AWS Well-Architected Framework

# Pilar de fiabilidad

# Pilar de fiabilidad: AWS Well-Architected Framework

# Table of Contents

# Pilar de fiabilidad: AWS Well-Architected Framework

Fecha de publicación: 27 de junio de 2024 (Revisiones del documento)

Este documento se centra en el pilar de fiabilidad de AWS Well-Architected Framework. Proporciona una guía para ayudar a los clientes a aplicar las prácticas recomendadas a la hora de diseñar, entregar y mantener los entornos de Amazon Web Services (AWS).

## Introducción

El marco AWS Well-Architected Framework le ayuda a comprender las ventajas y desventajas de las decisiones que toma al crear cargas de trabajo en AWS. El uso del marco le permitirá conocer las prácticas recomendadas de arquitectura para diseñar y operar cargas de trabajo en la nube que sean fiables, seguras, eficaces, rentables y sostenibles. Proporciona una forma de medir sus arquitecturas de forma constante en función de las prácticas recomendadas y de identificar áreas de mejora. Creemos que contar con cargas de trabajo bien diseñadas aumenta en gran medida la probabilidad de éxito empresarial.

AWS Well-Architected Framework se basa en seis pilares:

- Excelencia operativa
- Seguridad
- Fiabilidad
- Eficiencia del rendimiento
- Optimización de costes
- Sostenibilidad

El presente documento se centra en el pilar de fiabilidad y cómo aplicarlo a sus soluciones. Lograr la fiabilidad puede ser un reto en los entornos tradicionales de las instalaciones debido a factores de puntos únicos de errores, la falta de automatización y la falta de elasticidad. Al adoptar las prácticas de este documento, construirá arquitecturas con fundamentos sólidos, una arquitectura resiliente, una administración de cambios consistente y procesos de recuperación de errores comprobados.

Este documento está destinado a aquellos que ocupan puestos en tecnología, como los directores de tecnología (CTO), arquitectos, desarrolladores y miembros del equipo de operaciones. Después de leer este documento, comprenderá mejor las prácticas recomendadas y estrategias de AWS

que puede utilizar cuando diseñe arquitecturas en la nube para lograr que sean fiables. El presente documento incluye detalles de implementación de alto nivel y patrones de arquitectura, así como referencias a recursos adicionales.

# Fiabilidad

El pilar de fiabilidad abarca la capacidad de una carga de trabajo para realizar su función prevista de forma correcta y coherente cuando se espera que lo haga. Esto incluye la capacidad de utilizar y probar la carga de trabajo a lo largo de todo su ciclo de vida. En este documento se incluye orientación de prácticas recomendadas para la implementación de cargas de trabajo fiables en AWS.

Temas



## Modelo de responsabilidad compartida para la resiliencia

La resiliencia es una responsabilidad compartida entre AWS y usted. Es importante que entienda cómo la recuperación de desastres (DR) y la disponibilidad, como parte de la resiliencia, operan bajo este modelo compartido.

Responsabilidad de AWS: la resiliencia de la nube

AWS es responsable de la resiliencia de la infraestructura en la que se ejecutan todos los servicios que se ofrecen en la Nube de AWS. Esta infraestructura comprende el hardware, el software, las redes y las instalaciones que ejecutan los servicios de Nube de AWS. AWS realiza esfuerzos comercialmente razonables para que estos servicios de Nube de AWS estén disponibles, lo que garantiza que la disponibilidad del servicio cumpla o supere los acuerdos de nivel de servicio (SLA) de AWS.

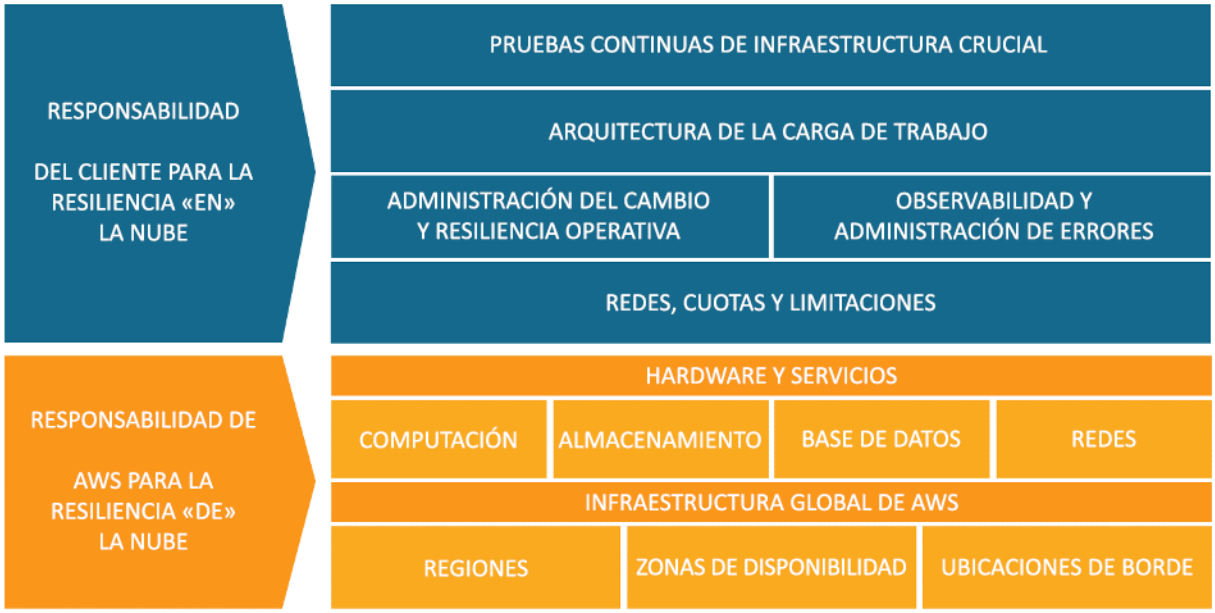
La infraestructura global de AWS se ha diseñado para permitir a los clientes crear arquitecturas de cargas de trabajo altamente resilientes. Cada Región de AWS está totalmente aislada y comprende numerosas zonas de disponibilidad, que son particiones físicamente aisladas de la infraestructura. Las zonas de disponibilidad aíslan errores que podrían afectar la resiliencia de la carga de trabajo, y les impide repercutir en otras zonas en la región. Al mismo tiempo, todas las AZ de una Región de AWS están interconectadas con ancho de banda alto y redes de baja latencia, y utilizan fibra metropolitana dedicada y totalmente redundante, lo que ofrece una conexión de red de baja latencia y alto rendimiento entre las zonas. Todo el tráfico entre las zonas está cifrado. El rendimiento de

la red es suficiente para realizar la replicación sincrónica entre zonas. Cuando una aplicación está particionada en varias AZ, las empresas estarán mejor aisladas y contarán con mayor protección frente a problemas como interrupciones de alimentación eléctrica, tormentas eléctricas, tornados, huracanes, etc.

Responsabilidad del cliente: la seguridad en la nube

Su responsabilidad vendrá determinada por los servicios de Nube de AWS que elija. Esto determina la cantidad de trabajo de configuración que debe realizar como parte de sus responsabilidades de resiliencia. Por ejemplo, un servicio como Amazon Elastic Compute Cloud (Amazon EC2) exige que el cliente lleve a cabo todas las tareas necesarias de configuración y administración de la resiliencia. Los clientes que despliegan instancias Amazon EC2 son responsables de desplegar las instancias Amazon EC2 en numerosas ubicaciones (como las zonas de disponibilidad de AWS), implementar la autorrecuperación mediante servicio como Auto Scaling y usar las prácticas recomendadas de arquitectura de carga de trabajo resiliente para las aplicaciones instaladas en las instancias. En el caso de los servicios administrados, como Amazon S3 y Amazon DynamoDB, AWS se encarga de administrar la capa de infraestructura, el sistema operativo y las plataformas, mientras que los clientes acceden a los puntos de conexión para guardar y recuperar información. Usted es responsable de administrar la resiliencia de sus datos, incluidas las estrategias de copia de seguridad, control de versiones y replicación.

El despliegue de la carga de trabajo en varias zonas de disponibilidad de una Región de AWS forma parte de una estrategia de alta disponibilidad diseñada para proteger las cargas de trabajo al aislar los problemas en una zona de disponibilidad, que utiliza la redundancia de las demás zonas de disponibilidad para seguir atendiendo las solicitudes. Una arquitectura de varias zonas de disponibilidad también forma parte de una estrategia de DR diseñada para que las cargas de trabajo estén mejor aisladas y protegidas de problemas como interrupciones de alimentación eléctrica, tormentas eléctricas, tornados, terremotos, etc. Las estrategias de DR también pueden hacer uso de varias Regiones de AWS. Por ejemplo, en una configuración activa-pasiva, el servicio para la carga de trabajo pasa de su región activa a su región de DR si la región activa ya no puede atender solicitudes.




Responsabilidad de la resiliencia en la nube y de la nube para los clientes y AWS.

Puede utilizar los servicios de AWS para lograr sus objetivos de resiliencia. Como cliente, es responsable de la administración de los siguientes aspectos de su sistema para lograr la resiliencia en la nube. Para obtener más información sobre cada servicio en particular, consulte la documentación de AWS.

Redes, cuotas y limitaciones

- Las prácticas recomendadas para esta área del modelo de responsabilidad compartida se describen en detalle en Fundamentos.
- Al planificar su arquitectura deje espacio suficiente para escalar y comprenda las cuotas de servicio y las limitaciones de los servicios que incluya, basándose en los aumentos de solicitud de carga previstos cuando proceda.
- Diseñe su topología de red para que sea altamente disponible, redundante y escalable.

Administración del cambio y resiliencia operativa

- La administración del cambio incluye cómo introducir el cambio en su entorno y cómo administrarlo. La implementación del cambio exige crear y mantener actualizados runbooks y estrategias de despliegue para su aplicación e infraestructura.

- Una estrategia resiliente para supervisar los recursos de carga de trabajo tiene en cuenta todos los componentes, entre ellos, las métricas técnicas y empresariales, las notificaciones, la automatización y el análisis.
- Las cargas de trabajo en la nube deben adaptarse a los cambios en el escalado de la demanda como reacción a las deficiencias o fluctuaciones en el uso.

### Observabilidad y administración de errores

- Observar los errores a través de la supervisión es necesario para automatizar la recuperación, de modo que sus cargas de trabajo puedan resistir los errores de los componentes.
- La administración de errores requiere copias de seguridad de los datos, aplicar las prácticas recomendadas para que la carga de trabajo resista los fallos de los componentes y planificar la recuperación de desastres.

### Arquitectura de la carga de trabajo

- La arquitectura de la carga de trabajo incluye el diseño de servicios en torno a dominios empresariales, la aplicación de SOA y el diseño de sistemas distribuidos para evitar errores, y la incorporación de capacidades como limitación, reintentos, administración de colas, tiempos de espera y recursos de emergencia.
- Confíe en las soluciones probadas de AWS, la Amazon Builders Library y los patrones sin servidor para alinearse con las prácticas recomendadas y poner en marcha las implementaciones.
- Utilice la mejora continua para descomponer su sistema en servicios distribuidos para escalar e innovar con más rapidez. Use la guía de microservicios de AWS y las opciones de servicios administrados para simplificar y acelerar su capacidad de introducir cambios e innovar.

### Pruebas continuas de infraestructura crucial

- Probar la fiabilidad significa realizar pruebas a nivel funcional, de rendimiento y de caos, así como adoptar prácticas de análisis de incidentes y de día de juego para adquirir experiencia en la resolución de problemas que no se comprenden bien.
- Tanto para las aplicaciones en la nube como para las híbridas, saber cómo se comporta su aplicación cuando surgen problemas o se caen los componentes le permite recuperarse de las interrupciones de forma rápida y fiable.

- Cree y documente experimentos repetibles para comprender cómo se comporta su sistema cuando las cosas no funcionan como se esperaba. Estas pruebas demostrarán la eficacia de su capacidad de recuperación general y proporcionarán un bucle de retroalimentación para sus procedimientos operativos antes de enfrentarse a escenarios de error reales.

## Principios de diseño

En la nube, hay algunos principios que pueden contribuir a aumentar la fiabilidad. Téngalos presentes cuando hablemos de las prácticas recomendadas:

- Recupérese de un error automáticamente: al supervisar los indicadores clave de rendimiento (KPI) de una carga de trabajo, se puede desencadenar la automatización cuando se supera un umbral. Estos KPI deben ser una medida del valor de negocio, no de los aspectos técnicos del funcionamiento del servicio. De este modo, se permite la notificación y el seguimiento automático de los errores, así como los procesos de recuperación automatizada que pueden solucionar o corregir el error. Con una automatización más sofisticada, es posible anticipar y solucionar errores antes de que sucedan.
- Pruebe los procedimientos de recuperación: en un entorno local, a menudo se realizan pruebas para ver si una carga de trabajo funciona en una situación concreta. Normalmente, las pruebas no se usan para comprobar estrategias de recuperación. En la nube, puede probar los errores de la carga de trabajo y validar los procedimientos de recuperación. Puede usar la automatización para simular diferentes errores o recrear escenarios que anteriormente han producido algún error. Esto expone vías de error que puede probar y arreglar antes de que se produzca un escenario de error real, lo que permite reducir el riesgo.
- Escale de manera horizontal para aumentar la disponibilidad agregada de la carga de trabajo: reemplace un gran recurso por varios recursos pequeños para reducir el efecto de un solo error en toda la carga de trabajo. Distribuya las solicitudes a través de varios recursos más pequeños para garantizar que no compartan el mismo error.
- Deje de adivinar la capacidad: un factor habitual de errores de los sistemas locales es la saturación de recursos, que se produce cuando las demandas que se hacen a una carga de trabajo superan su capacidad (este es a menudo el objetivo de los ataques de denegación de servicio). En la nube, se puede supervisar la demanda y el uso de la carga de trabajo, además de automatizar la incorporación o eliminación de recursos de forma automatizada para mantener un nivel óptimo y satisfacer la demanda sin tener un aprovisionamiento excesivo o insuficiente. Aún hay límites, pero algunas cuotas se pueden controlar, mientras que otras se pueden administrar (consulte Administración de cuotas de servicio y restricciones).

- Administre cambios a través de la automatización: los cambios en su infraestructura deben hacerse mediante la automatización. Entre los cambios que se deben administrar se encuentran los de la automatización, de los que, posteriormente, se puede hacer un seguimiento y una revisión.

# Definiciones

En este documento técnico se trata la fiabilidad en la nube y se describen las prácticas recomendadas para estas cuatro áreas:

- Fundamentos
- Arquitectura de la carga de trabajo
- Administración de cambios
- Administración de errores

Para lograr la fiabilidad hay que empezar por los cimientos: un entorno en el que las cuotas de servicio y la topología de la red se adapten a la carga de trabajo. La arquitectura de la carga de trabajo del sistema distribuido debe estar diseñada para prevenir y mitigar los errores. La carga de trabajo debe gestionar los cambios en la demanda o los requisitos, y debe estar diseñada para detectar errores y repararse automáticamente.

Temas



## La resiliencia y los componentes de la fiabilidad

La fiabilidad de una carga de trabajo en la nube depende de varios factores, el principal de los cuales es la resiliencia:

- La resiliencia es la habilidad de una carga de trabajo de recuperarse de interrupciones de infraestructura o servicio, adquirir dinámicamente recursos de computación para satisfacer la demanda y mitigar interrupciones tales como configuraciones incorrectas o problemas de red transitorios.

Los otros factores que influyen en la fiabilidad de la carga de trabajo son:

- Excelencia operativa, que incluye la automatización de los cambios, el uso de manuales para responder a los errores y las revisiones de disponibilidad operativa (ORR) para confirmar que las aplicaciones están listas para las operaciones de producción.
- La seguridad, que incluye la prevención de daños a los datos o a la infraestructura por parte de infractores, lo que afectaría a la disponibilidad. Por ejemplo, cifrar las copias de seguridad para garantizar la seguridad de los datos.
- Eficiencia del rendimiento, que incluye el diseño para obtener las máximas tasas de solicitudes y minimizar las latencias para su carga de trabajo.
- Optimización de costes, que incluye compensaciones tales como si se debe gastar más en instancias EC2 para lograr la estabilidad estática o confiar en el escalado automático cuando se necesita más capacidad.

La resiliencia es el objetivo principal de este documento técnico.

Los otros cuatro aspectos también son importantes y se tratan en sus respectivos pilares de AWS Well-Architected Framework. En muchas de estas prácticas recomendadas también se tratan esos aspectos de la fiabilidad, pero el enfoque se centra en la resiliencia.

## Zona de

La disponibilidad (también conocida como disponibilidad del servicio) es tanto una métrica que se utiliza habitualmente para medir de forma cuantitativa la resiliencia, como un objetivo de resiliencia.

- La disponibilidad es el porcentaje de tiempo que una carga de trabajo está disponible para su uso.

La disponibilidad de uso significa que realiza correctamente su función convenida cuando se requiere.

Este porcentaje se calcula en periodos de tiempo, por ejemplo, mensual, anual o trienal. Al aplicar la interpretación más estricta posible, la disponibilidad se reduce siempre que la aplicación no funciona con normalidad, ya sea con interrupciones programadas o no programadas. Definimos la disponibilidad como sigue:

$$Disponibilidad = \frac{Tiempo\ de\ disponibilidad\ de\ uso}{Tiempo\ total}$$

- La disponibilidad es un porcentaje de tiempo de actividad (como el 99,9 %) durante un periodo de actividad (normalmente un mes o un año)
- La denominación común se refiere únicamente al «número de nueves» (por ejemplo, «cinco nueves» se traduce en una disponibilidad del 99,999 %).
- Algunos clientes optan por excluir el tiempo de inactividad del servicio programado (por ejemplo, el mantenimiento previsto) del tiempo total en la formula. Sin embargo, esto no es aconsejable, ya que es probable que sus usuarios quieran utilizar su servicio durante esas horas.

A continuación, se presenta una tabla con los objetivos comunes de diseño de disponibilidad de aplicaciones y la duración máxima en el que de las interrupciones se pueden producir en un año sin dejar de cumplir el objetivo. En la tabla figuran ejemplos de los tipos de aplicaciones que se suelen ver en cada nivel de disponibilidad. A lo largo del documento, trataremos estos valores.

| Disponibilidad | Falta de disponibilidad máxima (por año) | Categorías de aplicación |
|---|---|---|
| 99 % | 3 días 15 horas | Procesamiento en lotes, extracción de datos, transferencia y trabajos de carga |
| 99,9 % | 8 horas 45 minutos | Herramientas internas como la administración del conocimiento o el seguimiento de proyectos |
| 99,95 % | 4 horas 22 minutos | Comercio en línea, punto de venta |
| 99,99 % | 52 minutos | Distribución de videos, cargas de trabajo de transmisión |

| Disponibilidad | Falta de disponibilidad máxima (por año) | Categorías de aplicación |
| --- | --- | --- |
| 99,999 % | 5 minutos | Transacciones en cajeros automáticos, cargas de trabajo de telecomunicaciones |

Medir la disponibilidad en función de las solicitudes. Para su servicio, puede ser más fácil contar las solicitudes correctas y erróneas en lugar del «tiempo disponible para su uso». En este caso, se puede utilizar el siguiente cálculo:

$$Disponibilidad = \frac{Respuestas\ correctas}{Solicitudes\ válidas}$$

Suele medirse por periodos de uno o cinco minutos. Se puede calcular un porcentaje de tiempo de actividad mensual (medición de la disponibilidad en función del tiempo) a partir de la media de estos periodos. Si no se recibe ninguna solicitud en un periodo determinado, se cuenta con el 100 % disponible para ese tiempo.

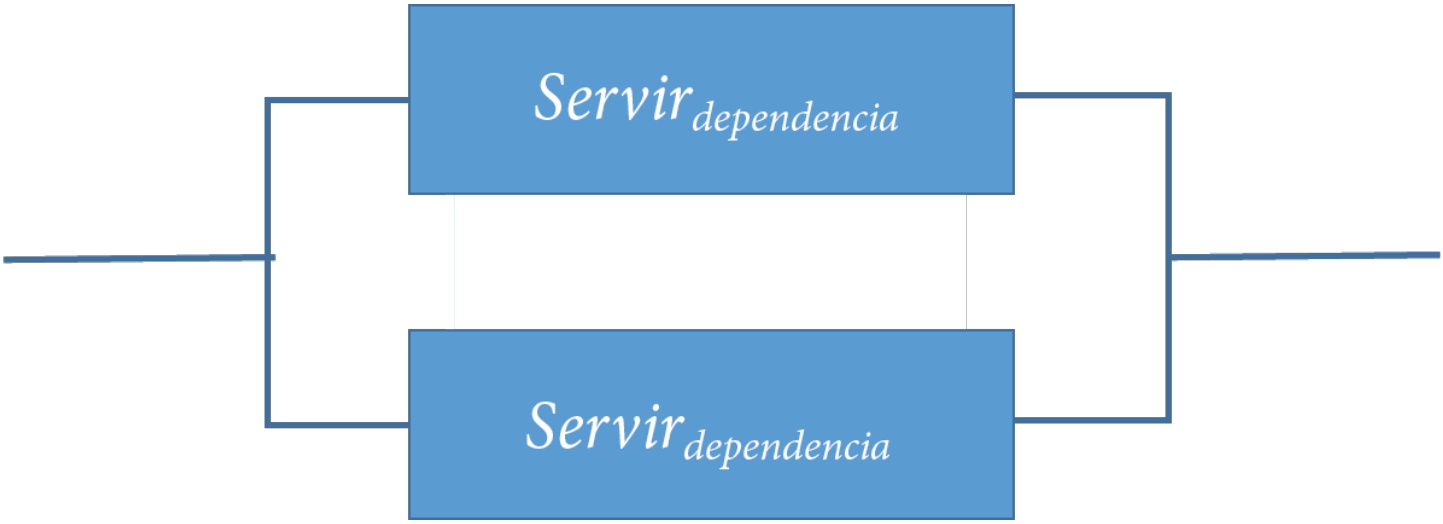
Cálculo de la disponibilidad con gran dependencia. Muchos sistemas dependen de otros, por lo que la interrupción de un sistema dependiente se interpreta directamente como una interrupción en el sistema de invocación. Esto se contrapone a una dependencia flexible, en la que un error del sistema dependiente se compensa en la aplicación. Cuando se producen las dependencias estrictas, la invocación de la disponibilidad del sistema es el producto de las disponibilidades de los sistemas dependientes. Por ejemplo, si se tiene un sistema diseñado para una disponibilidad del 99,99 % que tenga una dependencia estricta de otros dos sistemas independientes que están diseñados para una disponibilidad del 99,99 % cada uno, la carga de trabajo puede teóricamente lograr una disponibilidad del 99,97 %:

$Disponibilidad_{invocación} \times Disponibilidad_{dep1} \times Disponibilidad_{dep2} = Disponibilidad_{carga\ de\ trabajo}$

99,99 % × 99,99 % × 99,99 % = 99,97 %

Por lo tanto, es importante entender sus dependencias y sus objetivos de diseño de disponibilidad al realizar los cálculos.

Cálculo de la disponibilidad con componentes redundantes. Cuando un sistema implica el uso de componentes independientes y redundantes (por ejemplo, recursos redundantes en diferentes zonas de disponibilidad), la disponibilidad teórica se calcula como el 100 % menos el producto de las tasas de error de los componentes. Por ejemplo, si un sistema utiliza dos componentes independientes, cada uno con una disponibilidad del 99,9 %, la disponibilidad efectiva de esta dependencia es 99,9999 %:




$Disponibilidad_{rentabilidad} = Disponibilidad_{MÁX.} - ((100\ \% - Disponibilidad_{dependencia}) \times (100\ \% - Disponibilidad_{dependencia}))$

99,9999 % = 100 % − (0,1 %×0,1 %)

Cálculo abreviado: si las disponibilidades de todos los componentes de su cálculo consisten únicamente en el dígito nueve, puede sumar el recuento del número de dígitos de nueve para obtener la respuesta. En el ejemplo anterior, dos componentes redundantes e independientes con una disponibilidad de tres nueves da como resultado seis nueves.

Cálculo de la disponibilidad de las dependencias. Algunas dependencias proporcionan orientación sobre su disponibilidad, por ejemplo, los objetivos de diseño de disponibilidad de muchos servicios de AWS. Pero en los casos en que no se dispone de ella (por ejemplo, un componente en el que el fabricante no publica la información sobre la disponibilidad), una forma de estimarla es determinar el tiempo medio entre errores (MTBF) y el tiempo medio de recuperación (MTTR). Se puede estimar la disponibilidad:

$$Avail_{EST} = \frac{MTBF}{MTBF + MTTR}$$

Por ejemplo, si el MTBF es de 150 días y el MTTR es de 1 hora, la disponibilidad estimada será del 99,97 %.

Para obtener detalles adicionales, consulte Availability and Beyond: Understanding and improving the resilience of distributed systems on AWS (Disponibilidad y más allá: comprender y mejorar la resiliencia de los sistemas distribuidos en AWS), que puede ayudarle a calcular su disponibilidad.

Costos de disponibilidad. El diseño de aplicaciones para niveles más altos de disponibilidad suele aumentar los costes, por lo que es conveniente identificar las verdaderas necesidades de disponibilidad antes de iniciar el diseño de la aplicación. Los altos niveles de disponibilidad exigen requisitos más estrictos para el ensayo y la validación en situaciones de error exhaustivas. Requieren la automatización para la recuperación de todo tipo de errores y necesitan que todos los aspectos de las operaciones del sistema se desarrollen y prueben de forma similar con los mismos estándares. Por ejemplo, el aumento o la disminución de la capacidad, la implementación o restauración de software actualizado o cambios en la configuración, o la migración de los datos del sistema se deben realizar hasta alcanzar el objetivo de disponibilidad deseado. Al sumar los costos de desarrollo de los programas informáticos con niveles de disponibilidad muy elevados, la innovación se ve afectada por la necesidad de avanzar más lentamente en el despliegue de los sistemas. Por lo tanto, la orientación debe ser minuciosa al aplicar las normas y considerar el objetivo de disponibilidad adecuado para todo el ciclo de vida del sistema.

La selección de dependencias es otra forma con la que aumentan los costos en los sistemas que funcionan con objetivos de diseño de mayor disponibilidad. En estos objetivos superiores, el conjunto de programas o servicios que se pueden elegir como dependencias disminuye en función de cuáles de estos servicios han tenido las inversiones más importantes que hemos descrito anteriormente. A medida que aumenta el objetivo de diseño de la disponibilidad, es habitual encontrar menos servicios de uso múltiple (como una base de datos relacional) y más servicios de uso específico. Esto se debe a que los servicios de uso específico son más fáciles de evaluar, probar y automatizar, además de que muestran menos posibilidades de interacciones inesperadas con funcionalidades incluidas, pero sin utilizar.

## Objetivos de recuperación de desastres (DR)

Además de los objetivos de disponibilidad, su estrategia de resiliencia debe incluir también objetivos de recuperación de desastres (DR) basados en estrategias para recuperar la carga de trabajo en caso de que se produzca un desastre. La recuperación de desastres se centra en objetivos de recuperación puntuales en respuesta a catástrofes naturales, errores técnicos a gran escala o amenazas humanas como ataques o errores. Es distinto de la disponibilidad, que mide la resiliencia media durante un periodo de tiempo en respuesta a los errores de los componentes, los picos de carga o los errores de software.

Objetivo de tiempo de recuperación (RTO) definido por la organización. El RTO es el retraso máximo aceptable entre la interrupción del servicio y su restablecimiento. Esto determina lo que se considera un intervalo de tiempo aceptable cuando el servicio no está disponible.

Objetivo de punto de recuperación (RPO) definido por la organización. El RPO es el periodo de tiempo máximo aceptable desde el último punto de recuperación de datos. Determina lo que se considera una pérdida aceptable de datos entre el último punto de recuperación y la interrupción del servicio.


La relación entre el RPO (objetivo de punto de recuperación), el RTO (objetivo de tiempo de recuperación) y el evento de desastre.

El RTO es similar al MTTR (tiempo medio de recuperación) en el sentido de que ambos miden el tiempo transcurrido entre el inicio de una interrupción y la recuperación de la carga de trabajo.

Sin embargo, el MTTR es un valor medio tomado a lo largo de varios eventos que afectan a la disponibilidad durante un periodo de tiempo, mientras que el RTO es un objetivo, o valor máximo permitido, para un evento único que afecta a la disponibilidad.

## Comprensión de las necesidades de disponibilidad

Es común pensar inicialmente en la disponibilidad de una aplicación como un solo objetivo para la aplicación en su totalidad. Sin embargo, al analizarlo más detenidamente, con frecuencia encontramos que ciertos aspectos de una aplicación o servicio tienen requisitos de disponibilidad diferentes. Por ejemplo, algunos sistemas podrían dar prioridad a la capacidad de recibir y almacenar nuevos datos antes que a la de recuperar datos ya existentes. Otros sistemas dan prioridad a las operaciones en tiempo real sobre las operaciones que cambian la configuración o el entorno de un sistema. Los servicios pueden tener requisitos de disponibilidad muy elevados durante ciertas horas del día, pero pueden tolerar periodos mucho más largos de interrupción fuera de esas horas. Estas son algunas de las formas en que se puede separar una sola aplicación en las unidades funcionales y evaluar los requisitos de disponibilidad de cada una de ellas. El beneficio de hacerlo consiste en centrar los esfuerzos (y los gastos) en la disponibilidad de acuerdo con las necesidades específicas, en lugar de diseñar todo el sistema según los requisitos de mayor exigencia.

| Recomendación |
| --- |
| Evalúe de forma crítica los aspectos exclusivos de sus aplicaciones y, según corresponda, determine los objetivos de diseño de la disponibilidad y de la recuperación de desastres para reflejar las necesidades de su empresa. |

En AWS, normalmente dividimos los servicios en el «plano de datos» y el «plano de control». El plano de datos se encarga de entregar el servicio en tiempo real mientras que el plano de control se utiliza para configurar el entorno. Por ejemplo, las instancias Amazon EC2, las bases de datos de Amazon RDS y las operaciones de escritura de tablas de Amazon DynamoDB son operaciones de plano de datos. Por el contrario, el lanzamiento de nuevas instancias EC2 o bases de datos RDS o bien agregar o cambiar los metadatos de las tablas en DynamoDB se consideran operaciones de plano de control. Si bien los altos niveles de disponibilidad son importantes para todas estas capacidades, los planos de datos suelen tener objetivos de diseño de disponibilidad más altos que los planos de control. Por lo tanto, las cargas de trabajo con requisitos de alta disponibilidad deben evitar la dependencia en tiempo de ejecución de las operaciones de plano de control.

Muchos clientes de AWS adoptan un enfoque similar para evaluar de forma crítica sus aplicaciones e identificar los subcomponentes con diferentes necesidades de disponibilidad. Los objetivos de diseño de la disponibilidad se adaptan a los diferentes aspectos y se realizan los debidos estudios de ingeniería del sistema. AWS cuenta con gran experiencia en aplicaciones de ingeniería con un intervalo de objetivos de diseño de disponibilidad, incluidos servicios con una disponibilidad del 99,999 % o mayor. Los arquitectos de soluciones (SA) de AWS pueden ayudarle a crear el diseño adecuado para lograr sus objetivos de disponibilidad. Involucrar a AWS en la fase inicial de su proceso de diseño aumenta nuestra capacidad para ayudarle a cumplir sus objetivos de disponibilidad. La planificación de la disponibilidad no solo se hace antes de lanzar la carga de trabajo. También se hace continuamente para perfeccionar su diseño a medida que se adquiere experiencia operativa, se aprende de los eventos reales y se toleran errores de diferentes tipos. De esta forma, podrá aplicar el esfuerzo de trabajo adecuado para mejorar su aplicación.

Las necesidades de disponibilidad que se requieren para una carga de trabajo deben estar alineadas con la necesidad empresarial y la criticidad. Al definir primero el marco de criticidad empresarial con RTO, RPO y disponibilidad definidos, podrá evaluar cada carga de trabajo. Este enfoque requiere que los implicados en la implementación de la carga de trabajo conozcan el marco y el impacto que su carga de trabajo tiene en las necesidades empresariales.

# Fundamentos

Los requisitos fundamentales son aquellos cuyo ámbito va más allá de una única carga de trabajo o proyecto. Antes de diseñar la arquitectura de cualquier sistema, los requisitos básicos que afectan a la fiabilidad deberían estar aplicados. Por ejemplo, es preciso que haya suficiente ancho de banda de la red en el centro de datos.

En un entorno en las instalaciones, estos requisitos pueden alargar los plazos por las dependencias y, por lo tanto, se tienen que incorporar durante la planificación inicial. Sin embargo, AWS ya incorpora la mayor parte de estos requisitos básicos. Si no lo están, permite que estos se traten según corresponda. La nube está diseñada para que sea, en esencia, ilimitada, por lo que la responsabilidad de brindar suficiente capacidad de cómputo y de red recae en AWS. Al mismo tiempo, será libre de cambiar la asignación y el tamaño de los recursos según lo necesite.

En las siguientes secciones se explican las prácticas recomendadas que se centran en estas cuestiones de fiabilidad.

Temas



## Administración de cuotas de servicio y restricciones

Para las arquitecturas de carga de trabajo basadas en la nube, existen cuotas de servicio (que también se denominan límites de servicio). Estas cuotas existen para evitar aprovisionar por accidente más recursos de los necesarios y para limitar las tasas de solicitud en las operaciones de la API, de modo que los servicios queden protegidos ante posibles abusos. También existen restricciones de recursos, por ejemplo, la velocidad a la que se pueden introducir bits en un cable de fibra óptica o la cantidad de almacenamiento de un disco físico.

Si usa aplicaciones de AWS Marketplace, debe comprender las limitaciones de las aplicaciones. Si usa servicios web o software de terceros como un servicio, también debe ser consciente de esos límites.

Prácticas recomendadas

- REL01-BP02 Administrar cuotas de servicio en cuentas y regiones
- REL01-BP03 Adaptar las cuotas de servicio fijas y las restricciones a través de la arquitectura
- REL01-BP04 Supervisar y administrar cuotas
- REL01-BP05 Automatizar la administración de cuotas
- REL01-BP06 Garantizar que exista una diferencia suficiente entre las cuotas actuales y el uso máximo para permitir la conmutación por error

## REL01-BP01 Conocimiento de las cuotas y restricciones del servicio

Conozca las cuotas predeterminadas y administre las solicitudes de aumento de cuota para su arquitectura de carga de trabajo. Sepa qué restricciones de recursos en la nube, como el disco o la red, pueden causar impacto.

Resultado deseado: los clientes pueden evitar el deterioro o la interrupción del servicio en sus Cuentas de AWS mediante la implementación de directrices adecuadas para la supervisión de las métricas clave, las revisiones de la infraestructura y la automatización de los pasos de corrección, a fin de verificar que no se alcanzan las cuotas y restricciones de los servicios que podrían provocar el deterioro o la interrupción del servicio.

Antipatrones usuales:

- Desplegar una carga de trabajo sin conocer las cuotas estrictas o flexibles y sus límites para los servicios utilizados.
- Desplegar una carga de trabajo de reemplazo sin analizar ni volver a configurar las cuotas necesarias o sin contactar previamente con el servicio de asistencia.
- Suponer que los servicios en la nube no tienen límites y que los servicios pueden utilizarse sin tener en cuenta tarifas, límites, recuentos o cantidades.
- Suponer que las cuotas se incrementarán automáticamente.
- Desconocer el proceso y la cronología de las solicitudes de cuotas.
- Suponer que la cuota de servicio predeterminada en la nube es la misma para todos los servicios en diferentes regiones.
- Suponer que se pueden incumplir las restricciones del servicio y que los sistemas se escalarán automáticamente o añadirán un aumento del límite más allá de las restricciones del recurso.
- No probar la aplicación en picos de tráfico para estresar el uso de sus recursos.

