 中：`--tag-keys`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

要移除之標籤的標籤金鑰 (名稱)。

回應

- 無回應參數。

錯誤

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ApplyPendingMaintenanceAction (動作)

此 API 的 AWS CLI 名稱是：`apply-pending-maintenance-action`。

將待處理維護動作套用到資源 (例如，資料庫執行個體)。

請求

- ApplyAction (在 CLI 中：`--apply-action`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要套用至此資源的待處理維護動作。

  有效值：`system-update`、`db-upgrade`
- OptInType (在 CLI 中：`--opt-in-type`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  值，指定選擇使用請求的類型，或復原選擇使用請求。無法復原 `immediate` 類型的選擇使用請求。

  有效值：

  - `immediate` - 立即套用維護動作。
  - `next-maintenance` - 在下次資源維護時段期間套用維護動作。
  - `undo-opt-in` - 取消任何現有的 `next-maintenance` 選擇使用請求。
- ResourceIdentifier (在 CLI 中：`--resource-identifier`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

要套用待處理維護動作之資源的 Amazon Resource Name (ARN)。如需建立 ARN 的資訊，請參閱建構 Amazon Resource Name (ARN)。

回應

描述資源的待處理維護動作。

- PendingMaintenanceActionDetails – 一個 PendingMaintenanceAction 物件陣列。

  清單，提供資源待處理維護動作的詳細資訊。
- ResourceIdentifier – 字串，類型為：`string` (UTF-8 編碼的字串)。

  具有待處理維護動作的資源 ARN。

錯誤

- ResourceNotFoundFault

DescribePendingMaintenanceActions (動作)

此 API 的 AWS CLI 名稱是：`describe-pending-maintenance-actions`。

傳回至少有一個待處理維護動作的資源清單 (例如，資料庫執行個體)。

請求

- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  篩選條件，指定傳回待處理維護動作的一或多種資源。

  支援的篩選條件：

  - `db-cluster-id` - 接受資料庫叢集識別符及資料庫叢集 Amazon Resource Name (ARN)。結果清單只包含這些 ARN 找到的資料庫叢集待處理維護動作。
  - `db-instance-id` - 接受資料庫執行個體識別符和資料庫執行個體 ARN。結果清單只包含這些 ARN 找到的資料庫執行個體待處理維護動作。
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

前一個 DescribePendingMaintenanceActions 請求提供的選用分頁字符。如已指定此參數，則回應僅包含超出標記的記錄，不超過 MaxRecords 指定的記錄數量。

- MaxRecords (在 CLI 中：--max-records) – IntegerOptional，類型為：integer (帶正負號的 32 位元整數)。

  要在回應中包含的記錄數量上限。若存在的記錄比 MaxRecords 值指定的更多，則稱為「標記」的分頁字符會包含在回應中，讓您可以擷取剩餘的結果。

  預設：100

  限制條件：最小 20，最大 100。

- ResourceIdentifier (在 CLI 中：--resource-identifier) – 字串，類型為：string (UTF-8 編碼的字串)。

  傳回待處理維護動作的資源 ARN。

回應

- Marker – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個 DescribePendingMaintenanceActions 請求提供的選用分頁字符。如已指定此參數，則回應僅包含超出標記的記錄，不超過 MaxRecords 指定的記錄數量。

- PendingMaintenanceActions – 一個 ResourcePendingMaintenanceActions 物件陣列。

  資源的待處理維護動作清單。

錯誤

- ResourceNotFoundFault

DescribeDBEngineVersions (動作)

此 API 的 AWS CLI 名稱是：describe-db-engine-versions。

傳回可用的資料庫引擎清單。

請求

- DBParameterGroupFamily (在 CLI 中：`--db-parameter-group-family`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回詳細資訊的特定資料庫參數群組系列名稱。

  約束:

  - 如果提供，則必須符合現有的 DBParameterGroupFamily。
- DefaultOnly (在 CLI 中:`--default-only`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指出只會傳回指定引擎或引擎預設版本和主要版本的組合。
- Engine (在 CLI 中：`--engine`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回的資料庫引擎。
- EngineVersion (在 CLI 中：`--engine-version`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回的資料庫引擎版本。

  範例：`5.1.49`
- Filters (在 CLI 中：`--filters`) – 篩選條件 物件的陣列。

  目前不支援。
- ListSupportedCharacterSets (在 CLI 中:`--list-supported-character-sets`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如已指定此參數，而請求的引擎支援 `CreateDBInstance` 的 `CharacterSetName` 參數，則回應會包含每個引擎版本支援的字元集清單。
- ListSupportedTimezones (在 CLI 中:`--list-supported-timezones`) – BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  如已指定此參數，而請求的引擎支援 `CreateDBInstance` 的 `TimeZone` 參數，則回應會包含每個引擎版本支援的時區清單。
- Marker (在 CLI 中：`--marker`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。
- MaxRecords (在 CLI 中：`--max-records`) – IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

要在回應中包含的記錄數量上限。如有 MaxRecords 以上的值可用，則稱為標記的分頁字符會包含在回應中，以便擷取下列結果。

預設：100

限制條件：最小 20，最大 100。

回應

- DBEngineVersions – 一個 DBEngineVersion 物件陣列。

  DBEngineVersion 元素的清單。
- Marker – 字串，類型為：string (UTF-8 編碼的字串)。

  前一個請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 MaxRecords 指定的值為止。

結構：

DBEngineVersion (結構)

此資料類型在 the section called “DescribeDBEngineVersions” 動作中會用來作為回應元素。

欄位

- DBEngineDescription - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫引擎描述。
- DBEngineVersionDescription - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫引擎版本描述。
- DBParameterGroupFamily - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫引擎的資料庫參數群組系列名稱。
- Engine - 這是字串，類型為：string (UTF-8 編碼的字串)。

  資料庫引擎名稱。
- EngineVersion - 這是字串，類型為：string (UTF-8 編碼的字串)。

資料庫引擎版本編號。

- ExportableLogTypes - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫引擎能提供匯出到 CloudWatch Logs 的日誌類型。

- SupportedTimezones - 這是 時區 物件的陣列。

  此引擎針對 `CreateDBInstance` 動作之 `Timezone` 參數支援的時區清單。

- SupportsGlobalDatabases - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  一值，指示您是否可以使用 Aurora 全球資料庫，搭配特定的資料庫引擎版本。

- SupportsLogExportsToCloudwatchLogs - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  值，指出引擎版本是否支援將 ExportableLogTypes 提定的日誌類型匯出到 CloudWatch Logs。

- SupportsReadReplica - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指出資料庫引擎版本是否支援僅供讀取複本。

- ValidUpgradeTarget - 這是 UpgradeTarget 物件的陣列。

  此資料庫引擎版本可升級的引擎版本清單。

EngineDefaults (結構)

包含成功呼叫 the section called “DescribeEngineDefaultParameters” 動作的結果。

欄位

- DBParameterGroupFamily - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定要套用引擎預設參數的資料庫參數群組系列名稱。

- Marker - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  前一個 EngineDefaults 請求提供的選用分頁字符。若指定此參數，則回應只會包含超過標記的記錄，直到 `MaxRecords` 指定的值為止。

- Parameters - 這是 參數 物件的陣列。

  包含引擎預設參數的清單。

EngineDefaults 會用來做為以下項目的回應元素：

- DescribeEngineDefaultParameters
- DescribeEngineDefaultClusterParameters

PendingMaintenanceAction (結構)

提供資源待處理維護動作的資訊。

欄位

- Action - 這是字串，類型為：string (UTF-8 編碼的字串)。

  可供資源使用的待處理維護動作類型。

- AutoAppliedAfterDate - 這是 TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  套用動作時的維護時段日期。維護動作會在此日期後的第一個維護時段中，套用到資源。如已指定此日期，則忽略任何 next-maintenance 選擇使用請求。

- CurrentApplyDate - 這是 TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  待處理維護動作套用至資源時的有效日期。此日期會考慮從 the section called "ApplyPendingMaintenanceAction" API、AutoAppliedAfterDate 和 ForcedApplyDate 收到的選擇使用請求。如未收到選擇使用請求，且沒有任何項目指定為 AutoAppliedAfterDate 或 ForcedApplyDate，則此值為空白。

- Description - 這是字串，類型為：string (UTF-8 編碼的字串)。

  提供維護動作更多詳細資訊的描述。

- ForcedApplyDate - 這是 TStamp，類型為：timestamp (一個時間點，通常定義為 1970-01-01 午夜的偏移量)。

  自動套用維護動作時的日期。維護動作會在此日期套用至資源，無論資源的維護時段為何。如已指定此日期，則忽略任何 immediate 選擇使用請求。

- OptInStatus - 這是字串，類型為：string (UTF-8 編碼的字串)。

  指出資源已收到的選擇使用請求類型。

ResourcePendingMaintenanceActions (結構)

描述資源的待處理維護動作。

欄位

- PendingMaintenanceActionDetails - 這是 PendingMaintenanceAction 物件的陣列。

  清單，提供資源待處理維護動作的詳細資訊。
- ResourceIdentifier - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  具有待處理維護動作的資源 ARN。

`ResourcePendingMaintenanceActions` 會用來做為以下項目的回應元素：

- ApplyPendingMaintenanceAction

UpgradeTarget (結構)

資料庫執行個體可升級的資料庫引擎版本。

欄位

- AutoUpgrade - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  值，指出目標版本是否套用到已將 AutoMinorVersionUpgrade 設定為 true 的任何來源資料庫執行個體。
- Description - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體可升級的資料庫引擎版本。
- Engine - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  升級目標資料庫引擎名稱。
- EngineVersion - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  升級目標資料庫引擎的版本編號。
- IsMajorVersionUpgrade - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  值，指出資料庫引擎是否升級到主要版本。

- SupportsGlobalDatabases – 這是 BooleanOptional，類型為：`boolean` (布林值 (true 或 false))。

  一值，指示您是否可以使用 Neptune 全球資料庫，搭配目標引擎版本。

Tag (結構)

中繼資料，指派給由金鑰/值對組成的 Amazon Neptune 資源。

欄位

- Key - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  鍵是標籤的必要名稱。字串值長度可以是 1 到 128 個 Unicode 字元，不可在前面加上 `aws:` 或 `rds:`。該字串僅能包含一組 Unicode 字母、數字、空格、「_」、「.」、「/」、「=」、「+」、「-」(Java regex："^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")。
- Value - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  值是標籤的選用值。字串值長度可以是 1 到 256 個 Unicode 字元，不可在前面加上 `aws:` 或 `rds:`。該字串僅能包含一組 Unicode 字母、數字、空格、「_」、「.」、「/」、「=」、「+」、「-」(Java regex："^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")。

常見的 Neptune 資料類型

結構：

- AvailabilityZone (結構)
- DBSecurityGroupMembership (結構)
- DomainMembership (結構)
- DoubleRange (結構)
- Endpoint (結構)
- Filter (結構)
- Range (結構)
- ServerlessV2ScalingConfiguration (結構)
- ServerlessV2ScalingConfigurationInfo (結構)
- Timezone (結構)

- VpcSecurityGroupMembership (結構)

AvailabilityZone (結構)

指定可用區域。

欄位

- Name - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  可用區域的名稱。

DBSecurityGroupMembership (結構)

指定指定資料庫安全群組中的成員資格。

欄位

- DBSecurityGroupName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫安全群組的名稱。
- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫安全群組的狀態。

DomainMembership (結構)

與資料庫執行個體相關聯的 Active Directory 網域成員資格記錄。

欄位

- Domain - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  Active Directory 網域的識別符。
- FQDN - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  Active Directory 網域的完整網域名稱。
- IAMRoleName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  對 Directory Service 執行 API 呼叫時要使用的 IAM 角色名稱。

- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資料庫執行個體 Active Directory 網域成員資格的狀態，例如已加入、加入待定、故障等等。

DoubleRange (結構)

double 值的範圍。

欄位

- From - 這是 Double，類型為：`double` (雙精度 IEEE 754 浮點數)。

  範圍內的最小值。
- To - 這是 Double，類型為：`double` (雙精度 IEEE 754 浮點數)。

  範圍內的最大值。

Endpoint (結構)

指定連線端點。

如需代表 Amazon Neptune 資料庫叢集端點的資料結構，請參閱 `DBClusterEndpoint`。

欄位

- Address - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定資料庫執行個體的 DNS 地址。
- HostedZoneId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指定當您建立託管區域時，Amazon Route 53 指派的 ID。
- Port - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指定資料庫引擎接聽的連接埠。

Filter (結構)

目前不支援此類型。

欄位

- Name - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  目前不支援此參數。
- Values - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  目前不支援此參數。

Range (結構)

整數值範圍。

欄位

- From - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  範圍內的最小值。
- Step – 這是 IntegerOptional，類型為：`integer` (帶正負號的 32 位元整數)。

  範圍的間距值。例如，如果範圍是從 5,000 到 10,000，間距值為 1,000，則有效值為自 5,000 開始，每次加 1,000 的值。即使 7,500 在範圍內，卻不是範圍內的有效值。有效值為 5,000、6,000、7,000、8,000...
- To - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  範圍內的最大值。

ServerlessV2ScalingConfiguration (結構)

包含 Neptune Serverless 資料庫叢集的擴展組態。

如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

欄位

- MaxCapacity - 這是 DoubleOptional，類型為：`double` (雙精度 IEEE 754 浮點數)。

  Neptune Serverless 叢集中資料庫執行個體的 Neptune 容量單位 (NCU) 數量上限。指定 NCU 值時可使用 0.5 的增量，例如 40、40.5、41 等。