- Aprovisionar el recurso sin analizar el tamaño de recurso requerido.
- Sobreaprovisionar la capacidad mediante la elección de tipos de recursos que van mucho más allá de las necesidades reales o de los picos previstos.
- No evaluar las necesidades de capacidad para nuevos niveles de tráfico antes de que se produzca un nuevo evento con un cliente o de desplegar una nueva tecnología.

Beneficios de establecer esta práctica recomendada: la supervisión y la administración automatizada de las cuotas de servicio y las restricciones de recursos pueden reducir los errores de forma proactiva. Los cambios en los patrones de tráfico para el servicio de un cliente pueden provocar una interrupción o un deterioro si no se siguen las prácticas recomendadas. Con la supervisión y la administración de estos valores en todas las regiones y en todas las cuentas, las aplicaciones pueden tener una mayor resiliencia ante acontecimientos adversos o imprevistos.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

Service Quotas es un servicio de AWS que le ayuda a administrar sus cuotas para más de 250 servicios de AWS desde una sola ubicación. Además de consultar los valores de las cuotas, también puede solicitar y realizar un seguimiento de los aumentos de las cuotas desde la consola de Service Quotas o mediante el SDK de AWS. AWS Trusted Advisor ofrece una comprobación de las cuotas de servicio que muestra su uso y las cuotas para ciertos aspectos de algunos servicios. Las cuotas de servicio predeterminadas por servicio también están en la documentación de AWS del servicio respectivo (por ejemplo, consulte Cuotas de Amazon VPC).

Algunos límites de servicio, como los límites de velocidad en las API limitadas se establecen en Amazon API Gateway mediante la configuración de un plan de uso. Algunos límites que se establecen como configuración en sus respectivos servicios son las IOPS aprovisionadas, el almacenamiento de Amazon RDS asignado y las asignaciones de volumen Amazon EBS. Amazon Elastic Compute Cloud tiene su propio panel de límites de servicio que puede ayudarle a administrar sus límites de instancia, Amazon Elastic Block Store y de dirección IP elástica. Si tiene un caso de uso en el que las cuotas de servicio repercuten en el rendimiento de su aplicación y no se ajustan a sus necesidades, contacte con AWS Support para ver si existen mitigaciones.

Las cuotas de servicio pueden ser específicas de una región o también pueden tener carácter global. El uso de un servicio de AWS que alcance su cuota no actuará del modo previsto en un uso normal y puede provocar la interrupción o el deterioro del servicio. Por ejemplo, una cuota de servicio limita

el número de DL Amazon EC2 que pueden utilizarse en una región y ese límite puede alcanzarse durante un evento de escalamiento de tráfico mediante grupos de Auto Scaling (ASG).

Las cuotas de servicio de cada cuenta deben evaluarse periódicamente para determinar cuáles podrían ser los límites de servicio adecuados para esa cuenta. Estas cuotas de servicio existen como barreras de protección operativas para evitar aprovisionar por accidente más recursos de los necesarios. También sirven para limitar las tasas de solicitudes en las operaciones de API para proteger los servicios del abuso.

Las restricciones de servicio son diferentes de las cuotas de servicio. Las restricciones de servicio representan los límites de un recurso concreto, tal y como los define ese tipo de recurso. Pueden ser la capacidad de almacenamiento (por ejemplo, gp2 tiene un límite de tamaño de 1 GB - 16 TB) o el rendimiento del disco (10 0000 iops). Es esencial que las restricciones de un tipo de recurso se diseñen y evalúen constantemente para detectar un uso que pueda alcanzar su límite. Si se alcanza una restricción de forma inesperada, las aplicaciones o servicios de la cuenta pueden deteriorarse o interrumpirse.

Si existe un caso de uso en el que las cuotas de servicio repercuten en el rendimiento de una aplicación y no pueden ajustarse a las necesidades requeridas, contacte con AWS Support para ver si existen mitigaciones. Para obtener más detalles sobre el ajuste de cuotas fijas, consulte REL01-BP03 Adaptar las cuotas de servicio fijas y las restricciones a través de la arquitectura.

Hay una serie de servicios y herramientas de AWS con las que se puede supervisar y administrar Service Quotas. El servicio y las herramientas se deben utilizar para proporcionar comprobaciones automatizadas o manuales de los niveles de cuota.

- AWS Trusted Advisor ofrece una comprobación de cuotas de servicio que muestra su uso y las cuotas para ciertos aspectos de algunos servicios. Puede ayudar a identificar los servicios que están cerca de la cuota.
- AWS Management Console proporciona métodos para mostrar los valores de las cuotas de los servicios, administrar, solicitar nuevas cuotas, supervisar el estado de las solicitudes de cuotas y mostrar el historial de cuotas.
- AWS CLI y los CDK ofrecen métodos programáticos para administrar y supervisar automáticamente los niveles de cuota de servicio y el uso.

Pasos para la implementación

Para Service Quotas:

- Revise AWS Service Quotas.
- Para conocer sus cuotas de servicio existentes, determine los servicios (como IAM Access Analyzer) que se utilizan. Hay aproximadamente 250 servicios de AWS controlados por cuotas de servicio. A continuación, determine el nombre específico de la cuota de servicio que podría utilizarse en cada cuenta y región. Hay aproximadamente 3000 nombres de cuotas de servicio por región.
- Aumente este análisis de cuotas con AWS Config para encontrar todos los recursos de AWS utilizados en sus Cuentas de AWS.
- Use los datos de AWS CloudFormation para determinar los recursos de AWS utilizados. Examine los recursos que se han creado en la AWS Management Console o con el comando `list-stack-resources` de la AWS CLI. También puede ver los recursos configurados para desplegarse en la propia plantilla.
- Consulte el código de despliegue para determinar todos los servicios que necesita su carga de trabajo.
- Determine las cuotas de servicio que se aplican. Use la información accesible mediante programación de Trusted Advisor y Service Quotas.
- Establezca un método de supervisión automatizado (consulte REL01-BP02 Administrar cuotas de servicio en cuentas y regiones y REL01-BP04 Supervisar y administrar cuotas) para alertar e informar si las cuotas de servicio están cerca de su límite o lo han alcanzado.
- Establezca un método automatizado y programático para comprobar si se ha modificado una cuota de servicio en una región pero no en otras regiones de la misma cuenta (consulte REL01-BP02 Administrar cuotas de servicio en cuentas y regiones y REL01-BP04 Supervisar y administrar cuotas).
- Automatice el análisis de los registros y las métricas de las aplicaciones para determinar si existen errores de cuotas o restricciones de servicio. Si se producen estos errores, envíe alertas al sistema de supervisión.
- Establezca procedimientos de ingeniería para calcular el cambio necesario en la cuota (consulte REL01-BP05 Automatizar la administración de cuotas) una vez que se haya identificado que se necesitan cuotas mayores para servicios específicos.
- Cree un flujo de trabajo de aprovisionamiento y aprobación para solicitar cambios en la cuota de servicio. Debería incluir un flujo de trabajo de excepciones en caso de denegación de la solicitud o de aprobación parcial.

- Cree un método de ingeniería para efectuar una revisión de las cuotas de servicio previa al aprovisionamiento y el uso de nuevos servicios de AWS antes de desplegarlos en entornos de producción o de carga (por ejemplo, una cuenta de pruebas de carga).

Para las restricciones de servicio:

- Establezca métodos de supervisión y medición para alertar de la lectura de los recursos próximos a sus restricciones. Use CloudWatch según proceda para las métricas o la supervisión de registros.
- Establezca umbrales de alerta para cada recurso con una restricción significativa para la aplicación o el sistema.
- Cree procedimientos de administración de flujos de trabajo e infraestructuras para cambiar el tipo de recurso si la restricción está próxima a su uso. Como práctica recomendada, este flujo de trabajo debe incluir pruebas de carga para verificar que el nuevo tipo sea el tipo de recurso correcto con las nuevas restricciones.
- Migre el recurso identificado al nuevo tipo de recurso recomendado, mediante los procedimientos y los procesos existentes.

## Recursos

Prácticas recomendadas relacionadas:

- REL01-BP02 Administrar cuotas de servicio en cuentas y regiones
- REL01-BP03 Adaptar las cuotas de servicio fijas y las restricciones a través de la arquitectura
- REL01-BP04 Supervisar y administrar cuotas
- REL01-BP05 Automatizar la administración de cuotas
- REL01-BP06 Garantizar que exista una diferencia suficiente entre las cuotas actuales y el uso máximo para permitir la conmutación por error
- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores
- REL11-BP03 Automatizar la reparación en todas las capas
- REL12-BP05 Probar la resiliencia mediante la ingeniería del caos

Documentos relacionados:

- Pilar de fiabilidad de AWS Well-Architected Framework: disponibilidad
- AWS Service Quotas (denominados anteriormente límites de servicio)
- Comprobaciones de prácticas recomendadas de AWS Trusted Advisor (consulte la sección Límites de servicio)
- AWS Limit Monitor en AWS Answers
- Límites de servicio de Amazon EC2
- ¿Qué es Service Quotas?
- Cómo solicitar un aumento de cuota
- Puntos de conexión y cuotas de servicio
- Guía del usuario de Service Quotas
- Supervisor de cuotas para AWS
- Límites de aislamiento de errores de AWS
- Disponibilidad con redundancia
- AWS para datos
- ¿Qué es la integración continua?
- ¿Qué es la entrega continua?
- Socio de APN: socios que pueden ayudar con la administración de la configuración
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(Administración del ciclo de vida de las cuentas en entornos SaaS de cuenta por inquilino en AWS)
- Managing and monitoring API throttling in your workloads (Administrar y supervisar la limitación de las API en sus cargas de trabajo)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(Ver recomendaciones de AWS Trusted Advisor a escala con AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(Automatización de los aumentos del límite de servicio y asistencia a empresas con AWS Control Tower)

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas

- View and Manage Quotas for AWS Services Using Service Quotas (Ver y administrar cuotas para AWS Services con Service Quotas)
- AWS IAM Quotas Demo (Demostración de las cuotas de AWS IAM)

Herramientas relacionadas:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

## REL01-BP02 Administrar cuotas de servicio en cuentas y regiones

Si utiliza múltiples cuentas o regiones, solicite las cuotas pertinentes en todos los entornos en los que se ejecutan sus cargas de trabajo de producción.

Resultado deseado: los servicios y las aplicaciones no deberían verse afectados si se agota la cuota de servicio en las configuraciones que abarcan cuentas o regiones, o que tienen diseños de resiliencia mediante la conmutación por error de zonas, regiones o cuentas.

Antipatrones usuales:

- Permitir que aumente el uso de recursos en una región aislada sin ningún mecanismo para mantener la capacidad en las demás.
- Configurar manualmente todas las cuotas de forma independiente en regiones aisladas.
- No considerar el efecto de las arquitecturas de resiliencia (activa o pasiva) en las futuras necesidades de cuota durante un deterioro de la región no principal.

- No evaluar las cuotas periódicamente ni realizar los cambios necesarios en cada región y cuenta donde se ejecuta la carga de trabajo.
- No utilizar las plantillas de solicitud de cuota para solicitar incrementos en varias regiones y cuentas.
- No actualizar las cuotas de servicio por pensar erróneamente que el aumento de las cuotas tiene implicaciones de coste, como las solicitudes de reserva de computación.

Beneficios de establecer esta práctica recomendada: verificar que puede gestionar su carga actual en regiones o cuentas secundarias si los servicios regionales dejan de estar disponibles. Esto puede reducir el número de errores o los niveles de deterioro que se producen durante la pérdida de una región.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

El seguimiento de las cuotas de servicio se realiza por cuenta. A no ser que se especifique lo contrario, cada cuota es específica de una Región de AWS. Además de los entornos de producción, administre también las cuotas en todos los entornos que no sean de producción aplicables, de modo que las pruebas y el desarrollo no se vean limitados. El mantenimiento de un elevado nivel de resiliencia requiere que las cuotas de servicio se evalúen continuamente (ya sea de forma automatizada o manual).

Al haber más cargas de trabajo que abarcan regiones debido a la implementación de diseños que utilizan los enfoques Activo/Activo, Activo/Pasivo - En caliente, Activo/Pasivo - En frío y Activo/Pasivo - Luz piloto, es esencial comprender todos los niveles de cuotas de regiones y cuentas. Los patrones de tráfico anteriores no siempre son un buen indicador de si la cuota de servicio está configurada correctamente.

Igualmente importante es el hecho de que el límite de nombres de cuota de servicio no es siempre el mismo para todas las regiones. En una región, el valor podría ser cinco, y en otra, diez. La administración de estas cuotas debe abarcar los mismos servicios, cuentas y regiones para proporcionar una resiliencia coherente bajo carga.

Concilie todas las diferencias de cuota de servicio entre las distintas regiones (región activa o región pasiva) y cree procesos para conciliar continuamente estas diferencias. Los planes de prueba de las conmutaciones por error pasivas de las regiones en muy pocas ocasiones se escalan a la capacidad

activa máxima, lo que significa que los ejercicios del día de juego o de mesa pueden no encontrar diferencias en las cuotas de servicio entre las regiones y tampoco mantener los límites correctos.

Es muy importante controlar y evaluar la desviación de cuota de servicio, la situación en la que los límites de la cuota de servicio para una determinada cuota con nombre se modifican en una región y no en todas las regiones. Debe considerarse la posibilidad de cambiar la cuota en las regiones con tráfico o con posibilidad de tener tráfico.

- Seleccione las cuentas y regiones que correspondan según sus requisitos de servicio, latencia, normativos y de recuperación de desastres (DR).
- Identifique las cuotas de servicio en todas las cuentas, regiones y zonas de disponibilidad pertinentes. Los límites se determinan por cuenta y región. Estos valores deben compararse para detectar diferencias.

Pasos para la implementación

- Revise los valores de Service Quotas que podrían haber superado el nivel de riesgo de uso. AWS Trusted Advisor proporciona alertas si superan los umbrales del 80 % y el 90 %.
- Revise los valores de las cuotas de servicio en cualquier región pasiva (en un diseño activo/pasivo). Verifique que la carga se ejecutará correctamente en las regiones secundarias si se produce un error en la región principal.
- Automatice la evaluación de si se ha producido alguna desviación de la cuota de servicio entre regiones de la misma cuenta y actúe en consecuencia para modificar los límites.
- Si las unidades organizativas (UO) del cliente están estructuradas de la forma admitida, las plantillas de cuotas de servicio deberán actualizarse para reflejar los cambios en las cuotas que deban aplicarse a varias regiones y cuentas.
  - Cree una plantilla y asocie regiones al cambio de cuota.
  - Revise todas las plantillas de cuota de servicio existentes por si fuera necesario realizar algún cambio (región, límites y cuentas).

## Recursos

Prácticas recomendadas relacionadas:

- REL01-BP01 Conocimiento de las cuotas y restricciones del servicio
- REL01-BP03 Adaptar las cuotas de servicio fijas y las restricciones a través de la arquitectura

- REL01-BP04 Supervisar y administrar cuotas
- REL01-BP05 Automatizar la administración de cuotas
- REL01-BP06 Garantizar que exista una diferencia suficiente entre las cuotas actuales y el uso máximo para permitir la conmutación por error
- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores
- REL11-BP03 Automatizar la reparación en todas las capas
- REL12-BP05 Probar la resiliencia mediante la ingeniería del caos

Documentos relacionados:

- Pilar de fiabilidad de AWS Well-Architected Framework: disponibilidad
- AWS Service Quotas (denominados anteriormente límites de servicio)
- Comprobaciones de prácticas recomendadas de AWS Trusted Advisor (consulte la sección Límites de servicio)
- AWS Limit Monitor en AWS Answers
- Límites de servicio de Amazon EC2
- ¿Qué es Service Quotas?
- Cómo solicitar un aumento de cuota
- Puntos de conexión y cuotas de servicio
- Guía del usuario de Service Quotas
- Supervisor de cuotas para AWS
- Límites de aislamiento de errores de AWS
- Disponibilidad con redundancia
- AWS para datos
- ¿Qué es la integración continua?
- ¿Qué es la entrega continua?
- Socio de APN: socios que pueden ayudar con la administración de la configuración
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(Administración del ciclo de vida de las cuentas en entornos SaaS de cuenta por inquilino en AWS)

- Managing and monitoring API throttling in your workloads (Administrar y supervisar la limitación de las API en sus cargas de trabajo)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(Ver recomendaciones de AWS Trusted Advisor a escala con AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(Automatización de los aumentos del límite de servicio y asistencia a empresas con AWS Control Tower)

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Ver y administrar cuotas para AWS Services con Service Quotas)
- AWS IAM Quotas Demo (Demostración de las cuotas de AWS IAM)

Servicios relacionados:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

# REL01-BP03 Adaptar las cuotas de servicio fijas y las restricciones a través de la arquitectura

Sea consciente de las cuotas de servicio inalterables, las restricciones de servicio y los límites de recursos físicos. Diseñe arquitecturas para aplicaciones y servicios que eviten que estos límites afecten a la fiabilidad.

Algunos ejemplos son el ancho de banda de la red, el tamaño de la carga útil de la invocación de funciones sin servidor, la tasa de ráfagas de aceleración para una puerta de enlace de API y las conexiones simultáneas de usuarios a una base de datos.

Resultado deseado: la aplicación o servicio funciona como se espera en condiciones normales y de alto tráfico. Se han diseñado para funcionar dentro de las limitaciones fijadas para ese recurso o cuotas de servicio.

Antipatrones usuales:

- Elegir un diseño que utilice un recurso de un servicio, sin saber que existen restricciones de diseño que provocarán que este diseño producirá un error en el escalado.
- Realizar una evaluación comparativa que no es realista y que alcanzará las cuotas fijas del servicio durante la evaluación. Por ejemplo, ejecutar pruebas con un límite de ráfagas, pero durante un período prolongado.
- Elegir un diseño que no pueda escalarse ni modificarse si se van a superar las cuotas de servicio fijas. Por ejemplo, un tamaño de carga útil SQS de 256 KB.
- No diseña ni implementa la observabilidad para supervisar y avisar sobre umbrales de cuotas de servicio que podrían estar en riesgo durante eventos de alto tráfico.

Beneficios de establecer esta práctica recomendada: verificación de que la aplicación funcionará bajo todos los niveles de carga de servicios previstos sin interrupciones ni degradaciones.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

A diferencia de las cuotas de servicio blandas o los recursos que pueden sustituirse por unidades de mayor capacidad, las cuotas fijas de los servicios de AWS no pueden modificarse. Esto significa que todos estos tipos de servicios de AWS deben evaluarse para detectar posibles límites estrictos de capacidad cuando se utilizan en el diseño de una aplicación.

Los límites estrictos se muestran en la consola de Service Quotas. Si las columnas muestran `AJUSTABLE = No`, el servicio tiene un límite estricto. Los límites estrictos también se muestran en algunas páginas de configuración de recursos. Por ejemplo, Lambda tiene límites estrictos específicos que no se pueden ajustar.

Por ejemplo, cuando se diseña una aplicación python para que se ejecute en una función Lambda, es necesario evaluar la aplicación para determinar si existe alguna posibilidad de que Lambda se ejecute durante más de 15 minutos. Si el código puede ejecutarse durante más tiempo de este límite de cuota de servicio, deben considerarse tecnologías o diseños alternativos. Si se alcanza el límite después del despliegue en producción, la aplicación sufrirá degradación e interrupciones hasta que pueda remediarse. A diferencia de las cuotas flexibles, no existe ningún método para cambiar estos límites, ni siquiera en caso de eventos de emergencia de gravedad 1.

Una vez que la aplicación se ha desplegado en un entorno de pruebas, se deben utilizar estrategias para averiguar si se puede alcanzar algún límite estricto. Las pruebas de estrés, las pruebas de carga y las pruebas de caos deben formar parte del plan de pruebas de introducción.

Pasos para la implementación

- Revise la lista completa de servicios de AWS que podrían utilizarse en la fase de diseño de la aplicación.
- Revise los límites de cuotas flexibles y estrictos para todos estos servicios. En la consola de Service Quotas no se muestran todos los límites. Algunos servicios describen estos límites en ubicaciones alternativas.
- Al diseñar su aplicación, revise los impulsores empresariales y tecnológicos de la carga de trabajo, como los resultados empresariales, el caso de uso, los sistemas dependientes, los objetivos de disponibilidad y los objetos de recuperación de desastres. Permita que sus impulsores empresariales y tecnológicos guíen el proceso para identificar el sistema distribuido adecuado para su carga de trabajo.
- Analice la carga de servicio en todas las regiones y cuentas. Muchos límites estrictos se basan en la región para los servicios. Sin embargo, algunos límites se basan en las cuentas.
- Analice el uso de recursos de las arquitecturas de resistencia durante un error zonal y un error regional. En la progresión de los diseños multirregión que utilizan enfoques activo/activo, activo/pasivo: en caliente, activo/pasivo: en frío y activo/pasivo: luz piloto, estos casos de error provocarán un mayor uso. Esto crea un caso de uso potencial para alcanzar límites estrictos.

## Recursos

Prácticas recomendadas relacionadas:

- REL01-BP01 Conocimiento de las cuotas y restricciones del servicio
- REL01-BP02 Administrar cuotas de servicio en cuentas y regiones
- REL01-BP04 Supervisar y administrar cuotas
- REL01-BP05 Automatizar la administración de cuotas
- REL01-BP06 Garantizar que exista una diferencia suficiente entre las cuotas actuales y el uso máximo para permitir la conmutación por error
- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores
- REL11-BP03 Automatizar la reparación en todas las capas
- REL12-BP05 Probar la resiliencia mediante la ingeniería del caos

Documentos relacionados:

- Pilar de fiabilidad de AWS Well-Architected Framework: disponibilidad
- AWS Service Quotas (denominados anteriormente límites de servicio)
- Comprobaciones de prácticas recomendadas de AWS Trusted Advisor (consulte la sección Límites de servicio)
- AWS Limit Monitor en AWS Answers
- Límites de servicio de Amazon EC2
- ¿Qué es Service Quotas?
- Cómo solicitar un aumento de cuota
- Puntos de conexión y cuotas de servicio
- Guía del usuario de Service Quotas
- Supervisor de cuotas para AWS
- Límites de aislamiento de errores de AWS
- Disponibilidad con redundancia

- AWS para datos
- ¿Qué es la integración continua?
- ¿Qué es la entrega continua?
- Socio de APN: socios que pueden ayudar con la administración de la configuración
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(Administración del ciclo de vida de las cuentas en entornos SaaS de cuenta por inquilino en AWS)
- Managing and monitoring API throttling in your workloads (Administrar y supervisar la limitación de las API en sus cargas de trabajo)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(Ver recomendaciones de AWS Trusted Advisor a escala con AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(Automatización de los aumentos del límite de servicio y asistencia a empresas con AWS Control Tower)
- Acciones, recursos y claves de condición de los servicios de Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Ver y administrar cuotas para AWS Services con Service Quotas)
- AWS IAM Quotas Demo (Demostración de las cuotas de AWS IAM)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Cerrar los bucles y abrir las mentes: cómo asumir el control de los sistemas grandes y pequeños)

Herramientas relacionadas:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config

- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

## REL01-BP04 Supervisar y administrar cuotas

Evalúe el uso potencial y aumente las cuotas pertinentemente, lo que permitirá un crecimiento planificado del uso.

Resultado deseado: se despliegan sistemas activos y automatizados que administran y supervisan. Estas soluciones operativas garantizan que los umbrales de uso de las cuotas están a punto de alcanzarse. Esto se solucionaría de forma proactiva solicitando cambios en las cuotas.

Antipatrones usuales:

- No se configura la supervisión para comprobar el umbral de cuota de servicio.
- No se configura la supervisión de los límites estrictos, aunque esos valores no puedan modificarse.
- Se supone que el tiempo necesario para solicitar y asegurar un cambio de cuota flexible es inmediato o un de corta duración.
- Se configuran alarmas de aproximación para cuotas de servicio sin contar con ningún proceso para responder a una alerta.
- Solo se configuran alarmas para servicios compatibles con AWS Service Quotas y no se supervisan otros servicios de AWS.
- No se considera la administración de cuotas para diseños de resiliencia multirregional, como los enfoques activo/activo, activo/pasivo: en caliente, activo/pasivo: en frío y activo/pasivo: luz piloto.
- No se evalúan las diferencias de cuota entre regiones.
- No se evalúan las necesidades de cada región para una solicitud específica de aumento de cuota.
- No se aprovechan las plantillas de administración de cuotas multirregión.

Beneficios de establecer esta práctica recomendada: el seguimiento automático de las cuotas de servicio de AWSService Quotas y la supervisión del uso en comparación con dichas cuotas le permitirá comprobar cuándo se acerca al límite de una cuota. También puede utilizar estos datos de supervisión para ayudar a limitar cualquier degradación debida al agotamiento de cuotas.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Para los servicios admitidos, puede supervisar las cuotas por medio de la configuración de varios servicios diferentes que pueden evaluar y luego enviar alertas o alarmas. Esto puede ayudar a supervisar el uso y alertarle de que se aproxima a las cuotas. Estas alarmas se pueden desencadenar con AWS Config, las funciones de Lambda, Amazon CloudWatch o con AWS Trusted Advisor. También puede usar filtros de métricas en CloudWatch Logs para buscar y extraer patrones en registros para determinar si el uso se aproxima a los umbrales de las cuotas.

Pasos para la implementación

Para la supervisión:

- Capture el consumo de recursos actual (por ejemplo, buckets o instancias). Utilice operaciones de API de servicio, como la API `DescribeInstances` de Amazon EC2 para recopilar el consumo actual de recursos.
- Capture las cuotas actuales que son esenciales y aplicables a los servicios que utiliza:
  - AWS Service Quotas
  - AWS Trusted Advisor
  - Documentación de AWS
  - Páginas específicas de los servicios de AWS
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Utilice AWS Service Quotas, un servicio de AWS que le ayuda a administrar sus cuotas para más de 250 servicios de AWS desde una ubicación.
- Utilice los límites de servicio de Trusted Advisor para supervisar sus límites de servicio actuales en diversos umbrales.
- Utilice el historial de cuotas de servicio (consola o AWS CLI) para comprobar los aumentos regionales.
- Compare los cambios de cuota de servicio en cada región y cada cuenta para crear equivalencias, si es necesario.

Para la administración:

- Automatizada: configure una regla personalizada de AWS Config para analizar las cuotas de servicio en todas las regiones y comparar las diferencias.
- Automatizada: configure una función programada de Lambda para analizar las cuotas de servicio en todas las regiones y comparar las diferencias.
- Manual: analice la cuota de servicios a través de la AWS CLI, la API o la consola de AWS para analizar las cuotas de servicio en todas las regiones y comparar las diferencias. Informe de las diferencias.
- Si se identifican diferencias en las cuotas entre las regiones, solicite un cambio de cuota, si es necesario.
- Revise el resultado de todas las solicitudes.

## Recursos

Prácticas recomendadas relacionadas:

- REL01-BP01 Conocimiento de las cuotas y restricciones del servicio
- REL01-BP02 Administrar cuotas de servicio en cuentas y regiones
- REL01-BP03 Adaptar las cuotas de servicio fijas y las restricciones a través de la arquitectura
- REL01-BP05 Automatizar la administración de cuotas
- REL01-BP06 Garantizar que exista una diferencia suficiente entre las cuotas actuales y el uso máximo para permitir la conmutación por error
- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores
- REL11-BP03 Automatizar la reparación en todas las capas
- REL12-BP05 Probar la resiliencia mediante la ingeniería del caos

Documentos relacionados:

- Pilar de fiabilidad de AWS Well-Architected Framework: disponibilidad
- AWS Service Quotas (denominados anteriormente límites de servicio)
- Comprobaciones de prácticas recomendadas de AWS Trusted Advisor (consulte la sección Límites de servicio)
- AWS Limit Monitor en AWS Answers

- Límites de servicio de Amazon EC2
- ¿Qué es Service Quotas?
- Cómo solicitar un aumento de cuota
- Puntos de conexión y cuotas de servicio
- Guía del usuario de Service Quotas
- Supervisor de cuotas para AWS
- Límites de aislamiento de errores de AWS
- Disponibilidad con redundancia
- AWS para datos
- ¿Qué es la integración continua?
- ¿Qué es la entrega continua?
- Socio de APN: socios que pueden ayudar con la administración de la configuración
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(Administración del ciclo de vida de las cuentas en entornos SaaS de cuenta por inquilino en AWS)
- Managing and monitoring API throttling in your workloads (Administrar y supervisar la limitación de las API en sus cargas de trabajo)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(Ver recomendaciones de AWS Trusted Advisor a escala con AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(Automatización de los aumentos del límite de servicio y asistencia a empresas con AWS Control Tower)
- Acciones, recursos y claves de condición de los servicios de Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Ver y administrar cuotas para AWS Services con Service Quotas)
- AWS IAM Quotas Demo (Demostración de las cuotas de AWS IAM)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Cerrar los bucles y abrir las mentes: cómo asumir el control de los sistemas grandes y pequeños)

Herramientas relacionadas:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

## REL01-BP05 Automatizar la administración de cuotas

Implemente herramientas para alertarle cuando se acerque a los límites. Puede automatizar las solicitudes de incremento de cuotas utilizando las API de AWS Service Quotas y automatizar las solicitudes de incremento de cuotas.

Si integra su base de datos de administración de configuraciones (CMDB) o su sistema de emisión de tiques con Service Quotas, puede automatizar el seguimiento de las solicitudes de aumento de cuotas y las cuotas actuales. Además del SDK de AWS, Service Quotas ofrece automatización utilizando AWS Command Line Interface (AWS CLI).

Patrones de uso no recomendados comunes:

- Realizar el seguimiento de las cuotas y el uso en hojas de cálculo.
- Ejecutar informes de uso cada día, semana o mes y después comparar el uso con las cuotas.

Beneficios de establecer esta práctica recomendada: El control automático de las cuotas de servicio de AWS y la supervisión del uso en comparación con dichas cuotas le permite comprobar cuándo se acerca a una cuota. Puede configurar la automatización para que le ayude a solicitar un aumento de cuota cuando resulte necesario. Es posible que quiera plantearse la reducción de algunas cuotas cuando su uso adopte una tendencia opuesta para materializar los beneficios de un menor riesgo (en caso de que sus credenciales se hayan visto comprometidas) y el ahorro de costes.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

## Guía para la implementación

- Configure una supervisión automatizada. Implemente herramientas con SDK para alertarle cuando se acerque a los límites.
  - Utilice Service Quotas y potencie el servicio con una solución de supervisión de cuotas automatizada, como AWS Limit Monitor o una oferta de AWS Marketplace.
    - ¿Qué es Service Quotas?
    - Supervisor de cuotas en AWS: solución de AWS
  - Configure respuestas desencadenadas en función de umbrales de cuotas con las API de Amazon SNS y AWS Service Quotas.
  - Automatización de pruebas.
    - Configure umbrales de límites.
    - Integre con eventos de cambio de AWS Config, canalizaciones de despliegue, Amazon EventBridge o terceros.
    - Defina de forma artificial umbrales de cuota bajos para probar las respuestas.
    - Configure desencadenadores para realizar acciones pertinentes en las notificaciones y póngase en contacto con AWS Support cuando sea necesario.
    - Desencadene manualmente eventos de cambio.
    - Ejecute un día de juego para probar el proceso de cambio de aumento de cuotas.

## Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudar con la administración de la configuración
- AWS Marketplace: productos de CMDB que ayudan a hacer un seguimiento de los límites
- AWS Service Quotas (denominadas anteriormente límites de servicio)
- Comprobaciones de prácticas recomendadas de AWS Trusted Advisor (consulte la sección Límites de servicio)
- Supervisor de cuotas en AWS: solución de AWS
- Límites de servicio de Amazon EC2
- ¿Qué es Service Quotas?

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas

## REL01-BP06 Garantizar que exista una diferencia suficiente entre las cuotas actuales y el uso máximo para permitir la conmutación por error

Cuando un recurso falla o es inaccesible, ese recurso puede seguir computando para una cuota dada hasta que se finalice correctamente. Compruebe que sus cuotas cubran el solapamiento de los recursos averiados o inaccesibles y sus sustitutos. A la hora de calcular esta brecha debe tener en cuenta casos de uso como errores de red, errores de la zona de disponibilidad o errores regionales.

Resultado deseado: los errores pequeños o grandes en los recursos o en la accesibilidad de los recursos pueden cubrirse dentro de los umbrales de servicio actuales. En la planificación de recursos se tienen en cuenta los errores de zona, de red o, incluso, regionales.

Antipatrones usuales:

- Se establecen cuotas de servicio sobre la base de las necesidades actuales sin tener en cuenta los casos de conmutación por error.
- No se tienen en cuenta los principios de estabilidad estática al calcular la cuota máxima de un servicio.
- No se tiene en cuenta el potencial de recursos inaccesibles al calcular la cuota total necesaria para cada región.
- No se tienen en cuenta los límites de aislamiento de errores del servicio de AWS para algunos servicios y sus posibles patrones de uso anómalos.

Beneficios de establecer esta práctica recomendada: cuando un evento de interrupción del servicio afecta a la disponibilidad de la aplicación, la nube le permite implementar estrategias para mitigar o recuperarse de estos eventos. Estas estrategias suelen incluir la creación de recursos adicionales para sustituir aquellos que han experimentado algún error o a los que no se puede acceder. La estrategia de cuotas se adaptaría a estas condiciones de conmutación por error y no introduciría degradaciones adicionales debidas al agotamiento de los límites de servicio.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Al evaluar los límites de cuota, considere los casos de conmutación por error que podrían producirse debido a alguna degradación. Deben tenerse en cuenta los siguientes tipos de casos de conmutación por error:

- Una VPC interrumpida o inaccesible.
- Una subred inaccesible.
- Una zona de disponibilidad que se ha degradado lo suficiente como para afectar a la accesibilidad de muchos recursos.
- Varias rutas de red o puntos de entrada y salida bloqueados o modificados.
- Una región que se ha degradado lo suficiente como para afectar a la accesibilidad de muchos recursos.
- Hay numerosos recursos, pero no todos se ven afectados por un error en una región o zona de disponibilidad.

Los errores como los de la lista anterior podrían ser el detonante del inicio de un evento de conmutación por error. La decisión de conmutar por error es única para cada situación y cliente, ya que el efecto empresarial puede variar drásticamente. Sin embargo, cuando operacionalmente se decide conmutar por error aplicaciones o servicios, la planificación de la capacidad de los recursos en la ubicación de la conmutación por error y sus cuotas correspondientes deben abordarse antes del evento.

Revise las cuotas de servicio para cada servicio teniendo en cuenta los picos más altos de lo normal que puedan producirse. Estos picos pueden estar relacionados con recursos a los que no se puede acceder debido a la red o a los permisos, pero que siguen activos. Los recursos activos no finalizados seguirán contando para el límite de cuota de servicio.

Pasos para la implementación

- Asegúrese de que haya una diferencia suficiente entre la cuota de servicio y el uso máximo para permitir la conmutación por error o una pérdida de accesibilidad.
- Determine sus cuotas de servicio, teniendo en cuenta sus patrones de despliegue, los requisitos de disponibilidad y el crecimiento del consumo.
- Solicite aumentos de la cuota si fuera necesario. Planifique el tiempo necesario para que se cumplan las solicitudes de aumentos de cuotas.

- Determine sus requisitos de fiabilidad (también conocidos como «número de nueves»).
- Establezca sus escenarios de error (por ejemplo, la pérdida de componentes, una zona de disponibilidad o una región).
- Establezca su metodología de despliegue (por ejemplo, valor controlado, azul-verde, rojo-negro o continua).
- Incluya un búfer adecuado (por ejemplo, del 15 %) en el límite actual.
- Incluya cálculos de estabilidad estática (zonal y regional) cuando proceda.
- Planifique el crecimiento de consumo (por ejemplo, supervise sus tendencias de consumo).
- Considere la repercusión de la estabilidad estática para las cargas de trabajo más fundamentales. Evalúe los recursos conforme a un sistema estáticamente estable en todas las regiones y zonas de disponibilidad.
- Considere el uso de reservas de capacidad bajo demanda para programar la capacidad antes de que se produzca una conmutación por error. Puede ser una estrategia útil durante las programaciones comerciales más cruciales para reducir los riesgos potenciales de obtener la cantidad y el tipo correctos de recursos durante la conmutación por error.

## Recursos

Prácticas recomendadas relacionadas:

- REL01-BP01 Conocimiento de las cuotas y restricciones del servicio
- REL01-BP02 Administrar cuotas de servicio en cuentas y regiones
- REL01-BP03 Adaptar las cuotas de servicio fijas y las restricciones a través de la arquitectura
- REL01-BP04 Supervisar y administrar cuotas
- REL01-BP05 Automatizar la administración de cuotas
- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores
- REL11-BP03 Automatizar la reparación en todas las capas
- REL12-BP05 Probar la resiliencia mediante la ingeniería del caos

Documentos relacionados:

- Pilar de fiabilidad de AWS Well-Architected Framework: disponibilidad

- AWS Service Quotas (denominados anteriormente límites de servicio)
- Comprobaciones de prácticas recomendadas de AWS Trusted Advisor (consulte la sección Límites de servicio)
- AWS Limit Monitor en AWS Answers
- Límites de servicio de Amazon EC2
- ¿Qué es Service Quotas?
- Cómo solicitar un aumento de cuota
- Puntos de conexión y cuotas de servicio
- Guía del usuario de Service Quotas
- Supervisor de cuotas para AWS
- Límites de aislamiento de errores de AWS
- Disponibilidad con redundancia
- AWS para datos
- ¿Qué es la integración continua?
- ¿Qué es la entrega continua?
- Socio de APN: socios que pueden ayudar con la administración de la configuración
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(Administración del ciclo de vida de las cuentas en entornos SaaS de cuenta por inquilino en AWS)
- Managing and monitoring API throttling in your workloads (Administrar y supervisar la limitación de las API en sus cargas de trabajo)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(Ver recomendaciones de AWS Trusted Advisor a escala con AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(Automatización de los aumentos del límite de servicio y asistencia a empresas con AWS Control Tower)
- Acciones, recursos y claves de condición de los servicios de Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Ver y administrar cuotas para AWS Services con Service Quotas)

- AWS IAM Quotas Demo (Demostración de las cuotas de AWS IAM)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Cerrar los bucles y abrir las mentes: cómo asumir el control de los sistemas grandes y pequeños)

Herramientas relacionadas:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

## Planificar su topología de red

Suele haber cargas de trabajo en distintos entornos. Entre estos se incluyen los entornos de la nube (tanto públicamente accesibles como privados), y posiblemente la infraestructura del centro de datos existente. Los planes deben incluir consideraciones, como la conectividad dentro de los sistemas y entre ellos, la administración de las direcciones IP públicas, la administración de las direcciones IP privadas y la resolución de nombres de dominio.

Si diseña sistemas que utilizan redes basadas en direcciones IP, tendrá que planificar la topología y el direccionamiento de la red anticipándose a posibles fallos, y permitiendo el crecimiento futuro y la integración con otros sistemas y sus redes.

Amazon Virtual Private Cloud (Amazon VPC) le permite aprovisionar una sección privada y aislada de forma lógica de la nube de AWS donde puede lanzar recursos de AWS en una red virtual.

Prácticas recomendadas

- REL02-BP01 Usar conectividad de red de alta disponibilidad para los puntos de conexión públicos de la carga de trabajo
- REL02-BP02 Aprovisionar conectividad redundante entre las redes privadas en la nube y los entornos locales
- REL02-BP03 Garantizar que la asignación de subredes IP tenga en cuenta la expansión y la disponibilidad
- REL02-BP04 Preferir topologías radiales (hub-and-spoke) a una conexión en malla de varios a varios
- REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

## REL02-BP01 Usar conectividad de red de alta disponibilidad para los puntos de conexión públicos de la carga de trabajo

La creación de una conectividad de red de alta disponibilidad para los puntos de conexión públicos de las cargas de trabajo puede ayudarle a reducir el tiempo de inactividad debido a la pérdida de conectividad y mejorar la disponibilidad y el SLA de su carga de trabajo. Para conseguirlo, use DNS, redes de entrega de contenido (CDN), puertas de enlace de API, un equilibrador de carga o proxies inversos altamente disponibles.

Resultado deseado: es fundamental planificar, construir y poner en funcionamiento una conectividad de red de alta disponibilidad para sus puntos de conexión públicos. Si la carga de trabajo resulta inaccesible debido a una pérdida de conectividad, incluso si la carga de trabajo está en funcionamiento y disponible, los clientes verán su sistema como caído. Al combinar una conectividad de red de alta disponibilidad y resistente para los puntos de conexión públicos de la carga de trabajo, junto con una arquitectura resistente para la propia carga de trabajo, puede proporcionar la mejor disponibilidad y nivel de servicio posibles a sus clientes.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, las URL de función de AWS Lambda, las API de AWS AppSync y Elastic Load Balancing (ELB) ofrecen puntos de conexión públicos de alta disponibilidad. Amazon Route 53 proporciona un servicio DNS de alta disponibilidad para la resolución de nombres de dominio con el fin de verificar que las direcciones de los puntos de conexión públicos se puedan resolver.

También puede evaluar las aplicaciones de software de AWS Marketplace que proporcionen equilibrio de carga o uso de proxies.

Antipatrones usuales:

- Diseñar una carga de trabajo de alta disponibilidad sin planificar el DNS y la conectividad de red para alta disponibilidad.
- Usar direcciones de internet públicas en instancias o contenedores individuales y administrar la conectividad a ellas a con DNS.
- Usar direcciones IP en lugar de nombres de dominio para localizar los servicios.
- No hacer pruebas de escenarios en que se pierda la conectividad con sus puntos de conexión públicos.
- No analizar las necesidades de rendimiento de la red y los patrones de distribución.
- No hacer pruebas ni planificar escenarios en los que la conectividad de la red de internet a sus puntos de conexión públicos de la carga de trabajo pueda verse interrumpida.
- Proporcionar contenido (como páginas web, activos estáticos o archivos multimedia) a una gran área geográfica y no usar una red de entrega de contenido.
- No planificar en caso de que se produzcan ataques de denegación de servicio distribuido (DDoS). Los ataques DDoS corren el riesgo de cerrar el tráfico legítimo y reducir la disponibilidad para sus usuarios.

Beneficios de establecer esta práctica recomendada: se diseña una conectividad de red resistente y de alta disponibilidad que garantiza que la carga de trabajo esté accesible y disponible para los usuarios.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

El enrutamiento del tráfico es el núcleo de la creación de una conectividad de red de alta disponibilidad para sus puntos de conexión públicos. Para verificar que el tráfico puede llegar a los puntos de conexión, el DNS debe ser capaz de resolver los nombres de dominio en sus direcciones IP correspondientes. Utilice un sistema de nombres de dominio (DNS) escalable y de alta disponibilidad como Amazon Route 53 para administrar los registros DNS de su dominio. También puede utilizar las comprobaciones de estado proporcionadas por Amazon Route 53. Las comprobaciones de estado verifican que la aplicación sea accesible, esté disponible y funcione; se pueden configurar de manera que imiten el comportamiento de su usuario, como la solicitud de una página web o una URL concreta. En caso de error, Amazon Route 53 responde a las solicitudes de resolución de DNS y dirige el tráfico únicamente a los puntos de conexión de estado. También puede

plantearse el uso de las capacidades de DNS geográfico y enrutamiento basado en la latencia que ofrece Amazon Route 53.

Para comprobar que la carga de trabajo en sí sea de alta disponibilidad, utilice Elastic Load Balancing (ELB). Amazon Route 53 se puede utilizar para dirigir el tráfico a ELB, que distribuye el tráfico a las instancias de computación de destino. También puede utilizar Amazon API Gateway junto con AWS Lambda para una solución sin servidor. Los clientes también pueden ejecutar cargas de trabajo en varias Regiones de AWS. Con el patrón activo/activo multisitio, la carga de trabajo puede atender tráfico de varias regiones. Con un patrón activo/pasivo multisitio, la carga de trabajo atiende tráfico desde la región activa, mientras que los datos se replican en la región secundaria y se activan en caso de error en la región principal. Las comprobaciones de estado de Route 53 se pueden utilizar para controlar la conmutación por error de DNS desde cualquier punto de conexión en una región principal y a un punto de conexión en una región secundaria, lo que verifica que la carga de trabajo esté accesible y disponible para los usuarios.

Amazon CloudFront proporciona una API sencilla para distribuir contenido con baja latencia y altas velocidades de transferencia de datos atendiendo las solicitudes mediante una red de ubicaciones periféricas en todo el mundo. Las redes de entrega de contenido (CDN) atienden a los clientes proporcionándoles contenido ubicado o almacenado en caché en una ubicación cercana al usuario. Esto también mejora la disponibilidad de su aplicación, ya que la carga de contenido se desplaza de sus servidores a las ubicaciones periféricas de CloudFront. Las ubicaciones periféricas y las cachés periféricas regionales mantienen copias en caché de su contenido cerca de sus usuarios, lo que permite una recuperación rápida y aumenta la accesibilidad y la disponibilidad de su carga de trabajo.

Para cargas de trabajo con usuarios distribuidos geográficamente, AWS Global Accelerator ayuda a mejorar la disponibilidad y el rendimiento de las aplicaciones. AWS Global Accelerator proporciona direcciones IP estáticas de difusión por proximidad que sirven como punto de entrada fijo a su aplicación alojada en una o más Regiones de AWS. Esto permite que el tráfico entre en la red global de AWS lo más cerca posible de sus usuarios, lo que mejora la accesibilidad y disponibilidad de su carga de trabajo. AWS Global Accelerator también supervisa el estado de los puntos de conexión de su aplicación mediante comprobaciones de estado de TCP, HTTP y HTTPS. Cualquier cambio en el estado o la configuración de sus puntos de conexión activa el redireccionamiento del tráfico de usuario a puntos de conexión en buen estado que ofrezcan el mejor rendimiento y disponibilidad a los usuarios. Además, AWS Global Accelerator cuenta con un diseño de aislamiento de errores que utiliza dos direcciones IPv4 estáticas atendidas por zonas de red independientes que aumentan la disponibilidad de las aplicaciones.

Para ayudar a proteger a los clientes de ataques DDoS, AWS proporciona AWS Shield Standard. Shield Standard se activa automáticamente y protege de los ataques habituales a la infraestructura (capas 3 y 4), como las inundaciones de SYN/UDP y los ataques de reflexión, para respaldar la alta disponibilidad de sus aplicaciones en AWS. Para obtener protecciones adicionales contra ataques más sofisticados y grandes (como inundaciones de UDP) y ataques de agotamiento de estado (como inundaciones de TCP SYN), y para ayudar a proteger sus aplicaciones que se ejecutan en Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator y Route 53, puede considerar el uso de AWS Shield Advanced. Para la protección contra ataques en la capa de aplicación como HTTP POST o inundaciones de GET, utilice AWS WAF. AWS WAF puede utilizar condiciones de direcciones IP, encabezados HTTP, cuerpos HTTP, cadenas de URI, inyección de código SQL y scripting entre sitios para determinar si una solicitud debe bloquearse o permitirse.

Pasos para la implementación

1. Configure DNS de alta disponibilidad: Amazon Route 53 es un servicio web de sistema de nombres de dominio (DNS) (DNS) altamente disponible y escalable. Route 53 conecta las solicitudes de los usuarios con las aplicaciones de Internet que se ejecutan en AWS o localmente. Para obtener más información, consulte Configuración de Amazon Route 53 como servicio DNS.
2. Configure comprobaciones de estado: cuando utilice Route 53, verifique que solo se puedan resolver los destinos en buen estado. Empiece por Creación de comprobaciones de estado de Route 53 y configuración de la conmutación por error a nivel de DNS. Es importante tener en cuenta los siguientes aspectos a la hora de configurar las comprobaciones de estado:
   a. Cómo determina Amazon Route 53 si la comprobación de estado es correcta
   b. Creación, actualización y eliminación de comprobaciones de estado
   c. Supervisar el estado de la comprobación de estado y recibir notificaciones
   d. Prácticas recomendadas de DNS de Amazon Route 53
3. Conecte su servicio DNS a sus puntos de conexión.
   a. Al utilizar Elastic Load Balancing como destino de su tráfico, cree un registro de alias mediante Amazon Route 53 que apunte al punto de conexión regional de su equilibrador de carga. Durante la creación del registro de alias, establezca la opción de evaluación de estado del destino a Sí.
   b. Para cargas de trabajo sin servidor o API privadas cuando se utilice API Gateway, utilice Route 53 para dirigir el tráfico a API Gateway.
4. Decida la red de entrega de contenido.

   a. A la hora de entregar contenido mediante las ubicaciones periféricas más cercanas al usuario, comience por comprender cómo CloudFront entrega el contenido.
   b. Empiece con una distribución sencilla de CloudFront. CloudFront sabrá entonces desde dónde desea que se entregue el contenido, así como los detalles sobre cómo realizar el seguimiento y administrar la entrega de contenido. Es importante comprender y tener en cuenta los siguientes aspectos al configurar la distribución de CloudFront:
      i. Cómo funciona el almacenamiento en caché con ubicaciones periféricas de CloudFront
      ii. Incrementar la proporción de solicitudes que se atienden directamente desde las cachés de CloudFront (tasa de aciertos de caché)
      iii. Uso de Amazon CloudFront Origin Shield
      iv. Optimización de alta disponibilidad con conmutación por error de origen de CloudFront
5. Configure la protección de la capa de aplicación: AWS WAF le ayuda a protegerse contra ataques web y bots habituales que pueden afectar a la disponibilidad, comprometer la seguridad o consumir demasiados recursos. Para obtener una comprensión más profunda, revise cómo funciona AWS WAF y, cuando esté listo para implementar protecciones contra inundaciones de HTTP POST Y GET en la capa de aplicación, revise Introducción a AWS WAF. También puede utilizar AWS WAF con CloudFront. Consulte la documentación sobre cómo funciona AWS WAF con las características de Amazon CloudFront.
6. Configure protección DDoS adicional: de forma predeterminada, todos los clientes de AWS reciben protección frente a los ataques DDoS más habituales y frecuentes de la capa de red y transporte dirigidos a su sitio web o aplicación con AWS Shield Standard y sin ningún cargo adicional. Para obtener protección adicional de las aplicaciones orientadas a Internet que se ejecutan en Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator y Amazon Route 53 considere AWS Shield Advanced y revise los ejemplos de arquitecturas resistentes a DDoS. Para proteger su carga de trabajo y sus puntos de conexión públicos de ataques DDoS, consulte Introducción a AWS Shield Advanced.

## Recursos

Prácticas recomendadas relacionadas:

- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones
- REL10-BP02 Seleccionar las ubicaciones adecuadas para el despliegue en varias ubicaciones
- REL11-BP04 Confiar en el plano de datos y no en el plano de control durante la recuperación
- REL11-BP06 Enviar notificaciones cuando los eventos afecten a la disponibilidad

Documentos relacionados:

- Socio de APN: socios que pueden ayudarle a planificar sus redes
- AWS Marketplace for Network Infrastructure (AWS Marketplace para la infraestructura de red)
- ¿Qué es AWS Global Accelerator?
- ¿Qué es Amazon CloudFront?
- ¿Qué es Amazon Route 53?
- ¿Qué es Elastic Load Balancing?
- Network Connectivity capability - Establishing Your Cloud Foundations (Capacidad de conectividad de red: establecimiento de las bases de su nube)
- What is Amazon API Gateway? (¿Qué es Amazon API Gateway?)
- What are AWS WAF, AWS Shield, and AWS Firewall Manager? (¿Qué son AWS WAF, AWS Shield y AWS Firewall Manager?)
- What is Amazon Route 53 Application Recovery Controller? (¿Qué es el Controlador de recuperación de aplicaciones de Amazon Route 53?)
- Configurar las comprobaciones de estado personalizadas para la conmutación por error de DNS

Vídeos relacionados:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator (AWS re:Invent 2022: Mejorar el rendimiento y la disponibilidad con AWS Global Accelerator)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53 (AWS re:Invent 2020: Administración de tráfico global con Amazon Route 53)
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications (AWS re:Invent 2022: Funcionamiento de aplicaciones multi-AZ de alta disponibilidad)
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure (AWS re:Invent 2022: Profundización en la infraestructura de red de AWS)
- AWS re:Invent 2022 - Building resilient networks (AWS re:Invent 2022: Creación de redes resistentes)

Ejemplos relacionados:

- Disaster Recovery with Amazon Route 53 Application Recovery Controller (ARC) (Recuperación de desastres con el controlador de recuperación de aplicaciones [ARC] de Amazon Route 53)

- Reliability Workshops (Talleres de fiabilidad)
- AWS Global Accelerator Workshop (Taller de AWS Global Accelerator)

## REL02-BP02 Aprovisionar conectividad redundante entre las redes privadas en la nube y los entornos locales

Implemente la redundancia en las conexiones entre redes privadas en la nube y entornos locales para lograr la resiliencia de la conectividad. Esto se puede lograr mediante el despliegue de dos o más enlaces y rutas de tráfico, lo que permite mantener la conectividad en el caso de que se produzcan errores en la red.

Antipatrones usuales:

- Depende de una única conexión de red, lo que crea un único punto de error.
- Utiliza únicamente un túnel de VPN o varios túneles que terminan en la misma zona de disponibilidad.
- Confía en un único proveedor de servicios de Internet (ISP) para la conectividad de VPN, lo que puede provocar un fallo total de la conexión durante las interrupciones del ISP.
- No implementa protocolos de enrutamiento dinámico como BGP, que son fundamentales para redirigir el tráfico durante las interrupciones de la red.
- Ignora las limitaciones de ancho de banda de los túneles de VPN y sobrestima sus capacidades de copia de seguridad.

Beneficios de establecer esta práctica recomendada: al implementar la conectividad redundante entre el entorno en la nube y el entorno corporativo o local, los servicios dependientes entre los dos entornos se pueden comunicar sin problemas.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

### Guía para la implementación

Al utilizar AWS Direct Connect para conectar la red local a AWS, puede lograr la máxima resiliencia de la red (SLA del 99,99 %) mediante el uso de conexiones independientes que terminan en distintos dispositivos en más de una ubicación local y en más de una ubicación de AWS Direct Connect. Esta topología ofrece resiliencia frente a los errores de los dispositivos, los problemas de conectividad y las interrupciones totales de la conexión producidas en la ubicación. Como alternativa, puede lograr una alta resiliencia (SLA del 99,9 %) mediante el uso de dos conexiones individuales a varias

ubicaciones (cada ubicación local conectada a una única ubicación de Direct Connect). Este enfoque protege frente a las interrupciones de conectividad provocadas por cortes en la fibra o errores en los dispositivos y ayuda a mitigar los fallos totales de la conexión producidos en la ubicación. El AWS Direct Connect Resiliency Toolkit puede ayudarle a diseñar su topología de AWS Direct Connect.

También puede plantearse la posibilidad de utilizar AWS Site-to-Site VPN que finaliza en una AWS Direct Connect como una opción de conexión alternativa y rentable en el caso de que surjan problemas con la conexión principal de AWS Transit Gateway. Esta configuración permite el enrutamiento de múltiples rutas de igual coste (ECMP) a través de varios túneles de VPN, lo que permite un rendimiento de hasta 50 Gbps, aunque cada túnel de VPN tenga un límite de 1,25 Gbps. Sin embargo, es importante tener en cuenta que AWS Direct Connect sigue siendo la opción más eficaz para reducir al mínimo las interrupciones producidas en la red y proporcionar una conectividad estable.

Al utilizar VPN a través de Internet para conectar el entorno de nube al centro de datos local, configure dos túneles de VPN como parte de una única conexión de Site-to-Site VPN. Cada túnel debe terminar en una zona de disponibilidad diferente para lograr una alta disponibilidad y usar hardware redundante con el fin de evitar errores en los dispositivos locales. Asimismo, tenga en cuenta la posibilidad de establecer varias conexiones a Internet de varios proveedores de servicios de Internet (ISP) en su ubicación local para evitar una interrupción total de la conectividad de la VPN debido a una única interrupción del ISP. La selección de ISP con enrutamientos e infraestructuras diversos, especialmente aquellos con rutas físicas independientes a los puntos de enlace de AWS, proporciona una alta disponibilidad de la conectividad.

Además de la redundancia física con varias conexiones de AWS Direct Connect y varios túneles de VPN (o una combinación de ambos), también es fundamental implementar el enrutamiento dinámico de protocolo de puerta de enlace fronteriza (BGP). El BGP dinámico permite redireccionar automáticamente el tráfico de una ruta a otra en función de las condiciones de la red en tiempo real y las políticas configuradas. Este comportamiento dinámico es especialmente beneficioso para mantener la disponibilidad de la red y la continuidad del servicio en caso de que se produzcan errores de enlace o en la red. Selecciona rápidamente rutas alternativas, lo que mejora la resiliencia y la fiabilidad de la red.

Pasos para la implementación

- Establezca una conectividad de alta disponibilidad entre AWS y el entorno local.
  - Use varias conexiones de AWS Direct Connect o túneles de VPN entre redes privadas desplegadas por separado.

  - Use varias ubicaciones de AWS Direct Connect para contar con alta disponibilidad.
  - Si utiliza varias Regiones de AWS, cree redundancia en al menos dos de ellas.
- Utilice AWS Transit Gateway, cuando sea posible, para finalizar la conexión de VPN.
- Evalúe los dispositivos de AWS Marketplace para eliminar las VPN o amplíe su SD-WAN a AWS. Si utiliza dispositivos de AWS Marketplace, despliegue instancias redundantes para obtener alta disponibilidad en diferentes zonas de disponibilidad.
- Proporcione una conexión redundante al entorno local.
  - Es posible que necesite conexiones redundantes a varias Regiones de AWS para cubrir sus necesidades de disponibilidad.
  - Utilice el AWS Direct Connect Resiliency Toolkit para comenzar.

## Recursos

Documentos relacionados:

- Recomendaciones sobre resiliencia de AWS Direct Connect
- Using Redundant Site-to-Site VPN Connections to Provide Failover
- Políticas de enrutamiento y comunidades de BGP
- Active/Active and Active/Passive Configurations in AWS Direct Connect
- Socio de APN: socios que pueden ayudarle a planificar sus redes
- AWS Marketplace for Network Infrastructure (AWS Marketplace para la infraestructura de red)
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure (Creación de una infraestructura de red de AWS con varias VPC escalable y segura)
- Using redundant Site-to-Site VPN connections to provide failover
- Using the AWS Direct Connect Resiliency Toolkit to get started
- Puntos de conexión de VPC y servicios de punto de conexión de VPC (AWS PrivateLink)
- ¿Qué es Amazon VPC?
- What is a transit gateway?
- What is AWS Site-to-Site VPN?
- Working with Direct Connect gateways

Vídeos relacionados:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs

## REL02-BP03 Garantizar que la asignación de subredes IP tenga en cuenta la expansión y la disponibilidad

Los intervalos de direcciones IP de Amazon VPC deben ser lo suficientemente amplios como para dar cabida a los requisitos de las cargas de trabajo, como la posible expansión futura y la asignación de direcciones IP a las subredes de las zonas de disponibilidad. Esto incluye equilibradores de carga, instancias de EC2 y aplicaciones basadas en contenedores.

Cuando planifica la topología de su red, el primer paso es definir el espacio de la dirección IP. Se deben asignar rangos de direcciones IP privadas para cada VPC (siguiendo las directrices de la RFC 1918). Facilite los siguientes requisitos como parte de este proceso:

- Permita los espacios de direcciones IP para más de una VPC por región.
- En una VPC, deje espacio para varias subredes para poder cubrir varias zonas de disponibilidad.
- Considere la posibilidad de dejar siempre un espacio de bloque de CIDR sin usar en una VPC para posibles expansiones futuras.
- Asegúrese de que haya espacio de direcciones IP suficiente como para satisfacer las necesidades de flotas transitorias de instancias de Amazon EC2 que podría usar, como flotas de spot para el machine learning, clústeres de Amazon EMR o clústeres de Amazon Redshift. Se debe prestar una atención similar a los clústeres de Kubernetes, como Amazon Elastic Kubernetes Service (Amazon EKS), ya que a cada pod de Kubernetes se le asigna una dirección enrutable desde el bloque de CIDR de la VPC de forma predeterminada.
- Tenga en cuenta que las primeras cuatro direcciones IP y la última dirección IP de cada bloque CIDR de subred están reservadas y no están disponibles para que las use.
- Tenga en cuenta que el bloque de CIDR de la VPC inicial asignado a su VPC no debe cambiar ni eliminarse, pero puede añadir bloques de CIDR que no se solapen a la VPC. Los CIDR IPv4 de subred no se pueden cambiar; sin embargo, los CIDR IPv6 sí.
- El bloque de CIDR de VPC más grande posible es un /16 y el más pequeño es un /28.
- Tenga en cuenta otras redes conectadas (VPC, locales u otros proveedores de nube) y asegúrese de que el espacio de direcciones IP no se superponga. Para obtener más información, consulte

REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected.

Resultado deseado: Una subred IP escalable puede ayudarle a adaptarse al crecimiento futuro y evitar desperdicios innecesarios.

Patrones comunes de uso no recomendados:

- Si no se tiene en cuenta el crecimiento futuro, los bloques de CIDR serán demasiado pequeños y habrá que reconfigurarlos, lo que puede provocar tiempos de inactividad.
- Calcular incorrectamente cuántas direcciones IP puede usar un equilibrador de carga elástico.
- Implementar muchos equilibradores de carga de tráfico intenso en las mismas subredes.
- Usar mecanismos de escalado automatizados sin monitorizar el consumo de direcciones IP.
- Definir rangos de CIDR excesivamente grandes que superen con creces las expectativas de crecimiento futuro, lo que puede generar dificultades para conectarse con otras redes con rangos de direcciones superpuestos.

Beneficios de establecer esta práctica recomendada: De esta forma, se asegurará de dar cabida al crecimiento de sus cargas de trabajo y seguir proporcionando disponibilidad cuando aumente la capacidad.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio

## Guía para la implementación

Planificar su red para que se adapte al crecimiento, la conformidad normativa y la integración con otros. El crecimiento se puede subestimar, la conformidad normativa puede variar y las adquisiciones y conexiones de redes privadas pueden ser difíciles de realizar sin una planificación adecuada.

- Seleccione las regiones y Cuentas de AWS que correspondan según sus requisitos normativos y de servicio, latencia y recuperación ante desastres (DR).
- Identifique sus necesidades para despliegues regionales de VPC.
- Identifique el tamaño de las VPC.
  - Determine si va a implementar la conectividad de varias VPC.
    - What Is a Transit Gateway?
    - Conectividad de varias VPC en una sola región

- Determinar si necesita redes segregadas conforme a los requisitos normativos.
- Cree VPC con bloques de CIDR del tamaño adecuado para adaptarse a sus necesidades actuales y futuras.
  - Si desconoce sus proyecciones de crecimiento, es posible que desee optar por bloques de CIDR más grandes para no tener que volver a configurarlos en el futuro.
- Piense en la posibilidad de usar el direccionamiento IPv6 para subredes como parte de una VPC de doble pila. IPv6 es ideal en subredes privadas que contienen flotas de instancias o contenedores efímeros que, de otro modo, requerirían un gran número de direcciones IPv4.

## Recursos

Prácticas recomendadas por Well-Architected:

- REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

Documentos relacionados:

- Socio de APN: socios que pueden ayudarle a planificar sus redes
- AWS Marketplace para la infraestructura de red
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Conectividad de red de alta disponibilidad en varios centros de datos
- Conectividad de varias VPC en una sola región
- What Is Amazon VPC?
- IPv6 en AWS
- IPv6 en arquitecturas de referencia
- Amazon Elastic Kubernetes Service launches IPv6 support

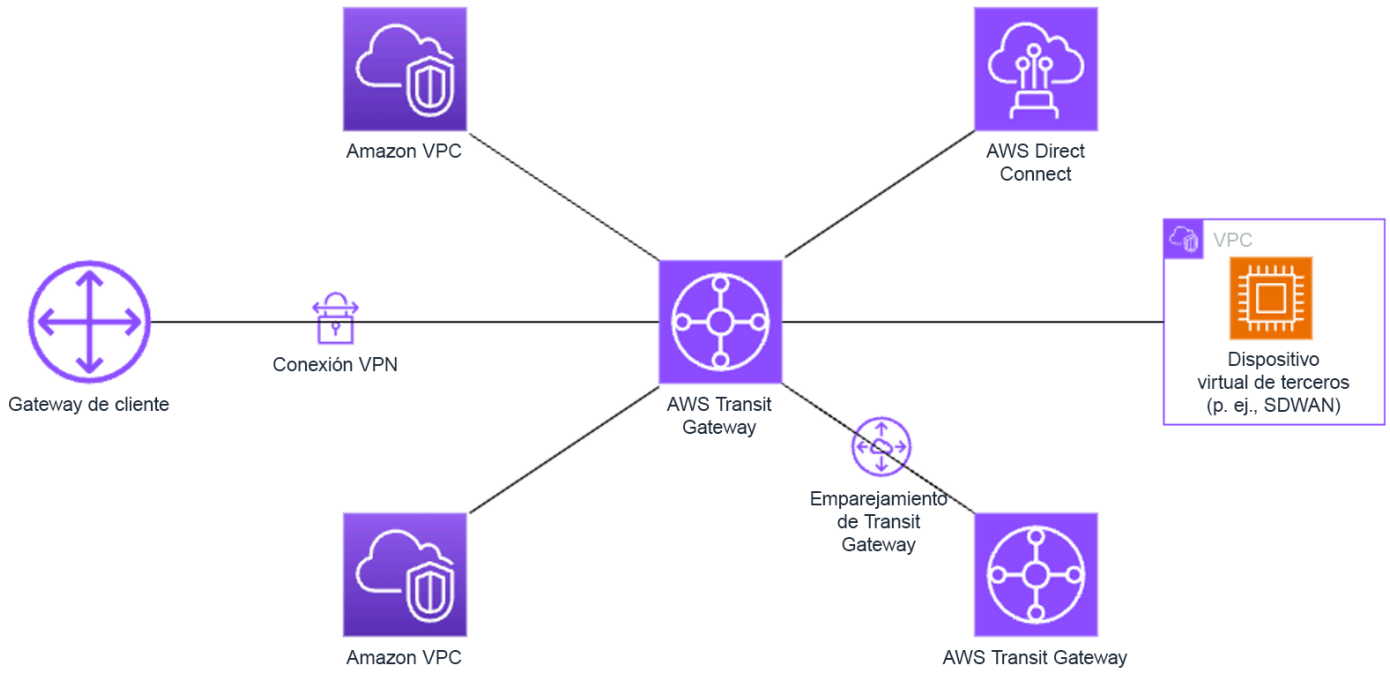
Vídeos relacionados:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)
- AWS re:Invent 2023: AWS Ready for what's next? Designing networks for growth and flexibility (NET310)

# REL02-BP04 Preferir topologías radiales (hub-and-spoke) a una conexión en malla de varios a varios

Al conectar varias redes privadas, como nubes virtuales privadas (VPC) y redes locales, opte por una topología radial (hub-and-spoke) en lugar de una en malla. A diferencia de las topologías en malla, en las que cada red se conecta directamente a las demás y aumenta la complejidad y la sobrecarga de administración, la arquitectura radial (hub-and-spoke) centraliza las conexiones a través de un único hub. Esta centralización simplifica la estructura de la red y mejora su operatividad, escalabilidad y control.

AWS Transit Gateway es un servicio administrado, escalable y de alta disponibilidad diseñado para la construcción de redes radiales (hub-and-spoke) en AWS. Sirve como hub central de la red que proporciona una segmentación de la red, un enrutamiento centralizado y una conexión simplificada a los entornos locales y en la nube. La siguiente figura muestra cómo puede utilizar AWS Transit Gateway para crear su topología radial (hub-and-spoke).




Antipatrones usuales:

- Las políticas de enrutamiento se complican en exceso en una arquitectura radial (hub-and-spoke), lo que reduce la eficiencia de la red y complica tanto la solución de problemas como la administración proactiva.

- Una segmentación insuficiente basada en el enrutamiento dentro del hub podría dar lugar a vulnerabilidades y eso podría exponer la red a un acceso no autorizado.
- Si no se realiza optimización cuidadosa, el tráfico que pasa por el hub puede generar mayores costes de transferencia de datos, especialmente para el tráfico que cruza zonas de disponibilidad y regiones. Es esencial disponer de estrategias eficaces de administración del tráfico para controlar los gastos.

Beneficios de establecer esta práctica recomendada:  a medida que aumenta la cantidad de redes conectadas, la administración y la expansión de la conectividad en malla es cada vez más complicada. AWS Transit Gateway ofrece un hub administrado escalable y fiable para crear y operar topologías radiales (hub-and-spoke) . Cuando usa AWS Transit Gateway, puede establecer conexiones y centralizar el enrutamiento del tráfico en varias redes.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

- Planifique su red.
- Cree su AWS Transit Gateway.
- Conecte sus VPC.
- Si es necesario, cree conexiones VPN o puertas de enlace de Direct Connect y asócielas a Transit Gateway.
- Defina cómo se enruta el tráfico entre las VPC conectadas y otras conexiones mediante la configuración de las tablas de enrutamiento de Transit Gateway.
- Utilice Amazon CloudWatch para supervisar y ajustar las configuraciones según sea necesario para optimizar el rendimiento y los costes.

## Recursos

Documentos relacionados:

- «What Is a Transit Gateway?»
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure (Creación de una infraestructura de red de AWS con varias VPC escalable y segura)
- «Building a global network using AWS Transit Gateway Inter-Region peering»

- «Amazon Virtual Private Cloud Connectivity Options»
- Socio de APN: socios que pueden ayudarle a planificar sus redes
- AWS Marketplace for Network Infrastructure (AWS Marketplace para la infraestructura de red)

Vídeos relacionados:

- «AWS re:Invent 2023 - AWS networking foundations»
- «AWS re:Invent 2023 - Advanced VPC designs and new capabilities»

## REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

Los intervalos de direcciones IP de cada VPC no deben solaparse si se emparejan o conectan mediante Transit Gateway o VPN. Evite conflictos de direcciones IP entre una VPC y los entornos locales o con otros proveedores de servicios en la nube que utilice. También debe tener una forma de asignar intervalos de direcciones IP privadas cuando sea necesario. Un sistema de administración de direcciones IP (IPAM) puede ayudar en esta automatización.

Resultado deseado:

- No hay conflictos de intervalo de direcciones IP entre VPC, entornos locales u otros proveedores de servicios en la nube.
- La administración adecuada de las direcciones IP permite escalar de forma más sencilla la infraestructura de red para adaptarse al crecimiento y los cambios en los requisitos de la red.