- MinCapacity - 這是 DoubleOptional，類型為：`double` (雙精度 IEEE 754 浮點數)。

  Neptune Serverless 叢集中資料庫執行個體的 Neptune 容量單位 (NCU) 數量下限。指定 NCU 值時可使用 0.5 的增量，例如 8、8.5、9 等。

ServerlessV2ScalingConfigurationInfo (結構)

顯示 Neptune Serverless 資料庫叢集的擴展組態。

如需詳細資訊，請參閱《Amazon Neptune 使用者指南》中的使用 Amazon Neptune Serverless。

欄位

- MaxCapacity - 這是 DoubleOptional，類型為：`double` (雙精度 IEEE 754 浮點數)。

  Neptune Serverless 叢集中資料庫執行個體的 Neptune 容量單位 (NCU) 數量上限。指定 NCU 值時可使用 0.5 的增量，例如 40、40.5、41 等。
- MinCapacity - 這是 DoubleOptional，類型為：`double` (雙精度 IEEE 754 浮點數)。

  Neptune Serverless 叢集中資料庫執行個體的 Neptune 容量單位 (NCU) 數量下限。指定 NCU 值時可使用 0.5 的增量，例如 8、8.5、9 等。

Timezone (結構)

與 the section called “DBInstance” 相關聯的時區。

欄位

- TimezoneName - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  時區的名稱。

VpcSecurityGroupMembership (結構)

此資料類型在 VPC 安全群組成員資格中用為查詢的回應元素。

欄位

- Status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

VPC 安全群組的狀態。

- VpcSecurityGroupId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  VPC 安全群組的名稱。

針對個別 API 的 Neptune 例外狀況

例外狀況：

- AuthorizationAlreadyExistsFault (結構)
- AuthorizationNotFoundFault (結構)
- AuthorizationQuotaExceededFault (結構)
- CertificateNotFoundFault (結構)
- DBClusterAlreadyExistsFault (結構)
- DBClusterNotFoundFault (結構)
- DBClusterParameterGroupNotFoundFault (結構)
- DBClusterQuotaExceededFault (結構)
- DBClusterRoleAlreadyExistsFault (結構)
- DBClusterRoleNotFoundFault (結構)
- DBClusterRoleQuotaExceededFault (結構)
- DBClusterSnapshotAlreadyExistsFault (結構)
- DBClusterSnapshotNotFoundFault (結構)
- DBInstanceAlreadyExistsFault (結構)
- DBInstanceNotFoundFault (結構)
- DBLogFileNotFoundFault (結構)
- DBParameterGroupAlreadyExistsFault (結構)
- DBParameterGroupNotFoundFault (結構)
- DBParameterGroupQuotaExceededFault (結構)
- DBSecurityGroupAlreadyExistsFault (結構)
- DBSecurityGroupNotFoundFault (結構)
- DBSecurityGroupNotSupportedFault (結構)

- DBSecurityGroupQuotaExceededFault (結構)
- DBSnapshotAlreadyExistsFault (結構)
- DBSnapshotNotFoundFault (結構)
- DBSubnetGroupAlreadyExistsFault (結構)
- DBSubnetGroupDoesNotCoverEnoughAZs (結構)
- DBSubnetGroupNotAllowedFault (結構)
- DBSubnetGroupNotFoundFault (結構)
- DBSubnetGroupQuotaExceededFault (結構)
- DBSubnetQuotaExceededFault (結構)
- DBUpgradeDependencyFailureFault (結構)
- DomainNotFoundFault (結構)
- EventSubscriptionQuotaExceededFault (結構)
- GlobalClusterAlreadyExistsFault (結構)
- GlobalClusterNotFoundFault (結構)
- GlobalClusterQuotaExceededFault (結構)
- InstanceQuotaExceededFault (結構)
- InsufficientDBClusterCapacityFault (結構)
- InsufficientDBInstanceCapacityFault (結構)
- InsufficientStorageClusterCapacityFault (結構)
- InvalidDBClusterEndpointStateFault (結構)
- InvalidDBClusterSnapshotStateFault (結構)
- InvalidDBClusterStateFault (結構)
- InvalidDBInstanceStateFault (結構)
- InvalidDBParameterGroupStateFault (結構)
- InvalidDBSecurityGroupStateFault (結構)
- InvalidDBSnapshotStateFault (結構)
- InvalidDBSubnetGroupFault (結構)
- InvalidDBSubnetGroupStateFault (結構)
- InvalidDBSubnetStateFault (結構)
- InvalidEventSubscriptionStateFault (結構)

- InvalidGlobalClusterStateFault (結構)
- InvalidOptionGroupStateFault (結構)
- InvalidRestoreFault (結構)
- InvalidSubnet (結構)
- InvalidVPCNetworkStateFault (結構)
- KMSKeyNotAccessibleFault (結構)
- OptionGroupNotFoundFault (結構)
- PointInTimeRestoreNotEnabledFault (結構)
- ProvisionedIopsNotAvailableInAZFault (結構)
- ResourceNotFoundFault (結構)
- SNSInvalidTopicFault (結構)
- SNSNoAuthorizationFault (結構)
- SNSTopicArnNotFoundFault (結構)
- SharedSnapshotQuotaExceededFault (結構)
- SnapshotQuotaExceededFault (結構)
- SourceNotFoundFault (結構)
- StorageQuotaExceededFault (結構)
- StorageTypeNotSupportedFault (結構)
- SubnetAlreadyInUse (結構)
- SubscriptionAlreadyExistFault (結構)
- SubscriptionCategoryNotFoundFault (結構)
- SubscriptionNotFoundFault (結構)

AuthorizationAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

指定的 CIDRIP 或 EC2 安全群組已獲指定的資料庫安全群組授權。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

說明問題詳細資訊的訊息。

AuthorizationNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

指定的 CIDRIP 或 EC2 安全群組未獲指定的資料庫安全群組授權。

Neptune 可能不會透過 IAM 授權代您執行必要的動作。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

AuthorizationQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

已達到資料庫安全群組授權配額。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

CertificateNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

CertificateIdentifier 不代表現有的憑證。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

使用者的資料庫叢集已有指定的識別符。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBClusterIdentifier 不代表現有的資料庫叢集。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterParameterGroupNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBClusterParameterGroupName 不代表現有的資料庫叢集參數群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：403。

使用者嘗試建立新的資料庫叢集，但使用者已經達到允許的資料庫叢集配額上限。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterRoleAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

指定的 IAM 角色 Amazon Resource Name (ARN) 與指定的資料庫叢集相關聯。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterRoleNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

指定的 IAM 角色 Amazon Resource Name (ARN) 與指定的資料庫叢集不相關聯。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterRoleQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

您已超過可與指定資料庫叢集相關聯之 IAM 角色的最大數量。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterSnapshotAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

使用者的資料庫叢集快照已有指定的識別符。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBClusterSnapshotNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBClusterSnapshotIdentifier 不代表現有的資料庫叢集快照。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBInstanceAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

使用者的資料庫執行個體已有指定的識別符。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBInstanceNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBInstanceIdentifier 不代表現有的資料庫執行個體。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBLogFileNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

LogFileName 不代表現有的資料庫日誌檔案。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBParameterGroupAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

有同名的資料庫參數群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBParameterGroupNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBParameterGroupName 不代表現有的資料庫參數群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBParameterGroupQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

請求會造成使用者超出允許的資料庫參數群組數目。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSecurityGroupAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

已有 DBSecurityGroupName 指定名稱的資料庫安全群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSecurityGroupNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBSecurityGroupName 不代表現有的資料庫安全群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSecurityGroupNotSupportedFault (結構)

傳回的 HTTP 狀態碼：400。

資料庫安全群組不允許此動作。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSecurityGroupQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

請求會造成使用者超出允許的資料庫安全群組數目。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSnapshotAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

現有的快照已使用 DBSnapshotIdentifier。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSnapshotNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBSnapshotIdentifier 不代表現有的資料庫快照。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSubnetGroupAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

現有的資料庫子網路群組已使用 DBSubnetGroupName。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSubnetGroupDoesNotCoverEnoughAZs (結構)

傳回的 HTTP 狀態碼：400。

除非只有一個可用區域，否則資料庫子網路群組中的子網路至少應覆蓋兩個可用區域。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSubnetGroupNotAllowedFault (結構)

傳回的 HTTP 狀態碼：400。

指出建立僅供讀取複本時不應指定 DBSubnetGroup，此複本與來源執行個體位於相同的區域。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSubnetGroupNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

DBSubnetGroupName 不代表現有的資料庫子網路群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSubnetGroupQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

請求會造成使用者超出允許的資料庫子網路群組數目。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBSubnetQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

請求會造成使用者超出資料庫子網路允許的子網路數目。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DBUpgradeDependencyFailureFault (結構)

傳回的 HTTP 狀態碼：400。

因為無法修改資料庫依賴的資源，所以資料庫升級失敗。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

DomainNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

Domain 不表示現有的 Active Directory 網域。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

EventSubscriptionQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

您已超過可以訂閱的事件數目。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

GlobalClusterAlreadyExistsFault (結構)

傳回的 HTTP 狀態碼：400。

`GlobalClusterIdentifier` 已存在。選擇新的全球資料庫識別符 (唯一名稱)，以建立新的全球資料庫叢集。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

GlobalClusterNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

GlobalClusterIdentifier 不會參考現有的全球資料庫叢集。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

GlobalClusterQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

此帳戶的全球資料庫叢集數目已達允許的上限。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InstanceQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

請求會造成使用者超出允許的資料庫執行個體數目。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InsufficientDBClusterCapacityFault (結構)

傳回的 HTTP 狀態碼：403。

資料庫叢集沒有足夠的容量，無法執行目前的操作。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

說明問題詳細資訊的訊息。

InsufficientDBInstanceCapacityFault (結構)

傳回的 HTTP 狀態碼：400。

指定的資料庫執行個體類別無法提供指定的可用區域使用。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InsufficientStorageClusterCapacityFault (結構)

傳回的 HTTP 狀態碼：400。

目前的動作沒有足夠的儲存體可用。您可以更新您的子網路群組，使用有更多儲存體可用的不同可用區域，來解決這個錯誤。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBClusterEndpointStateFault (結構)

傳回的 HTTP 狀態碼：400。

端點處於此狀態時，無法在端點上執行請求的操作。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBClusterSnapshotStateFault (結構)

傳回的 HTTP 狀態碼：400。

提供的值不是有效的資料庫叢集快照狀態。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBClusterStateFault (結構)

傳回的 HTTP 狀態碼：400。

資料庫叢集不為有效狀態。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBInstanceStateFault (結構)

傳回的 HTTP 狀態碼：400。

指定的資料庫執行個體不為「可用」狀態。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBParameterGroupStateFault (結構)

傳回的 HTTP 狀態碼：400。

資料庫參數群組正在使用，或為無效狀態。如果您想嘗試刪除參數群組，您無法刪除此狀態的參數群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBSecurityGroupStateFault (結構)

傳回的 HTTP 狀態碼：400。

資料庫安全群組的狀態不允許刪除。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBSnapshotStateFault (結構)

傳回的 HTTP 狀態碼：400。

資料庫快照的狀態不允許刪除。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBSubnetGroupFault (結構)

傳回的 HTTP 狀態碼：400。

指出 DBSubnetGroup 和現有的跨區域僅供讀取複本，雖然有相同的來源執行個體，但不屬於同一個 VPC。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBSubnetGroupStateFault (結構)

傳回的 HTTP 狀態碼：400。

資料庫子網路群組因為正在使用，所以無法刪除。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidDBSubnetStateFault (結構)

傳回的 HTTP 狀態碼：400。

資料庫子網路不為「可用」狀態。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidEventSubscriptionStateFault (結構)

傳回的 HTTP 狀態碼：400。

事件訂閱為無效狀態。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidGlobalClusterStateFault (結構)

傳回的 HTTP 狀態碼：400。

全域叢集處於無效狀態，無法執行請求的作業。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidOptionGroupStateFault (結構)

傳回的 HTTP 狀態碼：400。

選項群組不為「可用」狀態。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidRestoreFault (結構)

傳回的 HTTP 狀態碼：400。

無法從 vpc 備份還原到非 vpc 資料庫執行個體。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidSubnet (結構)

傳回的 HTTP 狀態碼：400。

請求的子網路無效，或已請求的多個子網路，不全在共同的 VPC 中。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

InvalidVPCNetworkStateFault (結構)

傳回的 HTTP 狀態碼：400。

因為使用者的變更，資料庫子網路群組在建立後不會覆蓋所有可用區域。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

KMSKeyNotAccessibleFault (結構)

傳回的 HTTP 狀態碼：400。

存取 KMS 金鑰發生錯誤。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

OptionGroupNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

找不到指定的選項群組。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

PointInTimeRestoreNotEnabledFault (結構)

傳回的 HTTP 狀態碼：400。

SourceDBInstanceIdentifier 是指 BackupRetentionPeriod 等於 0 的資料庫執行個體。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

ProvisionedIopsNotAvailableInAZFault (結構)

傳回的 HTTP 狀態碼：400。

在指定的可用區域中無法使用佈建 IOPS。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

ResourceNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

找不到指定的資源 ID。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SNSInvalidTopicFault (結構)

傳回的 HTTP 狀態碼：400。

SNS 主題無效。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SNSNoAuthorizationFault (結構)

傳回的 HTTP 狀態碼：400。

沒有 SNS 授權。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SNSTopicArnNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

找不到 SNS 主題的 ARN。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SharedSnapshotQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

您已超過可共享手動資料庫快照的帳戶數量上限。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SnapshotQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

請求會造成使用者超出允許的資料庫快照數目。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SourceNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