Antipatrones usuales:

- Usar el mismo intervalo de direcciones IP en la VPC que en el entorno local, en la red corporativa o en otros proveedores de servicios en la nube
- No controlar los intervalos de direcciones IP de las VPC usadas para implementar sus cargas de trabajo
- Confiar en los procesos manuales de administración de direcciones IP, como, por ejemplo, las hojas de cálculo
- Sobredimensionar o infradimensionar bloques de CIDR, lo que provoca un derroche en el uso de direcciones IP o un espacio insuficiente para las direcciones de la carga de trabajo

Beneficios de establecer esta práctica recomendada: la planificación activa de la red garantizará que no tenga varias instancias de la misma dirección IP en las redes interconectadas. Con esto evitará que se produzcan problemas de enrutamiento en las partes de la carga de trabajo que usan las diferentes aplicaciones.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Utilice un administrador de direcciones IP (IPAM), como, por ejemplo, Amazon VPC IP Address Manager, para supervisar y administrar el uso de CIDR. En AWS Marketplace también hay disponibles varios IPAM. Evalúe su potencial de uso en AWS, añada intervalos de CIDR a las VPC existentes y cree VPC para permitir un crecimiento planificado del uso.

Pasos para la implementación

- Capture el consumo actual de CIDR (por ejemplo, VPC y subredes).
  - Use operaciones de la API de servicio para recopilar el consumo actual de CIDR.
  - Consulte Amazon VPC IP Address Manager para descubrir recursos.
- Registre el uso actual de la subred.
  - Use operaciones de la API de servicio para capturar subredes por VPC en cada región
  - Consulte Amazon VPC IP Address Manager para descubrir recursos.
- Registre el uso actual.
- Determine si ha creado intervalos de direcciones IP superpuestos.
- Calcule la capacidad de reserva.
- Identifique los intervalos de direcciones IP superpuestos. Puede migrar a un nuevo intervalo de direcciones o plantearse el uso de técnicas como una puerta de enlace de NAT privada o AWS PrivateLink si necesita conectar los intervalos superpuestos.

## Recursos

Prácticas recomendadas relacionadas:

- Protección de redes

Documentos relacionados:

- Socio de APN: socios que pueden ayudarle a planificar sus redes
- AWS Marketplace for Network Infrastructure (AWS Marketplace para la infraestructura de red)
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Conectividad de red de alta disponibilidad en varios centros de datos
- Connecting Networks with Overlapping IP Ranges
- ¿Qué es Amazon VPC?
- ¿Qué es IPAM?

Vídeos relacionados:

- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs
- AWS re:Invent 2023 - Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2021 - {New Launch} Manage your IP addresses at scale on AWS

# Arquitectura de la carga de trabajo

Una carga de trabajo fiable comienza por tomar decisiones de diseño anticipadas tanto para el software como para la infraestructura. Sus elecciones respecto a la arquitectura tendrán un impacto sobre el comportamiento de su carga de trabajo en los seis pilares de Well-Architected. En lo tocante a la fiabilidad, hay patrones específicos que debe seguir.

En las siguientes secciones se explican las prácticas recomendadas que se deben usar en estos patrones en términos de fiabilidad.

Temas



## Diseñar la arquitectura de servicio de su carga de trabajo

Desarrolle cargas de trabajo sumamente escalables y fiables utilizando una arquitectura orientada a servicios (SOA) o una arquitectura de microservicios. La SOA es la práctica de hacer que los componentes de software se puedan reutilizar mediante interfaces de servicio. La arquitectura de microservicios va más allá, para hacer que los componentes sean más pequeños y sencillos.

Las interfaces de la SOA usan estándares de comunicación comunes, de modo que pueden incorporarse rápidamente a nuevas cargas de trabajo. La SOA sustituye la práctica de desarrollar arquitecturas monolíticas, compuestas por unidades interdependientes e indivisibles.

En AWS, siempre hemos utilizado la SOA, pero ahora hemos decidido desarrollar nuestros sistemas utilizando microservicios. Si bien los microservicios tienen varias cualidades atractivas, el beneficio más importante para la disponibilidad es que estos son más pequeños y sencillos. Permiten diferenciar la disponibilidad requerida de diferentes servicios y, por lo tanto, centrar las inversiones más específicamente en los microservicios que tienen mayores necesidades de disponibilidad. Por ejemplo, para entregar páginas de información de productos en Amazon.com («páginas de detalles»), se invocan cientos de microservicios para crear distintas partes de la página. Si bien hay algunos servicios que deben estar disponibles para proporcionar el precio y los detalles del producto, la gran mayoría del contenido de la página puede simplemente excluirse si el servicio no está disponible. Incluso las fotos y las reseñas no son necesarias para proporcionar una experiencia en la que un cliente pueda comprar un producto.

Prácticas recomendadas

- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL03-BP02 Desarrollar servicios centrados en funcionalidades y dominios empresariales específicos
- REL03-BP03 Facilitar contratos de servicio por cada API

## REL03-BP01 Elegir cómo segmentar su carga de trabajo

La segmentación de la carga de trabajo es importante a la hora de determinar los requisitos de resiliencia de su aplicación. La arquitectura monolítica debe evitarse siempre que sea posible. En su lugar, considere detenidamente qué componentes de la aplicación pueden dividirse en microservicios. Según los requisitos de su aplicación, esto puede terminar siendo una combinación de una arquitectura orientada a servicios (SOA) con microservicios cuando sea posible. Las cargas de trabajo que son capaces de no tener estado son más capaces desplegarse como microservicios.

Resultado deseado: Las cargas de trabajo deben ser soportables, escalables y estar tan poco acopladas como sea posible.

A la hora de elegir cómo segmentar la carga de trabajo, hay que sopesar las ventajas frente a las complejidades. Lo que puede ser adecuado para un nuevo producto encaminado a su primer lanzamiento es diferente a lo que necesita una carga de trabajo creada para escalarse desde el principio. Al refactorizar un monolito existente, tendrá que considerar en qué medida soportará la aplicación una descomposición hacia la falta de estado. Dividir los servicios en partes más pequeñas permite que equipos pequeños y bien definidos los desarrollen y administren. No obstante, los servicios más pequeños pueden introducir complejidades que incluyen un aumento de la latencia, una depuración más compleja y un mayor lastre operativo.

Patrones comunes de uso no recomendados:

- El microservicio Death Star es una situación en la que los componentes atómicos son tan interdependientes que el error de uno de ellos provoca un error mucho mayor, haciendo que los componentes sean tan rígidos y frágiles como un monolito.

Beneficios de establecer esta práctica:

- Los segmentos más específicos conducen a una mayor agilidad, flexibilidad organizativa y escalabilidad.

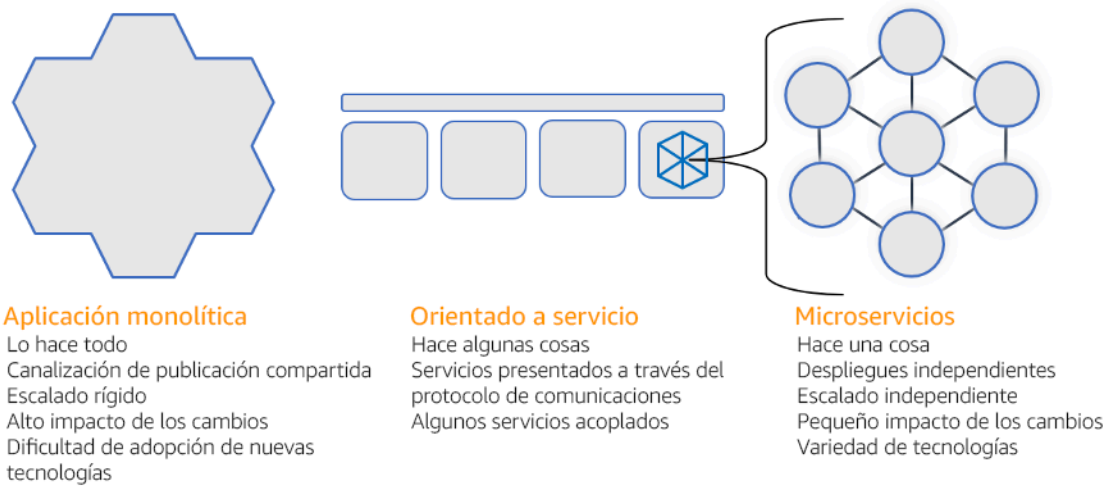
- Reducción del impacto de las interrupciones del servicio.
- Los componentes de la aplicación pueden tener diferentes requisitos de disponibilidad, que pueden soportarse mediante una segmentación más atómica.
- Responsabilidades bien definidas para los equipos que apoyan la carga de trabajo.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Seleccione el tipo de arquitectura en función de cómo va a segmentar su carga de trabajo. Seleccione una SOA o una arquitectura de microservicios (o, en algunos casos raros, una arquitectura monolítica). Incluso si decide empezar con una arquitectura monolítica, debe asegurarse de que sea modular y de que pueda evolucionar hacia SOA o microservicios de forma definitiva, a medida que su producto escala con la adopción por parte de los usuarios. La SOA y los microservicios ofrecen respectivamente una segmentación más pequeña, lo que resulta preferible como arquitectura moderna escalable y confiable, pero existen compensaciones a tener en cuenta, especialmente al desplegar una arquitectura de microservicios.

Una compensación principal es que se dispone de una arquitectura de informática distribuida que puede dificultar el cumplimiento de los requisitos de latencia del usuario y existe una complejidad adicional en la depuración y el rastreo de las interacciones del usuario. Puede utilizar AWS X-Ray para ayudarle a resolver este problema. Otro efecto que hay que tener en cuenta es el aumento de la complejidad operativa a medida que aumenta el número de aplicaciones que se administran, lo que requiere el despliegue de componentes con varias independencias.



Arquitecturas monolíticas, orientadas al servicio y de microservicios

## Pasos para la aplicación

- Determine la arquitectura adecuada para refactorizar o desarrollar su aplicación. La SOA y los microservicios ofrecen respectivamente una segmentación más pequeña, lo que resulta preferible como arquitectura moderna escalable y confiable. La SOA puede ofrecer un término intermedio ideal para conseguir una segmentación más pequeña y, a la vez, evitar algunas de las complejidades de los microservicios. Para obtener más información, consulte Microservice Trade-Offs (Compensaciones de microservicios).
- Si su carga de trabajo lo admite y su organización puede permitírselo, debería usar una arquitectura de microservicios para conseguir la mejor agilidad y fiabilidad. Para obtener más información, consulte Implementación de microservicios en AWS
- Tenga en cuenta seguir el patrón de Strangler Fig para refactorizar un monolito en componentes más pequeños. Se trata de sustituir gradualmente componentes específicos de la aplicación por nuevas aplicaciones y servicios. AWS Migration Hub Refactor Spaces actúa como punto de partida para la refactorización incremental. Para obtener más información, consulte Seamlessly migrate on-premises legacy workloads using a strangler pattern (Migrar sin problemas las cargas de trabajo heredadas localmente utilizando un patrón estrangulador).
- La implementación de microservicios puede requerir un mecanismo de detección de servicios para permitir que estos servicios distribuidos se comuniquen entre sí. AWS App Mesh se puede usar con arquitecturas orientadas a servicios para ofrecer una detección y un acceso confiable a los servicios. AWS Cloud Map también puede utilizarse para la detección dinámica de servicios basada en DNS.
- Si está migrando de un monolito a SOA, Amazon MQ puede ayudar a salvar la brecha como un bus de servicio cuando se rediseñan las aplicaciones heredadas en la nube.
- Para los monolitos existentes con una única base de datos compartida, elija cómo reorganizar los datos en segmentos más pequeños. Puede ser por unidad de negocio, patrón de acceso o estructura de datos. En este punto del proceso de refactorización, debe elegir entre una base de datos de tipo relacional o no relacional (NoSQL). Para obtener más información, consulte From SQL to NoSQL (De SQL a NoSQL).

Nivel de esfuerzo para el plan de implementación: Alto

## Recursos

Prácticas recomendadas relacionadas:

- REL03-BP02 Desarrollar servicios centrados en funcionalidades y dominios empresariales específicos

Documentos relacionados:

- Amazon API Gateway: Configuración de una API de REST con OpenAPI
- ¿Qué es la arquitectura orientada a servicios?
- Contexto limitado (un patrón central en un diseño basado en dominios)
- Implementación de microservicios en AWS
- Microservice Trade-Offs (Compensaciones de microservicios)
- Microservicios: definición de este nuevo término de arquitectura
- Microservicios en AWS
- ¿Qué es AWS App Mesh?

Ejemplos relacionados:

- Taller de modernización de aplicaciones iterativas

Vídeos relacionados:

- Delivering Excellence with Microservices on AWS (Proporcionar excelencia con microservicios en AWS)

## REL03-BP02 Desarrollar servicios centrados en funcionalidades y dominios empresariales específicos

La arquitectura orientada a servicios (SOA) define servicios con funciones bien delineadas que están determinadas por necesidades empresariales. Los microservicios utilizan modelos de dominio y contextos delimitados para trazar los límites de los servicios en los límites del contexto empresarial. Centrarse en los dominios y las funcionalidades empresariales ayuda a los equipos a definir requisitos de fiabilidad independientes para sus servicios. Los contextos delimitados aíslan y encapsulan la lógica empresarial, lo que permite a los equipos razonar mejor la forma de gestionar los errores.

Resultado deseado: los ingenieros y las partes interesadas de la empresa definen conjuntamente los contextos delimitados y los utilizan para diseñar sistemas como servicios que cumplan funciones empresariales específicas. Estos equipos utilizan prácticas establecidas, como las tormentas de eventos, para definir los requisitos. Las nuevas aplicaciones se diseñan como límites bien definidos de servicios y con acoplamiento flexible. Los monolitos existentes se descomponen en contextos delimitados y los diseños de sistemas se mueven a arquitecturas SOA o microservicios. Cuando los monolitos se refactorizan, se aplican enfoques establecidos, como contextos de burbujas y patrones de descomposición de monolitos.

Los servicios orientados al dominio se ejecutan como uno o más procesos que no comparten el estado. Responden de forma independiente a las fluctuaciones de la demanda y gestionan los escenarios de error teniendo en cuenta los requisitos específicos del dominio.

Patrones comunes de uso no recomendados:

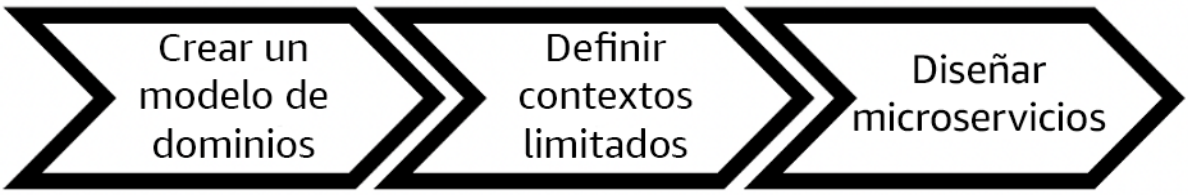
- Se forman equipos en torno a dominios técnicos específicos, como la interfaz de usuario y la experiencia de usuario, el middleware o la base de datos, en lugar de formarse en torno a dominios empresariales específicos.
- Las aplicaciones abarcan las responsabilidades del dominio. Los servicios que abarcan contextos delimitados pueden ser más difíciles de mantener, exigen más pruebas y requieren la participación de equipos de varios dominios en las actualizaciones del software.
- Las dependencias de dominio, como las bibliotecas de entidades de dominio, se comparten entre los servicios, de modo que los cambios en un dominio de servicio requieren cambios en otros dominios de servicio.
- Los contratos de servicio y la lógica empresarial no expresan las entidades en un lenguaje de dominio común y coherente, lo que genera capas de traducción que complican los sistemas e incrementan los esfuerzos de depuración.

Beneficios de establecer esta práctica recomendada: las aplicaciones se diseñan como servicios independientes delimitados por dominios empresariales y utilizan un lenguaje empresarial común. Los servicios se pueden probar y desplegar de forma independiente. Los servicios cumplen los requisitos de resiliencia específicos del dominio implementado.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

El enfoque de decisiones impulsadas por dominio (DDD) es el enfoque fundamental para diseñar y crear software en torno a los dominios empresariales. Resulta útil trabajar con un marco existente a la hora de crear servicios centrados en dominios empresariales. Si trabaja con aplicaciones monolíticas existentes, puede utilizar patrones de descomposición que proporcionan técnicas establecidas para modernizar las aplicaciones y convertirlas en servicios.




Decisión impulsada por dominio

## Pasos para la implementación

- Los equipos pueden realizar talleres de tormentas de eventos para identificar rápidamente eventos, comandos, agregados y dominios en un formato de notas adhesivas más ligero.
- Una vez que se hayan elaborado las entidades y funciones de dominio en el contexto de un dominio, puede dividir su dominio en servicios mediante un contexto delimitado, en el que se agrupan las entidades que comparten características y atributos similares. Si el modelo está dividido en contextos, tendrá una plantilla para limitar los microservicios.
  - Por ejemplo, las entidades del sitio web de Amazon.com podrían incluir el empaquetado, la entrega, la programación, el precio, el descuento y la divisa.
  - El empaquetado, la entrega y la programación se agrupan en el contexto del envío, mientras que el precio, el descuento y la divisa se agrupan en el contexto de los precios.
- En Decomposing monoliths into microservices (Descomposición de monolitos en microservicios), se describen patrones para refactorizar microservicios. El uso de patrones de descomposición por capacidad empresarial, subdominio o transacción se ajusta bien a los enfoques basados en dominios.
- Existen técnicas estratégicas, como el contexto de burbuja , que permiten introducir DDD en aplicaciones existentes o heredadas sin necesidad de reescrituras iniciales ni confirmaciones completas de las DDD. En un enfoque de contexto de burbujas, se establece un contexto pequeño y delimitado mediante la asignación y coordinación de servicios, o una capa anticorrupción, que protege el modelo de dominio recién definido de las influencias externas.

Una vez que los equipos hayan analizado el dominio y hayan definido las entidades y los contratos de servicio, pueden utilizar los servicios de AWS para implementar su diseño basado en dominio como servicios basados en la nube.

- Para comenzar el desarrollo, defina pruebas en las que se utilicen las reglas empresariales de su dominio. El desarrollo basado en pruebas (TDD) y el desarrollo basado en comportamiento (BDD) ayudan a los equipos a mantener los servicios centrados en resolver problemas empresariales.
- Seleccione los servicios de AWS que mejor se ajusten a los requisitos de su dominio empresarial y arquitectura de microservicios:
  - AWS sin servidor permite a su equipo centrarse en una lógica de dominio específica en lugar de administrar servidores e infraestructuras.
  - Los contenedores de AWS simplifican la administración de su infraestructura para que pueda centrarse en los requisitos de su dominio.
  - Las bases de datos personalizadas le ayudan a adaptar los requisitos de su dominio al tipo de base de datos más adecuado.
- En Building hexagonal architectures on AWS (Desarrollo de arquitecturas hexagonales en AWS), se describe un marco para integrar la lógica empresarial en los servicios que funcionan de manera inversa desde un dominio empresarial para cumplir los requisitos funcionales y, a continuación, asociar los adaptadores de integración. Los patrones que separan los detalles de la interfaz de la lógica empresarial con los servicios de AWS ayudan a los equipos a centrarse en la funcionalidad del dominio y a mejorar la calidad del software.

## Recursos

Prácticas recomendadas relacionadas:

- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL03-BP03 Facilitar contratos de servicio por cada API

Documentos relacionados:

- AWS Microservicios
- Implementing Microservices on AWS (Implementación de microservicios en AWS)
- How to break a Monolith into Microservices (Cómo descomponer un sistema monolítico en microservicios)

- Getting Started with DDD when Surrounded by Legacy Systems (Introducción al DDD en un ambiente lleno de sistemas heredados)
- Domain-Driven Design: Tackling Complexity in the Heart of Software (Diseño basado en dominios: abordar la complejidad en el corazón del software)
- En Building hexagonal architectures on AWS (Desarrollo de arquitecturas hexagonales en AWS),
- Decomposing monoliths into microservices (Descomposición de monolitos en microservicios),
- Event Storming (Tormentas de eventos)
- Messages Between Bounded Contexts (Mensajes entre contextos delimitados)
- Microservicios
- Test-driven development (Desarrollo basado en pruebas)
- Behavior-driven development (Desarrollo basado en comportamiento)

Ejemplos relacionados:

- Enterprise Cloud Native Workshop (Taller sobre entornos nativos de la nube empresarial)
- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop) (Diseño de microservicios nativos en la nube en AWS [de DDD/EventStormingWorkshop])

Herramientas relacionadas:

- Bases de datos en la Nube de AWS
- Sin servidor en AWS
- Contenedores de AWS

## REL03-BP03 Facilitar contratos de servicio por cada API

Los contratos de servicio son acuerdos documentados entre los productores y los consumidores de las API que se encuentran en una definición de API legible por máquina. Una estrategia de control de versiones permite a los clientes seguir usando la API existente y migrar sus aplicaciones a la nueva API cuando estén listas. El despliegue del productor puede realizarse en cualquier momento, siempre y cuando se cumpla el contrato. Los equipos del servicio pueden usar la pila tecnológica que prefieran para cumplir el contrato de la API.

Resultado deseado:

Patrones comunes de uso no recomendados: las aplicaciones creadas con arquitecturas orientadas a servicios o de microservicios pueden funcionar de forma independiente y, al mismo tiempo, tener integrada una dependencia de la versión ejecutable. Los cambios desplegados en un consumidor o productor de API no interrumpen la estabilidad del sistema general cuando ambas partes utilizan el mismo contrato de API. Los componentes que se comunican a través de las API de servicio pueden realizar lanzamientos funcionales independientes, actualizar las dependencias de versiones ejecutables o realizar conmutaciones por error a un sitio de recuperación de desastres (DR) con poco o ningún impacto entre sí. Además, los servicios discretos pueden escalarse de forma independiente y absorber la demanda de recursos sin que sea necesario que otros servicios se escalen al unísono.

- Crear API de servicio sin esquemas estrictamente asignados. Como consecuencia, las API no se pueden usar para generar enlaces de API y las cargas útiles no se pueden validar mediante programación.
- No adoptar una estrategia de control de versiones, lo que obliga a los usuarios de la API a actualizarla y lanzarla; de lo contrario, fallará cuando los contratos de servicio evolucionen.
- Mensajes de error que filtran detalles de la implementación del servicio subyacente en lugar de describir los errores de integración en el contexto y el lenguaje del dominio.
- No utilizar contratos de API para desarrollar casos de prueba ni simulaciones de implementaciones de API para probar de forma independiente los componentes del servicio.

Beneficios de establecer esta práctica recomendada: los sistemas distribuidos que constan de componentes que se comunican a través de contratos de servicio de API pueden mejorar la fiabilidad. Los desarrolladores pueden detectar posibles problemas al principio del proceso de desarrollo mediante la comprobación de tipos durante la compilación para comprobar que las solicitudes y las respuestas cumplen el contrato de la API y que los campos obligatorios están presentes. Los contratos de la API proporcionan una interfaz clara y autodocumentada para las API y mejoran la interoperabilidad entre diferentes sistemas y lenguajes de programación.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio

## Guía para la implementación

Una vez que hayan identificado los dominios empresariales y determinado la segmentación de la carga de trabajo, podrá desarrollar las API de sus servicios. Primero, defina contratos de servicio legibles por máquina para las API y, a continuación, implemente una estrategia de control de versiones de API. Cuando esté preparado para integrar servicios a través de protocolos comunes,

como REST, GraphQL o eventos asíncronos, podrá incorporar servicios de AWS a su arquitectura para integrar sus componentes con contratos de API estrictamente asignados.

Servicios de AWS para contratos de API de servicios

Incorpore servicios de AWS, como Amazon API Gateway, AWS AppSyncy Amazon EventBridge, en su arquitectura para usar contratos de servicio de API en su aplicación. Amazon API Gateway le ayuda a integrarse directamente con servicios de AWS nativos y otros servicios web. API Gateway admite la especificación de OpenAPI y el control de versiones. AWS AppSync es un punto de conexión de GraphQL administrado que se configura definiendo un esquema de GraphQL para definir una interfaz de servicio para consultas, mutaciones y suscripciones. Amazon EventBridge usa esquemas de eventos para definir eventos y generar enlaces de código para sus eventos.

## Pasos para la implementación

- Primero, defina un contrato para su API. En un contrato, se expresan las capacidades de una API y se definen objetos y campos de datos estrictamente asignados para la entrada y la salida de la API.
- Cuando configure las API en API Gateway, puede importar y exportar las especificaciones de OpenAPI para sus puntos de conexión.
  - La importación de una definición de OpenAPI simplifica la creación de su API y se puede integrar con la infraestructura de AWS, como las herramientas de código AWS Serverless Application Model y AWS Cloud Development Kit (AWS CDK).
  - La exportación de una definición de API simplifica la integración con las herramientas de prueba de API y proporciona a los consumidores de servicios una especificación de la integración.
- Puede definir y administrar las API de GraphQL con AWS AppSync mediante la definición de un archivo de esquema de GraphQL para generar la interfaz del contrato y simplificar la interacción con modelos REST complejos, múltiples tablas de bases de datos o servicios heredados.
- Los proyectos de AWS Amplify que están integrados con AWS AppSync generan archivos de consulta de JavaScript estrictamente asignados para usarlos en su aplicación, así como una biblioteca de clientes de AWS AppSync GraphQL para tablas de Amazon DynamoDB .
- Cuando se consumen eventos de servicio de Amazon EventBridge, los eventos se ajustan a esquemas que ya existen en el registro de esquemas o que usted define con la especificación de OpenAPI. Si tiene un esquema definido en el registro, también puede generar enlaces de clientes desde el contrato de esquema para integrar el código con los eventos.

- Amplíe o realice un control de versiones de la API. Ampliar una API es la opción más sencilla cuando se añaden campos que se pueden configurar con campos opcionales o valores predeterminados para los campos obligatorios.
  - Los contratos basados en JSON para protocolos como REST y GraphQL pueden ser una buena opción para la ampliación del contrato.
  - Los contratos basados en XML para protocolos como SOAP deben probarse con los consumidores de servicios para determinar la viabilidad de la ampliación del contrato.
- Al realizar el control de versiones de una API, considere la posibilidad de implementar un control de versiones por proxy en el que se utilice una fachada para admitir las versiones, de modo que la lógica se pueda mantener en una única base de código.
  - Con API Gateway, puede usar mapeos de solicitudes y respuestas para simplificar la absorción de los cambios en los contratos mediante el establecimiento de una fachada que proporcione valores predeterminados para los campos nuevos o para quitar los campos eliminados de una solicitud o respuesta. Con este enfoque, el servicio subyacente puede mantener una única base de código.

## Recursos

Prácticas recomendadas relacionadas:

- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL03-BP02 Desarrollar servicios centrados en funcionalidades y dominios empresariales específicos
- REL04-BP02 Implementar dependencias con acoplamiento flexible
- REL05-BP03 Controlar y limitar las llamadas de reintento
- REL05-BP05 Definir tiempos de espera del cliente

Documentos relacionados:

- ¿Qué es una API?
- Implementing Microservices on AWS (Implementación de microservicios en AWS)
- Microservice Trade-Offs (Compensaciones de microservicios)
- Microservicios: definición de este nuevo término de arquitectura
- Microservicios en AWS

- Trabajar con extensiones de API Gateway para OpenAPI
- OpenAPI-Specification (Especificación de OpenAPI)
- GraphQL: Schemas and Types (GraphQL: esquemas y tipos)
- Amazon EventBridge code bindings (Enlaces de código de EventBridge)

Ejemplos relacionados:

- Amazon API Gateway: Configuración de una API de REST con OpenAPI
- Amazon API Gateway to Amazon DynamoDB CRUD application using OpenAPI (Aplicación CRUD de Amazon API Gateway en Amazon DynamoDB mediante OpenAPI)
- Modern application integration patterns in a serverless age: API Gateway Service Integration (Patrones de integración de aplicaciones modernos en una era sin servidores: integración de servicios de API Gateway)
- Implementing header-based API Gateway versioning with Amazon CloudFront (Implementación del control de versiones de API Gateway basado en encabezados con Amazon CloudFront)
- AWS AppSync: Building a client application (Creación de una aplicación cliente)

Vídeos relacionados:

- Using OpenAPI in AWS SAM to manage API Gateway (Uso de OpenAPI en AWS SAM para administrar API Gateway)

Herramientas relacionadas:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge,

## Diseñe interacciones en un sistema distribuido para evitar errores

Los sistemas distribuidos dependen de las redes de comunicaciones para interconectar componentes, como servidores o servicios. Su carga de trabajo debe funcionar de manera fiable aunque se pierdan datos o haya latencia en estas redes. Los componentes del sistema distribuido deben funcionar de forma que no repercutan negativamente en otros componentes ni en la carga de

trabajo. Estas prácticas recomendadas evitan que se produzcan errores y mejoran el tiempo medio entre errores (MTBF).

Prácticas recomendadas

- REL04-BP01 Identificar el tipo de sistemas distribuidos de los que depende
- REL04-BP02 Implementar dependencias con acoplamiento flexible
- REL04-BP03 Realizar un trabajo constante
- REL04-BP04 Hacer que todas las respuestas sean idempotentes

## REL04-BP01 Identificar el tipo de sistemas distribuidos de los que depende

Los sistemas distribuidos pueden ser síncronos, asíncronos o por lotes. Los sistemas síncronos deben procesar las solicitudes lo más rápido posible y comunicarse entre sí mediante llamadas síncronas de solicitud y respuesta mediante protocolos HTTP/S, REST o de llamada a procedimiento remoto (RPC). Los sistemas asíncronos se comunican entre sí mediante el intercambio de datos de forma asíncrona a través de un servicio intermediario sin acoplar sistemas individuales. Los sistemas por lotes reciben un gran volumen de datos de entrada, ejecutan procesos de datos automatizados sin intervención humana y generan datos de salida.

Resultado deseado: diseña una carga de trabajo que interactúe eficazmente con las dependencias síncronas, asíncronas y por lotes.

Patrones comunes de uso no recomendados:

- La carga de trabajo espera indefinidamente una respuesta de sus dependencias, lo que podría provocar que se agote el tiempo de espera de los clientes de la carga de trabajo sin saber si su solicitud se ha recibido.
- La carga de trabajo utiliza una cadena de sistemas dependientes que se llaman entre sí de forma síncrona. Para ello, cada sistema debe estar disponible y procesar correctamente una solicitud para que toda la cadena pueda tener éxito, lo que se traduce en un comportamiento y una disponibilidad general potencialmente frágiles.
- La carga de trabajo se comunica con sus dependencias de forma asíncrona y confía en que la entrega de mensajes exactamente una vez está garantizada, cuando muchas veces sigue siendo posible que se reciban mensajes duplicados.
- La carga de trabajo no utiliza herramientas adecuadas de programación por lotes y permite la ejecución simultánea del mismo trabajo por lotes.

Beneficios de establecer esta práctica recomendada: es habitual que una determinada carga de trabajo implemente uno o más estilos de comunicación, ya sea síncrono, asíncrono o por lotes. Esta práctica recomendada le ayuda a identificar las diferentes ventajas y desventajas asociadas a cada estilo de comunicación para que su carga de trabajo pueda tolerar las interrupciones en cualquiera de sus dependencias.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

Las siguientes secciones contienen una guía de implementación general y específica para cada tipo de dependencia.

### Guía general

- Asegúrese de que los objetivos de nivel de servicio (SLO) de rendimiento y fiabilidad que ofrecen sus dependencias cumplan los requisitos de rendimiento y fiabilidad de su carga de trabajo.
- Use servicios de observabilidad de AWS para supervisar los tiempos de respuesta y las tasas de error con el fin de asegurarse de que su dependencia atienda los niveles que necesita su carga de trabajo.
- Identifique los posibles desafíos a los que puede enfrentarse su carga de trabajo al comunicarse con sus dependencias. Los sistemas distribuidos presentan una amplia gama de desafíos que pueden aumentar la complejidad de la arquitectura, la carga operativa y el coste. Entre los desafíos comunes, se incluyen la latencia, las interrupciones de la red, la pérdida de datos, el escalamiento y el retardo en la replicación de datos.
- Implemente una gestión de errores y un registro sólidos que le ayuden a solucionar problemas cuando su dependencia experimente problemas.

### Dependencia síncrona

En las comunicaciones síncronas, la carga de trabajo envía una solicitud a su dependencia y bloquea la operación en espera de una respuesta. Cuando la dependencia recibe la solicitud, intenta gestionarla lo antes posible y envía una respuesta a su carga de trabajo. Un problema importante de la comunicación síncrona es que provoca un acoplamiento temporal, por lo que la carga de trabajo y las dependencias de esta deben estar disponibles al mismo tiempo. Cuando la carga de trabajo necesite comunicarse de forma síncrona con sus dependencias, tenga en cuenta lo siguiente:

- La carga de trabajo no debe depender de varias dependencias síncronas para realizar una sola función. Esta cadena de dependencias aumenta la fragilidad general, porque todas las dependencias de la ruta deben estar disponibles para que la solicitud se complete correctamente.
- Cuando una dependencia no esté en buen estado o no esté disponible, determine sus estrategias de gestión de errores y reintentos. Evite utilizar un comportamiento bimodal. El comportamiento bimodal se produce cuando la carga de trabajo presenta un comportamiento diferente en los modos normal y de error. Para obtener más información sobre el comportamiento bimodal, consulte REL11-BP05 Usar la estabilidad estática para evitar el comportamiento bimodal.
- Tenga en cuenta que responder rápido a los errores es mejor que hacer esperar a la carga de trabajo. Por ejemplo, en la Guía para desarrolladores de AWS Lambda se describe cómo gestionar los reintentos y los fallos al invocar funciones de Lambda.
- Establezca tiempos de espera cuando la carga de trabajo llame a su dependencia. Esta técnica evita esperar demasiado o indefinidamente una respuesta. Para obtener información útil sobre este tema, consulte «Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications».
- Minimice la cantidad de llamadas que se realizan desde la carga de trabajo a su dependencia para atender una sola solicitud. Si hay una conversación demasiado intensa entre ellos, aumenta el acoplamiento y la latencia.

Dependencia asíncrona

Para desvincular temporalmente la carga de trabajo de su dependencia, deben comunicarse de forma asíncrona. Con un enfoque asíncrono, la carga de trabajo puede continuar con cualquier otro procesamiento sin tener que esperar a que su dependencia o cadena de dependencias envíe una respuesta.

Cuando la carga de trabajo necesite comunicarse de forma asíncrona con su dependencia, tenga en cuenta lo siguiente:

- Determine si va a utilizar la mensajería o la transmisión de eventos en función de su caso de uso y sus requisitos. La mensajería permite que la carga de trabajo se comunique con su dependencia mediante el envío y la recepción de mensajes a través de un agente de mensajes. La transmisión de eventos permite que la carga de trabajo y su dependencia utilicen un servicio de transmisión para publicar y suscribirse a eventos, que se entregan como flujos continuos de datos que deben procesarse lo antes posible.

- La mensajería y la transmisión de eventos gestionan los mensajes de manera diferente, por lo que debe decidir si compensan en función de:
  - Prioridad de mensajes: los agentes de mensajes pueden procesar los mensajes de alta prioridad antes que los mensajes normales. En la transmisión de eventos, todos los mensajes tienen la misma prioridad.
  - Consumo de mensajes: los agentes de mensajes se aseguran de que los consumidores reciban el mensaje. Los consumidores de la transmisión de eventos deben llevar un registro del último mensaje que han leído.
  - Orden de los mensajes: en el sistema de mensajería, no se garantiza la recepción de los mensajes en el orden exacto en que se envían, a menos que se utilice un enfoque FIFO (primero en entrar, primero en salir). La transmisión de eventos siempre mantiene el orden en que se produjeron los datos.
  - Eliminación de mensajes: en el sistema de mensajería, el consumidor debe eliminar el mensaje después de procesarlo. El servicio de transmisión de eventos añade el mensaje a una transmisión y permanece allí hasta que venza el período de retención. Esta política de eliminación hace que la transmisión de eventos sea adecuada para volver a reproducir mensajes.
- Defina la forma en que la carga de trabajo sabe cuándo su dependencia ha terminado el trabajo. Por ejemplo, cuando la carga de trabajo invoca una función de Lambda de forma asíncrona, Lambda coloca el evento en una cola e indica que se ha realizado correctamente en la respuesta sin información adicional. Una vez finalizado el procesamiento, la función de Lambda puede enviar el resultado a un destino, que se puede configurar en función de si se ha realizado correctamente o no.
- Aumente la carga de trabajo para gestionar los mensajes duplicados utilizando la idempotencia. La idempotencia significa que los resultados de la carga de trabajo no cambian aunque esta se genere más de una vez para el mismo mensaje. Es importante señalar que los servicios de mensajería o transmisión volverán a entregar un mensaje si se produce un fallo de la red o si no se recibe la confirmación de la recepción.
- Si la carga de trabajo no recibe una respuesta de su dependencia, debe volver a enviar la solicitud. Considere la posibilidad de limitar el número de reintentos para conservar los recursos de CPU, memoria y red de la carga de trabajo para gestionar otras solicitudes. En la documentación de AWS Lambda se indica cómo gestionar los errores de la invocación asíncrona.
- Utilice las herramientas de observabilidad, depuración y rastreo adecuadas para administrar y utilizar la comunicación asíncrona de la carga de trabajo con su dependencia. Puede utilizar Amazon CloudWatch para supervisar los servicios de mensajería y transmisión de eventos.

También puede instrumentar la carga de trabajo con AWS X-Ray para obtener rápidamente información para la solución de problemas.

Dependencia por lotes

Los sistemas por lotes toman los datos de entrada, inician una serie de trabajos para procesarlos y producen algunos datos de salida, sin intervención manual. Según el tamaño de los datos, los trabajos pueden durar desde unos minutos hasta, en algunos casos, varios días. Cuando la carga de trabajo se comunique con su dependencia por lotes, tenga en cuenta lo siguiente:

- Defina el intervalo de tiempo en el que la carga de trabajo debe ejecutar el trabajo por lotes. La carga de trabajo puede configurar un patrón de recurrencia para invocar un sistema por lotes como, por ejemplo, cada hora o al final de cada mes.
- Determine la ubicación de la entrada de datos y la salida de los datos procesados. Elija un servicio de almacenamiento, como Amazon Simple Storage Service (Amazon S3), Amazon Elastic File System (Amazon EFS) y Amazon FSx for Lustre, que permita que su carga de trabajo lea y escriba archivos a escala.
- Si su carga de trabajo necesita invocar varios trabajos por lotes, puede utilizar AWS Step Functions para simplificar la orquestación de los trabajos por lotes que se ejecutan en AWS o de forma local. Este proyecto de ejemplo demuestra el uso de la orquestación de trabajos por lotes mediante Step Functions, AWS Batch y Lambda.
- Supervise los trabajos por lotes para detectar anomalías, como que un trabajo tarde más de lo debido en completarse. Puede utilizar herramientas como CloudWatch Container Insights para supervisar los entornos y los trabajos de AWS Batch. En este caso, su carga de trabajo impediría el inicio del siguiente trabajo e informaría al personal correspondiente de la excepción.

## Recursos

Documentos relacionados:

- Nube de AWS Operations: «Monitoreo y observabilidad»
- Amazon Builders' Library: «Desafíos de los sistemas distribuidos»
- REL11-BP05 Usar la estabilidad estática para evitar el comportamiento bimodal
- «Guía para desarrolladores de AWS Lambda: Control de errores y reintentos automáticos en AWS Lambda»

- «Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications»
- «Servicios de mensajería de AWS»
- «¿Qué son los datos de streaming?
- «Guía para desarrolladores de AWS Lambda: Invocación asincrónica»
- «Preguntas frecuentes sobre Amazon Simple Queue Service: Colas FIFO»
- «Amazon Kinesis Data Streams Developer Guide: Handling Duplicate Records»
- «Amazon Simple Queue Service Developer Guide: Available CloudWatch metrics for Amazon SQS»
- «Amazon Kinesis Data Streams Developer Guide: Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch»
- «AWS X-Ray Developer Guide: AWS X-Ray concepts»
- «AWS Samples on GitHub: AWS Step functions Complex Orchestrator App»
- «AWS Batch User Guide: AWS Batch CloudWatch Container Insights»

Vídeos relacionados:

- «AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310)»

Herramientas relacionadas:

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Services (Amazon S3)
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx for Lustre
- AWS Step Functions
- AWS Batch

# REL04-BP02 Implementar dependencias con acoplamiento flexible

Las dependencias, como los sistemas de colas, los sistemas de transmisión, los flujos de trabajo y los equilibradores de carga, tienen un acoplamiento flexible. El acoplamiento flexible ayuda a aislar el comportamiento de un componente de otros componentes que dependen de él, lo que aumenta la resiliencia y la agilidad.

En sistemas de acoplamiento ajustado, los cambios en un componente pueden requerir cambios en otros componentes que dependen de él, lo que reduce el rendimiento de todos los componentes. El acoplamiento flexible elimina esta dependencia, de forma que los componentes dependientes solo necesitan conocer la interfaz publicada y versionada. La implementación de un acoplamiento flexible entre las dependencias aísla un fallo en una de ellas para que no afecte a otra.

El acoplamiento flexible permite modificar el código o añadir características a un componente y, al mismo tiempo, minimizar el riesgo para otros componentes que dependen de él. También permite una resiliencia granular a nivel de componente, lo que permite escalar horizontalmente o incluso cambiar la implementación subyacente de la dependencia.

Para mejorar aún más la resiliencia mediante el acoplamiento flexible, haga que las interacciones entre componentes sean asincrónicas siempre que sea posible. Este modelo es adecuado para cualquier interacción que no necesite una respuesta inmediata y en la que baste con el reconocimiento de que una solicitud se ha registrado. Consta de un componente que genera eventos y de otro que los consume. Ambos componentes no se integran mediante una interacción directa de punto a punto, sino que normalmente emplean una capa de almacenamiento duradera intermedia, como una cola Amazon SQS o una plataforma de restringa de datos como Amazon Kinesis o AWS Step Functions.

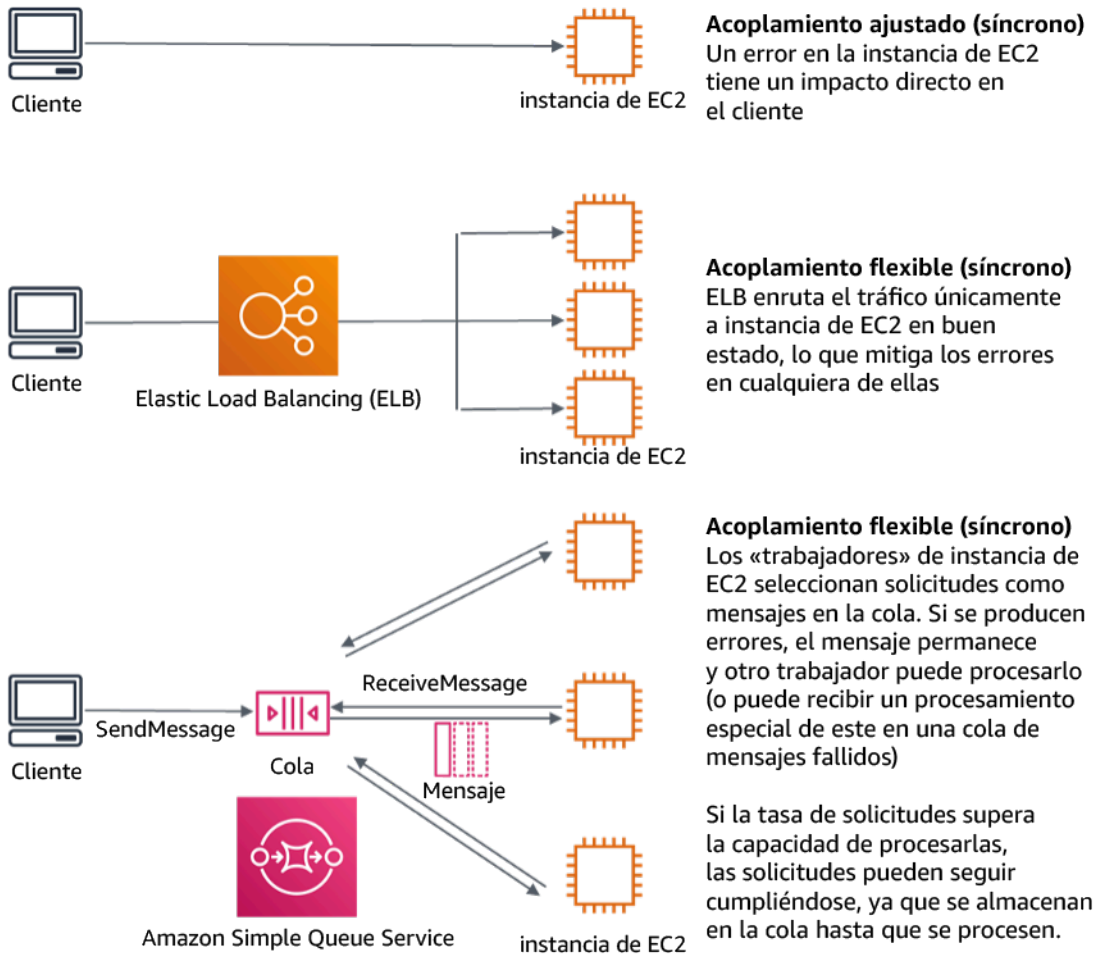



Figura 4: Las dependencias, como sistemas de colas y equilibradores de carga, tienen acoplamiento flexible

Las colas de Amazon SQS y los equilibradores de carga elásticos son solo dos formas de añadir una capa intermedia para el acoplamiento flexible. Las arquitecturas basadas en eventos también pueden integrarse en Nube de AWS utilizando Amazon EventBridge, lo que puede abstraer a los clientes (productores de eventos) de los servicios en los que confían (consumidores de eventos). Amazon Simple Notification Service (Amazon SNS) es una solución eficaz cuando se necesita una mensajería de alto rendimiento, basada en push y de varios a varios. Utilizando temas de Amazon SNS, los sistemas de su editor pueden repartir mensajes por una gran cantidad de puntos de conexión de suscriptores para procesarlos en paralelo.

Aunque las colas ofrecen varias ventajas, en la mayoría de sistemas inflexibles en tiempo real, las solicitudes que superan un umbral temporal (que suele ser de segundos) se consideran obsoletas (el cliente ha desistido y ya no espera una respuesta), por lo que no se procesan. De esta manera, se pueden procesar las solicitudes más recientes (y probablemente aún válidas) en su lugar.

Resultado deseado: la implementación de dependencias de acoplamiento flexible permite minimizar el área de superficie en caso de fallo a nivel de componente, lo que ayuda a diagnosticar y resolver problemas. También simplifica los ciclos de desarrollo, lo que permite a los equipos implementar cambios a nivel modular sin que eso afecte al rendimiento de otros componentes que dependen de él. Este enfoque ofrece la capacidad de escalar horizontalmente a nivel de componente en función de los recursos que sean necesarios, así como de utilizar un componente que contribuye a ahorrar costes.

Antipatrones usuales:

- Desplegar una carga de trabajo monolítica.
- Invocar directamente las API entre capas de la carga de trabajo sin la capacidad de conmutar por error ni procesar asincrónicamente la solicitud
- Utilizar un acoplamiento ajustado con datos compartidos. Los sistemas de acoplamiento flexible no deben compartir datos a través de bases de datos compartidas u otras formas de almacenamiento de datos de acoplamiento ajustado, que pueden reintroducir el acoplamiento ajustado y dificultar la escalabilidad.
- Ignorar la contrapresión. La carga de trabajo debe tener la capacidad de ralentizar o detener los datos entrantes cuando un componente no pueda procesarlos al mismo ritmo.

Ventajas de establecer esta práctica recomendada: el acoplamiento flexible ayuda a aislar el comportamiento de un componente de otros componentes que dependen de él, lo que aumenta la resiliencia y la agilidad. Un error en un componente está aislado de los demás componentes.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

Implemente dependencias con acoplamiento flexible. Existen varias soluciones que permiten crear aplicaciones con un acoplamiento flexible. Entre ellas, se incluyen servicios para implementar colas totalmente administradas, flujos de trabajo automatizados, reacción a eventos y API, entre otras, que pueden ayudar a aislar el comportamiento de los componentes de otros componentes y, por lo tanto, aumentar la resiliencia y la agilidad.

- Cree arquitecturas basadas en eventos: Amazon EventBridge le ayuda a crear arquitecturas basadas en eventos distribuidas y de acoplamiento flexible.
- Implemente colas en sistemas distribuidos: puede usar Amazon Simple Queue Service (Amazon SQS) para integrar y desacoplar sistemas distribuidos.

- Contenedorice los componentes como microservicios: los microservicios permiten a los equipos crear aplicaciones compuestas por pequeños componentes independientes que se comunican a través de API bien definidas. Amazon Elastic Container Service(Amazon ECS) y Amazon Elastic Kubernetes Service (Amazon EKS) pueden ayudarle a empezar a utilizar los contenedores con mayor rapidez.
- Administre los flujos de trabajo con Step Functions: Step Functions le ayudan a coordinar varios servicios de AWS para convertirlos en flujos de trabajo flexibles.
- Utilice las arquitecturas de mensajería de publicación y suscripción (pub/sub): Amazon Simple Notification Service (Amazon SNS) permite entregar mensajes de los editores a los suscriptores (también conocidos como productores y consumidores).

Pasos para la implementación

- Los componentes de una arquitectura basada en eventos se inician mediante eventos. Los eventos son acciones que ocurren en un sistema, como cuando un usuario añade un artículo a una cesta. Cuando una acción se realiza correctamente, se genera un evento que activa el siguiente componente del sistema.
  - «Building Event-driven Applications with Amazon EventBridge»
  - AWS «re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge»
- Los sistemas de mensajería distribuida tienen tres partes principales que deben implementarse para una arquitectura basada en colas. Incluyen los componentes del sistema distribuido, la cola que se usa para el desacoplamiento (distribuida en servidores de Amazon SQS) y los mensajes de la cola. Un sistema típico tiene productores que inician el mensaje en la cola y el consumidor que recibe el mensaje de la cola. La cola almacena los mensajes en varios servidores de Amazon SQS para garantizar la redundancia.
  - «Basic Amazon SQS architecture»
  - «Send Messages Between Distributed Applications with Amazon Simple Queue Service»
- Los microservicios, cuando se utilizan bien, facilitan el mantenimiento y aumentan la escalabilidad, ya que los componentes de acoplamiento flexible los administran equipos independientes. También permiten aislar los comportamientos en un solo componente en caso de que se realicen cambios.
  - «Implementing Microservices on AWS»
  - «Let's Architect! Architecting microservices with containers»

- Con AWS Step Functions, puede crear aplicaciones distribuidas, automatizar procesos y orquestar microservicios, entre otras cosas. La orquestación de varios componentes en un flujo de trabajo automatizado le permite desacoplar las dependencias de su aplicación.
  - «Create a Serverless Workflow with AWS Step Functions and AWS Lambda»
  - «Introducción a AWS Step Functions»

## Recursos

Documentos relacionados:

- «Amazon EC2: Ensuring Idempotency»
- La Amazon Builders' Library: Desafíos de los sistemas distribuidos
- La Amazon Builders' Library: Fiabilidad, trabajo constante y una buena taza de café
- ¿Qué es Amazon EventBridge?
- «¿Qué es Amazon Simple Queue Service?»
- «Break up with your monolith»
- «Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS»
- «Basic Amazon SQS architecture»
- «Queue-Based Architecture»

Vídeos relacionados:

- «AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)»
- «AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337» (incluye acoplamiento flexible, trabajo constante y estabilidad estática)
- «AWS re:Invent 2019: Moving to event-driven architectures (SVS308)»
- «AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304)»
- «AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda«
- «AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge«
- «AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices»

# REL04-BP03 Realizar un trabajo constante

Los sistemas pueden producir error cuando hay cambios rápidos grandes en la carga. Por ejemplo, si la carga de trabajo está realizando una comprobación de estado que supervisa el estado de miles de servidores, debería enviar siempre una carga del mismo tamaño (una instantánea completa del estado actual). Si no hay errores en ningún servidor, o hay errores en todos ellos, el sistema de comprobación de estado estará haciendo un trabajo constante sin rápidos cambios de gran tamaño.

Por ejemplo, si el sistema de comprobación de estado supervisa 100 000 servidores, la carga contenida en él es nominal con un porcentaje de errores del servidor normalmente bajo. Sin embargo, si un evento importante deja a la mitad de esos servidores en mal estado, el sistema de comprobación de estado se sobrecargaría intentando actualizar los sistemas de notificación y comunicar el estado a sus clientes. Por ello, el sistema de comprobación de estado debería enviar cada vez la instantánea completa del estado actual. 100 000 estados de servidor, cada uno representado por un bit, solo constituiría una carga de 12,5 KB. Si no hay errores en ningún servidor, o hay errores en todos ellos, el sistema de comprobación de estado estará haciendo un trabajo constante y los cambios rápidos de gran tamaño no pondrán en peligro la estabilidad del sistema. En realidad, así es como Amazon Route 53 gestiona las comprobaciones de estado de los puntos de conexión (como las direcciones IP) para determinar cómo se enruta a los usuarios finales.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Bajo

## Guía para la implementación

- Realice un trabajo para que los sistemas no tengan errores cuando hay cambios grandes y rápidos en la carga.
- Implemente dependencias con acoplamiento flexible. Las dependencias, como los sistemas de colas, los sistemas de transmisión, los flujos de trabajo y los equilibradores de carga, tienen un acoplamiento flexible. El acoplamiento flexible ayuda a aislar el comportamiento de un componente de otros componentes que dependen de él, lo que aumenta la resiliencia y la agilidad.
  - La Amazon Builders' Library: Fiabilidad, trabajo constante y una buena taza de café
  - AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work) (Cerrar los bucles y abrir las mentes: cómo tomar el control de los sistemas, grandes y pequeños ARC337 [incluye trabajo constante])
    - En el ejemplo de una sistema de comprobación de estado que supervisa 100 000 servidores, diseñe las cargas de trabajo de forma que los tamaños de la carga útil sean iguales independientemente del número de éxitos o fracasos.

### Recursos

Documentos relacionados:

- Amazon EC2: garantizar la idempotencia
- La Amazon Builders' Library: Desafíos de los sistemas distribuidos
- La Amazon Builders' Library: Fiabilidad, trabajo constante y una buena taza de café

Vídeos relacionados:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (Introducción a las arquitecturas basadas en eventos y Amazon EventBridge) (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work) (Cerrar los bucles y abrir las mentes: cómo asumir el control de los sistemas grandes y pequeños ARC337 [incluye trabajo constante])
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability) (Cerrar los bucles y abrir las mentes: cómo asumir el control de los sistemas grandes y pequeños ARC337 [incluye acoplamiento flexible, trabajo constante y estabilidad estática])
- AWS re:Invent 2019: Moving to event-driven architectures (Optar por arquitecturas basadas en eventos) (SVS308)

## REL04-BP04 Hacer que todas las respuestas sean idempotentes

Un servicio idempotente promete que cada solicitud se completará una y solo una vez, de tal forma que realizar varias solicitudes idénticas tiene el mismo efecto que realizar una sola solicitud. Un servicio idempotente permite que un cliente implemente fácilmente los reintentos sin el temor de que una solicitud se procese erróneamente varias veces. Para ello, los clientes pueden usar solicitudes de API con un token de idempotencia: se utiliza el mismo token siempre que se repite la solicitud. Una API de servicio idempotente usa el token para devolver una respuesta idéntica a la que se devolvió por primera vez cuando se completó la solicitud.

En un sistema distribuido, es fácil llevar a cabo una acción una vez como máximo (el cliente realiza solo una solicitud) o al menos una vez (sigue realizando la solicitud hasta que el cliente obtiene una confirmación del éxito). Sin embargo, es difícil garantizar que una acción es idempotente, lo que significa que se lleva a cabo exactamente una vez, de modo que realizar varias solicitudes idénticas

tiene el mismo efecto que realizar una sola solicitud. Al usar tokens de idempotencia en las API, los servicios pueden recibir una solicitud de migración una o más veces sin crear registros duplicados ni efectos secundarios.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

## Guía para la implementación

- Haga que todas las respuestas sean idempotentes. Un servicio idempotente promete que cada solicitud se completará una y solo una vez, de tal forma que realizar varias solicitudes idénticas tiene el mismo efecto que realizar una sola solicitud.
  - Los clientes pueden usar solicitudes de API con un token de idempotencia: se utiliza el mismo token siempre que se repite la solicitud. Una API de servicio idempotente usa el token para devolver una respuesta idéntica a la que se devolvió por primera vez cuando se completó la solicitud.
    - Amazon EC2: garantizar la idempotencia

## Recursos

Documentos relacionados:

- Amazon EC2: garantizar la idempotencia
- La Amazon Builders' Library: Desafíos de los sistemas distribuidos
- La Amazon Builders' Library: Fiabilidad, trabajo constante y una buena taza de café

Videos relacionados:

- Cumbre de AWS en Nueva York 2019: Introducción a las arquitecturas basadas en eventos y Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Bucles cerrados y mentes abiertas: cómo asumir el control de los sistemas grandes y pequeños ARC337 (incluye acoplamiento flexible, trabajo constante y estabilidad estática)
- AWS re:Invent 2019: Optar por arquitecturas basadas en eventos (SVS308)

# Diseñar las interacciones en un sistema distribuido para mitigar o tolerar los errores

Los sistemas distribuidos dependen de las redes de comunicaciones para interconectar componentes, como servidores o servicios. Su carga de trabajo debe funcionar de manera fiable aunque se pierdan datos o haya latencia en estas redes. Los componentes del sistema distribuido deben funcionar de forma que no repercutan negativamente en otros componentes ni en la carga de trabajo. Estas prácticas recomendadas permiten que las cargas de trabajo toleren el estrés o los errores, se recuperen más rápidamente de ellos y mitiguen el impacto de dichos errores. El resultado es un tiempo medio de recuperación (MTTR) mejor.

Estas prácticas recomendadas evitan que se produzcan errores y mejoran el tiempo medio entre errores (MTBF).

Prácticas recomendadas

- REL05-BP01 Implementar una degradación estable para transformar las dependencias estrictas en flexibles
- REL05-BP02 Limitar las solicitudes
- REL05-BP03 Controlar y limitar las llamadas de reintento
- REL05-BP04 Responder rápido a los errores y limitar las colas
- REL05-BP05 Definir tiempos de espera del cliente
- REL05-BP06 Crear sistemas sin estado cuando sea posible
- REL05-BP07 Implementar recursos de emergencia

## REL05-BP01 Implementar una degradación estable para transformar las dependencias estrictas en flexibles

Los componentes de la aplicación deben seguir desempeñando su función principal incluso si las dependencias dejan de estar disponibles. Es posible que proporcionen datos ligeramente obsoletos, datos alternativos o incluso ningún dato. Esto garantiza que los errores localizados solo impidan lo mínimo del funcionamiento general del sistema y, al mismo tiempo, se obtenga el valor empresarial central.

Resultado deseado: Cuando las dependencias de un componente no están en buen estado, el propio componente puede seguir funcionando, aunque con la capacidad mermada. Los modos de errores

de los componentes deben considerarse parte del funcionamiento normal. Los flujos de trabajo deben diseñarse de tal manera que dichos errores no produzcan un fallo total o, al menos, lleven a estados predecibles y recuperables.

Patrones comunes de uso no recomendados:

- No identificar la funcionalidad empresarial principal necesaria. No probar que los componentes funcionen, incluso durante los errores de dependencia.
- No proporcionar datos en caso de error o cuando solo una de las múltiples dependencias no está disponible y aún se pueden devolver resultados parciales.
- Crear un estado incoherente cuando una transacción falla parcialmente.
- No tener una forma alternativa de acceder a un almacén de parámetros central.
- Invalidar o vaciar un estado local como resultado de un fallo de actualización sin tener en cuenta las consecuencias.

Beneficios de establecer esta práctica recomendada: la degradación gradual mejora la disponibilidad del sistema en su conjunto y mantiene la funcionalidad de las funciones más importantes incluso cuando hay errores.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

La implementación de una degradación gradual ayuda a minimizar el impacto de los errores de dependencia en la función de los componentes. Lo ideal sería que un componente detectara los errores de dependencia y siguiese funcionando de una forma que afectara lo mínimo a otros componentes o clientes.

Diseñar una arquitectura que permita una degradación gradual implica considerar los posibles modos de errores durante el diseño de las dependencias. Para cada modo de error, disponga de una forma de ofrecer la mayoría o, al menos la funcionalidad más crítica del componente, a las personas que llaman o a los clientes. Estos factores pueden convertirse en requisitos adicionales que se pueden probar y verificar. Lo ideal es que un componente pueda realizar su función principal de manera aceptable incluso cuando falla una o varias dependencias.

Se trata tanto de un tema empresarial como técnico. Todos los requisitos empresariales son importantes y deben cumplirse si es posible. Sin embargo, es lógico preguntarse qué debe suceder

cuando no se puedan cumplir todos. Se puede diseñar un sistema para que esté disponible y sea coherente, pero en circunstancias en las que haya que eliminar un requisito, ¿cuál es más importante? En el caso del procesamiento de pagos, puede ser la coherencia. En una aplicación en tiempo real, puede ser la disponibilidad. En el caso de un sitio web orientado al cliente, la respuesta dependería de las expectativas del cliente.

Lo que esto significa depende de los requisitos del componente y de lo que deba considerarse su función principal. Por ejemplo:

- Un sitio web de comercio electrónico podría mostrar en su página de inicio los datos de varios sistemas diferentes, como las recomendaciones personalizadas, los productos mejor clasificados y el estado de los pedidos de los clientes. Cuando un sistema anterior falla, sigue siendo lógico mostrar todo lo demás en lugar de mostrar una página de error al cliente.
- Un componente que realiza escrituras por lotes puede seguir procesando un lote si se produce un error en una de las operaciones individuales. Implementar un mecanismo de reintento debería ser sencillo. Se puede hacer devolviendo a la persona que llama información sobre qué operaciones se han realizado correctamente, cuáles han fallado y por qué han fallado, o colocando las solicitudes que han fallado en una cola de mensajes fallidos para implementar reintentos asíncronos. También se debe registrar la información sobre las operaciones que han fallado.
- Un sistema que procese las transacciones debe verificar que se ejecuten todas o ninguna de las actualizaciones individuales. En el caso de las transacciones distribuidas, se puede usar el patrón Saga para revertir operaciones anteriores en caso de que falle una operación posterior de la misma transacción. En este caso, la función principal es mantener la coherencia.
- Los sistemas en los que el tiempo es crítico deberían poder gestionar de la manera oportuna las dependencias que no respondan. En estos casos, se puede utilizar el patrón del disyuntor. Cuando se agota el tiempo de espera de las respuestas de una dependencia, el sistema puede cambiar a un estado cerrado en el que no se realizan llamadas adicionales.
- Una aplicación puede leer parámetros de un almacén de parámetros. Puede resultar útil crear imágenes de contenedores con un conjunto predeterminado de parámetros y utilizarlos en caso de que ese almacén de parámetros no esté disponible.

Tenga en cuenta que las soluciones que se adopten en caso de fallo de un componente deben probarse y deben ser significativamente más sencillas que la solución principal. En general, deben evitarse estrategias alternativas.

## Pasos para la implementación

Identifique las dependencias externas e internas. Considere qué tipos de errores pueden producirse en ellas. Piense en formas de minimizar el impacto negativo en los sistemas anteriores y posteriores y en los clientes durante esos errores.

A continuación, tenemos una lista de dependencias y cómo degradar correctamente cuando fallan:

1. Fallo parcial de las dependencias: un componente puede realizar varias solicitudes a los sistemas posteriores, ya sean varias solicitudes a un sistema o una sola solicitud destinada a varios sistemas. Dependiendo del contexto empresarial, es posible que haya diferentes formas apropiadas de gestionar este problema (para obtener más información, consulte los ejemplos anteriores en la Guía de implementación).
2. Un sistema posterior no puede procesar las solicitudes debido a la alta carga: si las solicitudes a un sistema posterior fallan constantemente, no tiene sentido seguir intentándolo. Esto puede suponer una carga adicional para un sistema ya sobrecargado y dificultar la recuperación. Aquí se puede utilizar el patrón de disyuntor, que monitoriza las llamadas que han fallado al enviarlas a un sistema posterior. Si falla un gran número de llamadas, dejará de enviar más solicitudes al sistema posterior y solo permitirá ocasionalmente el paso de las llamadas para comprobar si el sistema posterior vuelve a estar disponible.
3. El almacén de parámetros no está disponible: para transformar un almacén de parámetros, se puede utilizar el almacenamiento en caché de dependencia flexible o los valores predeterminados en buen estado que se incluyen en las imágenes de contenedores o máquinas. Tenga en cuenta que estos valores predeterminados deben mantenerse actualizados e incluirse en los conjuntos de pruebas.
4. No hay disponible un servicio de monitorización u otra dependencia no funcional: si un componente no puede enviar registros, métricas o rastros de forma intermitente a un servicio de monitorización central, suele ser mejor seguir ejecutando las funciones empresariales como de costumbre. No registrar ni subir métricas de forma silenciosa durante mucho tiempo no suele ser aceptable. Además, algunos casos de uso pueden requerir entradas de auditoría completas para satisfacer los requisitos de cumplimiento.
5. Es posible que una instancia principal de una base de datos relacional no esté disponible: Amazon Relational Database Service, como casi todas las bases de datos relacionales, solo puede tener una instancia de escritor principal. Esto crea un único punto de error para las cargas de trabajo de escritura y dificulta el escalamiento. Este problema se puede mitigar parcialmente mediante el uso de una configuración multi-AZ para alta disponibilidad o Amazon Aurora sin servidor para mejorar el escalamiento. Cuando los requisitos de disponibilidad son muy altos, podría ser conveniente no

utiliza en absoluto el escritor principal. Para consultas que solo leen, se pueden utilizar réplicas de lectura, que proporcionan redundancia y capacidad de escalamiento horizontal, no solo vertical. Las escrituras se pueden almacenar en búfer, por ejemplo, en una cola de Amazon Simple Queue Service, de modo que las solicitudes de escritura de los clientes puedan seguir aceptándose incluso si la principal no está disponible temporalmente.

### Recursos

Documentos relacionados:

- Amazon API Gateway: Limitar las solicitudes de la API para mejorar el rendimiento
- CircuitBreaker (resumen del patrón de interruptor del libro «Release It!»)
- Error Retries and Exponential Backoff in AWS (Reintentos en caso de error y retroceso exponencial en AWS)
- Michael Nygard «Release It! Design and Deploy Production-Ready Software»
- La Amazon Builders' Library: Evitar el retroceso en sistemas distribuidos
- La Amazon Builders' Library: Evitar trabajos pendientes en colas insalvables
- La Amazon Builders' Library: Desafíos y estrategias del almacenamiento en caché
- La Amazon Builders' Library: Tiempos de espera, reintentos y retroceso con alteración

Vídeos relacionados:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders’ Library (DOP328) (Reintento, retroceso y fluctuación: AWS re:Invent 2019: Presentación de Amazon Builders’ Library [DOP328])

Ejemplos relacionados:

- Laboratorio de Well-Architected: Nivel 300: Implementación de comprobaciones de estado y administración de dependencias para mejorar la fiabilidad

## REL05-BP02 Limitar las solicitudes

Limite las solicitudes para mitigar el agotamiento de los recursos debido a aumentos inesperados de la demanda. Las solicitudes por debajo de los índices de limitación se procesan, pero las que

superan el límite definido se rechazan y se envía un mensaje que indica que la solicitud no se ha procesado a causa de la limitación.

Resultado deseado: la limitación de las solicitudes mitiga los grandes picos de volumen, ya sea debido a un aumento repentino del tráfico de clientes, a ataques por desbordamiento o a tormentas de reintentos, lo que permite que las cargas de trabajo sigan procesando de manera normal el volumen de solicitudes admitido.

Patrones comunes de uso no recomendados:

- Las limitaciones de puntos de conexión de la API no se implementan o se mantienen en los valores predeterminados sin tener en cuenta los volúmenes esperados.
- Los puntos de conexión de la API no se someten a pruebas de carga ni se prueban las limitaciones.
- Los índices de solicitudes se reducen sin tener en cuenta el tamaño o la complejidad de las solicitudes.
- Los índices o el tamaño máximos de las solicitudes se prueban, pero por separado.
- Los recursos no se aprovisionan con los mismos límites establecidos en las pruebas.
- No se han configurado ni considerado planes de uso para los consumidores de API de aplicación a aplicación (A2A).
- Los consumidores de cola que escalan horizontalmente no tienen configurado un valor máximo de simultaneidad.
- No se ha implementado la limitación de índices por dirección IP.

Beneficios de establecer esta práctica recomendada: las cargas de trabajo que establecen límites pueden funcionar con normalidad y procesar correctamente la carga de solicitudes aceptada en caso de que se produzcan picos de volumen inesperados. Los picos repentinos o sostenidos de solicitudes a las API y las colas se limitan y no agotan los recursos de procesamiento de solicitudes. Hay límites de índices que limitan a solicitantes individuales para que un gran volumen de tráfico desde una sola dirección IP o un único consumidor de API no agote los recursos y afecte a otros consumidores.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Los servicios deben diseñarse para procesar una capacidad de solicitudes conocida; esta capacidad se puede establecer mediante pruebas de carga. Si los índices de llegada de solicitudes superan los límites, se emite la respuesta correspondiente que indica que la solicitud no se ha procesado a causa de las limitaciones. Esto permite al consumidor gestionar el error y volver a intentarlo más tarde.

Cuando su servicio requiera la implementación de limitaciones, considere la posibilidad de implementar el algoritmo del bucket de tokens, en el que un token se refiere a una solicitud. Los tokens se recargan a un índice de limitación por segundo y se vacían de forma asíncrona a un ritmo de un token por solicitud.


El algoritmo del bucket de tokens.

Amazon API Gateway implementa el algoritmo del bucket de tokens de acuerdo con los límites de la cuenta y la región, y se puede configurar por cliente con planes de uso. Además, Amazon Simple Queue Service (Amazon SQS) y Amazon Kinesis pueden almacenar en búfer las solicitudes para aliviar el índice de solicitudes y permitir índices de limitaciones más altos para las solicitudes que se pueden atender. Por último, puede implementar la limitación de índices con AWS WAF para limitar a consumidores de API específicos que generan una carga inusualmente alta.

## Pasos para la implementación

Puede configurar API Gateway con limitaciones para sus API y devolver errores `429 Demasiadas solicitudes` cuando se superan los límites. Puede utilizar AWS WAF con sus puntos de conexión AWS AppSync y API Gateway para habilitar la limitación de índices por dirección IP. Además, si su

sistema tolera el procesamiento asíncrono, puede colocar los mensajes en una cola o secuencia para acelerar las respuestas a los clientes del servicio, lo que le permite ampliar los índices de limitación más altos.

Con el procesamiento asíncrono, cuando se haya configurado Amazon SQS como origen de eventos para AWS Lambda, puede configurar la simultaneidad máxima para evitar que los altos índices de eventos consuman la cuota de ejecución simultánea de la cuenta disponible que necesitan otros servicios de su carga de trabajo o cuenta.

Si bien API Gateway proporciona una implementación administrada del bucket de tokens, en los casos en que no pueda usar API Gateway, puede utilizar las implementaciones de código abierto específicas de cada lenguaje (consulte los ejemplos relacionados en Recursos) del bucket de tokens para sus servicios.