找不到來源。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

StorageQuotaExceededFault (結構)

傳回的 HTTP 狀態碼：400。

請求會造成使用者超過所有資料庫執行個體允許的可用儲存量。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

StorageTypeNotSupportedFault (結構)

傳回的 HTTP 狀態碼：400。

指定的 StorageType 無法與資料庫執行個體相關聯。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SubnetAlreadyInUse (結構)

傳回的 HTTP 狀態碼：400。

資料庫子網路已在可用區域中使用。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SubscriptionAlreadyExistFault (結構)

傳回的 HTTP 狀態碼：400。

此訂閱已存在。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SubscriptionCategoryNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

找不到指定的訂閱類別。

欄位

- message - 這是 ExceptionMessage，類型:string (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

SubscriptionNotFoundFault (結構)

傳回的 HTTP 狀態碼：404。

找不到指定的訂閱。

欄位

- message - 這是 ExceptionMessage，類型:`string` (UTF-8 編碼的字串)。

  說明問題詳細資訊的訊息。

Amazon Neptune 資料 API 參考

本章記載 Neptune 資料 API 操作，您可以使用這些操作，載入、存取和維護 Neptune 圖形中的資料。

Neptune 資料 API 為載入資料、執行查詢、取得資料相關資訊，以及執行機器學習操作提供 SDK 支援。它支援 Neptune 中的 Gremlin 和 OpenCypher 查詢語言，並且可在所有 SDK 語言中使用。它會自動簽署 API 請求，並大幅簡化將 Neptune 整合至應用程式的操作。

內容

- Neptune 資料平面大量載入器 API
  - StartLoaderJob (動作)
  - GetLoaderJobStatus (動作)
  - ListLoaderJobs (動作)
  - CancelLoaderJob (動作)
  - 大量載入結構：
  - LoaderIdResult (結構)
- Neptune 串流資料平面 API
  - GetPropertygraphStream (動作)
  - 串流資料結構：
  - PropertygraphRecord (結構)
  - PropertygraphData (結構)
- Neptune 資料平面統計資料和圖形摘要 API
  - GetPropertygraphStatistics (動作)
  - ManagePropertygraphStatistics (動作
  - DeletePropertygraphStatistics (動作)
  - GetPropertygraphSummary (動作)
  - 統計資料結構：
  - 統計資料 (結構)
  - StatisticsSummary (結構)
  - DeleteStatisticsValueMap (結構)
  - RefreshStatisticsIdMap (結構)
  - NodeStructure (結構)
  - EdgeStructure (結構)
  - SubjectStructure (結構)
  - PropertygraphSummaryValueMap (結構)
  - PropertygraphSummary (結構)
- Neptune ML 資料處理 API
  - StartMLDataProcessingJob (動作)
  - ListMLDataProcessingJobs (動作)

- ConcurrentModificationException (結構)
- ConstraintViolationException (結構)
- ExpiredStreamException (結構)
- FailureByQueryException (結構)
- IllegalArgumentException (結構)
- InternalFailureException (結構)
- InvalidArgumentException (結構)
- InvalidNumericDataException (結構)
- InvalidParameterException (結構)
- LoadUrlAccessDeniedException (結構)
- MalformedQueryException (結構)
- MemoryLimitExceededException (結構)
- MethodNotAllowedException (結構)
- MissingParameterException (結構)
- MLResourceNotFoundException (結構)
- ParsingException (結構)
- PreconditionsFailedException (結構)
- QueryLimitExceededException (結構)
- QueryLimitException (結構)
- QueryTooLargeException (結構)
- ReadOnlyViolationException (結構)
- S3Exception (結構)
- ServerShutdownException (結構)
- StatisticsNotAvailableException (結構)
- StreamRecordsNotFoundException (結構)
- ThrottlingException (結構)
- TimeLimitExceededException (結構)
- TooManyRequestsException (結構)
- UnsupportedOperationException (結構)
- UnloadUrlAccessDeniedException (結構)

Neptune 資料平面引擎、快速重設和一般結構 API

引擎操作：

- GetEngineStatus (動作)
- ExecuteFastReset (動作)

引擎操作結構：

- QueryLanguageVersion (結構)
- FastResetToken (結構)

GetEngineStatus (動作)

此 API 的 AWS CLI 名稱是：`get-engine-status`。

擷取主機上圖形資料庫的狀態。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，允許該叢集中的 neptune-db:GetEngineStatus 動作。

請求

- 無要求參數。

回應

- dbEngineVersion – 字串，類型為：`string` (UTF-8 編碼的字串)。

  設定為資料庫叢集上執行的 Neptune 引擎版本。若此引擎版本自發行以來已手動進行修補，則版本號碼會加上 `Patch-` 前綴。
- dfeQueryEngine – 字串，類型為：`string` (UTF-8 編碼的字串)。

  如果 DFE 引擎已完全啟用，則設定為 `enabled`，或者，如果 DFE 引擎僅與 `useDFE` 查詢提示設定為 `true` 的查詢搭配使用，則設定為 `viaQueryHint` (預設值)。
- features – 它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：`string` (UTF-8 編碼的字串)。

每個值都是文件，類型為：document (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

包含有關資料庫叢集上啟用之功能的狀態資訊。

- gremlin – QueryLanguageVersion 物件。

  包含叢集上可用之 Grimlin 查詢語言的相關資訊。具體而言，它包含一個版本欄位，指定引擎正在使用的目前 TinkerPop 版本。

- labMode – 它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：string (UTF-8 編碼的字串)。

  每個值都是字串，類型為：string (UTF-8 編碼的字串)。

  包含引擎正在使用的實驗室模式設定。

- opencypher – QueryLanguageVersion 物件。

  包含叢集上可用之 openCypher 查詢語言的相關資訊。具體而言，它包含一個版本欄位，指定引擎正在使用的目前 openCypher 版本。

- role – 字串，類型為：string (UTF-8 編碼的字串)。

  如果執行個體是僅供讀取複本，則設定為 reader，或者，如果執行個體是主要執行個體，則設為 writer。

- rollingBackTrxCount – 整數，類型為：integer (帶正負號的 32 位元整數)。

  如果有要復原的交易，則此欄位會設定為此類交易的數目。如果沒有，則欄位根本不會出現。

- rollingBackTrxEarliestStartTime – 字串，類型為：string (UTF-8 編碼的字串)。

  設定為正在復原之最早交易的開始時間。如果沒有任何交易正在復原，則欄位根本不會出現。

- settings – 它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：string (UTF-8 編碼的字串)。

  每個值都是字串，類型為：string (UTF-8 編碼的字串)。

  包含資料庫叢集上目前設定的相關資訊。例如，包含目前的叢集查詢逾時設定 (clusterQueryTimeoutInMs)。

- sparql – QueryLanguageVersion 物件。

包含叢集上可用之 SPARQL 查詢語言的相關資訊。具體而言，它包含一個版本欄位，指定引擎正在使用的目前 SPARQL 版本。

- startTime – 字串，類型為：`string` (UTF-8 編碼的字串)。

  設定為目前伺服器程序啟動的 UTC 時間。

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  如果執行個體沒有發生問題，則設定為 `healthy`。如果執行個體正在從當機復原，或是正在重新開機，而仍有從最近一次伺服器關機執行的作用中交易，則狀態會設為 `recovery`。

錯誤

- UnsupportedOperationException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

ExecuteFastReset (動作)

此 API 的 AWS CLI 名稱是：`execute-fast-reset`。

快速重設 REST API 可讓您快速輕鬆地重設 Neptune 圖形，同時刪除其所有資料。

Neptune 快速重設是兩步驟程序。首先您會在 `action` 設定為 `initiateDatabaseReset` 的情況下呼叫 `ExecuteFastReset`。這會傳回一個 UUID 權杖，然後在 `action` 設定為 `performDatabaseReset` 的情況下再次呼叫 `ExecuteFastReset` 時會包含此權杖。請參閱使用快速重設 API 清空 Amazon Neptune 資料庫叢集。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，允許該叢集中的 neptune-db:ResetDatabase 動作。

請求

- action (在 CLI 中：`--action`) – 必要：一種動作，類型為：`string` (UTF-8 編碼的字串)。

  快速重設動作。下列其中一值：

  - **`initiateDatabaseReset`** – 此動作會產生實際執行快速重設所需的唯一權杖。
  - **`performDatabaseReset`** – 此動作會使用 `initiateDatabaseReset` 動作產生的權杖來實際執行快速重設。

- token (在 CLI 中：`--token`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要啟動重設的快速重設權杖

回應

- payload – FastResetToken 物件。

  `payload` 只由 `initiateDatabaseReset` 動作傳回，並包含要與 `performDatabaseReset` 動作搭配使用以進行重設的唯一權杖。
- status – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  `status` 只會針對 `performDatabaseReset` 動作傳回，並指示是否接受快速重設請求。

錯誤

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- ServerShutdownException
- PreconditionsFailedException
- MethodNotAllowedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException

- MissingParameterException

引擎操作結構：

QueryLanguageVersion (結構)

表示查詢語言版本的結構。

欄位

- version - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  查詢語言的版本。

FastResetToken (結構)

一種結構，其中包含用來啟動快速重設的快速重設權杖。

欄位

- token - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  `initiateDatabaseReset` 動作中由資料庫產生的 UUID，然後由 `performDatabaseReset` 用來重設資料庫。

Neptune 查詢 API

Gremlin 查詢動作：

- ExecuteGremlinQuery (動作)
- ExecuteGremlinExplainQuery (動作)
- ExecuteGremlinProfileQuery (動作)
- ListGremlinQueries (動作)
- GetGremlinQueryStatus (動作)
- CancelGremlinQuery (動作)

openCypher 查詢動作：

- ExecuteOpenCypherQuery (動作)
- ExecuteOpenCypherExplainQuery (動作)
- ListOpenCypherQueries (動作)
- GetOpenCypherQueryStatus (動作)
- CancelOpenCypherQuery (動作)

查詢結構：

- QueryEvalStats (結構)
- GremlinQueryStatus (結構)
- GremlinQueryStatusAttributes (結構)

ExecuteGremlinQuery (動作)

此 API 的 AWS CLI 名稱是：`execute-gremlin-query`。

此命令會執行一個 Gremlin 查詢。Amazon Neptune 與 Apache TinkerPP3 和 Gremlin 相容，因此您可以使用 Gremlin 周遊語言來查詢圖形，如 Apache TinkerPop3 文件中的圖形下所述。更多詳細資訊也可以在使用 Gemlin 存取 Neptune 圖形中找到。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中啟用下列其中一個 IAM 動作，取決於查詢：

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:Gremlin IAM 條件金鑰，以限制 Grimlin 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- gremlinQuery (在 CLI 中：`--gremlin-query`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

使用此 API，您可以像使用 HTTP 端點一樣以字串格式執行 Gremlin 查詢。介面與您的資料庫叢集正在使用的任何 Gremlin 版本相容 (請參閱 Tinkerpop 用戶端一節以確定引擎版本支援的 Gemlin 版本）。

- serializer (在 CLI 中：`--serializer`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若為非 null，則會依此參數指定的格式，以序列化回應訊息傳回查詢結果。如需目前支援的格式清單，請參閱 TinkerPop 文件中的 GraphSON 一節。

回應

- meta – 文件，類型為：`document` (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

  關於 Gremlin 查詢的中繼資料。
- requestId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Gremlin 查詢的唯一識別符。
- result – 文件，類型為：`document` (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

  來自伺服器的 Gremlin 查詢輸出。
- status – GremlinQueryStatusAttributes 物件。

  Gremlin 查詢的狀態。

錯誤

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException

- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinExplainQuery (動作)

此 API 的 AWS CLI 名稱是：`execute-gremlin-explain-query`。

執行 Gremlin Explain 查詢。

Amazon Neptune 新增了名為 `explain` 的 Gremlin 功能，其是一種自助式工具，用於了解 Neptune 引擎為查詢採取的執行方法。您透過將 `explain` 參數新增到提交 Gremlin 查詢的 HTTP 呼叫，來進行叫用。

Explain 功能可提供查詢執行計劃邏輯結構的相關資訊。您可以使用此資訊來識別潛在的評估和執行瓶頸，並調校您的查詢，如調校 Gremlin 查詢中所述。您也可以使用查詢提示，以改善查詢執行計劃。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許下列其中一個 IAM 動作，取決於查詢：

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:Gremlin IAM 條件金鑰，以限制 Grimlin 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- gremlinQuery (在 CLI 中：`--gremlin-query`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  Gremlin Explain 查詢字串。

回應

- output – ReportAsText，類型為：`blob` (未解譯的二進位資料區塊)。

  包含 Gemlin Explain 結果的文字 Blob，如調校 Gremlin 查詢中所述。

錯誤

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException

- ConcurrentModificationException

ExecuteGremlinProfileQuery (動作)

此 API 的 AWS CLI 名稱是：`execute-gremlin-profile-query`。

執行 Gremlin 設定檔查詢，其會執行指定的 Gremlin 周遊、收集各種有關執行的指標，以及產生設定檔報告做為輸出。如需詳細資訊，請參閱 Neptune 中的 Gremlin 設定檔 API。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:ReadDataViaQuery IAM 動作。

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:Gremlin IAM 條件金鑰，以限制 Grimlin 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- chop (在 CLI 中：`--chop`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  若為非零，則會在該字元數處截斷結果字串。若設定為零，則字串會包含所有結果。
- gremlinQuery (在 CLI 中：`--gremlin-query`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要分析的 Gremlin 查詢字串。
- indexOps (在 CLI 中:`--index-ops`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果此旗標設定為 `TRUE`，則結果會包括查詢執行和序列化期間所發生之所有索引操作的詳細報告。
- results (在 CLI 中:`--results`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果此旗標設定為 `TRUE`，則會收集查詢結果，並將其顯示為設定檔報告的一部分。若為 `FALSE`，則只會顯示結果計數。
- serializer (在 CLI 中：`--serializer`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  若為非 null，則會依此參數指定的格式，以序列化回應訊息傳回所收集的結果。如需詳細資訊，請參閱 Neptune 中的 Gremlin 設定檔 API。