- Comprenda y configure limitaciones de API Gateway a nivel de cuenta por región, API por etapa y clave de API por niveles de planes de uso.
- Utilice reglas de limitación de índices de AWS WAF para puntos de conexión de API Gateway y AWS AppSync para protegerse contra ataques de desbordamiento y bloquear las IP malintencionadas. Las reglas de limitación de índices también se pueden configurar en las claves de API de AWS AppSync para los consumidores de A2A.
- Analice si necesita un control de limitación superior a la limitación de índices para las API de AWS AppSync y, de ser así, configure una API Gateway enfrente de su punto de conexión de AWS AppSync.
- Si las colas de Amazon SQS están configuradas como disparadores para los consumidores de colas de Lambda, defina la simultaneidad máxima en un valor que procese lo suficiente para cumplir los objetivos de nivel de servicio, pero que no consuma límites de simultaneidad que afecten a otras funciones Lambda. Considere la posibilidad de configurar la simultaneidad reservada en otras funciones Lambda de la misma cuenta y región cuando consuma colas con Lambda.
- Utilice API Gateway con integraciones de servicios nativos para Amazon SQS o Kinesis para almacenar en búfer las solicitudes.
- Si no puede utilizar API Gateway, consulte las bibliotecas específicas del lenguaje para implementar el algoritmo del bucket de tokens para su carga de trabajo. Consulte la sección de ejemplos e investigue por su cuenta para encontrar una biblioteca adecuada.
- Pruebe los límites que tiene pensado establecer o que va a permitir que se aumenten, y documente los límites probados.

- No aumente los límites por encima de lo que establezca en las pruebas. Cuando aumente un límite, antes de aplicar ese aumento, compruebe que los recursos aprovisionados son equivalentes o superiores a los de los escenarios de prueba.

## Recursos

Prácticas recomendadas relacionadas:

- REL04-BP03 Realizar un trabajo constante
- REL05-BP03 Controlar y limitar las llamadas de reintento

Documentos relacionados:

- Amazon API Gateway: Limitar las solicitudes de la API para mejorar el rendimiento
- AWS WAF: Rate-based rule statement (Declaración de la regla basada en índices)
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source (Introducción a la simultaneidad máxima de funciones AWS Lambda cuando Amazon SQS se utiliza como origen de los eventos)
- AWS Lambda: Simultaneidad máxima

Ejemplos relacionados:

- The three most important AWS WAF rate-based rules (Las tres reglas más importantes basadas en índices de AWS WAF)
- Java Bucket4j
- Python token-bucket (Bucket de tokens de Python)
- Node token-bucket (Bucket de tokens de nodos)
- .NET System Threading Rate Limiting (Limitación de índices de subprocesos del sistema .NET)

Vídeos relacionados:

- Implementing GraphQL API security best practices with AWS AppSync (Implementación de prácticas recomendadas de seguridad de la API GraphQL con AWS AppSync)

Herramientas relacionadas:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF

## REL05-BP03 Controlar y limitar las llamadas de reintento

Utilice un retroceso exponencial para reintentar las solicitudes a intervalos progresivamente más largos entre cada reintento. Introduzca una fluctuación entre reintentos para aleatorizar los intervalos de reintentos. Limite el número máximo de reintentos.

Resultado deseado: entre los componentes típicos de un sistema de software distribuido se incluyen servidores, equilibradores de carga, bases de datos y servidores DNS. Durante el funcionamiento normal, estos componentes pueden responder a las solicitudes con errores temporales o limitados, y también con errores que serían persistentes independientemente de los reintentos. Cuando los clientes realizan solicitudes a los servicios, esas solicitudes consumen recursos, como memoria, subprocesos, conexiones, puertos o cualquier otro recurso limitado. Controlar y limitar los reintentos es una estrategia para liberar y minimizar el consumo de recursos, de modo que los componentes del sistema sometidos a presión no se sobrecarguen.

Cuando se agota el tiempo de espera de las solicitudes del cliente o se reciben respuestas de error, deben determinar si deben volver a intentarlo o no. Si lo vuelven a intentar, lo hacen con un retroceso exponencial con fluctuaciones y un valor de reintento máximo. Como resultado, los servicios y procesos de backend tienen menos carga y más tiempo para recuperarse automáticamente, lo que se traduce en una recuperación más rápida y una tramitación satisfactoria de las solicitudes.

Patrones comunes de uso no recomendados:

- Implementar los reintentos sin añadir valores de retroceso exponencial, fluctuación y reintentos máximos. El retroceso y la fluctuación ayudan a evitar picos de tráfico artificiales debidos a reintentos coordinados involuntariamente a intervalos comunes.
- Implementar reintentos sin probar sus efectos o asumir que los reintentos ya están integrados en un SDK sin probar los escenarios de reintento.
- No entender los códigos de error publicados de las dependencias, lo que lleva a volver a intentar todos los errores, incluidos los que tienen una causa clara que indica una falta de permisos,

un error de configuración u otro problema que es de esperar que no se pueda resolver sin una intervención manual.

- No utilizar prácticas de observabilidad, como monitorización y alertas en caso de errores de servicio repetidos, para conocer problemas subyacentes y poder solucionarlos.
- Desarrollar mecanismos de reintento personalizados cuando son suficientes las capacidades de reintento integradas o de terceros.
- Realizar reintentos en varias capas de la pila de aplicaciones de una forma que se acumulen, lo que consume aún más recursos en una tormenta de reintentos. Asegúrese de entender cómo afectan estos errores a las dependencias en las que se basa y, a continuación, implemente los reintentos en un solo nivel.
- Reintentar llamadas de servicio que no son idempotentes, lo que provoca efectos secundarios inesperados, como resultados duplicados.

Beneficios de establecer esta práctica recomendada: los reintentos ayudan a los clientes a obtener los resultados deseados cuando las solicitudes fallan, pero también consumen más tiempo del servidor para obtener las respuestas satisfactorias que desean. Cuando los errores son poco frecuentes o transitorios, los reintentos funcionan bien. Cuando los errores se deben a una sobrecarga de recursos, los reintentos pueden empeorar las cosas. Añadir un retroceso exponencial con fluctuaciones para los reintentos de los clientes permite que los servidores se recuperen cuando los errores se deben a una sobrecarga de recursos. La fluctuación evita que haya picos de solicitudes y el retroceso disminuye el escalamiento de la carga provocado por la adición de reintentos a la carga normal de solicitudes. Por último, es importante configurar un número de reintentos máximo o un tiempo transcurrido máximo para evitar que se acumulen tareas pendientes que generen errores metaestables.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Controle y limite las llamadas de reintento. Use el retroceso exponencial para los reintentos tras intervalos cada vez más largos. Introduzca una fluctuación para aleatorizar los intervalos de reintento y limite el número máximo de reintentos.

Algunos SDK de AWS implementan los reintentos y el retroceso exponencial de forma predeterminada. Utilice estas implementaciones de AWS integradas cuando corresponda en su carga de trabajo. Implemente una lógica similar en su carga de trabajo cuando llame a servicios

que sean idempotentes y en los que los reintentos mejoren la disponibilidad de sus clientes. Decida cuáles son los tiempos de espera y cuándo dejar de reintentar según su caso de uso. Cree y realice escenarios de prueba para esos casos de uso de reintentos.

## Pasos para la implementación

- Determine la capa óptima de la pila de aplicaciones para implementar los reintentos de los servicios de los que depende su aplicación.
- Tenga en cuenta que los SDK existentes implementan estrategias de reintento probadas con retroceso exponencial y fluctuaciones para el lenguaje que elija, y dé preferencia a estas estrategias en lugar de escribir sus propias implementaciones de reintentos.
- Verifique que los servicios sean idempotentes antes de implementar los reintentos. Una vez implementados, asegúrese de que se prueben y se utilicen regularmente en producción.
- Al llamar a las API del servicio de AWS, utilice los SDK de AWS y AWS CLI y comprenda las opciones de configuración de reintentos. Determine si los valores predeterminados funcionan para su caso de uso, pruébelos y ajústelos según sea necesario.

## Recursos

Prácticas recomendadas relacionadas:

- REL04-BP04 Hacer que todas las respuestas sean idempotentes
- REL05-BP02 Limitar las solicitudes
- REL05-BP04 Responder rápido a los errores y limitar las colas
- REL05-BP05 Definir tiempos de espera del cliente
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores

Documentos relacionados:

- Error Retries and Exponential Backoff in AWS (Reintentos en caso de error y retroceso exponencial en AWS)
- La Amazon Builders' Library: Tiempos de espera, reintentos y retroceso con alteración
- Exponential backoff and jitter (Retroceso exponencial y fluctuación)
- Making retries safe with idempotent APIs (Hacer que los reintentos sean seguros con API idempotentes)

Ejemplos relacionados:

- Spring Retry (Reintento de Spring)
- Resilience4j Retry (Reintento de Resilience4j)

Vídeos relacionados:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328) (Reintento, retroceso y fluctuación: AWS re:Invent 2019: Presentación de Amazon Builders' Library [DOP328])

Herramientas relacionadas:

- AWS SDKs and Tools: Retry behavior (SDK y herramientas de AWS: comportamiento de reintento)
- AWS Command Line Interface: Reintentos de AWS CLI

## REL05-BP04 Responder rápido a los errores y limitar las colas

Cuando un servicio no pueda responder correctamente a una solicitud, responda rápido a los errores. Esto permite que se liberen los recursos asociados a una solicitud y que un servicio se recupere cuando se le agotan los recursos. La respuesta rápida a los errores es un patrón de diseño de software bien establecido que se puede utilizar para conseguir cargas de trabajo enormemente fiables en la nube. Las colas también son un patrón de integración empresarial bien establecido que puede suavizar la carga y permitir a los clientes liberar recursos cuando se pueda tolerar el procesamiento asíncrono. Cuando un servicio puede responder correctamente en condiciones normales, pero falla cuando el índice de solicitudes es demasiado alto, utilice una cola para almacenar en búfer las solicitudes. Sin embargo, no permita que se acumulen largas colas de tareas pendientes, ya que eso podría hacer que se procesaran solicitudes obsoletas a las que un cliente ya ha renunciado.

Resultado deseado: cuando los sistemas sufren contención de recursos, tiempos de espera, excepciones o errores grises que hacen que los objetivos de nivel de servicio sean inalcanzables, las estrategias de respuesta rápida a los errores permiten recuperar el sistema más rápido. Los sistemas que deben absorber los picos de tráfico y pueden adaptarse al procesamiento asíncrono pueden mejorar la fiabilidad al permitir a los clientes liberar rápidamente las solicitudes mediante el uso de colas para almacenar en búfer las solicitudes a los servicios de backend. Cuando las solicitudes a

las colas se almacenan en búfer, se implementan estrategias de administración de colas para evitar retrasos insuperables.

Patrones comunes de uso no recomendados:

- Implementar colas de mensajes, pero no configurar colas de mensajes fallidos (DLQ) ni alarmas en los volúmenes de DLQ para detectar cuándo está fallando un sistema.
- No medir la antigüedad de los mensajes de una cola, que es una medida de la latencia para saber cuándo los usuarios de la cola sufren retrasos o producen errores que dan lugar a reintentos.
- No borrar los mensajes pendientes de una cola cuando no sirve de nada procesar esos mensajes si la empresa ya no necesita hacerlo.
- Configurar colas de primero en entrar/primero en salir (FIFO) cuando las colas de último en entrar, primero en salir (LIFO) responderían mejor a las necesidades de los clientes, por ejemplo, cuando no se requieren pedidos estrictos y el procesamiento pendiente retrasa todas las solicitudes nuevas y urgentes, lo que hace que se infrinjan los niveles de servicio de todos los clientes.
- Exponer las colas internas a los clientes en lugar de exponer las API que administran la entrada de trabajo y colocan las solicitudes en colas internas.
- Combinar demasiados tipos de solicitudes de trabajo en una sola cola puede agravar las condiciones de las tareas pendientes al distribuir la demanda de recursos entre los tipos de solicitudes.
- Procesar solicitudes complejas y simples en la misma cola, a pesar de necesitar diferentes niveles de monitorización, tiempos de espera y asignaciones de recursos.
- No validar las entradas ni utilizar afirmaciones para implementar mecanismos de respuesta rápida a los errores en el software que envíen las excepciones a componentes de nivel superior que puedan gestionar los errores con facilidad.
- No eliminar los recursos que fallan del enrutamiento de solicitudes, especialmente cuando los errores grises emiten tanto éxitos como errores debido a bloqueos y reinicios, errores de dependencia intermitentes, una reducción de la capacidad o la pérdida de paquetes de red.

Beneficios de establecer esta práctica recomendada: los sistemas que responden rápido a los errores son más fáciles de depurar y corregir y, a menudo, revelan problemas de codificación y configuración antes de que las versiones se publiquen en producción. Los sistemas que incorporan estrategias de puesta en cola eficaces tienen una mayor resiliencia y fiabilidad a los picos de tráfico y a las condiciones de errores intermitentes del sistema.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Las estrategias de respuesta rápida a los errores pueden codificarse en soluciones de software y también configurarse en la infraestructura. Además de la respuesta rápida a los errores, las colas son una técnica arquitectónica sencilla pero potente para desacoplar los componentes del sistema sin problemas de carga. Amazon CloudWatch proporciona capacidades para monitorizar los errores y alertar en caso de que existan. Una vez que se sabe que un sistema está fallando, se pueden invocar estrategias de mitigación, como el alejamiento de los recursos deteriorados. Cuando los sistemas implementan colas con Amazon SQS y otras tecnologías de cola para facilitar la carga, deben considerar cómo administrar los atrasos en las colas, así como los errores en el consumo de mensajes.

## Pasos para la implementación

- Implemente afirmaciones programáticas o métricas específicas en su software y utilícelas para alertar explícitamente sobre problemas del sistema. Amazon CloudWatch le ayuda a crear métricas y alarmas basadas en el patrón de registro de la aplicación y la instrumentación del SDK.
- Utilice métricas y alarmas de CloudWatch para alejarse de los recursos deteriorados que aumentan la latencia del procesamiento o que no procesan las solicitudes de forma reiterada.
- Utilice el procesamiento asíncrono diseñando API que acepten solicitudes y las anexen a las colas internas mediante Amazon SQS y luego respondan al cliente que produce los mensajes con un mensaje de éxito, de modo que el cliente pueda liberar recursos y continuar con otras tareas mientras los consumidores de la cola del backend procesan las solicitudes.
- Mida y monitorice la latencia de procesamiento de las colas generando una métrica de CloudWatch cada vez que se retire un mensaje de una cola comparándolo en ese momento con la marca de tiempo del mensaje.
- Cuando los errores impidan procesar correctamente los mensajes o los picos de tráfico en los volúmenes que no se pueden procesar dentro de los acuerdos de nivel de servicio, aparte el tráfico antiguo o excesivo y colóquelo en una cola secundaria. Esto permite procesar de forma prioritaria los trabajos nuevos y dejar los antiguos para cuando haya capacidad disponible. Esta técnica es una aproximación al procesamiento LIFO y permite que el sistema procese normalmente todos los trabajos nuevos.
- Utilice colas de mensajes fallidos o redireccione las colas para sacar de la lista de espera los mensajes que no se puedan procesar y colocarlos en una ubicación que pueda investigarse y resolverse más adelante

- Vuelva a intentarlo o, cuando sea tolerable, elimine los mensajes antiguos comparándolos en ese momento con la marca de tiempo del mensaje y descartando los mensajes que ya no sean relevantes para el cliente que los ha solicitado.

## Recursos

Prácticas recomendadas relacionadas:

- REL04-BP02 Implementar dependencias con acoplamiento flexible
- REL05-BP02 Limitar las solicitudes
- REL05-BP03 Controlar y limitar las llamadas de reintento
- REL06-BP02 Definir y calcular métricas (agregación)
- REL06-BP07 Supervisar el seguimiento de las solicitudes de principio a fin en todo el sistema

Documentos relacionados:

- Cómo evitar demoras de colas insuperables
- Respuesta rápida a errores
- How can I prevent an increasing backlog of messages in my Amazon SQS queue? (¿Cómo puedo evitar que se acumulen mensajes en mi cola de Amazon SQS?
- Elastic Load Balancing: Zonal Shift (Cambio de zona)
- Amazon Route 53 Application Recovery Controller: Routing control for traffic failover (Amazon Route 53 Application Recovery Controller: control de enrutamiento para conmutación por error del tráfico)

Ejemplos relacionados:

- Enterprise Integration Patterns: Dead Letter Channel (Patrones de integración empresarial: canal de mensajes fallidos)

Vídeos relacionados:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications (AWS re:Invent 2022: Funcionamiento de aplicaciones multi-AZ de alta disponibilidad)

Herramientas relacionadas:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

## REL05-BP05 Definir tiempos de espera del cliente

Defina tiempos de espera adecuados para las conexiones y las solicitudes, verifíquelos sistemáticamente y no use los valores predeterminados, ya que no tienen en cuenta las características específicas de la carga de trabajo.

Resultado deseado: en los tiempos de espera de los clientes, se debe tener en cuenta el coste para el cliente, el servidor y la carga de trabajo asociados a la espera de las solicitudes que tardan un tiempo anormal en completarse. Dado que no es posible conocer la causa exacta de ningún tiempo de espera, los clientes deben utilizar el conocimiento de los servicios para fijar expectativas sobre las causas probables y los tiempos de espera adecuados.

El tiempo de espera de las conexiones del cliente se agota en función de los valores configurados. Cuando el tiempo de espera se agota, los clientes toman la decisión de dar marcha atrás y volver a intentarlo o abrir un disyuntor. Estos patrones evitan que se emitan solicitudes que puedan agravar una condición de error subyacente.

Patrones comunes de uso no recomendados:

- No estar al tanto de los tiempos de espera del sistema o de los tiempos de espera predeterminados.
- No estar al tanto del tiempo normal de finalización de las solicitudes.
- No conocer las posibles causas por las que las solicitudes tardan un tiempo anormalmente largo en completarse ni los costes para el rendimiento del cliente, el servicio o la carga de trabajo asociados a la espera a que se completen.
- No conocer la probabilidad de que la red deteriorada haga que una solicitud falle solo una vez que se haya agotado el tiempo de espera, ni de los costes que supone para el rendimiento del cliente y la carga de trabajo no utilizar un tiempo de espera más corto.
- No probar escenarios de tiempo de espera tanto para las conexiones como para las solicitudes.

- Definir tiempos de espera demasiado altos, lo que puede provocar tiempos de espera prolongados y aumentar la utilización de los recursos.
- Definir tiempos de espera demasiado bajos, lo que provoca errores artificiales.
- Pasar por alto los patrones para solucionar los errores de tiempo de espera de las llamadas remotas, como disyuntores y reintentos.
- No considerar la posibilidad de monitorizar los índices de errores de las llamadas de servicio, los objetivos de nivel de servicio referentes a la latencia y los valores atípicos de latencia. Estas métricas pueden proporcionar información sobre tiempos de espera agresivos o permisivos.

Beneficios de establecer esta práctica recomendada: los tiempos de espera de las llamadas remotas están configurados y los sistemas están diseñados para gestionar los tiempos de espera correctamente, de modo que los recursos se conserven cuando las llamadas remotas responden con una lentitud anormal y los clientes del servicio gestionan correctamente los errores de tiempo de espera.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Defina un tiempo de espera de conexión y un tiempo de espera de solicitud en cualquier llamada de dependencia del servicio y, normalmente, en todas las llamadas de los procesos. Muchos marcos integran capacidades de tiempo de espera, pero tenga cuidado, ya que algunos tienen valores predeterminados que son infinitos o superiores a lo aceptable para sus objetivos de servicio. Un valor demasiado alto reduce la utilidad del tiempo de espera porque se siguen consumiendo recursos mientras el cliente espera a que transcurra el tiempo de espera. Un valor demasiado bajo puede generar un aumento del tráfico en el backend y un aumento de la latencia debido a que las solicitudes realizan demasiados reintentos. En algunos casos, esto puede producir una interrupción completa si se reintentan todas las solicitudes.

Tenga en cuenta lo siguiente al determinar las estrategias de tiempo de espera:

- Las solicitudes pueden tardar más de lo normal en procesarse debido a su contenido, a deficiencias en un servicio de destino o a un error en la partición de la red.
- Las solicitudes con contenido anormalmente caro podrían consumir recursos innecesarios del servidor y del cliente. En este caso, si se agota el tiempo de espera de estas solicitudes y no se vuelven a intentar, se pueden conservar los recursos. Los servicios también deberían protegerse del contenido anormalmente caro con restricciones y tiempos de espera del lado del servidor.

- Se puede agotar el tiempo de espera y volver a intentar las solicitudes que tarden un tiempo anormalmente largo debido a una interrupción del servicio. Se deben tener en cuenta los costes del servicio de la solicitud y el reintento, pero si la causa es una deficiencia localizada, es probable que el reintento no sea caro y reduzca el consumo de recursos del cliente. El tiempo de espera también puede liberar recursos del servidor según la naturaleza de la deficiencia.
- Se puede agotar el tiempo de espera y volver a intentar las solicitudes que tarden mucho en completarse porque la red no ha podido entregar la solicitud o la respuesta. Como la solicitud o la respuesta no se han entregado, el resultado habría sido un error independientemente del tiempo de espera. En este caso, el tiempo de espera no liberará los recursos del servidor, pero sí liberará los recursos del cliente y mejorará el rendimiento de la carga de trabajo.

Aproveche patrones de diseño bien establecidos, como los reintentos y los disyuntores, para gestionar los tiempos de espera correctamente y ofrecer enfoques de respuesta rápida a los errores. Los SDK de AWS y AWS CLI permiten configurar los tiempos de espera de las conexiones y las solicitudes y los reintentos con un retroceso exponencial y fluctuaciones. Las funciones de AWS Lambda admiten la configuración de tiempos de espera, y con AWS Step Functions, puede crear disyuntores de poco código que utilicen integraciones predefinidas con los servicios y los SDK de AWS. AWS App Mesh Envoy incluye capacidades de tiempo de espera y de disyuntor.

## Pasos para la implementación

- Configure tiempos de espera en las llamadas de servicio remotas y aproveche las características integradas de tiempo de espera del lenguaje o las bibliotecas de tiempo de espera de código abierto.
- Cuando su carga de trabajo realice llamadas con un SDK de AWS, consulte la documentación para ver la configuración del tiempo de espera específica de cada lenguaje.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++

- Cuando utilice SDK de AWS o comandos de la AWS CLI en su carga de trabajo, defina los valores predeterminados de AWS al configurar los valores de tiempo de espera predeterminados para `connectTimeoutInMillis` y `tlsNegotiationTimeoutInMillis`.
- Utilice las opciones de línea de comandos `cli-connect-timeout` y `cli-read-timeout` para controlar comandos de la AWS CLI puntuales de los servicios de AWS.
- Monitorice las llamadas de servicio remotas para comprobar si hay tiempos de espera y configure alarmas en caso de errores persistentes para poder gestionar los escenarios de error de forma proactiva.
- Implemente Métricas de CloudWatch y detección de anomalías de CloudWatch en los índices de error de las llamadas, los objetivos de nivel de servicio en lo que se refiere a la latencia y los valores atípicos de latencia para proporcionar información sobre la administración de tiempos de espera demasiado agresivos o permisivos.
- Configure tiempos de espera en funciones de Lambda.
- Los clientes de API Gateway deben implementar sus propios reintentos al gestionar los tiempos de espera. API Gateway admite un tiempo de espera de integración de 50 milisegundos a 29 segundos para integraciones posteriores y no vuelve a intentarlo cuando se agota el tiempo de espera de las solicitudes de integración.
- Implemente el patrón de disyuntor para que no se realicen llamadas remotas cuando se agote el tiempo de espera. Abra el circuito para evitar llamadas fallidas y ciérrelo cuando las llamadas respondan normalmente.
- Para cargas de trabajo basadas en contenedores, consulte las características de App Mesh Envoy para utilizar los tiempos de espera y los disyuntores integrados.
- Utilice AWS Step Functions para crear disyuntores de poco código para las llamadas de servicio remotas, especialmente cuando se utilizan SDK nativos de AWS e integraciones de Step Functions compatibles para simplificar la carga de trabajo.

## Recursos

Prácticas recomendadas relacionadas:

- REL05-BP03 Controlar y limitar las llamadas de reintento
- REL05-BP04 Responder rápido a los errores y limitar las colas
- REL06-BP07 Supervisar el seguimiento de las solicitudes de principio a fin en todo el sistema

Documentos relacionados:

- AWS SDK: Retries and Timeouts (SDK de AWS: reintentos y tiempos de espera)
- La Amazon Builders' Library: Tiempos de espera, reintentos y retroceso con alteración
- Cuotas de Amazon API Gateway y notas importantes
- AWS Command Line Interface: Command line options (Opciones de línea de comandos)
- AWS SDK for Java 2.x: Configure API Timeouts (Configurar los tiempos de espera de la API)
- AWS Botocore using the config object and Config Reference (AWS Botocore: Uso del objeto config y referencia de configuración)
- AWS SDK for .NET: Retries and Timeouts (Reintentos y tiempos de espera)
- AWS Lambda: Configuring Lambda function options (Configuración de las opciones de la función Lambda)

Ejemplos relacionados:

- Using the circuit breaker pattern with AWS Step Functions and Amazon DynamoDB (Uso del patrón del disyuntor con AWS Step Functions y Amazon DynamoDB)
- Martin Fowler: CircuitBreaker (Disyuntor)

Herramientas relacionadas:

- SDK de AWS
- AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

## REL05-BP06 Crear sistemas sin estado cuando sea posible

Los sistemas deben o bien no requerir estado o bien descargar el estado, de forma que entre solicitudes de clientes distintos no haya dependencia en los datos almacenados localmente en disco y en memoria. Esto permite reemplazar los servidores a voluntad sin que la disponibilidad resulte afectada.

Cuando los usuarios o los servicios interactúan con una aplicación, suelen realizar una serie de interacciones que constituyen una sesión. Una sesión es un dato único para los usuarios que persiste entre las solicitudes mientras utilizan la aplicación. Una aplicación sin estado es aquella que no necesita conocer las interacciones anteriores y no almacena la información de la sesión.

Una vez se ha diseñado para no tener estado, puede utilizar servicios de computación sin servidor, como AWS Lambda o AWS Fargate.

Además del reemplazo del servidor, otro beneficio de las aplicaciones sin estado es que pueden escalar horizontalmente porque cualquiera de los recursos de computación disponibles (como las instancias de EC2 y las funciones de AWS Lambda) puede dar servicio a cualquier solicitud.

Beneficios de establecer esta práctica recomendada: los sistemas que se han diseñado para no tener estado se adaptan mejor al escalamiento horizontal, lo que permite añadir o eliminar capacidad en función de la fluctuación del tráfico y la demanda. También son intrínsecamente resistentes a los errores y proporcionan flexibilidad y agilidad en el desarrollo de aplicaciones.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Cree aplicaciones sin estado. Las aplicaciones sin estado permiten el escalamiento horizontal y toleran el error de un nodo individual. Analice y comprenda los componentes de la aplicación que mantienen el estado dentro de la arquitectura. Esto le ayuda a evaluar el posible impacto de la transición a un diseño sin estado. Una arquitectura sin estado desacopla los datos del usuario y descarga los datos de la sesión. Esto proporciona la flexibilidad para escalar cada componente de forma independiente para cumplir con las diferentes demandas de carga de trabajo y optimizar la utilización de los recursos.

Pasos para la implementación

- Identifique y comprenda los componentes con estado de la aplicación.
- Desacople los datos separando y administrando los datos de usuario de la lógica principal de la aplicación.
  - Amazon Cognito puede desacoplar los datos de usuario del código de la aplicación mediante características, tales como grupos de identidades, grupos de usuarios y Amazon Cognito Sync.
  - Puede usar AWS Secrets Manager para desacoplar los datos de usuario almacenando los secretos en una ubicación segura y centralizada. Esto significa que el código de la aplicación no necesita almacenar secretos, lo que la hace más segura.

    - Plantéese utilizar Amazon S3 para almacenar datos no estructurados y de gran volumen, como imágenes y documentos. La aplicación puede recuperar estos datos cuando sea necesario, lo que elimina la necesidad de almacenarlos en la memoria.
    - Utilice Amazon DynamoDB para almacenar información, como, por ejemplo, perfiles de usuario. La aplicación puede consultar estos datos prácticamente en tiempo real.
- Descargue los datos de la sesión en una base de datos, caché o archivos externos.
    - Amazon ElastiCache, Amazon DynamoDB, Amazon Elastic File System (Amazon EFS) y Amazon MemoryDB for Redis son ejemplos de servicios de AWS que puede usar para descargar datos de sesión.
- Diseñe una arquitectura sin estado después de identificar qué datos de estado y de usuario deben conservarse con la solución de almacenamiento que elija.

### Recursos

Prácticas recomendadas relacionadas:

- REL11-BP03 Automatizar la reparación en todas las capas

Documentos relacionados:

- La Amazon Builders' Library: Evitar el retroceso en sistemas distribuidos
- La Amazon Builders' Library: Evitar trabajos pendientes en colas insalvables
- La Amazon Builders' Library: Desafíos y estrategias del almacenamiento en caché
- «Best Practices for Stateless Web Tier on AWS»

## REL05-BP07 Implementar recursos de emergencia

Los recursos de emergencia son procesos rápidos que pueden mitigar el impacto en la disponibilidad de la carga de trabajo.

Los recursos de emergencia desactivan, limitan o cambian el comportamiento de componentes o dependencias mediante mecanismos conocidos y probados. Esto puede aliviar las deficiencias de la carga de trabajo causadas por el agotamiento de los recursos debido a los aumentos inesperados de la demanda y reducir el impacto de los fallos en los componentes no críticos de la carga de trabajo.

Resultado deseado: al implementar recursos de emergencia, puede establecer procesos que se sabe que son buenos para mantener la disponibilidad de los componentes críticos de su carga de trabajo. La carga de trabajo debe degradarse de forma estable y seguir realizando sus funciones críticas para la empresa durante la activación de un recurso de emergencia. Para obtener más información sobre la degradación estable, consulte «REL05-BP01 Implementar una degradación estable para transformar las dependencias estrictas en flexibles».

Antipatrones usuales:

- El fallo de las dependencias no críticas repercute en la disponibilidad de su carga de trabajo principal.
- No probar o verificar el comportamiento de los componentes críticos durante el deterioro de los componentes no críticos.
- No definir criterios claros y deterministas para la activación o desactivación de un recurso de emergencia.

Beneficios de establecer esta práctica recomendada: la implementación de recursos de emergencia puede mejorar la disponibilidad de los componentes críticos de su carga de trabajo al proporcionar a sus solucionadores procesos establecidos para responder a picos inesperados de demanda o fallos de dependencias no críticas.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

- Identifique los componentes críticos de su carga de trabajo.
- Diseñe y cree los componentes críticos de su carga de trabajo para que resistan los fallos de los componentes no críticos.
- Realice pruebas para validar el comportamiento de sus componentes críticos durante el fallo de los componentes no críticos.
- Defina y supervise las métricas o los factores desencadenantes relevantes para iniciar los procedimientos de recursos de emergencia.
- Defina los procedimientos (manuales o automáticos) que componen el recurso de emergencia.

Pasos para la implementación

- Identifique los componentes críticos para la empresa en su carga de trabajo.

    - Cada componente técnico de su carga de trabajo debe asignarse a su función empresarial relevante y clasificarse como crítico o no crítico. Para ver ejemplos de funciones críticas y no críticas de Amazon, lea «Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second».
    - Se trata de una decisión tanto técnica como empresarial, y varía según la organización y la carga de trabajo.
- Diseñe y cree los componentes críticos de su carga de trabajo para que resistan los fallos de los componentes no críticos.
    - Durante el análisis de dependencias, tenga en cuenta todos los modos de fallo potenciales y verifique que sus mecanismos de recursos de emergencia proporcionan la funcionalidad crítica a los componentes downstream.
- Realice pruebas para validar el comportamiento de sus componentes críticos durante la activación de sus recursos de emergencia.
    - Evite el comportamiento bimodal. Para obtener más información, consulte «REL11-BP05 Usar la estabilidad estática para evitar el comportamiento bimodal».
- Defina, supervise y alerte sobre las métricas relevantes para iniciar el procedimiento del recurso de emergencia.
    - Encontrar las métricas adecuadas para supervisar depende de su carga de trabajo. Algunos ejemplos de métricas son la latencia o el número de solicitudes fallidas a una dependencia.
- Defina los procedimientos (manuales o automáticos) que componen el recurso de emergencia.
    - Esto puede incluir mecanismos como el desbordamiento de carga, la limitación de solicitudes o la implementación de una degradación estable.

## Recursos

Prácticas recomendadas relacionadas:

- REL05-BP01 Implementar una degradación estable para transformar las dependencias estrictas en flexibles
- REL05-BP02 Limitar las solicitudes
- REL11-BP05 Usar la estabilidad estática para evitar el comportamiento bimodal

Documentos relacionados:

- Automatización de implementaciones seguras y sin intervención
- «Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second»

Vídeos relacionados:

- «AWS re:Invent 2020: Reliability, consistency, and confidence through immutability»

# Administración de cambios

Los cambios en su carga de trabajo o su entorno deben preverse y se les debe dar cabida para poder conseguir un funcionamiento fiable de la carga de trabajo. Los cambios incluyen aquellos que se imponen a su carga de trabajo, como los picos de demanda, además de los inherentes a ella, como los despliegues de funciones o los parches de seguridad.

En las siguientes secciones se explican las prácticas recomendadas para la administración de cambios.

Temas



## Supervisar los recursos de la carga de trabajo

Los registros y las métricas son una potente herramienta para obtener información sobre el estado de sus cargas de trabajo. Puede configurar su carga de trabajo de forma que supervise registros y métricas, y envíe notificaciones cuando se crucen ciertos umbrales o se produzcan eventos importantes. La supervisión permite que su carga de trabajo reconozca cuándo se cruzan umbrales de bajo rendimiento o se producen errores, para que pueda recuperarse de los errores rápidamente una vez recibida una respuesta.

La supervisión es vital para garantizar el cumplimiento de los requisitos de disponibilidad. El monitoreo debe detectar los errores de manera efectiva. El peor modo de error es el «silencioso», en el que la funcionalidad ya no sirve, pero no hay forma de detectarlo, excepto indirectamente. El cliente lo sabe antes que usted. Alertar cuando tiene problemas es una de las principales razones por las que monitorea. Las alertas deben estar desacopladas del sistema en la medida de lo posible. Si la interrupción del servicio elimina la posibilidad de generar alertas, habrá un periodo de interrupción más largo.

En AWS, instrumentamos nuestras aplicaciones en múltiples niveles. Registramos la latencia, las tasas de error y la disponibilidad para cada solicitud, para todas las dependencias y para las operaciones clave dentro del proceso. También registramos métricas de operación exitosa. Esto permite ver los problemas inminentes antes de que sucedan. No nos limitamos a tener en cuenta

la latencia media. Nos centramos más estrechamente en las divergencias de latencia, como el percentil 99,9 o el 99,99. Esto se debe a que, si una solicitud de entre 1000 o 10 000 es lenta, sigue provocando una experiencia deficiente. Además, aunque el promedio sea aceptable, si una de cada 100 solicitudes causa una latencia extrema, acabará convirtiéndose en un problema cuando el tráfico crezca.

La supervisión en AWS consta de cuatro fases distintas:

1. Generación: supervisar todos los componentes de la carga de trabajo
2. Agregación: definir y calcular métricas
3. Procesamiento y alarmas en tiempo real: enviar notificaciones y automatizar respuestas
4. Almacenamiento y análisis

Prácticas recomendadas

- REL06-BP01 Supervisar todos los componentes de la carga de trabajo (generación)
- REL06-BP02 Definir y calcular métricas (agregación)
- REL06-BP03 Enviar notificaciones (procesamiento y alarmas en tiempo real)
- REL06-BP04 Automatizar las respuestas (procesamiento y alarmas en tiempo real)
- REL06-BP05 Análisis
- REL06-BP06 Realizar revisiones con frecuencia
- REL06-BP07 Supervisar el seguimiento de las solicitudes de principio a fin en todo el sistema

## REL06-BP01 Supervisar todos los componentes de la carga de trabajo (generación)

Supervise los componentes de la carga de trabajo con Amazon CloudWatch o herramientas de terceros. Supervise los servicios de AWS con el panel de AWS Health.