回應

- output – ReportAsText，類型為：`blob` (未解譯的二進位資料區塊)。

包含 Gremlin 設定檔結果的文字 Blob。如需詳細資訊，請參閱 Neptune 中的 Gremlin 設定檔 API。

錯誤

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListGremlinQueries (動作)

此 API 的 AWS CLI 名稱是：`list-gremlin-queries`。

列出作用中的 Gremlin 查詢。如需輸出的詳細資訊，請參閱 Gremlin 查詢狀態 API。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetQueryStatus IAM 動作。

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:Gremlin IAM 條件金鑰，以限制 Grimlin 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- includeWaiting (在 CLI 中:`--include-waiting`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `TRUE`，則傳回的清單會包含等待中查詢。預設值為 `FALSE`；

回應

- acceptedQueryCount – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  已接受但尚未完成的查詢數目，包括佇列中的查詢。
- queries – 一個 GremlinQueryStatus 物件陣列。

  目前查詢的清單。
- runningQueryCount – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  目前執行中 Gremlin 查詢的數量。

錯誤

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException

- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetGremlinQueryStatus (動作)

此 API 的 AWS CLI 名稱是：`get-gremlin-query-status`。

取得所指定 Gemlin 查詢的狀態。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetQueryStatus IAM 動作。

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:Gremlin IAM 條件金鑰，以限制 Grimlin 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- queryId (在 CLI 中：`--query-id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  識別 Gemlin 查詢的唯一識別符。

回應

- queryEvalStats – QueryEvalStats 物件。

  Gremlin 查詢的評估狀態。
- queryId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回其狀態之查詢的 ID。
- queryString – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Gremlin 查詢字串。

錯誤

- BadRequestException

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelGremlinQuery (動作)

此 API 的 AWS CLI 名稱是：`cancel-gremlin-query`。

取消 Gremlin 查詢。如需詳細資訊，請參閱 Gemlin 查詢取消。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:CancelQuery IAM 動作。

請求

- queryId (在 CLI 中：`--query-id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  識別要取消之查詢的唯一識別符。

回應

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

取消的狀態

錯誤

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

openCypher 查詢動作：

ExecuteOpenCypherQuery (動作)

此 API 的 AWS CLI 名稱是：`execute-open-cypher-query`。

執行 openCypher 查詢。如需詳細資訊，請參閱使用 OpenCypher 存取 Neptune 圖形。

Neptune 支援使用 OpenCypher 建置圖形，OpenCypher 是使用圖形資料庫的開發人員之間目前最熱門的其中一個查詢語言。開發人員、商務分析師和資料科學家都愛用 OpenCypher 的宣告式、SQL 啟發語法，因為它提供了查詢屬性圖的熟悉結構。

OpenCypher 語言最初是由 Neo4j 開發，然後在 2015 年成為開放原始碼，並在 Apache 2 開放原始碼授權下投入 OpenCypher 專案。

請注意，在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許下列其中一個 IAM 動作，取決於查詢：

- neptune-db:ReadDataViaQuery
- neptune-db:WriteDataViaQuery
- neptune-db:DeleteDataViaQuery

另請注意，可以在政策文件中使用 neptune-db:QueryLanguage:OpenCypher IAM 條件金鑰，以限制 openCypher 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- openCypherQuery (在 CLI 中：`--open-cypher-query`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要執行的 openCypher 查詢字串。
- parameters (在 CLI 中：`--parameters`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  用於查詢執行的 openCypher 查詢參數。如需詳細資訊，請參閱 OpenCypher 參數化查詢的範例。

回應

- results – 必要：文件，類型為：`document` (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

  openCypher 查詢結果。

錯誤

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException

- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteOpenCypherExplainQuery (動作)

此 API 的 AWS CLI 名稱是：`execute-open-cypher-explain-query`。

執行 openCypher `explain` 請求。如需詳細資訊，請參閱 openCypher Explain 功能。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:ReadDataViaQuery IAM 動作。

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:OpenCypher IAM 條件金鑰，以限制 openCypher 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- explainMode (在 CLI 中：`--explain-mode`) – 必要：OpenCypherExplainMode，類型為：`string` (UTF-8 編碼的字串)。

  openCypher `explain` 模式。可以是下列其中一個：`static`、`dynamic` 或 `details`。
- openCypherQuery (在 CLI 中：`--open-cypher-query`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

openCypher 查詢字串。

- parameters (在 CLI 中：`--parameters`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  openCypher 查詢參數。

回應

- results – 必要：Blob，類型為：`blob` (未解譯的二進位資料區塊)。

  包含 openCypher `explain` 結果的文字 Blob。

錯誤

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException

- MissingParameterException
- ConcurrentModificationException

ListOpenCypherQueries (動作)

此 API 的 AWS CLI 名稱是：`list-open-cypher-queries`。

列出作用中的 openCypher 查詢。如需詳細資訊，請參閱 Neptune OpenCypher 狀態端點。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetQueryStatus IAM 動作。

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:OpenCypher IAM 條件金鑰，以限制 openCypher 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- includeWaiting (在 CLI 中:`--include-waiting`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  當設定為 TRUE 且其他參數不存在時，這會導致針對等待中查詢以及執行中查詢傳回狀態資訊。

回應

- acceptedQueryCount – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  已接受但尚未完成的查詢數目，包括佇列中的查詢。
- queries – 一個 GremlinQueryStatus 物件陣列。

  目前 openCypher 查詢的清單。
- runningQueryCount – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  目前執行中的 openCypher 查詢數量。

錯誤

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException

- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetOpenCypherQueryStatus (動作)

此 API 的 AWS CLI 名稱是：`get-open-cypher-query-status`。

擷取所指定 OpenCypher 查詢的狀態。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetQueryStatus IAM 動作。

請注意，可以在政策文件中使用 neptune-db:QueryLanguage:OpenCypher IAM 條件金鑰，以限制 openCypher 查詢的使用 (請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰)。

請求

- queryId (在 CLI 中：`--query-id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要擷取其查詢狀態之 OpenCypher 查詢的唯一 ID。

回應

- queryEvalStats – QueryEvalStats 物件。

openCypher 查詢評估狀態。

- queryId – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回其狀態之查詢的唯一 ID。

- queryString – 字串，類型為：`string` (UTF-8 編碼的字串)。

  openCypher 查詢字串。

錯誤

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelOpenCypherQuery (動作)

此 API 的 AWS CLI 名稱是：`cancel-open-cypher-query`。

取消指定的 openCypher 查詢。如需詳細資訊，請參閱 Neptune OpenCypher 狀態端點。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:CancelQuery IAM 動作。

請求

- queryId (在 CLI 中：`--query-id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要取消之 openCypher 查詢的唯一 ID。
- silent (在 CLI 中:`--silent`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `TRUE`，則會導致以無訊息方式取消 OpenCypher 查詢。

回應

- payload – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  用於 openCypher 查詢的取消承載。
- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  openCypher 查詢的取消狀態。

錯誤

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException

- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

查詢結構：

QueryEvalStats (結構)

要擷取查詢統計資料的結構，例如執行、接受或等待的查詢數目及其詳細資料。

欄位

- cancelled - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果查詢已取消，則設定為 TRUE，否則設定為 FALSE。
- elapsed - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  到目前為止查詢已執行的毫秒數。
- subqueries - 這是一份文件，類型為：`document` (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

  此查詢中的子查詢數目。
- waited - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  指出查詢已等待多長時間 (以毫秒為單位)。

GremlinQueryStatus (結構)

擷取 Gremlin 查詢的狀態 (請參閱 Gremlin 查詢狀態 API 頁面)。

欄位

- queryEvalStats - 這是一個 QueryEvalStats 物件。

  Gremlin 查詢的狀態統計資料。
- queryId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  Gremlin 查詢的 ID。

- queryString - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  Gremlin 查詢的查詢字串。

GremlinQueryStatusAttributes (結構)

包含 Gremlin 查詢的狀態元件。

欄位

- attributes - 這是一份文件，類型為：`document` (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

  Gremlin 查詢狀態的屬性。
- code - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  從 Gremlin 查詢請求傳回的 HTTP 回應代碼。
- message - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  狀態訊息。

Neptune 資料平面大量載入器 API

大量載入動作：

- StartLoaderJob (動作)
- GetLoaderJobStatus (動作)
- ListLoaderJobs (動作)
- CancelLoaderJob (動作)

大量載入結構：

- LoaderIdResult (結構)

StartLoaderJob (動作)

此 API 的 AWS CLI 名稱是：`start-loader-job`。

啟動 Neptune 大量載入器工作，將資料從 Amazon S3 儲存貯體載入至 Neptune 資料庫執行個體。請參閱使用 Amazon Neptune 大量載入器擷取資料。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:StartLoaderJob IAM 動作。

請求

- dependencies (在 CLI 中：`--dependencies`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  這是選用參數，可在佇列中一或多個先前工作順利完成時，發出排入佇列的載入請求。

  Neptune 最多可以一次將 64 個載入請求排入佇列，前提是其 `queueRequest` 參數設為 `"TRUE"`。`dependencies` 參數可讓您在一或多個佇列中指定的先前請求順利完成時，執行這類佇列請求。

  例如，如果載入 `Job-A` 和 `Job-B` 彼此獨立，但載入 `Job-C` 須先完成 `Job-A` 和 `Job-B` 才能開始，請依照指示執行：

  1. 以任何順序一個接著一個提交 `load-job-A` 和 `load-job-B`，並儲存其載入 ID。
  2. 使用兩任務其 `dependencies` 欄位中的載入 ID 提交 `load-job-C`：

  Example

  ```
  "dependencies" : ["(job_A_load_id)", "(job_B_load_id)"]
  ```

  由於 `dependencies` 參數的關係，大量載入器在 `Job-A` 和 `Job-B` 順利完成之前，將不會啟動 `Job-C`。如果其中任何一個失敗，則不執行 Job-C，且其狀態將設為 `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`。

  您可以使用這種方式設定多個相依性層級，如此一來，一項任務若失敗，將導致直間接仰賴該任務的所有請求遭取消。

- failOnError (在 CLI 中:`--fail-on-error`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  `failOnError` – 一種旗標，用來切換錯誤時完全停止。

  允許的值：`"TRUE"`、`"FALSE"`。

  預設值：`"TRUE"`。

  此參數設為 `"FALSE"` 時，載入器會嘗試載入指定位置中的所有資料，並略過任何有錯誤的項目。

此參數設為 "TRUE" 時，載入器會在遇到錯誤時立即停止。到該時間點載入的資料仍然存在。

- format (在 CLI 中：`--format`) – 必要：一種格式，類型為：`string` (UTF-8 編碼的字串)。

  資料的格式。如需有關 Neptune `Loader` 命令的資料格式詳細資訊，請參閱載入資料格式。

  允許的值

  - **`csv`** 表示 Gremlin CSV 資料格式。
  - **`opencypher`** 表示 openCypher CSV 資料格式。
  - **`ntriples`** 表示 N-Triples RDF 資料格式。
  - **`nquads`** 表示 N-Quads RDF 資料格式。
  - **`rdfxml`** 表示 RDF\XML RDF 資料格式。
  - **`turtle`** 表示 Turtle RDF 資料格式。
- iamRoleArn (在 CLI 中：`--iam-role-arn`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 資料庫執行個體為了存取 S3 儲存貯體 而擔任之 IAM 角色的 Amazon Resource Name (ARN)。這裡提供的 IAM 角色 ARN 應附加至資料庫叢集 (請參閱將 IAM 角色新增至 Amazon Neptune 叢集。
- mode (在 CLI 中：`--mode`) – 一種模式，類型為：`string` (UTF-8 編碼的字串)。

  載入任務模式。

  允許的值：`RESUME`、`NEW`、`AUTO`。

  預設值：`AUTO`。

  - `RESUME` – 在 RESUME 模式下，載入器會從此來源尋找先前的載入，若找到某載入工作，則會繼續該工作。如果找不到先前的載入任務，載入器便會停止。

    載入程式可避免重新載入已順利載入先前任務的檔案。它只會嘗試處理失敗的檔案。如果從 Neptune 叢集捨棄先前載入的資料，則不會在此模式下重新載入該資料。如果先前的載入工作已從相同來源順利載入所有檔案，則不會重新載入任何工作，且載入器會傳回成功。
  - `NEW` – 在 NEW 模式下，建立新的載入請求，無論先前有任何的載入。您可以使用此模式，在捨棄 Neptune 叢集先前載入的資料之後，從來源重新載入所有資料，或者載入同一來源的可用新資料。

- AUTO – 在 AUTO 模式下，載入器會從相同來源尋找先前的載入任務，若找到一項載入任務，則會繼續該任務，如同 RESUME 模式一般。

  若載入器未能從相同來源找到先前的載入任務，則會從該來源載入所有資料，如同 NEW 模式那樣。

- parallelism (在 CLI 中：`--parallelism`) – 一種平行處理，類型為：`string` (UTF-8 編碼的字串)。

  您可以設定選用 `parallelism` 參數，以減少大量載入程序所使用的執行緒數目。

  允許的值：

  - LOW – 使用的執行緒數目是可用 vCPU 除以 8 後的數字。
  - MEDIUM – 使用的執行緒數目是可用 vCPU 除以 2 後的數字。
  - HIGH – 使用的執行緒數目與可用 vCPU 的數目相同。
  - OVERSUBSCRIBE – 使用的執行緒數目是可用 vCPU 乘以 2 後的數字。如果使用此值，大量載入器會佔用所有可用的資源。