Debería supervisar todos los componentes de su carga de trabajo, incluidos los niveles del front-end, la lógica empresarial y el almacenamiento. Defina métricas claves, describa cómo extraerlas de los registros (si fuera necesario) y establezca umbrales para desencadenar los eventos de alarma correspondientes. Asegúrese de que las métricas sean pertinentes para los indicadores clave de rendimiento (KPI) de su carga de trabajo, y utilice métricas y registros para identificar signos de advertencia tempranos de degradación del servicio. Por ejemplo, una métrica relacionada con los resultados empresariales como el número de pedidos procesado satisfactoriamente por minuto,

puede indicar problemas con la carga de trabajo más rápido que una métrica técnica, como el uso de la CPU. Utilice el panel de AWS Health para obtener una vista personalizada sobre el rendimiento y la disponibilidad de los servicios de AWS subyacentes a sus recursos de AWS.

La supervisión en la nube ofrece nuevas oportunidades. La mayoría de proveedores en la nube han desarrollado enlaces personalizables y pueden proporcionar conocimientos para ayudarle a supervisar varias capas de su carga de trabajo. Los servicios de AWS como Amazon CloudWatch aplican algoritmos estadísticos y de machine learning para analizar continuamente las métricas de los sistemas y aplicaciones, determinar las bases de referencia normales y hacer aflorar anomalías con una intervención mínima del usuario. Los algoritmos de detección de anomalías tienen en cuenta la estacionalidad y los cambios en las tendencias de las métricas.

AWS pone a disposición una gran cantidad de información de supervisión y registro para el consumo que se puede usar para definir métricas específicas de la carga de trabajo, procesos de cambio en la demanda y adoptar técnicas de machine learning independientemente de los conocimientos sobre ML.

Además, puede supervisar todos sus puntos de conexión externos para asegurarse de que sean independientes de su implementación base. Esta supervisión activa se puede llevar a cabo con transacciones sintéticas (a las que a veces se denomina «canaries» de usuario, y que no deben confundirse con los despliegues de valores controlados o «canary»), que ejecutan periódicamente varias tareas comunes que se ajustan a las acciones realizadas por los clientes de la carga de trabajo. Mantenga una duración breve para estas tareas y asegúrese de no sobrecargar sus cargas de trabajo durante las pruebas. Amazon CloudWatch Synthetics le permite: crear pruebas de transacciones o «canaries» sintéticas para supervisar sus puntos de conexión y API. También puede combinar los nodos de cliente de la «canary» sintética con la consola de AWS X-Ray para detectar qué «canaries» sintéticas están teniendo problemas de errores, fallos o limitaciones para el periodo de tiempo seleccionado.

Resultado deseado:

Recopilar y usar métricas esenciales de todos los componentes de la carga de trabajo para garantizar la fiabilidad de la carga de trabajo y una experiencia de usuario óptima. Detectar que una carga de trabajo no consigue los resultados empresariales le permite declarar rápidamente una situación de desastre y recuperarse de un incidente.

Patrones de uso no recomendados comunes:

- Supervisar solamente las interfaces externas con su carga de trabajo

- No generar métricas específicas de una carga de trabajo y basarse solamente en las métricas que proporcionan los servicios de AWS que usa su carga de trabajo.
- Usar exclusivamente métricas técnicas en su carga de trabajo y no supervisar las métricas relacionadas con KPI no técnicos a los que contribuye la carga de trabajo.
- Basarse en el tráfico de producción y las comprobaciones de estado sencillas para supervisar y evaluar el estado de las cargas de trabajo.

Beneficios de establecer esta práctica recomendada: La supervisión de todos los niveles de la carga de trabajo le permite prever y resolver los problemas rápidamente en los componentes de la carga de trabajo.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

1. Habilite el registro cuando esté disponible. La supervisión de los datos debe obtenerse a partir de todos los componentes de las cargas de trabajo. Active métodos de registro adicionales, como los registros de acceso de S3, y permita que su carga de trabajo registre datos específicos de la carga de trabajo. Recopile métricas para los promedios de CPU, E/S de red y E/S de disco de servicios como Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling y Amazon EMR. Consulte Servicios de AWS que publican métricas de CloudWatch para consultar una lista de servicios de AWS que publican métricas en CloudWatch.
2. Revise todas las métricas predeterminadas y explore las carencias en cuanto a recopilación de datos. Todos los servicios generan métricas predeterminadas. La recopilación de métricas predeterminadas le permite comprender mejor las dependencias entre los componentes de la carga de trabajo, y cómo la fiabilidad y el rendimiento de los componentes afectan a la carga de trabajo. También puede crear y publicar sus propias métricas en CloudWatch utilizando la AWS CLI o una API. Esto
3. Evalúe todas las métricas para decidir sobre cuáles alertar en cada servicio de AWS en su carga de trabajo. Puede decidir seleccionar un subconjunto de métricas que tenga un impacto importante en la fiabilidad de la carga de trabajo. Al centrarse en las métricas y umbrales críticos, podrá refinar el número de alertas de emergencia y contribuir a reducir al mínimo los falsos positivos.
4. Defina las alertas y los procesos de recuperación para su carga de trabajo una vez que se active la alerta. La definición de alertas le permite notificar, escalar y seguir los pasos necesarios rápidamente para recuperarse de un incidente y cumplir el objetivo de tiempo de recuperación

(RTO) prescrito. Puede usar alarmas de Amazon CloudWatch para invocar flujos de trabajo automatizados e iniciar procedimientos de recuperación basados en los umbrales definidos.

5. Explore el uso de transacciones sintéticas para recopilar datos relevantes sobre el estado de las cargas de trabajo. La supervisión sintética sigue las mismas rutas y lleva a cabo las mismas acciones que un cliente, lo que le permite verificar continuamente su experiencia de usuario incluso si no tiene tráfico de cliente en sus cargas de trabajo. Al usar transacciones sintéticas, puede detectar los problemas antes de que lo hagan los clientes.

## Recursos

Prácticas recomendadas relacionadas:

- REL11-BP03 Automatizar la reparación en todas las capas

Documentos relacionados:

- Introducción al panel de AWS Health: estado de su cuenta
- Servicios de AWS que publican métricas de CloudWatch
- Registros de acceso para su Network Load Balancer
- Registros de acceso para su Application Load Balancer
- Acceso a Amazon CloudWatch Logs para AWS Lambda
- Registro de acceso al servidor de Amazon S3
- Habilitar los registros de acceso para su Classic Load Balancer
- Exportación de datos de registro a Amazon S3
- Instalar el agente de CloudWatch en una instancia Amazon EC2
- Publicar métricas personalizadas
- Uso de paneles de Amazon CloudWatch
- Uso de métricas de Amazon CloudWatch
- Uso de «canaries» (Amazon CloudWatch Synthetics)
- ¿Qué son Amazon CloudWatch Logs?

Guías del usuario:

- Cree un registro de seguimiento
- Supervisión de memoria y métricas del disco para las instancias Linux de Amazon EC2

- Uso de CloudWatch Logs con instancias de contenedor
- Registros de flujo de VPC
- ¿Qué es Amazon DevOps Guru?
- ¿Qué es AWS X-Ray?

Blogs relacionados:

- Depuración con Amazon CloudWatch Synthetics y AWS X-Ray

Ejemplos relacionados y talleres:

- Laboratorios de AWS Well-Architected: excelencia operativa - supervisión de dependencias
- La Amazon Builders' Library: Instrumentación de los sistemas distribuidos para la visibilidad de las operaciones
- Taller sobre observabilidad

## REL06-BP02 Definir y calcular métricas (agregación)

Almacene los datos de registro y aplique filtros cuando sea necesario para calcular métricas, como las veces que se produce un evento de registro específico o la latencia calculada a partir de las marcas temporales del evento de registro.

Amazon CloudWatch y Amazon S3 sirven como las capas principales de agregación y almacenamiento. En algunos servicios, como AWS Auto Scaling y Elastic Load Balancing, las métricas predeterminadas se proporcionan listas para usar para la carga de CPU o la latencia promedio de solicitudes en un clúster o instancia. En servicios de streaming, como VPC Flow Logs o AWS CloudTrail, los datos del evento se envían a CloudWatch Logs y debe definir y aplicar filtros para extraer las métricas de los datos del evento. Esto le presenta datos sobre las series temporales, que pueden servir como entradas para las alarmas de CloudWatch que defina para activar las alertas.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

- Defina y calcule métricas (agregación). Almacene los datos de registro y aplique filtros cuando sea necesario para calcular métricas, como las veces que se produce un evento de registro específico o la latencia calculada de las marcas temporales del evento de registro.
  - Los filtros de métricas definen los términos y patrones que analizar en los datos de registro a medida que se envían a CloudWatch Logs. CloudWatch Logs usa estos filtros para convertir los datos de registro en métricas numéricas de CloudWatch que puede representar en gráficas o a partir de las cuales puede establecer alarmas.
    - Buscar y filtrar datos de registro
  - Use un tercero de confianza para agregar registros.
    - Siga las instrucciones de la solución externa. La mayoría de los productos de terceros se integran con CloudWatch y Amazon S3.
  - Algunos servicios de AWS pueden publicar registros directamente en Amazon S3. Si su requisito principal para los registros es almacenarlos en Amazon S3, puede hacer que el servidor que crea los registros los envíe directamente a Amazon S3 sin instalar infraestructura adicional.
    - Enviar registros directamente a Amazon S3

## Recursos

Documentos relacionados:

- Consultas de ejemplo de Amazon CloudWatch Logs Insights
- Depuración con Amazon CloudWatch Synthetics y AWS X-Ray
- Taller sobre observabilidad
- Buscar y filtrar datos de registro
- Enviar registros directamente a Amazon S3
- La Amazon Builders' Library: Instrumentación de los sistemas distribuidos para la visibilidad de las operaciones

# REL06-BP03 Enviar notificaciones (procesamiento y alarmas en tiempo real)

Cuando las organizaciones detectan posibles problemas, envían notificaciones y alertas en tiempo real al personal y los sistemas correspondientes para poder responder de manera rápida y eficaz a estos problemas.

Resultado deseado: es posible responder rápidamente a los eventos operativos con la configuración de las alarmas correspondientes en función de las métricas del servicio y la aplicación. Cuando se superan los umbrales de alarma, se avisa al personal y a los sistemas adecuados para que puedan abordar los problemas subyacentes.

Patrones comunes de uso no recomendados:

- Las alarmas están configuradas con un umbral excesivamente alto, lo que impide que se envíen notificaciones vitales.
- Las alarmas están configuradas con un umbral demasiado bajo, lo que provoca inacción en las alertas importantes por el ruido que genera el exceso de notificaciones.
- Las alarmas y los umbrales no se actualizan cuando hay cambios de uso.
- En el caso de las alarmas que se abordan mejor con acciones automatizadas, en lugar de generar la acción automatizada, se envían notificaciones al personal, lo que provoca un exceso de notificaciones.

Beneficios de establecer esta práctica recomendada: enviar notificaciones y alertas en tiempo real al personal y a los sistemas adecuados permite detectar problemas de forma temprana y responder rápidamente a los incidentes operativos.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Las cargas de trabajo deben estar equipadas con sistemas de procesamiento y generación de alarmas en tiempo real que permitan mejorar la capacidad de detección de problemas que podrían afectar a la disponibilidad de la aplicación y actúen como desencadenantes de una respuesta automatizada. Las organizaciones pueden realizar el procesamiento y generar alarmas en tiempo real creando alertas con métricas definidas para recibir notificaciones siempre que ocurran eventos importantes o una métrica supere un umbral.

Amazon CloudWatch le permite crear alarmas de métricas y alarmas compuestas mediante las alarmas de CloudWatch basadas en un umbral estático, la detección de anomalías y otros criterios. Para obtener más información sobre los tipos de alarmas que puede configurar con CloudWatch, consulte la sección de alarmas de la documentación de CloudWatch.

Puede crear vistas personalizadas de las métricas y alertas de los recursos de AWS para sus equipos mediante los paneles de CloudWatch. Las páginas de inicio personalizables de la consola de CloudWatch le permiten supervisar los recursos a través de una única vista de las diferentes regiones.

Las alarmas pueden realizar una o varias acciones, como enviar una notificación a un tema de Amazon SNS, realizar una acción de Amazon EC2 o de Amazon EC2 Auto Scaling o crear un elemento OpsItem o bien un incidente en AWS Systems Manager.

Amazon CloudWatch usa Amazon SNS para enviar notificaciones cuando la alarma cambia de estado, lo que permite que los editores (productores) envíen mensajes a los suscriptores (consumidores). Para obtener más detalles sobre la configuración de las notificaciones de Amazon SNS, consulte Configuración de Amazon SNS.

CloudWatch envía eventos EventBridge siempre que se crea, se actualiza, se elimina o cambia de estado un alarma de CloudWatch. Puede usar EventBridge con estos eventos para crear reglas que realicen acciones, como avisarle cada vez que cambie el estado de una alarma o que activen eventos en la cuenta de forma automática mediante la automatización de Systems Manager.

¿Cuándo debe usar EventBridge o Amazon SNS?

Tanto EventBridge como Amazon SNS se pueden utilizar para desarrollar aplicaciones basadas en eventos, así que la elección de uno u otro dependerá de sus necesidades específicas.

Se recomienda usar Amazon EventBridge si desea crear una aplicación que reaccione a los eventos de sus propias aplicaciones, de aplicaciones SaaS y de servicios de AWS. EventBridge es el único servicio basado en eventos que se integra directamente con socios de SaaS de terceros. EventBridge también ingiere automáticamente eventos de más de 200 servicios de AWS sin necesidad de que los desarrolladores tengan que crear ningún recurso en la cuenta.

EventBridge utiliza una estructura definida basada en JSON para los eventos y le ayuda a crear reglas que se aplican a todo el cuerpo del evento para seleccionar los eventos que se van a reenviar a un destino. Actualmente, EventBridge admite más de 20 servicios de AWS como destino, entre los que se incluyen AWS Lambda, Amazon SQS, Amazon SNS, Amazon Kinesis Data Streamsy Amazon Data Firehose.

Se recomienda usar Amazon SNS con aplicaciones que necesiten una gran distribución (miles o millones de puntos de conexión). Un patrón habitual que vemos con frecuencia es que los clientes usan Amazon SNS como destino de la regla para filtrar los eventos que necesitan y distribuirlos a diversos puntos de conexión.

Los mensajes no están estructurados y pueden tener cualquier formato. Amazon SNS permite reenviar mensajes a seis tipos diferentes de destinos, como Lambda, Amazon SQS, puntos de conexión HTTP/S, SMS, notificaciones push móviles y correo electrónico. La latencia normal de Amazon SNS es inferior a 30 milisegundos. Hay un gran número de servicios de AWS que envían mensajes de Amazon SNS si se configuran para ello (hay más de 30, incluidos Amazon EC2, Amazon S3y Amazon RDS).

Pasos para la implementación

1. Cree una alarma con las alarmas de Amazon CloudWatch.
    a. Las alarmas de métricas supervisan una única métrica de CloudWatch o una expresión que depende de las métricas de CloudWatch. La alarma activa una o varias acciones en función del valor de la métrica o de la expresión en comparación con un umbral durante varios intervalos de tiempo. La acción puede consistir en enviar una notificación a un tema de Amazon SNS, realizar una acción de Amazon EC2 o de Amazon EC2 Auto Scaling o crear un elemento OpsItem o bien un incidente en AWS Systems Manager.
    b. Una alarma compuesta es una expresión de regla que tiene en cuenta las condiciones de otras alarmas que se han creado. La alarma compuesta solo entra en estado de alarma si se cumplen todas las condiciones de la regla. Las alarmas especificadas en la expresión de la regla de una alarma compuesta pueden ser alarmas de métricas y otras alarmas compuestas. Las alarmas compuestas pueden enviar notificaciones de Amazon SNS cuando su estado cambia y pueden crear elementos OpsItem de Systems Manager o bien incidentes cuando entran en estado de alarma, pero no pueden realizar ninguna acción de Amazon EC2 o Auto Scaling.
2. Configure Notificaciones de Amazon SNS. Al crear una alarma de CloudWatch, puede incluir un tema de Amazon SNS para enviar una notificación cuando la alarma cambie de estado.
3. Cree reglas en EventBridge que coincidan con las alarmas de CloudWatch especificadas. Cada regla admite varios destinos, incluidas las funciones de Lambda. Por ejemplo, puede definir una alarma que se inicie cuando el espacio disponible en disco se esté agotando, lo que desencadenará una función de Lambda mediante una regla de EventBridge para limpiar el espacio. Para obtener más información sobre los destinos de EventBridge, consulte EventBridge targets.

## Recursos

Prácticas recomendadas por Well-Architected:

- REL06-BP01 Supervisar todos los componentes de la carga de trabajo (generación)
- REL06-BP02 Definir y calcular métricas (agregación)
- REL12-BP01 Usar guías de estrategias para investigar los errores

Documentos relacionados:

- Amazon CloudWatch
- CloudWatch Logs insights
- Using Amazon CloudWatch alarms
- Using Amazon CloudWatch dashboards
- Using Amazon CloudWatch metrics
- Setting up Amazon SNS notifications
- detección de anomalías de CloudWatch
- Protección de datos en CloudWatch Logs
- Amazon EventBridge
- Amazon Simple Notification Service

Vídeos relacionados:

- Vídeos sobre observabilidad de reinvent 2022
- AWS re:Invent 2022 - Observability best practices at Amazon

Ejemplos relacionados:

- Taller sobre observabilidad
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

# REL06-BP04 Automatizar las respuestas (procesamiento y alarmas en tiempo real)

Use la automatización para actuar cuando se detecte un evento, por ejemplo, para sustituir componentes defectuosos.

El procesamiento automatizado de las alarmas en tiempo real se implementa para que los sistemas puedan tomar medidas correctivas rápidas e intentar evitar fallos o que el servicio se degrade cuando se activan las alarmas. Entre las respuestas automatizadas a las alarmas, se podría incluir la sustitución de los componentes que fallan, el ajuste de la capacidad de computación, el redireccionamiento del tráfico a hosts, zonas de disponibilidad u otras regiones en buen estado y la notificación a los operadores.

Resultado deseado: se identifican las alarmas en tiempo real y se configura el procesamiento automatizado de las alarmas para invocar las acciones apropiadas que se necesitan para mantener los objetivos de nivel de servicio y los acuerdos de nivel de servicio (SLA). La automatización puede abarcar desde actividades de autorreparación de componentes individuales hasta la conmutación por error de todo el sitio.

Antipatrones usuales:

- No tener un inventario o catálogo claros de las principales alarmas en tiempo real.
- No tener respuestas automatizadas en las alarmas críticas (por ejemplo, cuando los recursos de computación están a punto de agotarse, se produce un escalamiento automático).
- Acciones de respuesta a alarmas contradictorias.
- No tener procedimientos operativos estándar (SOP) que los operadores puedan seguir cuando reciben notificaciones de alerta.
- No supervisar los cambios de configuración, ya que los cambios de configuración no detectados pueden provocar un tiempo de inactividad en las cargas de trabajo.
- No tener una estrategia para deshacer los cambios de configuración no deseados.

Ventajas de establecer esta práctica recomendada: la automatización del procesamiento de alarmas puede mejorar la resiliencia del sistema. El sistema aplica las medidas correctivas automáticamente, lo que reduce las actividades manuales que dan lugar a intervenciones humanas que son más susceptibles a errores. Las operaciones de carga de trabajo cumplen los objetivos de disponibilidad y reducen la interrupción del servicio.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Para administrar eficazmente las alertas y automatizar su respuesta, clasifique las alertas en función de su importancia y repercusión, documente los procedimientos de respuesta y planifique las respuestas antes de clasificar las tareas.

Identifique las tareas que requieren medidas específicas (suelen detallarse en los runbooks) y examine todos los runbooks y guías de estrategias para determinar qué tareas se pueden automatizar. Si se pueden definir acciones, estas suelen poderse automatizar. Si las acciones no se pueden automatizar, documente los pasos manuales en un SOP y forme a los operadores sobre ellos. Analice continuamente los procesos manuales en busca de oportunidades de automatización en las que pueda establecer y mantener un plan para automatizar las respuestas a las alertas.

Pasos para la implementación

1. Cree un inventario de alarmas: para obtener una lista de todas las alarmas, puede utilizar la AWS CLI con el comando de Amazon CloudWatch `describe-alarms`. Según el número de alarmas que haya configurado, es posible que tenga que usar la paginación para recuperar un subconjunto de alarmas para cada llamada o, alternativamente, puede usar el SDK de AWS para obtener las alarmas mediante una llamada a la API.
2. Documente todas las acciones de las alarmas: actualice un runbook con todas las alarmas y sus acciones, independientemente de si son manuales o automatizadas. AWS Systems Manager proporciona runbooks predefinidos. Para obtener más información sobre los runbooks, consulte «Creación de sus propios manuales de procedimientos». Para obtener más información sobre cómo ver el contenido del runbook, consulte «View runbook content».
3. Configure y administre las acciones de las alarmas: para cualquiera de las alarmas que requieran una acción, especifique la acción automatizada mediante el SDK de CloudWatch. Por ejemplo, puede cambiar el estado de sus instancias de Amazon EC2 automáticamente en función de una alarma de CloudWatch. Para ello, cree y habilite acciones en una alarma o deshabilite acciones en una alarma.

   También se puede utilizar Amazon EventBridge para responder automáticamente a los eventos del sistema, como los problemas de disponibilidad de las aplicaciones o los cambios en los recursos. Puede crear reglas para indicar qué eventos le interesan y las acciones que se deben realizar cuando un evento coincide con una regla. Entre las acciones que se pueden iniciar automáticamente, se incluye invocar una función AWS Lambda, invocar el comando de ejecución

de Amazon EC2, transmitir el evento a Amazon Kinesis Data Streams y ver cómo se automatiza Amazon EC2 con EventBridge.

4. Procedimientos operativos estándar (SOP): en función de los componentes que tenga su aplicación, AWS Resilience Hub recomienda varias plantillas de SOP. Puede utilizar estos SOP para documentar todos los procesos que debe seguir un operador en caso de que se genere una alerta. También puede crear un SOP basado en recomendaciones de Resilience Hub cuando necesite una aplicación Resilience Hub con una política de resiliencia asociada, así como una evaluación de resiliencia histórica en relación con esa aplicación. Las recomendaciones para su SOP provienen de la evaluación de resiliencia.

   Resilience Hub funciona con Systems Manager para automatizar los pasos de sus SOP al proporcionar una serie de SSMdocumentos que puede utilizar como base para esos SOP. Por ejemplo, Resilience Hub puede recomendar un SOP para añadir espacio en disco basándose en un documento de automatización de SSM existente.

5. Realice acciones automatizadas con Amazon DevOps Guru: puede utilizar Amazon DevOps Guru para supervisar automáticamente los recursos de la aplicación en busca de un comportamiento anómalo y ofrecer recomendaciones específicas para reducir el tiempo de identificación y resolución de problemas. ConDevOps Guru, puede supervisar secuencias de datos operativos casi en tiempo real desde múltiples orígenes, como métricas de Amazon CloudWatch, AWS Config, AWS CloudFormation y AWS X-Ray. También puede utilizar DevOps Guru para crear automáticamente OpsItems en OpsCenter y enviar eventos a EventBridge para una automatización adicional.

## Recursos

Prácticas recomendadas relacionadas:

- REL06-BP01 Supervisar todos los componentes de la carga de trabajo (generación)
- REL06-BP02 Definir y calcular métricas (agregación)
- REL06-BP03 Enviar notificaciones (procesamiento y alarmas en tiempo real)
- REL08-BP01 Usar runbooks para actividades estándares como la implementación

Documentos relacionados:

- «AWS Systems Manager Automation»
- «Creating an EventBridge Rule That Triggers on an Event from an AWS Resource»

- Taller sobre observabilidad
- La Amazon Builders' Library: Instrumentación de los sistemas distribuidos para la visibilidad de las operaciones
- «What IsAmazon DevOps Guru?»
- Trabajar con documentos de automatización (guías de estrategias)

Vídeos relacionados:

- AWS re:Invent 2022 - Observability best practices at Amazon (AWS re:Invent 2022: Prácticas recomendadas de observabilidad en Amazon)
- «AWS re:Invent 2020: Automate anything with AWS Systems Manager»
- «Introduction to AWS Resilience Hub»
- «Create Custom Ticket Systems for Amazon DevOps Guru Notifications»
- «Enable Multi-Account Insight Aggregation with Amazon DevOps Guru»

Ejemplos relacionados:

- Reliability Workshops (Talleres de fiabilidad)
- «Amazon CloudWatch and Systems Manager Workshop»

## REL06-BP05 Análisis

Recopile archivos de registros e historiales de métricas y analícelos para identificar tendencias e información sobre las cargas de trabajo.

Amazon CloudWatch Logs Insights es compatible con un lenguaje de consultas sencillo pero potente que puede usar para analizar datos de registro. Amazon CloudWatch Logs también admite suscripciones que permiten a los datos dirigirse de forma fluida hacia Amazon S3, donde podrá usar Amazon Athena para consultar los datos. También es compatible con consultas en una gran variedad de formatos. Consulte Formatos de SerDes y datos compatibles en la Guía del usuario de Amazon Athena para obtener más información. Para los análisis de conjuntos de archivos de registro enormes, puede ejecutar un clúster de Amazon EMR para ejecutar análisis en la escala de los petabytes.

Hay una serie de herramientas proporcionadas por socios de AWS y terceros que permiten la agregación, procesamiento, almacenamiento y análisis. Entre estas herramientas se incluyen New

Relic, Splunk, Loggly, Logstash, CloudHealth y Nagios. Sin embargo, la generación fuera de los registros del sistema y las aplicaciones es exclusiva de cada proveedor de la nube y, a menudo, exclusiva de cada servicio.

Una parte del proceso de monitoreo que a menudo se pasa por alto es la gestión de datos. Necesita determinar los requisitos de retención para supervisar los datos y, luego, aplicar las políticas del ciclo de vida correspondientemente. Amazon S3 permite la administración del ciclo de vida en el nivel del bucket de S3. Esta gestión del ciclo de vida se puede aplicar de manera diferente a diferentes rutas en el bucket. Hacia el final del ciclo de vida, puede realizar la transición de datos a Amazon S3 Glacier para el almacenamiento a largo plazo y vencimiento, una vez alcanzado el final del periodo de retención. La clase de almacenamiento de S3 Intelligent-Tiering está diseñado para optimizar los costos trasladando automáticamente los datos al nivel de acceso más eficiente, sin que se vea afectado el rendimiento ni los gastos generales operativos.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

## Guía para la implementación

- CloudWatch Logs Insights le permite buscar y analizar de forma interactiva sus datos de registro en Amazon CloudWatch Logs.
  - Análisis de los datos de registro con CloudWatch Logs Insights
  - Consultas de ejemplo de Amazon CloudWatch Logs Insights
- Use Amazon CloudWatch Logs para enviar registros a Amazon S3, donde puede usar Amazon Athena para consultar los datos.
  - ¿Cómo analizo mis registros de acceso al servidor de Amazon S3 mediante Athena?
    - Cree una política de ciclo de vida de S3 para su bucket de registros de acceso al servidor. Configure la política de ciclo de vida para que se eliminen periódicamente los archivos de registros. De esta forma, reducirá la cantidad de datos que analiza Athena en cada consulta.
      - ¿Cómo creo una política de ciclo de vida para un bucket de S3?

## Recursos

Documentos relacionados:

- Consultas de ejemplo de Amazon CloudWatch Logs Insights
- Análisis de los datos de registro con CloudWatch Logs Insights
- Depuración con Amazon CloudWatch Synthetics y AWS X-Ray

- ¿Cómo creo una política de ciclo de vida para un bucket de S3?
- ¿Cómo analizo mis registros de acceso al servidor de Amazon S3 mediante Athena?
- Taller sobre observabilidad
- La Amazon Builders' Library: Instrumentación de los sistemas distribuidos para la visibilidad de las operaciones

## REL06-BP06 Realizar revisiones con frecuencia

Revise frecuentemente cómo está implementada la supervisión de cargas de trabajo y actualícela en función de eventos y cambios importantes.

La supervisión efectiva se basa en métricas empresariales claves. Asegúrese de que estas métricas tengan cabida en su carga de trabajo a medida que cambien las prioridades empresariales.

La auditoría de su supervisión le permite asegurarse de que sabrá cuándo cumple una aplicación con sus objetivos de disponibilidad. El análisis de las causas raíces requiere la capacidad de descubrir qué ha ocurrido cuando se produce un error. AWS facilita servicios que le permiten realizar un seguimiento del estado de sus servicios durante un incidente:

- Amazon CloudWatch Logs: puede almacenar sus registros en este servicio e inspeccionar sus contenidos.
- Amazon CloudWatch Logs Insights: es un servicio totalmente administrado que le permite analizar registros inmensos en segundos. Le ofrece consultas y visualizaciones rápidas e interactivas.
- AWS Config: puede ver qué infraestructura de AWS se ha estado utilizando en diferentes momentos.
- AWS CloudTrail: puede ver qué API de AWS se invocaron en qué momento y desde qué entidad principal.

En AWS, realizamos una reunión semanal para revisar el rendimiento operativo y compartir lo que hemos aprendido entre los equipos. Como hay tantos equipos en AWS, creamos La rueda para elegir al azar una carga de trabajo que revisar. El establecimiento de una cadencia regular para las revisiones de rendimiento operativo y el intercambio de conocimientos mejorará su capacidad para lograr un mayor rendimiento de sus equipos operativos.

Patrones de uso no recomendados comunes:

- Recopilar solo métricas predeterminadas

- Establecer una estrategia de supervisión y no revisarla nunca
- No considerar la supervisión cuando se implementan cambios importantes

Beneficios de establecer esta práctica recomendada: la revisión periódica de la supervisión le permite anticiparse a los posibles problemas en lugar de reaccionar a las notificaciones cuando se produzca un problema previsto.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

## Guía para la implementación

- Cree varios paneles para la carga de trabajo. Debe tener un panel general que contenga las principales métricas del negocio, así como las métricas técnicas que ha identificado como más relevantes para el estado previsto de la carga de trabajo conforme cambie su uso. También debe tener paneles para los distintos niveles y dependencias de la aplicación que puedan inspeccionarse.
  - Uso de paneles de Amazon CloudWatch
- Programe y realice revisiones periódicas de los paneles de cargas de trabajo. Realice una inspección periódica de los paneles. Puede tener diferentes cadencias para el alcance de la inspección.
  - Inspeccione las tendencias en las métricas. Compare los valores de las métricas con los valores históricos para saber si hay tendencias que puedan indicar que algo necesita ser investigado. Algunos ejemplos son un aumento de la latencia, una reducción de la función empresarial principal y un aumento de las respuestas a los errores.
  - Inspeccione valores atípicos o anomalías en las métricas. Los promedios o las medianas pueden ocultar valores atípicos y anomalías. Examine los valores más altos y más bajos durante el período de tiempo e investigue las causas de los valores extremos. Mientras elimina estas causas, la relajación de la definición de «extremo» le permitirá seguir mejorando la sistematicidad del rendimiento de sus cargas de trabajo.
  - Busque cambios bruscos en el comportamiento. Un cambio inmediato en la cantidad o en la dirección de una métrica podría indicar que se ha producido un cambio en la aplicación o factores externos que podrían necesitar la inclusión de métricas adicionales para su seguimiento.

### Recursos

Documentos relacionados:

- Consultas de ejemplo de Amazon CloudWatch Logs Insights
- Depuración con Amazon CloudWatch Synthetics y AWS X-Ray
- Taller sobre observabilidad
- La Amazon Builders' Library: Instrumentación de los sistemas distribuidos para la visibilidad de las operaciones
- Uso de paneles de Amazon CloudWatch

## REL06-BP07 Supervisar el seguimiento de las solicitudes de principio a fin en todo el sistema

Realice un seguimiento de las solicitudes a medida que se procesan a través de los componentes del servicio para que los equipos de producto puedan analizar y depurar los problemas con mayor facilidad y mejorar el rendimiento.

Resultado deseado: las cargas de trabajo con un rastreo exhaustivo en todos los componentes son fáciles de depurar, lo que mejora el tiempo medio de resolución (MTTR) de los errores y la latencia al simplificar la detección de la causa raíz. El rastreo integral reduce el tiempo necesario para descubrir los componentes afectados y analizar detalladamente las causas raíz de los errores o la latencia.

Patrones comunes de uso no recomendados:

- El rastreo se utiliza para algunos componentes, pero no para todos. Por ejemplo, si no se rastreara AWS Lambda, es posible que los equipos no entendieran con claridad la latencia que producen los arranques en frío en una carga de trabajo con picos.
- Los valores controlados sintéticos o la monitorización de usuarios reales (RUM) no tienen configurado el rastreo. Sin valores controlados ni RUM, la telemetría de interacción con el cliente se omite del análisis del rastreo, lo que da lugar a un perfil de rendimiento incompleto.
- Las cargas de trabajo híbridas incluyen herramientas de rastreo nativas en la nube y de terceros, pero no se han tomado medidas para integrar por completo una única solución de rastreo. En función de la solución de rastreo elegida, se deben utilizar SDK de rastreo nativos en la nube para instrumentar componentes que no sean nativos en la nube o se deben configurar herramientas de terceros para ingerir la telemetría de rastreo nativa en la nube.

Beneficios de establecer esta práctica recomendada: Cuando los equipos de desarrollo reciben alertas sobre los problemas, ven una imagen completa de las interacciones entre los componentes del sistema, incluida la correlación componente por componente con el registro, el rendimiento y los errores. Dado que el rastreo facilita la identificación visual de las causas raíz, se dedica menos tiempo a investigar estas causas. Los equipos que conocen bien las interacciones de los componentes toman decisiones mejores y más rápidas a la hora de resolver problemas. Las decisiones, como cuándo invocar una conmutación por error de recuperación de desastres (DR) o cuál es la mejor forma de implementar las estrategias de autorreparación, se pueden mejorar analizando los rastros de los sistemas y, en última instancia, puede mejorar la satisfacción del cliente con sus servicios.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio

## Guía para la implementación

Los equipos que utilizan aplicaciones distribuidas pueden utilizar herramientas de rastreo para establecer un identificador de correlación, recopilar rastros de las solicitudes y crear mapas de servicio de los componentes conectados. Todos los componentes de la aplicación deben incluirse en los rastros de solicitudes, como las puertas de enlace de middleware, los buses de eventos y los clientes del servicio, los componentes de computación y el almacenamiento, incluidos los almacenes de valores clave y las bases de datos. Incluya valores controlados sintéticos y la monitorización de usuarios reales en su configuración de rastreo integral para medir las interacciones y la latencia de los clientes remotos, de modo que pueda evaluar con precisión el rendimiento de sus sistemas en función de sus acuerdos y objetivos de nivel de servicio.

Puede usar los servicios de instrumentación de monitorización de aplicaciones AWS X-Ray y Amazon CloudWatch para ofrecer una visión completa de las solicitudes a medida que pasan por su aplicación. X-Ray recopila la telemetría de las aplicaciones y le permite visualizarla y filtrarla en cargas útiles, funciones, rastros, servicios y API, y se puede activar para los componentes del sistema sin código o con poco código. La monitorización de aplicaciones de CloudWatch incluye ServiceLens para integrar sus rastros con métricas, registros y alarmas. La monitorización de aplicaciones de CloudWatch también incluye elementos sintéticos para monitorizar los puntos de conexión y las API, así como la monitorización de usuarios reales para instrumentar los clientes de sus aplicaciones web.