    不過，這並不表示 OVERSUBSCRIBE 設定會產生 100% CPU 使用率。由於載入操作受 I/O 限制，因此預期的最高 CPU 使用率在 60% 到 70% 的範圍內。

  預設值：HIGH

  載入 OpenCypher 資料時，`parallelism` 設定有時可能會導致執行緒之間發生死鎖。發生這種情況時，Neptune 會傳回 LOAD_DATA_DEADLOCK 錯誤。通常，您可以透過將 `parallelism` 設定為較低的設定，並重試載入命令來修正此問題。

- parserConfiguration (在 CLI 中：`--parser-configuration`) – 它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：`string` (UTF-8 編碼的字串)。

  每個值都是字串，類型為：`string` (UTF-8 編碼的字串)。

  `parserConfiguration` – 包含額外剖析器組態值的選用物件。也可選用每個子參數：

  - **`namedGraphUri`** – 未指定圖形時，所有 RDF 格式的預設圖形 (適用於非 quads 格式及無圖形的 NQUAD 項目)。

    預設值為 `https://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`。

- **baseUri** – RDF/XML 和 Turtle 格式的基本 URI。

  預設值為 https://aws.amazon.com/neptune/default。

- **allowEmptyStrings** – 載入 CSV 資料時，Gemlin 使用者必須能夠傳遞空字串值 ("") 做為節點和邊緣屬性。如果 allowEmptyStrings 設定為 false (預設值)，則這類空字串會被視為 null 而不會將其載入。

  如果 allowEmptyStrings 設定為 true，載入器會將空字串視為有效的屬性值，並相應地載入它們。

- queueRequest (在 CLI 中:--queue-request) – 布林值，類型為：boolean (布林值 (true 或 false))。

  此為選用旗標參數，其會指示載入請求能否排入佇列。

  您不必等到載入工作完成才能發出另一個載入工作，因為 Neptune 最多可以一次將 64 個工作排入佇列，前提是其 queueRequest 參數都設為 "TRUE"。工作的佇列順序將為先進先出 (FIFO)。

  如果 queueRequest 參數遭省略或設為 "FALSE"，且已有其他執行中的載入任務，則此載入請求將失敗。

  允許的值："TRUE"、"FALSE"。

  預設值："FALSE"。

- s3BucketRegion (在 CLI 中：--s-3-bucket-region) – 必要：S3BucketRegion，類型為：string (UTF-8 編碼的字串)。

  S3 儲存貯體的 Amazon 區域。這必須與資料庫叢集的 Amazon 區域相符。

- source (在 CLI 中：--source) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  source 參數接受可識別單一檔案、多個檔案、一個資料夾或多個資料夾的 S3 URI。Neptune 會載入任何所指定資料夾中的每個資料檔案。

  URI 可以是下列任何格式。

  - 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)

URI 的 object-key-name 元素相當於 S3 ListObjects API 呼叫中的 prefix 參數。它會識別所指定 S3 儲存貯體中其名稱以該字首開頭的所有物件。這可以是單一檔案或資料夾，也可以是多個檔案和/或資料夾。

一個或多個指定的資料夾可以包含多個頂點檔案和多個邊緣檔案。

- updateSingleCardinalityProperties (在 CLI 中:--update-single-cardinality-properties) – 布林值，類型為：boolean (布林值 (true 或 false))。

  updateSingleCardinalityProperties 是選用參數，可控制大量載入器如何處理單一基數頂點或邊緣屬性的新值。不支援將此用於載入 openCypher 資料。

  允許的值："TRUE"、"FALSE"。

  預設值："FALSE"。

  根據預設，或當 updateSingleCardinalityProperties 明確設定為 "FALSE" 時，載入器會將新值視為錯誤，因為它違反單一基數。

  另一方面，當 updateSingleCardinalityProperties 設為 "TRUE" 時，大量載入器會以新的值取代現有的值。如果在要載入的來源檔案中提供多個邊緣或單一基數頂點屬性值，則大量載入結尾的最終值可以是這些新值的其中之一。載入器只會保證現有的值已由其中一個新值取代。

- userProvidedEdgeIds (在 CLI 中:--user-provided-edge-ids) – 布林值，類型為：boolean (布林值 (true 或 false))。

  只有在載入包含關係 ID 的 OpenCypher 資料時，才需要這個參數。在載入資料中明確提供 OpenCypher 關係 ID 時，必須包含它並設定為 True (建議使用)。

  如果 userProvidedEdgeIds 不存在或設定為 True，則載入中的每個關係檔案中都必須存在一個 :ID 資料行。

  當 userProvidedEdgeIds 存在且設定為 False 時，載入中的關係檔案不得包含 :ID 資料行。相反地，Neptune 載入器會自動為每個關係產生一個 ID。

  明確提供關係 ID 很有用，如此一來，載入器就可以在 CSV 資料中的錯誤完成修正之後繼續載入，而不必重新載入任何已載入的關係。如果尚未明確指派關係 ID，則載入器無法繼續失敗的載入 (如果任何關係檔案必須更正的話)，反而必須重新載入所有關係。

回應

- payload – 必要：它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：`string` (UTF-8 編碼的字串)。

  每個值都是字串，類型為：`string` (UTF-8 編碼的字串)。

  包含 `loadId` 名稱值對，其會提供載入操作的識別符。
- status – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  指示載入工作狀態的傳回碼。

錯誤

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- S3Exception
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetLoaderJobStatus (動作)

此 API 的 AWS CLI 名稱是：`get-loader-job-status`。

取得有關所特定載入工作的狀態資訊。Neptune 會追蹤最近 1,024 個大量載入工作，並會儲存每個工作的最後 10,000 個錯誤詳細資訊。

如需詳細資訊，請參閱 Neptune 載入器 Get-Status API。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetLoaderJobStatus IAM 動作。

請求

- details (在 CLI 中:`--details`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示是否要包含整體狀態之外的詳細資訊的旗標 (TRUE 或 FALSE；預設值為 FALSE)。
- errors (在 CLI 中:`--errors`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示是否要包含發生的錯誤清單的旗標 (TRUE 或FALSE；預設值為 FALSE)。

  錯誤清單會分頁。`page` 和 `errorsPerPage` 參數可讓您翻閱所有錯誤頁面。
- errorsPerPage (在 CLI 中：`--errors-per-page`) – PositiveInteger，類型為：`integer` (帶有正負號的 32 位元整數)，至少 1 ?st?。

  每一頁傳回的錯誤數目 (正整數；預設值為 10)。僅在 `errors` 參數設定為 TRUE 時才有效。
- loadId (在 CLI 中：`--load-id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要取得其狀態之載入工作的載入 ID。
- page (在 CLI 中：`--page`) – PositiveInteger，類型為：`integer` (帶有正負號的 32 位元整數)，至少 1 ?st?。

  錯誤頁號碼 (正整數；預設值為 1)。僅在 `errors` 參數設定為 TRUE 時才有效。

回應

- payload – 必要：文件，類型為：`document` (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

  有關載入工作的狀態資訊，其配置可以如下所示：

  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 – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  請求的 HTTP 回應代碼。

錯誤

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ListLoaderJobs (動作)

此 API 的 AWS CLI 名稱是：`list-loader-jobs`。

為所有作用中的載入器操作擷取 `loadIds` 的清單。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:ListLoaderJobs IAM 動作。

請求

- includeQueuedLoads (在 CLI 中:`--include-queued-loads`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  選用參數，在將此參數設定為 `FALSE`，請求載入 ID 清單時，此參數可以用來排除佇列中載入請求的載入 ID。預設值為 `TRUE`。
- limit (在 CLI 中：`--limit`) – ListLoaderJobsInputLimitInteger，類型為：`integer` (帶有正負號的 32 位元整數)，不小於 1 或大於 100 ?st?s。

  要列出的載入 ID 數目。必須是大於零且不超過 `100` (預設值) 的正整數。

回應

- payload – 必要：LoaderIdResult 物件。

  要求的 工作 ID 清單。
- status – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  傳回工作清單請求的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException

CancelLoaderJob (動作)

> 此 API 的 AWS CLI 名稱是：`cancel-loader-job`。

取消指定的載入工作。這是一個 HTTP `DELETE` 請求。如需詳細資訊，請參閱 Neptune 載入器 Get-Status API。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:CancelLoaderJob IAM 動作。

請求

- loadId (在 CLI 中：`--load-id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要刪除之載入工作的 ID。

回應

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  取消狀態。

錯誤

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

大量載入結構：

LoaderIdResult (結構)

包含載入 ID 的清單。

欄位

- loadIds - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

載入 ID 的清單。

Neptune 串流資料平面 API

串流存取動作：

- GetPropertygraphStream (動作)

串流資料結構：

- PropertygraphRecord (結構)
- PropertygraphData (結構)

GetPropertygraphStream (動作)

此 API 的 AWS CLI 名稱是：`get-propertygraph-stream`。

取得屬性圖的串流。

使用 Neptune 串流功能，您可以產生一系列完整的變更日誌項目，記錄對圖形資料所做的每項變更。`GetPropertygraphStream` 可讓您為屬性圖收集這些變更日誌項目。

必須在您的 Neptune 資料庫叢集上啟用 Neptune 串流功能。若要啟用串流，請將 neptune_streams 資料庫叢集參數設定為 1。

請參閱使用 Neptune 串流即時擷取圖形變更。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetStreamRecords IAM 動作。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，啟用下列其中一個 IAM 動作，取決於查詢：

請注意，您可以使用下列 IAM 內容金鑰來限制屬性圖查詢：

- neptune-db:QueryLanguage:Gremlin
- neptune-db:QueryLanguage:OpenCypher

請參閱 Neptune IAM 資料存取政策陳述式中可用的條件金鑰。

請求

- commitNum (在 CLI 中：`--commit-num`) – Long 整數，類型為：`long` (帶正負號的 64 位元整數)。

  要從 change-log 串流讀取之開始記錄的遞交編號。當 `iteratorType` 是 `AT_SEQUENCE_NUMBER` 或 `AFTER_SEQUENCE_NUMBER` 時，需要此參數，當 `iteratorType` 是 `TRIM_HORIZON` 或 `LATEST` 時，會忽略此參數。
- encoding (在 CLI 中：`--encoding`) – 一種編碼，類型為：`string` (UTF-8 編碼的字串)。

  如果設定為 TRUE，Neptune 會使用 gzip 編碼壓縮回應。
- iteratorType (在 CLI 中：`--iterator-type`) – 一種 IteratorType，類型為：`string` (UTF-8 編碼的字串)。

  可為下列其中一個：
  - `AT_SEQUENCE_NUMBER` – 指出讀取應該從 `commitNum` 和 `opNum` 參數共同指定的事件序號開始。
  - `AFTER_SEQUENCE_NUMBER` – 指出讀取應該在 `commitNum` 和 `opNum` 參數共同指定的事件序號之後立即開始。
  - `TRIM_HORIZON` – 指出讀取應該從系統中的最後一個未修整記錄開始，這是變更日誌串流中最舊的未過期 (尚未刪除) 記錄。
  - `LATEST` – 指出讀取應該從系統中的最新記錄開始，這是變更日誌串流中最新的未過期 (尚未刪除) 記錄。
- limit (在 CLI 中：`--limit`) – GetPropertygraphStreamInputLimitLong，類型為：`long` (帶正負號的 64 位元整數)，不小於 1 或大於 100000 ?st?s。

  指定要傳回的記錄數上限。回應也有無法修改的 10 MB 大小限制，而且會優先於 `limit` 參數中指定的記錄數量。如果達到 10 MB 限制，回應會包含違反閾值的記錄。

  `limit` 的範圍是 1 到 100,000，預設值為 10。
- opNum (在 CLI 中：`--op-num`) – Long 整數，類型為：`long` (帶正負號的 64 位元整數)。

  所指定遞交內的操作序號，即在變更日誌串流資料中開始讀取之處。預設值為 1。

回應

- format – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要傳回之變更記錄的序列化格式。目前，僅支援的值為 `PG_JSON`。

- lastEventId – 必要：它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：`string` (UTF-8 編碼的字串)。

  每個值都是字串，類型為：`string` (UTF-8 編碼的字串)。

  串流回應中上次變更的序列識別符。

  事件 ID 由兩個欄位組成：`commitNum` (識別已變更圖形的交易)，以及 `opNum` (識別該交易內的特定操作)。

  Example

  ```
  "eventId": {
            "commitNum": 12,
            "opNum": 1
          }
  ```

- lastTrxTimestampInMillis – 必要：Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  已請求遞交交易的時間，以毫秒為單位，從 Unix epoch 開始。

- records – 必要：一個 PropertygraphRecord 物件。

  回應中包含之序列化變更日誌串流記錄的陣列。

- totalRecords – 必要：整數，類型為：`integer` (帶正負號的 32 位元整數)。

  回應中的記錄總數。

錯誤

- UnsupportedOperationException
- ExpiredStreamException
- InvalidParameterException
- MemoryLimitExceededException
- StreamRecordsNotFoundException
- ClientTimeoutException

- PreconditionsFailedException
- ThrottlingException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

串流資料結構：

PropertygraphRecord (結構)

屬性圖記錄的結構。

欄位

- commitTimestampInMillis - 這是必要：Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  已請求遞交交易的時間，以毫秒為單位，從 Unix epoch 開始。
- data - 這是必要：PropertygraphData 物件。

  序列化的 Gremlin 或 openCypher 變更記錄。
- eventId - 這是必要：它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：`string` (UTF-8 編碼的字串)。

  每個值都是字串，類型為：`string` (UTF-8 編碼的字串)。

  串流變更記錄中的序列識別符。
- isLastOp - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  僅當此操作是其交易中的最後一個操作時才存在。如果存在，其會設定為 true。這對於確保整個交易被耗用非常有用。
- op - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  已建立變更的操作。

PropertygraphData (結構)

Gremlin 或 openCypher 變更記錄。

欄位

- from - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  如果這是邊緣 (type= e)，則為對應 `from` 頂點或來源節點的 ID。
- id - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  Gremlin 或 openCypher 元素的 ID。
- key - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  屬性名稱 對於元素標籤，這是 `label`。
- to - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  如果這是邊緣 (type= e)，則為對應 `to` 頂點或目標節點的 ID。
- type - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  此 Gremlin 或 openCypher 元素的類型。必須是下列其中一個：
  - **v1** - Gremlin 的頂點標籤，或 openCypher 的節點標籤。
  - **vp** - Gremlin 的頂點屬性，或 openCypher 的節點屬性。
  - **e** - Gremlin 的邊緣和邊緣標籤，或 OpenCypher 的關係和關係類型。
  - **ep** - Gremlin 的邊緣屬性，或 openCypher 的關係屬性。
- value - 這是必要：文件，類型為：`document` (與通訊協定無關的開放內容，由類似 JSON 的資料模型表示)。

  這是 JSON 物件，其中包含值本身的 value 欄位，以及該值之 JSON 資料類型的 datatype 欄位。

  Example

```
"value": {
          "value": "(the new value"),
          "dataType": "(the JSON datatypenew value")
        }
```

Neptune 資料平面統計資料和圖形摘要 API

屬性圖統計資料動作：

- GetPropertygraphStatistics (動作)
- ManagePropertygraphStatistics (動作
- DeletePropertygraphStatistics (動作)
- GetPropertygraphSummary (動作)

統計資料結構：

- 統計資料 (結構)
- StatisticsSummary (結構)
- DeleteStatisticsValueMap (結構)
- RefreshStatisticsIdMap (結構)
- NodeStructure (結構)
- EdgeStructure (結構)
- SubjectStructure (結構)
- PropertygraphSummaryValueMap (結構)
- PropertygraphSummary (結構)

GetPropertygraphStatistics (動作)

> 此 API 的 AWS CLI 名稱是：`get-propertygraph-statistics`。

取得屬性圖統計資料 (Gremlin 和 openCypher)。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetStatisticsStatus IAM 動作。

請求

- 無要求參數。

回應

- payload – 必要：統計資料 物件。

  屬性圖資料的統計資料。

- status – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  請求的 HTTP 傳回碼。如果請求成功，則傳回碼為 200。如需常見錯誤的清單，請參閱 DFE 統計資料請求的常見錯誤代碼。

錯誤

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ManagePropertygraphStatistics (動作

此 API 的 AWS CLI 名稱是：`manage-propertygraph-statistics`。

管理屬性圖統計資料的產生和使用。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:ManageStatistics IAM 動作。

請求

- mode (在 CLI 中：`--mode`) – StatisticsAutoGenerationMode，類型為：`string` (UTF-8 編碼的字串)。

  統計資料產生模式。其中一個：`DISABLE_AUTOCOMPUTE`、`ENABLE_AUTOCOMPUTE` 或 `REFRESH`，最後一個會手動觸發 DFE 統計資料產生。

回應

- payload – RefreshStatisticsIdMap 物件。

  僅針對重新整理模式傳回此項。
- status – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  請求的 HTTP 傳回碼。如果請求成功，則傳回碼為 200。

錯誤

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

DeletePropertygraphStatistics (動作)

此 API 的 AWS CLI 名稱是：`delete-propertygraph-statistics`。

刪除 Gremlin 和 openCypher (屬性圖) 資料的統計資料。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:DeleteStatistics IAM 動作。

請求

- 無要求參數。

回應

- payload – DeleteStatisticsValueMap 物件。

  刪除承載。
- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  取消狀態。
- statusCode – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  HTTP 回應代碼：如果刪除成功，則為 200，或者，如果沒有要刪除的統計資料，則為 204。

錯誤

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException

- MissingParameterException

GetPropertygraphSummary (動作)

此 API 的 AWS CLI 名稱是：`get-propertygraph-summary`。

取得屬性圖的圖形摘要。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetGraphSummary IAM 動作。

請求

- mode (在 CLI 中：`--mode`) – GraphSummaryType，類型為：`string` (UTF-8 編碼的字串)。

  模式可以採取以下兩個值之一：`BASIC` (預設值) 和 `DETAILED`。

回應

- payload – PropertygraphSummaryValueMap 物件。

  包含屬性圖摘要回應的承載。
- statusCode – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  請求的 HTTP 傳回碼。如果請求成功，則傳回碼為 200。

錯誤

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException

- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

統計資料結構：

統計資料 (結構)

包含統計資料資訊。DFE 引擎會在規劃查詢執行時，使用 Neptune 圖形中的資料相關資訊，進行有效的權衡。這項資訊採取統計資料的形式，其中包括所謂的特性集和述詞統計資料，可以引導查詢規劃。請參閱管理要供 Neptune DFE 使用的統計資料。

欄位

- active - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示是否完全啟用 DFE 統計資料產生。
- autoCompute - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  指示是否啟用自動產生統計資料。
- date - 這是 SyntheticTimestamp_date_time，類型為：`string` (UTF-8 編碼的字串)。

  最近產生 DFE 統計資料的 UTC 時間。
- note - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  關於統計資料無效情況下問題的說明。
- signatureInfo - 這是一個 StatisticsSummary 物件。

  包含下列資料的 StatisticsSummary 結構：
  - `signatureCount` - 所有特性集的簽章總數。
  - `instanceCount` - 特性集執行個體的總數。
  - `predicateCount` - 唯一述詞的總數。
- statisticsId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

報告目前統計資料產生執行的 ID。值 -1 指示尚未產生任何統計資料。

StatisticsSummary (結構)

統計資料中所產生之特性集的相關資訊。

欄位

- instanceCount - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  特性集執行個體的總數。
- predicateCount - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  唯一述詞的總數。
- signatureCount - 這是整數，類型為：`integer` (帶正負號的 32 位元整數)。

  所有特性集的簽章總數。

DeleteStatisticsValueMap (結構)

DeleteStatistics 的承載。

欄位

- active - 這是布林值，類型為：`boolean` (布林值 (true 或 false))。

  統計資料的目前狀態。
- statisticsId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  目前正在發生的統計資料產生執行的識別符。

RefreshStatisticsIdMap (結構)

`REFRESH` 模式的統計資料。

欄位

- statisticsId - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

目前正在發生的統計資料產生執行的識別符。

NodeStructure (結構)

節點結構

欄位

- count - 這是 Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  具有此特定結構的節點數目。
- distinctOutgoingEdgeLabels - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此特定結構中存在之不同傳出邊緣標籤的清單。
- nodeProperties - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此特定結構中存在之節點屬性的清單。

EdgeStructure (結構)

邊緣結構。

欄位

- count - 這是 Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  具有此特定結構的邊緣數目。
- edgeProperties - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此特定結構中存在之邊緣屬性的清單。

SubjectStructure (結構)

主旨結構。

欄位

- count - 這是 Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

此特定結構的出現次數。

- predicates - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此特定結構中存在之述詞的清單。

PropertygraphSummaryValueMap (結構)

屬性圖摘要回應的承載。

欄位

- graphSummary - 這是一個 PropertygraphSummary 物件。

  圖形摘要。

- lastStatisticsComputationTime - 這是 SyntheticTimestamp_date_time，類型為：`string` (UTF-8 編碼的字串)。

  Neptune 上次計算統計資料之時間的時間戳記 (採用 ISO 8601 格式)。

- version - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  此圖形摘要回應的版本。

PropertygraphSummary (結構)

圖形摘要 API 會傳回節點和邊緣標籤以及屬性索引鍵的唯讀清單，也會傳回節點、邊緣和屬性的計數。請參閱屬性圖 (PG) 的圖形摘要回應。

欄位

- edgeLabels - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  圖形中不同邊緣標籤的清單。

- edgeProperties - 這是 LongValuedMap 物件。它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：`string` (UTF-8 編碼的字串)。

  每個值都是 Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  圖形中不同邊緣屬性的清單，以及其中使用每個屬性的邊緣計數。

- edgeStructures - 這是 EdgeStructure 物件的陣列。

  僅在請求的模式為 DETAILED 時，此欄位才會存在。它包含邊緣結構的清單。

- nodeLabels - 這是字串，類型為：string (UTF-8 編碼的字串)。

  圖形中不同節點標籤的清單。

- nodeProperties - 這是 LongValuedMap 物件。它是金鑰值對的對應陣列，其中：

  每個金鑰都是字串，類型為：string (UTF-8 編碼的字串)。

  每個值都是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  圖形中不同節點屬性的數目。

- nodeStructures - 這是 NodeStructure 物件的陣列。

  僅在請求的模式為 DETAILED 時，此欄位才會存在。它包含節點結構的清單。

- numEdgeLabels - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  圖形中不同邊緣標籤的數目。

- numEdgeProperties - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  圖形中不同邊緣屬性的數目。

- numEdges - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  圖形中邊緣的數目。

- numNodeLabels - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  圖形中不同節點標籤的數目。

- numNodeProperties - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  圖形中不同節點屬性的清單，以及其中使用每個屬性的節點計數。

- numNodes - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  圖形中節點的數目。

- totalEdgePropertyValues - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

  所有邊緣屬性的使用總數。

- totalNodePropertyValues - 這是 Long 整數，類型為：long (帶有正負號的 64 位元整數)。

所有節點屬性的使用總數。

Neptune ML 資料處理 API

資料處理動作：

- StartMLDataProcessingJob (動作)
- ListMLDataProcessingJobs (動作)
- GetMLDataProcessingJob (動作)
- CancelMLDataProcessingJob (動作)

ML 一般用途結構：

- MlResourceDefinition (結構)
- MlConfigDefinition (結構)

StartMLDataProcessingJob (動作)

此 API 的 AWS CLI 名稱是：`start-ml-data-processing-job`。

建立新的 Neptune ML 資料處理工作，用於處理從 Neptune 匯出的圖形資料以進行訓練。請參閱 `dataprocessing 命令`。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:StartMLModelDataProcessingJob IAM 動作。

請求

- configFileName (在 CLI 中：`--config-file-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  描述如何載入所匯出圖形資料進行訓練的資料規格檔案。Neptune 匯出工具組會自動產生此檔案。預設值為 `training-data-configuration.json`。
- id (在 CLI 中：`--id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新工作的唯一識別符。預設值為自動產生的 UUID。

- inputDataS3Location (在 CLI 中：`--input-data-s3-location`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  Amazon S3 位置的 URI，您想要 SageMaker 在該位置下載執行資料處理工作所需的資料。

- modelType (在 CLI 中：`--model-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune ML 目前支援的兩種模型類型之一：異質圖形模型 (`heterogeneous`) 和知識圖譜 (`kge`)。預設為 none。如果未指定，Neptune ML 會根據資料自動選擇模型類型。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 Amazon Resource Name (ARN)，SageMaker 可以擔任這個 IAM 角色來代表您執行任務。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- previousDataProcessingJobId (在 CLI 中：`--previous-data-processing-job-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  在舊版資料上執行且已完成之資料處理工作的工作 ID。

- processedDataS3Location (在 CLI 中：`--processed-data-s3-location`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  Amazon S3 位置的 URI，您想要 SageMaker 在該位置儲存資料處理工作的結果。

- processingInstanceType (在 CLI 中：`--processing-instance-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料處理期間所使用的 ML 執行個體類型。它的記憶體應該大到足以保留處理後的資料集。預設值為最小的 ml.r5 類型，其記憶體十倍於磁碟上所匯出圖形資料的大小。

- processingInstanceVolumeSizeInGB (在 CLI 中：`--processing-instance-volume-size-in-gb`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  處理執行個體的磁碟區大小。輸入資料和處理後的資料都會儲存在磁碟上，因此磁碟區大小必須大到足以保留這兩個資料集。預設值為 0。如果未指定或指定 0，Neptune ML 會根據資料大小自動選擇磁碟區大小。

- processingTimeOutInSeconds (在 CLI 中：`--processing-time-out-in-seconds`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  資料處理工作的逾時 (以秒為單位)。預設值為 86,400 (1 天)。

- s3OutputEncryptionKMSKey (在 CLI 中：`--s-3-output-encryption-kms-key`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  SageMaker 用來加密處理工作輸出的 Amazon Key Management Service (Amazon KMS) 金鑰。預設為 none。

- sagemakerIamRoleArn (在 CLI 中：`--sagemaker-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  用於 SageMaker 執行之 IAM 角色的 ARN。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- securityGroupIds (在 CLI 中：`--security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  VPC 安全群組 ID。預設值為 None (無)。

- subnets (在 CLI 中：`--subnets`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune VPC 中子網路的 ID。預設值為 None (無)。

- volumeEncryptionKMSKey (在 CLI 中：`--volume-encryption-kms-key`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  SageMaker 用來加密儲存磁碟區上資料的 Amazon Key Management Service (Amazon KMS) 金鑰，而該儲存磁碟區附加到執行訓練工作的 ML 運算執行個體。預設值為 None (無)。

回應

- arn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料處理工作的 ARN。

- creationTimeInMillis - Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  建立新處理工作所需的時間，以毫秒為單位。

- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新資料處理工作的唯一 ID。

錯誤

- UnsupportedOperationException

- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLDataProcessingJobs (動作)

此 API 的 AWS CLI 名稱是：`list-ml-data-processing-jobs`。

傳回 Neptune ML 資料處理工作的清單。請參閱使用 Neptune ML dataprocessing 命令列出作用中的資料處理工作。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:ListMLDataProcessingJobs IAM 動作。

請求

- maxItems (在 CLI 中：`--max-items`) – ListMLDataProcessingJobsInputMaxItemsInteger，類型為：`integer` (帶有正負號的 32 位元整數)，不小於 1 或大於 1024 ?st?s。

  要傳回的項目數目上限 (從 1 到 1024；預設值為 10)。
- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- ids – 字串，類型為：`string` (UTF-8 編碼的字串)。

列出資料處理工作 ID 的頁面。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLDataProcessingJob (動作)

此 API 的 AWS CLI 名稱是：`get-ml-data-processing-job`。

擷取所指定資料處理工作的相關資訊。請參閱 `dataprocessing 命令`。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，允許該叢集中的 neptune-db:neptune-db:GetMLDataProcessingJobStatus 動作。

請求

- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要擷取之資料處理工作的唯一識別符。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此資料處理工作的唯一識別符。
- processingJob – MlResourceDefinition 物件。

  資料處理工作的定義。
- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  資料處理工作的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLDataProcessingJob (動作)

此 API 的 AWS CLI 名稱是：`cancel-ml-data-processing-job`。

取消 Neptune ML 資料處理工作。請參閱 `dataprocessing 命令`。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:CancelMLDataProcessingJob IAM 動作。

請求

- clean (在 CLI 中:`--clean`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `TRUE`，則此旗標指定在工作停止時應刪除所有 Neptune ML S3 成品。預設值為 `FALSE`。

- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  資料處理工作的唯一識別碼。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  取消請求的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ML 一般用途結構：

MlResourceDefinition (結構)

定義 Neptune ML 資源。

欄位

- arn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資源 ARN。
- cloudwatchLogUrl - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資源的 CloudWatch 日誌 URL。
- failureReason - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  失敗時的失敗原因。
- name - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資源名稱。
- outputLocation - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  輸出位置。
- status - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  資源狀態。

MlConfigDefinition (結構)

包含 Neptune ML 組態。

欄位

- arn - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  組態的 ARN。
- name - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  組態名稱。

Neptune ML 模型訓練 API

模型訓練動作：

- StartMLModelTrainingJob (動作)
- ListMLModelTrainingJobs (動作)
- GetMLModelTrainingJob (動作)
- CancelMLModelTrainingJob (動作)

模型訓練結構：

- CustomModelTrainingParameters (結構)

StartMLModelTrainingJob (動作)

此 API 的 AWS CLI 名稱是：`start-ml-model-training-job`。

建立新的 Neptune ML 模型訓練工作。請參閱使用 `modeltraining` 命令進行模型訓練。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:StartMLModelTrainingJob IAM 動作。

請求

- baseProcessingInstanceType (在 CLI 中：`--base-processing-instance-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  用於準備和管理 ML 模型訓練的 ML 執行個體類型。這是根據記憶體需求選擇的 CPU 執行個體，用於處理訓練資料和模型。
- customModelTrainingParameters (在 CLI 中：`--custom-model-training-parameters`) – CustomModelTrainingParameters 物件。

  自訂模型訓練的組態。這是 JSON 物件。
- dataProcessingJobId (在 CLI 中：`--data-processing-job-id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  已完成資料處理工作的工作 ID，該工作已建立訓練將使用的資料。

- enableManagedSpotTraining (在 CLI 中:`--enable-managed-spot-training`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  使用 Amazon Elastic Compute Cloud Spot 執行個體，將訓練機器學習模型的成本最佳化。預設值為 `False`。

- id (在 CLI 中：`--id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新工作的唯一識別符。預設值為自動產生的 UUID。

- maxHPONumberOfTrainingJobs (在 CLI 中：`--max-hpo-number-of-training-jobs`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  要對超參數調校工作啟動的訓練工作總數上限。預設為 2。Neptune ML 會自動調校機器學習模型的超參數。若要取得效能良好的模型，請至少使用 10 個工作 (換句話說，將 `maxHPONumberOfTrainingJobs` 設為 10)。一般來說，調校執行越多，結果越好。

- maxHPOParallelTrainingJobs (在 CLI 中：`--max-hpo-parallel-training-jobs`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  要對超參數調校工作啟動的並行訓練工作數目上限。預設為 2。您可以執行的並行工作數目受制於訓練執行個體上可用的資源。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- previousModelTrainingJobId (在 CLI 中：`--previous-model-training-job-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  已完成模型訓練工作的工作 ID，您想要根據更新的資料以增量方式更新此工作。

- s3OutputEncryptionKMSKey (在 CLI 中：`--s-3-output-encryption-kms-key`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  SageMaker 用來加密處理工作輸出的 Amazon Key Management Service (KMS) 金鑰。預設為 none。

- sagemakerIamRoleArn (在 CLI 中：`--sagemaker-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  SageMaker 執行的 IAM 角色的 ARN。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- securityGroupIds (在 CLI 中：`--security-group-ids`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  VPC 安全群組 ID。預設值為 None (無)。

- subnets (在 CLI 中：`--subnets`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  Neptune VPC 中子網路的 ID。預設值為 None (無)。

- trainingInstanceType (在 CLI 中：`--training-instance-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  用於模型訓練的 ML 執行個體類型。所有 Neptune ML 模型都支援 CPU、GPU 和多 GPU 訓練。預設值為 `ml.p3.2xlarge`。選擇適合訓練的執行個體類型取決於工作類型、圖形大小和您的預算。

- trainingInstanceVolumeSizeInGB (在 CLI 中：`--training-instance-volume-size-in-gb`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  訓練執行個體的磁碟區大小。輸入資料和輸出模型都會儲存在磁碟上，因此磁碟區大小必須大到足以保留這兩個資料集。預設值為 0。如果未指定或指定 0，Neptune ML 會根據資料處理步驟中產生的建議選取磁碟區大小。

- trainingTimeOutInSeconds (在 CLI 中：`--training-time-out-in-seconds`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  訓練工作的逾時 (以秒為單位)。預設值為 86,400 (1 天)。

- trainModelS3Location (在 CLI 中：`--train-model-s3-location`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  Amazon S3 中要儲存模型成品的位置。

- volumeEncryptionKMSKey (在 CLI 中：`--volume-encryption-kms-key`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  SageMaker 用來加密儲存磁碟區上資料的 Amazon Key Management Service (KMS) 金鑰，而該儲存磁碟區附加到執行訓練工作的 ML 運算執行個體。預設值為 None (無)。

回應

- arn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新模型訓練工作的 ARN。

- creationTimeInMillis - Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

模型訓練工作建立時間，以毫秒為單位。

- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新模型訓練工作的唯一 ID。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLModelTrainingJobs (動作)

此 API 的 AWS CLI 名稱是：`list-ml-model-training-jobs`。

列出 Neptune ML 模型訓練工作。請參閱使用 `modeltraining` 命令進行模型訓練。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:neptune-db:ListMLModelTrainingJobs IAM 動作。

請求

- maxItems (在 CLI 中：`--max-items`) – ListMLModelTrainingJobsInputMaxItemsInteger，類型為：`integer` (帶有正負號的 32 位元整數)，不小於 1 或大於 1024 ?st?s。

  要傳回的項目數目上限 (從 1 到 1024；預設值為 10)。
- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- ids – 字串，類型為：`string` (UTF-8 編碼的字串)。

  模型訓練工作 ID 的清單頁面。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLModelTrainingJob (動作)

此 API 的 AWS CLI 名稱是：`get-ml-model-training-job`。

擷取 Neptune ML 模型訓練工作的相關資訊。請參閱使用 `modeltraining` 命令進行模型訓練。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetMLModelTrainingJobStatus IAM 動作。

請求

- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

要擷取的模型訓練工作的唯一識別符。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- hpoJob – MlResourceDefinition 物件。

  HPO 工作。

- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  此模型訓練工作的唯一識別符。

- mlModels – 一個 MlConfigDefinition 物件陣列。

  所使用之 ML 模型的組態清單。

- modelTransformJob – MlResourceDefinition 物件。

  模型轉換工作。

- processingJob – MlResourceDefinition 物件。

  資料處理工作。

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  模型訓練工作的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException

- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLModelTrainingJob (動作)

此 API 的 AWS CLI 名稱是：`cancel-ml-model-training-job`。

取消 Neptune ML 模型訓練工作。請參閱使用 `modeltraining` 命令進行模型訓練。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:CancelMLModelTrainingJob IAM 動作。

請求

- clean (在 CLI 中:`--clean`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 TRUE，則此旗標指定在工作停止時應刪除所有 Amazon S3 成品。預設值為 FALSE。
- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要取消的模型訓練工作的唯一識別符。
- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  取消的狀態。

錯誤

- UnsupportedOperationException

- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

模型訓練結構：

CustomModelTrainingParameters (結構)

包含自訂模型訓練參數。請參閱 Neptune ML 中的自訂模型。

欄位

- sourceS3DirectoryPath - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  此路徑通往實作您模型之 Python 模組所在的 Amazon S3 位置。這必須指向有效的現有 Amazon S3 位置，其中至少包含訓練指令碼、轉換指令碼和 `model-hpo-configuration.json` 檔案。
- trainingEntryPointScript - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指令碼模組中的進入點名稱，該指令碼會執行模型訓練，並接受超參數作為命令列引數 (包括固定的超參數)。預設值為 `training.py`。
- transformEntryPointScript - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指令碼模組中的進入點名稱，該指令碼應在識別了超參數搜尋中的最佳模型之後執行，以計算模型部署所需的模型成品。它應該能夠在沒有命令列參數的情況下執行。預設值為 `transform.py`。

Neptune ML 模型轉換 API

模型轉換動作：

- StartMLModelTransformJob (動作)
- ListMLModelTransformJobs (動作)
- GetMLModelTransformJob (動作)
- CancelMLModelTransformJob (動作)

模型轉換結構：

- CustomModelTransformParameters (結構)

StartMLModelTransformJob (動作)

此 API 的 AWS CLI 名稱是：`start-ml-model-transform-job`。

建立新模型轉換工作。請參閱使用訓練後的模型來產生新的模型成品。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:StartMLModelTransformJob IAM 動作。

請求

- baseProcessingInstanceType (在 CLI 中：`--base-processing-instance-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  用於準備和管理 ML 模型訓練的 ML 執行個體類型。這是根據記憶體需求選擇的 ML 運算執行個體，用於處理訓練資料和模型。
- baseProcessingInstanceVolumeSizeInGB (在 CLI 中：`--base-processing-instance-volume-size-in-gb`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  訓練執行個體的磁碟區大小 (以 GB 為單位)。預設值為 0。輸入資料和輸出模型都會儲存在磁碟上，因此磁碟區大小必須大到足以保留這兩個資料集。如果未指定或指定 0，Neptune ML 會根據資料處理步驟中產生的建議選取磁碟區大小。
- customModelTransformParameters (在 CLI 中：`--custom-model-transform-parameters`) – CustomModelTransformParameters 物件。

  使用自訂模型進行模型轉換的組態資訊。`customModelTransformParameters` 物件包含下列欄位，其具有的值必須與訓練工作中儲存的模型參數相容：
- dataProcessingJobId (在 CLI 中：`--data-processing-job-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

已完成資料處理工作的工作 ID。您必須包含 dataProcessingJobId 和 mlModelTrainingJobId 或 trainingJobName。

- id (在 CLI 中：--id) – 字串，類型為：string (UTF-8 編碼的字串)。

  新工作的唯一識別符。預設值為自動產生的 UUID。

- mlModelTrainingJobId (在 CLI 中：--ml-model-training-job-id) – 字串，類型為：string (UTF-8 編碼的字串)。

  已完成模型訓練工作的工作 ID。您必須包含 dataProcessingJobId 和 mlModelTrainingJobId 或 trainingJobName。

- modelTransformOutputS3Location (在 CLI 中：--model-transform-output-s3-location) – 必要：字串，類型為：string (UTF-8 編碼的字串)。

  Amazon S3 中要儲存模型成品的位置。

- neptuneIamRoleArn (在 CLI 中：--neptune-iam-role-arn) – 字串，類型為：string (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- s3OutputEncryptionKMSKey (在 CLI 中：--s-3-output-encryption-kms-key) – 字串，類型為：string (UTF-8 編碼的字串)。

  SageMaker 用來加密處理工作輸出的 Amazon Key Management Service (KMS) 金鑰。預設為 none。

- sagemakerIamRoleArn (在 CLI 中：--sagemaker-iam-role-arn) – 字串，類型為：string (UTF-8 編碼的字串)。

  用於 SageMaker 執行之 IAM 角色的 ARN。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

- securityGroupIds (在 CLI 中：--security-group-ids) – 字串，類型為：string (UTF-8 編碼的字串)。

  VPC 安全群組 ID。預設值為 None (無)。

- subnets (在 CLI 中：--subnets) – 字串，類型為：string (UTF-8 編碼的字串)。

  Neptune VPC 中子網路的 ID。預設值為 None (無)。

- trainingJobName (在 CLI 中：`--training-job-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  已完成 SageMaker 訓練工作的名稱。您必須包含 `dataProcessingJobId` 和 `mlModelTrainingJobId` 或 `trainingJobName`。

- volumeEncryptionKMSKey (在 CLI 中：`--volume-encryption-kms-key`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  SageMaker 用來加密儲存磁碟區上資料的 Amazon Key Management Service (KMS) 金鑰，而該儲存磁碟區附加到執行訓練工作的 ML 運算執行個體。預設值為 None (無)。

回應

- arn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  模型轉換工作的 ARN。

- creationTimeInMillis - Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  模型轉換工作的建立時間，以毫秒為單位。

- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新模型轉換工作的唯一 ID。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException

ListMLModelTransformJobs (動作)

此 API 的 AWS CLI 名稱是：`list-ml-model-transform-jobs`。

傳回模型轉換工作 ID 的清單。請參閱使用訓練後的模型來產生新的模型成品。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:ListMLModelTransformJobs IAM 動作。

請求

- maxItems (在 CLI 中：`--max-items`) – ListMLModelTransformJobsInputMaxItemsInteger，類型為：`integer` (帶有正負號的 32 位元整數)，不小於 1 或大於 1024 ?st?s。

  要傳回的項目數目上限 (從 1 到 1024；預設值為 10)。
- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- ids – 字串，類型為：`string` (UTF-8 編碼的字串)。

  來自模型轉換 ID 清單中的頁面。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException

- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLModelTransformJob (動作)

此 API 的 AWS CLI 名稱是：`get-ml-model-transform-job`。

取得所指定模型轉換工作的相關資訊。請參閱使用訓練後的模型來產生新的模型成品。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetMLModelTransformJobStatus IAM 動作。

請求

- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要擷取之模型轉換工作的唯一識別符。
- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- baseProcessingJob – MlResourceDefinition 物件。

  基礎資料處理工作。
- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  要擷取之模型轉換工作的唯一識別符。
- models – 一個 MlConfigDefinition 物件陣列。

  所使用之模型的組態資訊清單。
- remoteModelTransformJob – MlResourceDefinition 物件。

遠端模型轉換工作。

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  模型轉換工作的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

CancelMLModelTransformJob (動作)

此 API 的 AWS CLI 名稱是：`cancel-ml-model-transform-job`。

取消指定的模型轉換工作。請參閱使用訓練後的模型來產生新的模型成品。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:CancelMLModelTransformJob IAM 動作。

請求

- clean (在 CLI 中:`--clean`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果此旗標設定為 `TRUE`，則應在工作停止時刪除所有 Neptune ML S3 成品。預設值為 `FALSE`。
- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  要取消之模型轉換工作的唯一 ID。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  取消的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

模型轉換結構：

CustomModelTransformParameters (結構)

包含自訂模型轉換參數。請參閱使用訓練後的模型來產生新的模型成品。

欄位

- sourceS3DirectoryPath - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

此路徑通往實作您模型之 Python 模組所在的 Amazon S3 位置。這必須指向有效的現有 Amazon S3 位置，其中至少包含訓練指令碼、轉換指令碼和 `model-hpo-configuration.json` 檔案。

- transformEntryPointScript - 這是字串，類型為：`string` (UTF-8 編碼的字串)。

  指令碼模組中的進入點名稱，該指令碼應在識別了超參數搜尋中的最佳模型之後執行，以計算模型部署所需的模型成品。它應該能夠在沒有命令列參數的情況下執行。預設值為 `transform.py`。

Neptune ML 推論端點 API

推論端點動作：

- CreateMLEndpoint (動作)
- ListMLEndpoints (動作)
- GetMLEndpoint (動作)
- DeleteMLEndpoint (動作)

CreateMLEndpoint (動作)

此 API 的 AWS CLI 名稱是：`create-ml-endpoint`。

建立新的 Neptune ML 推論端點，此推論端點可讓您查詢模型訓練程序所建構的特定模型。請參閱使用端點命令管理推論端點。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:CreateMLEndpoint IAM 動作。

請求

- id (在 CLI 中：`--id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新推論端點的唯一識別符。預設值為自動產生的時間戳記名稱。

- instanceCount (在 CLI 中：`--instance-count`) – 整數，類型為：`integer` (帶正負號的 32 位元整數)。

  要部署到端點進行預測的 Amazon EC2 執行個體數量下限。預設為 1

- instanceType (在 CLI 中：`--instance-type`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

用於線上服務的 Neptune ML 執行個體類型。預設值為 `ml.m5.xlarge`。為推論端點選擇 ML 執行個體，取決於任務類型、圖形大小以及您的預算。

- mlModelTrainingJobId (在 CLI 中：`--ml-model-training-job-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  已完成模型訓練工作的工作 ID，該工作已建立推論端點將指向的模型。您必須提供 `mlModelTrainingJobId` 或 `mlModelTransformJobId`。

- mlModelTransformJobId (在 CLI 中：`--ml-model-transform-job-id`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  已完成模型轉換工作的工作 ID。您必須提供 `mlModelTrainingJobId` 或 `mlModelTransformJobId`。

- modelName (在 CLI 中：`--model-name`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  用於訓練的模型類型。根據預設，Neptune ML 模型會自動以資料處理中使用的 `modelType` 為基礎，但您可以在這裡指定不同的模型類型。預設值為 `rgcn` 用於異質圖和 `kge` 用於知識圖譜。異質圖的唯一有效值是 `rgcn`。知識圖譜的有效值為：`kge`、`transe`、`distmult`、和 `rotate`。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會擲回錯誤。

- update (在 CLI 中:`--update`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果設定為 `true`，`update` 指示這是更新請求。預設值為 `false`。您必須提供 `mlModelTrainingJobId` 或 `mlModelTransformJobId`。

- volumeEncryptionKMSKey (在 CLI 中：`--volume-encryption-kms-key`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  SageMaker 用來加密儲存磁碟區上資料的 Amazon Key Management Service (Amazon KMS) 金鑰，而該儲存磁碟區附加到執行訓練工作的 ML 運算執行個體。預設值為 None (無)。

回應

- arn – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新推論端點的 ARN。

- creationTimeInMillis - Long 整數，類型為：`long` (帶有正負號的 64 位元整數)。

  端點建立時間，以毫秒為單位。

- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  新推論端點的唯一 ID。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

ListMLEndpoints (動作)

此 API 的 AWS CLI 名稱是：`list-ml-endpoints`。

列出現有的推論端點。請參閱使用端點命令管理推論端點。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:ListMLEndpoints IAM 動作。

請求

- maxItems (在 CLI 中：`--max-items`) – ListMLEndpointsInputMaxItemsInteger，類型為：`integer` (帶有正負號的 32 位元整數)，不小於 1 或大於 1024 ?st?s。

  要傳回的項目數目上限 (從 1 到 1024；預設值為 10)。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- ids – 字串，類型為：`string` (UTF-8 編碼的字串)。

  來自推論端點 ID 清單的頁面。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

GetMLEndpoint (動作)

此 API 的 AWS CLI 名稱是：`get-ml-endpoint`。

擷取有關推論端點的詳細資訊。請參閱使用端點命令管理推論端點。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:GetMLEndpointStatus IAM 動作。

請求

- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  推論端點的唯一識別符。

- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會發生錯誤。

回應

- endpoint – MlResourceDefinition 物件。

  端點定義。

- endpointConfig – MlConfigDefinition 物件。

  端點組態

- id – 字串，類型為：`string` (UTF-8 編碼的字串)。

  推論端點的唯一識別符。

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  推論端點的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException

- TooManyRequestsException

DeleteMLEndpoint (動作)

此 API 的 AWS CLI 名稱是：`delete-ml-endpoint`。

取消建立 Neptune ML 推論端點。請參閱使用端點命令管理推論端點。

在已啟用 IAM 身分驗證的 Neptune 叢集中調用此操作時，發出請求的 IAM 使用者或角色必須附加一個政策，在該叢集中允許 neptune-db:DeleteMLEndpoint IAM 動作。

請求

- clean (在 CLI 中:`--clean`) – 布林值，類型為：`boolean` (布林值 (true 或 false))。

  如果此旗標設定為 TRUE，則應在工作停止時刪除所有 Neptune ML S3 成品。預設值為 FALSE。
- id (在 CLI 中：`--id`) – 必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  推論端點的唯一識別符。
- neptuneIamRoleArn (在 CLI 中：`--neptune-iam-role-arn`) – 字串，類型為：`string` (UTF-8 編碼的字串)。

  IAM 角色的 ARN，此角色可讓 Neptune 存取 SageMaker 和 Amazon S3 資源。這必須列示在您的資料庫叢集參數群組中，否則會擲回錯誤。

回應

- status – 字串，類型為：`string` (UTF-8 編碼的字串)。

  取消的狀態。

錯誤

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException

- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Neptune 資料平面 API 例外狀況

例外狀況：

- AccessDeniedException (結構)
- BadRequestException (結構)
- BulkLoadIdNotFoundException (結構)
- CancelledByUserException (結構)
- ClientTimeoutException (結構)
- ConcurrentModificationException (結構)
- ConstraintViolationException (結構)
- ExpiredStreamException (結構)
- FailureByQueryException (結構)
- IllegalArgumentException (結構)
- InternalFailureException (結構)
- InvalidArgumentException (結構)
- InvalidNumericDataException (結構)
- InvalidParameterException (結構)
- LoadUrlAccessDeniedException (結構)
- MalformedQueryException (結構)
- MemoryLimitExceededException (結構)
- MethodNotAllowedException (結構)
- MissingParameterException (結構)
- MLResourceNotFoundException (結構)

- ParsingException (結構)
- PreconditionsFailedException (結構)
- QueryLimitExceededException (結構)
- QueryLimitException (結構)
- QueryTooLargeException (結構)
- ReadOnlyViolationException (結構)
- S3Exception (結構)
- ServerShutdownException (結構)
- StatisticsNotAvailableException (結構)
- StreamRecordsNotFoundException (結構)
- ThrottlingException (結構)
- TimeLimitExceededException (結構)
- TooManyRequestsException (結構)
- UnsupportedOperationException (結構)
- UnloadUrlAccessDeniedException (結構)

AccessDeniedException (結構)

在身分驗證或授權失敗的情況下引發。

欄位

- code - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：`string` (UTF-8 編碼的字串)。

  有問題請求的 ID。

BadRequestException (結構)

當提交無法處理的請求時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  錯誤請求的 ID。

BulkLoadIdNotFoundException (結構)

找不到指定的大量載入工作 ID 時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  找不到的大量載入工作 ID。

CancelledByUserException (結構)

使用者取消請求時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。

- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

ClientTimeoutException (結構)

請求在用戶端發生逾時時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

ConcurrentModificationException (結構)

當請求嘗試修改目前正由另一個程序修改的資料時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

ConstraintViolationException (結構)

當請求欄位中的值未滿足必要的限制條件時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

ExpiredStreamException (結構)

當請求嘗試存取已過期的串流時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

FailureByQueryException (結構)

請求失敗時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。

- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

IllegalArgumentException (結構)

當不支援請求中的參數時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

InternalFailureException (結構)

當請求的處理意外失敗時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

InvalidArgumentException (結構)

當請求中的引數具有無效值時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

InvalidNumericDataException (結構)

當服務請求時遇到無效的數值資料時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

InvalidParameterException (結構)

當參數值無效時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。

- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  包含無效參數之請求的 ID。

LoadUrlAccessDeniedException (結構)

當存取指定的載入 URL 遭拒時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

MalformedQueryException (結構)

當提交語法不正確或未通過額外驗證的查詢時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  格式錯誤的查詢請求 ID。

MemoryLimitExceededException (結構)

當請求由於記憶體資源不足而失敗時引發。可以重試該請求。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  失敗請求的 ID。

MethodNotAllowedException (結構)

當正要使用的端點不支援請求所使用的 HTTP 方法時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

MissingParameterException (結構)

缺少必要參數時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。

- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  缺少參數之請求的 ID。

MLResourceNotFoundException (結構)

當找不到指定的機器學習資源時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

ParsingException (結構)

遇到剖析問題時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

PreconditionsFailedException (結構)

當處理請求的前提條件未得到滿足時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

QueryLimitExceededException (結構)

當作用中查詢的數目超過了伺服器可以處理的數目時引發。當系統不太忙時，可以重試有問題的查詢。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  超出限制之請求的 ID。

QueryLimitException (結構)

當查詢的大小超過系統限制時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。

- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  超出限制之請求的 ID。

QueryTooLargeException (結構)

當查詢的本文太大時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  太大請求的 ID。

ReadOnlyViolationException (結構)

當請求嘗試寫入唯讀資源時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  缺少參數之請求的 ID。

S3Exception (結構)

當存取 Amazon S3 若發生問題時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

ServerShutdownException (結構)

當伺服器關閉，同時處理請求時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

StatisticsNotAvailableException (結構)

當滿足請求所需的統計資料無法取得時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。

- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

StreamRecordsNotFoundException (結構)

當找不到查詢請求的串流記錄時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

ThrottlingException (結構)

請求速率超出輸送量上限時引發。在遇到此例外狀況之後，可以重試請求。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  由於此原因而無法處理之請求的 ID。

TimeLimitExceededException (結構)

當操作超過允許的時間限制時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  由於此原因而無法處理之請求的 ID。

TooManyRequestsException (結構)

當正在處理的請求數目超過限制時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  由於此原因而無法處理之請求的 ID。

UnsupportedOperationException (結構)

當請求嘗試啟動不支援的操作時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。

- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

UnloadUrlAccessDeniedException (結構)

當存取本身為卸載目標的 URL 遭拒時引發。

欄位

- code - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  與例外狀況一起傳回的 HTTP 狀態碼。
- detailedMessage - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  描述問題的詳細訊息。
- requestId - 這是必要：字串，類型為：string (UTF-8 編碼的字串)。

  有問題請求的 ID。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。