## Pasos para la implementación

- Use AWS X-Ray en todos los servicios nativos admitidos, como Amazon S3, AWS Lambda y Amazon API Gateway. Estos servicios de AWS habilitan X-Ray con conmutadores de

configuración que utilizan la infraestructura como código, SDK de AWS o la AWS Management Console.

- Aplicaciones de instrumentos AWS Distro for Open Telemetry y X-Ray o agentes de recopilación de terceros.
- Revise en la Guía para desarrolladores de AWS X-Ray la implementación específica del lenguaje de programación. En estas secciones de la documentación, se detalla cómo instrumentar las solicitudes HTTP, las consultas SQL y otros procesos específicos del lenguaje de programación de su aplicación.
- Utilice el rastreo de X-Ray para los valores controlados de Amazon CloudWatch Synthetics y Amazon CloudWatch RUM para analizar la ruta de la solicitud desde el cliente de su usuario final a través de su infraestructura posterior de AWS.
- Configure métricas y alarmas de CloudWatch en función del estado de los recursos y la telemetría de valores controlados para que los equipos reciban alertas de los problemas rápidamente y, a continuación, puedan analizar en profundidad los rastros y los mapas de servicio con ServiceLens.
- Habilite la integración de X-Ray con herramientas de rastreo de terceros, como Datadog, New Relico Dynatrace si utiliza herramientas de terceros para su solución de rastreo principal.

## Recursos

Prácticas recomendadas relacionadas:

- REL06-BP01 Supervisar todos los componentes de la carga de trabajo (generación)
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores

Documentos relacionados:

- ¿Qué es AWS X-Ray?
- Amazon CloudWatch: Application Monitoring (Monitorización de aplicaciones)
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray (Depuración con Amazon CloudWatch Synthetics y AWS X-Ray)
- La Amazon Builders' Library: Instrumentación de los sistemas distribuidos para la visibilidad de las operaciones
- Integrating AWS X-Ray with other AWS services (Integración de AWS X-Ray con otros servicios de AWS)

- AWS Distro for OpenTelemetry and AWS X-Ray (AWS Distro for OpenTelemetry y AWS X-Ray)
- Amazon CloudWatch: Using synthetic monitoring (Uso de la monitorización sintética)
- Amazon CloudWatch: Use CloudWatch RUM (Uso de CloudWatch RUM)
- Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm (Configuración del valor controlado en Amazon CloudWatch Synthetics y la alarma de Amazon CloudWatch)
- Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS (Disponibilidad y más allá: comprender y mejorar la resiliencia de sistemas distribuidos en AWS)

Ejemplos relacionados:

- Taller sobre observabilidad

Vídeos relacionados:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts (Cómo monitorizar aplicaciones en varias cuentas)
- How to Monitor your AWS Applications (Cómo monitorizar sus aplicaciones de AWS)

Herramientas relacionadas:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

## Diseñar su carga de trabajo para que se adapte a los cambios en la demanda

Una carga de trabajo escalable proporciona elasticidad para agregar o eliminar recursos automáticamente a fin de que se ajusten todo lo posible a la demanda en cualquier momento dado.

Prácticas recomendadas

- REL07-BP01 Usar la automatización al obtener o escalar recursos
- REL07-BP02 Obtener recursos tras detectar un impedimento en una carga de trabajo

- REL07-BP03 Obtener recursos tras detectar que se necesitan más recursos para una carga de trabajo
- REL07-BP04 Realizar pruebas de su carga de trabajo

## REL07-BP01 Usar la automatización al obtener o escalar recursos

Cuando reemplace recursos deteriorados o escale su carga de trabajo, automatice el proceso mediante el uso de servicios de AWS administrados, como Amazon S3 y AWS Auto Scaling. También puede utilizar herramientas de terceros y los SDK de AWS para automatizar el escalado.

Los servicios de AWS administrados incluyen Amazon S3, Amazon CloudFront, AWS Auto Scaling, AWS Lambda, Amazon DynamoDB, AWS Fargate y Amazon Route 53.

AWS Auto Scaling le permite detectar y reemplazar las instancias deterioradas. También le permite crear planes de escalado para los recursos, como instancias Amazon EC2 y flotas de spot, tareas de Amazon ECS , tablas e índices de Amazon DynamoDB y réplicas de Amazon Aurora .

Al escalar las instancias EC2, asegúrese de que utiliza varias zonas de disponibilidad (preferiblemente tres como mínimo) y agregue o elimine capacidad para mantener el equilibrio entre estas zonas de disponibilidad. Las tareas de ECS o los pods de Kubernetes (cuando se utiliza Amazon Elastic Kubernetes Service) también se deben distribuir en varias zonas de disponibilidad.

Al utilizar AWS Lambda, las instancias se escalan automáticamente. Cada vez que se recibe una notificación de evento para su función, AWS Lambda localiza rápidamente la capacidad libre en su flota de computación y ejecuta su código hasta la simultaneidad asignada. Debe asegurarse de que la simultaneidad necesaria está configurada en Lambda específico y en su Service Quotas.

Amazon S3 se escala automáticamente para gestionar las altas tasas de solicitudes. Por ejemplo, su aplicación puede alcanzar al menos 3500 solicitudes PUT/COPY/POST/DELETE o 5500 solicitudes GET/HEAD por segundo y prefijo en un bucket. No hay límites en el número de prefijos de un bucket. Puede aumentar su rendimiento de lectura o escritura si ejecuta en paralelo las lecturas. Por ejemplo, si crea diez prefijos en un bucket de Amazon S3 para ejecutar en paralelo las lecturas, podría escalar su rendimiento de lectura a 55 000 solicitudes de lectura por segundo.

Configure y use Amazon CloudFront o una red de entrega de contenido (CDN) de confianza. Una CDN puede proporcionar tiempos de respuesta más rápidos al usuario final y puede servir solicitudes de contenido desde la caché, lo que reduce la necesidad de escalar su carga de trabajo.

Patrones de uso no recomendados comunes:

- Implementar grupos de escalado automático para la reparación automatizada, pero no implementar la elasticidad.
- Utilizar el escalado automático para responder a los grandes aumentos de tráfico.
- Desplegar aplicaciones con un alto nivel de estado, lo que elimina la opción de la elasticidad.

Beneficios de establecer esta práctica recomendada: la automatización elimina la posibilidad de cometer errores manuales al desplegar y retirar los recursos. La automatización elimina el riesgo de sobrecostes y de denegación de servicio debido a la lentitud de respuesta en las necesidades de despliegue o retirada.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

- Configure y use AWS Auto Scaling. Esto supervisa sus aplicaciones y ajusta automáticamente la capacidad para mantener un rendimiento constante y predecible al menor coste posible. Con AWS Auto Scaling, puede configurar el escalado de aplicaciones para múltiples recursos en varios servicios.
  - ¿Qué es AWS Auto Scaling?
    - Configure el escalado automático en sus instancias y flotas de spot de Amazon EC2, tareas de Amazon ECS, tablas e índices de Amazon DynamoDB, réplicas de Amazon Aurora y dispositivos de AWS Marketplace según corresponda.
      - Administración automática de la capacidad de rendimiento con el escalado automático de DynamoDB
        - Use las operaciones de la API de servicio para especificar las alarmas, las políticas de escalado, los tiempos de calentamiento y los tiempos de enfriamiento.
- Use Elastic Load Balancing. Los equilibradores de carga pueden distribuir la carga por ruta o por conectividad de red.
  - ¿Qué es Elastic Load Balancing?
    - Application Load Balancers puede distribuir la carga por ruta.
      - ¿Qué es un Application Load Balancer?
        - Configure un Application Load Balancer para distribuir el tráfico entre diferentes cargas de trabajo en función de la ruta que corresponde al nombre de dominio.
        - Los Application Load Balancers se pueden utilizar para distribuir las cargas de manera que se integren con AWS Auto Scaling para administrar la demanda.

        - Usar un equilibrador de carga con un grupo de escalado automático
    - Los equilibradores de carga de red pueden distribuir la carga por conexión.
        - ¿Qué es un equilibrador de carga de red?
            - Configure un equilibrador de carga de red para distribuir el tráfico entre diferentes cargas de trabajo mediante TCP o para tener un conjunto constante de direcciones IP para la carga de trabajo.
            - Los equilibradores de carga de red se pueden utilizar para distribuir la carga de manera que se integre con AWS Auto Scaling para administrar la demanda.
- Use un proveedor de DNS de alta disponibilidad. Los nombres DNS permiten a sus usuarios acceder a sus cargas de trabajo con nombres en lugar de direcciones IP. Esta información se distribuye en un ámbito definido, normalmente de forma global para los usuarios de la carga de trabajo.
    - Utilice Amazon Route 53 o un proveedor de DNS de confianza.
        - ¿Qué es Amazon Route 53?
    - Use Route 53 para administrar las distribuciones de CloudFront y los equilibradores de carga.
        - Determine los dominios y subdominios que va a administrar.
        - Cree conjuntos de registros apropiados mediante registros ALIAS o CNAME.
            - Trabajar con los registros
- Utilice la red global de AWS para optimizar la ruta desde sus usuarios a sus aplicaciones. AWS Global Accelerator supervisa continuamente el estado de los puntos de conexión de sus aplicaciones y redirige el tráfico a los puntos en estado correcto en menos de 30 segundos.
    - AWS Global Accelerator es un servicio que mejora la disponibilidad y el rendimiento de sus aplicaciones con usuarios locales o globales. Proporciona direcciones IP estáticas que actúan como punto de entrada fijo a los puntos de conexión de aplicaciones en una o varias Regiones de AWS, como sus Application Load Balancers, equilibradores de carga de red o instancias Amazon EC2.
        - ¿Qué es AWS Global Accelerator?
- Configure y use Amazon CloudFront o una red de entrega de contenido (CDN) de confianza. Una red de entrega de contenido puede ofrecer tiempos de respuesta más rápidos a los usuarios finales y atender solicitudes de contenido que pueden provocar un escalado innecesario de las cargas de trabajo.
    - ¿Qué es Amazon CloudFront?

- Configure distribuciones de Amazon CloudFront para sus cargas de trabajo o utilice una CDN de terceros.
  - Puede limitar el acceso a sus cargas de trabajo para que solo sean accesibles desde CloudFront mediante los intervalos de IP para CloudFront en sus grupos de seguridad de puntos de conexión o políticas de acceso.

### Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudarle a crear soluciones informáticas automatizadas
- AWS Auto Scaling: cómo funcionan los planes de escalado
- AWS Marketplace: productos que pueden usarse con escalo automático
- Administración automática de la capacidad de rendimiento con el escalado automático de DynamoDB
- Usar un equilibrador de carga con un grupo de escalado automático
- ¿Qué es AWS Global Accelerator?
- ¿Qué es Amazon EC2 Auto Scaling?
- ¿Qué es AWS Auto Scaling?
- ¿Qué es Amazon CloudFront?
- ¿Qué es Amazon Route 53?
- ¿Qué es Elastic Load Balancing?
- ¿Qué es un equilibrador de carga de red?
- ¿Qué es un Application Load Balancer?
- Trabajar con los registros

## REL07-BP02 Obtener recursos tras detectar un impedimento en una carga de trabajo

Escale recursos de forma retroactiva cuando sea necesario si la disponibilidad se ve afectada para restaurar la disponibilidad de la carga de trabajo.

Primero debe configurar las comprobaciones de estado y los criterios de dichas comprobaciones para indicar cuándo se ve afectada la disponibilidad por falta de recursos. A continuación, notifique al

personal pertinente para que escale manualmente el recurso o inicie la automatización, a fin de que el escalamiento se realice de forma automática.

La escala puede ajustarse manualmente para su carga de trabajo (por ejemplo, se puede cambiar el número de instancias de EC2 en un grupo de Auto Scaling o se puede modificar el rendimiento de una tabla de DynamoDB mediante la AWS Management Console o la AWS CLI). Sin embargo, la automatización debe usarse siempre que sea posible (consulte «Usar la automatización al obtener o escalar recursos»).

Resultado deseado: se inician las actividades de escalamiento (de forma automática o manual) para restablecer la disponibilidad al detectar un error o una experiencia del cliente degradada.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Implemente la observabilidad y la supervisión en todos los componentes de su carga de trabajo para supervisar la experiencia del cliente y detectar errores. Defina los procedimientos, manuales o automatizados, que escalan los recursos necesarios. Para obtener más información, consulte «REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores».

Pasos para la implementación

- Defina los procedimientos, manuales o automatizados, que escalan los recursos requeridos.
  - Los procedimientos de escalamiento dependen de cómo estén diseñados los distintos componentes de la carga de trabajo.
  - Estos procedimientos también varían según la tecnología subyacente que se utilice.
    - Los componentes que utilizan AWS Auto Scaling pueden usar planes de escalamiento para configurar un conjunto de instrucciones para escalar los recursos. Si trabaja con AWS CloudFormation o añade etiquetas a recursos de AWS, puede configurar planes de escalamiento para diferentes conjuntos de recursos por aplicación. Auto Scaling proporciona recomendaciones de estrategias de escalamiento personalizadas para cada recurso. Tras crear su plan de ajuste de escalamiento, Auto Scaling combina el escalamiento dinámico y los métodos de escalamiento predictivos para facilitar su estrategia de escalamiento. Para obtener más información, consulte «Cómo funcionan los planes de escalado».
    - Amazon EC2 Auto Scaling verifica que tiene el número correcto de instancias de Amazon EC2 disponibles para gestionar la carga de la aplicación. Debe crear colecciones de instancias de EC2, denominadas grupos de Auto Scaling. Puede especificar el número mínimo y máximo de

instancias en cada grupo de Auto Scaling y Amazon EC2 Auto Scaling garantiza que su grupo nunca tenga un tamaño por encima o por debajo de este límite. Para obtener más información, consulte «¿Qué es Amazon EC2 Auto Scaling?»

- El escalamiento automático de Amazon DynamoDB usa el servicio Application Auto Scaling para ajustar dinámicamente la capacidad de rendimiento aprovisionada en su nombre como respuesta a los patrones de tráfico reales. Esto permite que una tabla o un índice secundario global aumente su capacidad de lectura y escritura aprovisionada para afrontar los picos repentinos de tráfico sin limitación. Para obtener más información, consulte «Administración automática de la capacidad de rendimiento con la función Auto Scaling de DynamoDB».

### Recursos

Prácticas recomendadas relacionadas:

- «REL07-BP01 Usar la automatización al obtener o escalar recursos»
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores

Documentos relacionados:

- «AWS Auto Scaling: Cómo funcionan los planes de escalado»
- «Administración automática de la capacidad de rendimiento con la función Auto Scaling de DynamoDB»
- What Is Amazon EC2 Auto Scaling? (¿Qué es Amazon EC2 Auto Scaling?)

## REL07-BP03 Obtener recursos tras detectar que se necesitan más recursos para una carga de trabajo

Escale recursos de forma proactiva para satisfacer la demanda y evitar que la disponibilidad se vea impactada.

Muchos servicios de AWS se escalan automáticamente para satisfacer la demanda. Si usa instancias de Amazon EC2 o clústeres de Amazon ECS, puede configurar su escalado automático para que se lleve a cabo en función de métricas de uso que se correspondan con la demanda para su carga de trabajo. Para Amazon EC2, el uso medio de la CPU, el recuento de solicitudes al equilibrador de carga o el ancho de banda de la red se pueden usar para escalar (o desescalar) horizontalmente instancias de EC2. Para Amazon ECS, el uso medio de la CPU, el recuento de

solicitudes al equilibrador de carga o el uso de memoria se pueden usar para escalar (o desescalar) horizontalmente tareas de ECS. Al utilizar el escalado automático por objetivos en AWS, el escalador automático actúa como un termostato doméstico y agrega o retira recursos para mantener el valor objetivo (por ejemplo, un uso de la CPU del 70 %) que haya especificado.

AWS Auto Scaling también puede llevar a cabo escalado automático predictivo, que utiliza machine learning para analizar la carga de trabajo histórica de cada recurso y predice regularmente la carga futura para los próximos dos días.

La ley de Little ayuda a calcular cuántas instancias de computación (instancias de EC2, funciones Lambda simultáneas, etc.) necesitará.

$R = \lambda W$

L = número de instancias (o simultaneidad media en el sistema)

λ = promedio de la tasa de llegada de solicitudes (solicitudes/s)

W = promedio del tiempo que pasa cada solicitud en el sistema (s)

Por ejemplo, a 100 sps, si cada solicitud tarda 0,5 segundos en procesarse, necesitará 50 instancias para satisfacer la demanda.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

## Guía para la implementación

- Obtenga recursos tras detectar que se necesitan más recursos para una carga de trabajo. Escale recursos de forma proactiva para satisfacer la demanda y evitar que la disponibilidad se vea impactada.
  - Calcule cuántos recursos de computación necesitará (simultaneidad de computación) para afrontar una tasa de solicitudes dada.
    - Presentación de historias sobre la ley de Little
  - Cuando tenga un patrón de uso histórico, configure el escalado programado para el escalado automático de Amazon EC2.
    - Escalado programado para Amazon EC2 Auto Scaling
  - Use el escalado predictivo de AWS.
    - Predictive Scaling for EC2, Powered by Machine Learning

### Recursos

Documentos relacionados:

- AWS Auto Scaling: cómo funcionan los planes de escalado
- AWS Marketplace: productos que pueden usarse con Auto Scaling
- Administrar automáticamente la capacidad de rendimiento con Auto Scaling de DynamoDB
- Predictive Scaling for EC2, Powered by Machine Learning
- Escalado programado para Amazon EC2 Auto Scaling
- Presentación de historias sobre la ley de Little
- ¿Qué es Amazon EC2 Auto Scaling?

## REL07-BP04 Realizar pruebas de su carga de trabajo

Adopte una metodología de prueba de carga para medir si la actividad de escalado satisface los requisitos de la carga de trabajo.

Es importante realizar pruebas de carga sostenidas. Las pruebas de carga deben descubrir el punto de ruptura y probar el rendimiento de su carga de trabajo. AWS facilita la creación de entornos de prueba temporales que modelan la escala de su carga de trabajo de producción. En la nube, puede crear un entorno de prueba a escala de producción, completar sus pruebas y desmantelar los recursos. Debido a que solo paga por el entorno de prueba cuando se ejecuta, puede simular su entorno en directo por una fracción del coste de las pruebas en las instalaciones.

Las pruebas de carga en producción también deben considerarse como parte de los días de juego en los que se estresa el sistema de producción, durante las horas de menor uso por parte de los clientes, con todo el personal a mano para interpretar los resultados y abordar cualquier problema que surja.

Patrones comunes de uso no recomendados:

- Realizar pruebas de carga en implementaciones que no tienen la misma configuración que su producción.
- Realizar pruebas de carga solo en elementos individuales de su carga de trabajo y no en toda la carga.
- Realizar pruebas de carga con un subconjunto de solicitudes y no con un conjunto representativo de solicitudes reales.

- Realizar pruebas de carga con un pequeño factor de seguridad por encima de la carga prevista.

Beneficios de establecer esta práctica recomendada: Sabrá qué componentes de su arquitectura presentan errores bajo carga y podrá identificar qué métricas se deben vigilar para indicar que se está acercando a esa carga a tiempo para solucionar el problema, con lo que se evitará el impacto de ese error.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio

## Guía para la implementación

- Realice pruebas de carga para identificar qué aspecto de su carga de trabajo indica que debe agregar o eliminar capacidad. Las pruebas de carga deben tener un tráfico representativo similar al que se recibe en producción. Aumente la carga mientras vigila las métricas que ha instrumentado para determinar qué métrica indica cuándo debe agregar o eliminar recursos.
  - Pruebas de carga distribuida en AWS: simular miles de usuarios conectados
    - Identifique la combinación de solicitudes. Es posible que tenga una combinación variada de solicitudes, por lo que deberá tener en cuenta diversos periodos de tiempo a la hora de identificar la combinación de tráfico.
    - Implemente un controlador de carga. Puede utilizar software de código personalizado, de código abierto o comercial para implementar un controlador de carga.
    - Realice la prueba de carga inicialmente con una capacidad pequeña. Ve algunos efectos inmediatos al pasar la carga a una capacidad menor, posiblemente tan pequeña como una instancia o un contenedor.
    - Realice una prueba de carga con una capacidad mayor. Los efectos serán diferentes en una carga distribuida, por lo que debe realizar las pruebas en un entorno lo más parecido al del producto.

## Recursos

Documentos relacionados:

- Pruebas de carga distribuida en AWS: simular miles de usuarios conectados
- Aplicaciones de pruebas de carga

Vídeos relacionados:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing

# Implementar cambios

Los cambios controlados son necesarios para implementar nuevas funcionalidades y garantizar que las cargas de trabajo y el entorno operativo ejecuten software conocido y con los parches correspondientes. Si estos cambios no se controlan, puede ser difícil prever su efecto o abordar los problemas que surjan a raíz de ellos.

Prácticas recomendadas

- REL08-BP01 Usar runbooks para actividades estándares como la implementación
- REL08-BP02 Integrar las pruebas funcionales como parte de su despliegue
- REL08-BP03 Integrar las pruebas de resiliencia como parte de su despliegue
- REL08-BP04 Desplegar mediante una infraestructura inmutable
- REL08-BP05 Desplegar cambios con automatización

## REL08-BP01 Usar runbooks para actividades estándares como la implementación

Los runbooks son procedimientos predefinidos para obtener resultados concretos. Use runbooks para realizar actividades estándar manuales o automáticas. Algunos ejemplos incluyen implementar una carga de trabajo, aplicarle un parche a dicha carga de trabajo o realizar modificaciones de DNS.

Por ejemplo, se pueden implementar procesos para garantizar la seguridad de la restauración durante los despliegues. Tener la garantía de poder dar marcha atrás en un despliegue sin disrupciones para sus clientes es esencial a la hora de hacer que un servicio sea fiable.

Para los procedimientos de runbooks, empiece por un proceso manual efectivo válido, impleméntelo en el código y desencadene la ejecución automatizada cuando sea oportuno.

Incluso en el caso de cargas de trabajo sofisticadas con un alto nivel de automatización los runbooks siguen siendo útiles para ejecutar días de juego o ajustarse a los exhaustivos requisitos de presentación de informes y auditoría.

Tenga en cuenta que las guías de estrategias se usan en respuesta a incidentes específicos y los runbooks se usan para conseguir resultados determinados. A menudo, los runbooks se usan para

actividades rutinarias, mientras que las guías de estrategias se utilizan para responder a eventos no rutinarios.

Patrones de uso no recomendados comunes:

- Realizar cambios imprevistos en la configuración en producción
- Omitir pasos del plan para realizar una implementación más rápida, lo que da lugar a una implementación errónea.
- Realizar cambios sin probar la revocación del cambio

Beneficios de establecer esta práctica recomendada: La planificación eficaz de los cambios aumenta su capacidad de realizar correctamente el cambio, ya que sabrá qué sistemas resultarán afectados. Validar el cambio en los entornos de prueba aumenta la confianza.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

- Puede obtener respuestas sistemáticas e inmediatas a eventos conocidos si documenta los procedimientos en runbooks.
  - Conceptos del AWS Well-Architected Framework: runbook
- Use el principio de la infraestructura como código para definir la infraestructura. Si usa AWS CloudFormation (o un tercero de confianza) para definir su infraestructura, puede utilizar software de control de versiones para crear versiones y hacer seguimiento de los cambios.
  - Use AWS CloudFormation (o un proveedor tercero de confianza) para definir su infraestructura.
    - ¿Qué es AWS CloudFormation?
  - Cree plantillas singulares y desacopladas, usando buenos principios de diseño de software.
    - Determine los permisos, las plantillas y las partes responsables de su implementación.
      - Control de acceso con AWS Identity and Access Management
    - Use herramientas de control de código, como AWS CodeCommit o las de terceros de confianza, para llevar un control de las versiones.
      - ¿Qué es AWS CodeCommit?

## Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudarle a crear soluciones de implementación automatizadas
- AWS Marketplace: productos que pueden usarse para automatizar sus implementaciones
- Conceptos del AWS Well-Architected Framework: runbook
- ¿Qué es AWS CloudFormation?
- ¿Qué es AWS CodeCommit?

Ejemplos relacionados:

- Automatización de operaciones con guías de estrategias y runbooks

## REL08-BP02 Integrar las pruebas funcionales como parte de su despliegue

Las pruebas funcionales se ejecutan como parte de la implementación automatizada. Si no se satisfacen los criterios de éxito, la canalización se detiene o se revierte. Estas pruebas realizan en un entorno de preproducción, que se lleva a cabo de forma escalonada antes de la producción en la canalización. Idealmente, esto se realiza como parte de una canalización de despliegue.

Resultado deseado: utiliza la automatización para realizar pruebas funcionales y los datos de prueba asociados reducen la duración y los gastos de las pruebas y mejoran la precisión de sus resultados. Integra las pruebas funcionales como parte de su proceso de despliegue, lo que le ayuda a automatizar sus canalizaciones de lanzamiento para obtener actualizaciones rápidas y fiables de las aplicaciones y la infraestructura.

Antipatrones usuales:

- Las pruebas se realizan manualmente fuera de la canalización del despliegue.
- Omite los pasos de prueba de la automatización mediante flujos de trabajo de emergencia manuales.
- No sigue sus planes y procesos de prueba establecidos a favor de plazos más rápidos.

Beneficios de establecer esta práctica recomendada:  las pruebas funcionales validan que el sistema funciona de acuerdo con los requisitos especificados. Se usa para verificar de manera sistemática el orden de funcionamiento previsto de los componentes, como las interfaces de usuario, las API, las bases de datos y el código fuente. Al examinar estos componentes del sistema, las pruebas funcionales verifican que cada característica se comporta según lo esperado, lo que garantiza tanto las expectativas del usuario como la integridad del software. Integra las pruebas funcionales como

parte de su despliegue habitual y utiliza la automatización para desplegar todos los cambios, lo que reduce la posibilidad de que se produzcan errores humanos.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

Integre las pruebas funcionales como parte de su despliegue. Las pruebas funcionales se ejecutan como parte de la implementación automatizada. Si no se cumplen los criterios de éxito, la canalización se detiene o se revierte. AWS CodePipeline proporciona una canalización de entrega continua para las pruebas automatizadas, lo que permite a los evaluadores automatizar todo el proceso de prueba y despliegue. Se integra con servicios de AWS como AWS CodeBuild y AWS CodeDeploy para automatizar las fases de creación, prueba y despliegue del ciclo de vida del desarrollo de software.

Pasos para la implementación

- Configure su canalización: configure las etapas de origen, compilación, prueba y despliegue mediante la consola de AWS CodePipeline o la CLI de AWS Command Line Interface.
  - Defina su código fuente: con AWS CodePipeline, puede recuperar automáticamente el código fuente de sistemas de control de versiones como GitHub, AWS CodeCommit o Bitbucket, lo que garantiza que siempre se use el código más reciente para las pruebas.
  - Automatice las compilaciones y las pruebas:  AWS CodeBuild puede compilar y probar automáticamente su código y generar informes de prueba. Es compatible con marcos de prueba populares como JUnit, NUnit y TestNG.
  - Despliegue su código: una vez que el código se haya creado y probado, AWS CodeDeploy puede desplegarlo en su entorno de prueba, incluso en las instancias de Amazon EC2, las funciones de AWS Lambda o los servidores locales.
  - Supervise las canalizaciones:  AWS CodePipeline puede realizar un seguimiento del progreso de su canalización y el estado de cada etapa. También puede usar controles de calidad para bloquear la canalización según el estado de ejecución de la prueba. También puede recibir notificaciones cuando se produzca un fallo en una etapa de canalización o cuando se haya completado la canalización.

## Recursos

Documentos relacionados:

- «Use AWS CodePipeline with AWS CodeBuild to test code and run builds»
- «Logging and monitoring in AWS CodeBuild»
- «Indicators for functional testing»

## REL08-BP03 Integrar las pruebas de resiliencia como parte de su despliegue

Integre las pruebas de resiliencia introduciendo fallos en el sistema de forma intencionada para medir su capacidad en caso de situaciones disruptivas. Las pruebas de resiliencia son diferentes de las pruebas unitarias y funcionales que suelen estar integradas en los ciclos de despliegue, ya que se centran en la identificación de fallos imprevistos en el sistema. Aunque es seguro comenzar con la integración de pruebas de resiliencia en la fase de preproducción, fíjese el objetivo de implementar estas pruebas en producción como parte de sus GameDays.

Resultado deseado: las pruebas de resiliencia ayudan a aumentar la confianza en la capacidad del sistema para resistir la degradación en producción. Los experimentos identifican los puntos débiles que podrían provocar fallos, lo que le ayuda a mejorar su sistema para mitigar de manera automática y eficiente los fallos y la degradación.

Antipatrones usuales:

- Falta de observabilidad y supervisión en los procesos de despliegue
- Confianza en los seres humanos para resolver fallos del sistema
- Mecanismos de análisis de mala calidad
- Centrarse en los problemas conocidos de un sistema y falta de experimentación para identificar cualquier elemento desconocido
- Identificación de fallos, pero sin ninguna resolución
- No hay documentación de los resultados ni runbooks

Beneficios de establecer las prácticas recomendadas:  las pruebas de resiliencia integradas en sus despliegues ayudan a identificar problemas desconocidos en el sistema que, de otro modo, pasarían desapercibidos y podrían provocar tiempos de inactividad en la producción. La identificación de estos elementos desconocidos en un sistema le ayuda a documentar los resultados, integrar las pruebas en su proceso de CI/CD y crear runbooks, lo que simplifica la mitigación mediante mecanismos eficientes y repetibles.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Los formularios de pruebas de resiliencia más comunes que se pueden integrar en los despliegues de su sistema son la recuperación ante desastres y la ingeniería del caos.

- Incluya actualizaciones en sus planes de recuperación ante desastres y procedimientos operativos estándar (SOP) en cualquier despliegue importante.
- Integre las pruebas de fiabilidad en sus canalizaciones de despliegue automatizadas. Los servicios como AWS Resilience Hub se pueden integrar en su canalización de CI/CD para establecer evaluaciones de resiliencia continuas que se realicen automáticamente como parte de cada despliegue.
- Defina sus aplicaciones en AWS Resilience Hub. Las evaluaciones de resiliencia generan fragmentos de código que le ayudan a crear procedimientos de recuperación como documentos de AWS Systems Manager para sus aplicaciones y proporcionan una lista de monitores y alarmas recomendados de Amazon CloudWatch.
- Una vez actualizados los planes de recuperación ante desastres y los procedimientos operativos estándar, realice las pruebas de recuperación ante desastres para verificar que son efectivos. Las pruebas de recuperación ante desastres le ayudan a determinar si puede restaurar el sistema después de un evento y recuperar el funcionamiento normal. Puede simular varias estrategias de recuperación ante desastres e identificar si su planificación es suficiente para cumplir sus requisitos de tiempo de actividad. Las estrategias comunes de recuperación ante desastres incluyen copias de seguridad y restauración, el enfoque de luz piloto, la espera en frío, la espera semiactiva, la espera activa y la estrategia activa-activa, y todas difieren en coste y complejidad. Antes de realizar las pruebas de recuperación ante desastres, le recomendamos que defina su objetivo de tiempo de recuperación (RTO) y su objetivo de punto de recuperación (RPO) para simplificar la elección de la estrategia de simulación. AWS ofrece herramientas de recuperación ante desastres como AWS Elastic Disaster Recovery que ayudan a comenzar con la planificación y las pruebas.
- Los experimentos de ingeniería del caos introducen disrupciones en el sistema, como cortes de la red y fallos del servicio. Al realizar simulaciones con fallos controlados, puede descubrir las vulnerabilidades de su sistema y, al mismo tiempo, contener la repercusión de los fallos inyectados. Al igual que en las demás estrategias, ejecute simulaciones de fallos controlados en entornos que no sean de producción utilizando servicios como AWS Fault Injection Service para ganar confianza antes de desplegarlos en producción.

### Recursos

Documentos relacionados:

- «Experiment with failure using resilience testing to build recovery preparedness»
- «Continually assessing application resilience with AWS Resilience Hub and AWS CodePipeline»
- «Disaster recovery (DR) architecture on AWS, part 1: Strategies for recovery in the cloud»
- «Verify the resilience of your workloads using Chaos Engineering»
- Principios de la ingeniería del caos
- «Chaos Engineering Workshop»

Vídeos relacionados:

- «AWS re:Invent 2020: Testing Resilience using Chaos Engineering»
- «Improve Application Resilience with AWS Fault Injection Service»
- «Prepare & Protect Your Applications From Disruption With AWS Resilience Hub»

## REL08-BP04 Desplegar mediante una infraestructura inmutable

La infraestructura inmutable es un modelo que exige que no haya actualizaciones, parches de seguridad ni cambios de configuración en las cargas de trabajo de producción. Cuando es necesario realizar un cambio, la arquitectura se integra en una nueva infraestructura y se implementa en producción.

Utilice una estrategia de despliegue de infraestructura inmutable para aumentar la fiabilidad, la coherencia y la reproducibilidad de los despliegues de sus cargas de trabajo.

Resultado deseado: con una infraestructura inmutable, no está permitido realizar modificaciones in situ para ejecutar los recursos de infraestructura dentro de una carga de trabajo. En su lugar, cuando es necesario realizar un cambio, se despliega en paralelo un nuevo conjunto de recursos de infraestructura actualizados que contienen todos los cambios que es necesario realizar en los recursos existentes. Este despliegue se valida automáticamente y, si se realiza correctamente, el tráfico se desplaza gradualmente al nuevo conjunto de recursos.

Esta estrategia de despliegue se aplica a las actualizaciones de software, los parches de seguridad, los cambios de infraestructura, las actualizaciones de la configuración y las actualizaciones de las aplicaciones, entre otros.

Antipatrones usuales:

- Implementar cambios in situ en los recursos de infraestructura en ejecución.

Beneficios de establecer esta práctica recomendada:

- Mayor coherencia entre todos los entornos: dado que no hay diferencias en los recursos de infraestructura entre los entornos, se aumenta la coherencia y se simplifican las pruebas.
- Reducción de las desviaciones de la configuración: al reemplazar los recursos de la infraestructura por una configuración conocida y controlada por versiones, la infraestructura se establece en un estado conocido, probado y fiable, lo que evita desviaciones de la configuración.
- Despliegues atómicos fiables: los despliegues se realizan correctamente o no cambia nada, lo que aumenta la coherencia y la fiabilidad en el proceso de despliegue.
- Despliegues simplificados: los despliegues se simplifican porque no tienen que ser compatibles con las mejoras. Las mejoras son simplemente nuevos despliegues.
- Despliegues más seguros con procesos de recuperación y restauración rápidos: los despliegues son más seguros porque la versión activa anterior no se cambia. Puede restaurarla si se detecta algún error.
- Postura de seguridad mejorada: al no permitir cambios en la infraestructura, es posible deshabilitar los mecanismos de acceso remoto (como SSH). Esto reduce el vector de ataque y mejora la postura de seguridad de su organización.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

### Automatización

A la hora de definir una estrategia de despliegue de infraestructura inmutable, se recomienda utilizar la automatización en la medida de lo posible para aumentar la reproducibilidad y minimizar la posibilidad de que se cometan errores humanos. Para obtener más información, consulte «REL08-BP05 Desplegar cambios con automatización» y «Automatización de implementaciones seguras y sin intervención».

Con la infraestructura como código (IaC), los pasos de aprovisionamiento, orquestación y despliegue de la infraestructura se definen de forma programática, descriptiva y declarativa, y se almacenan en un sistema de control del código fuente. El uso de la infraestructura como código simplifica

la automatización del despliegue de la infraestructura y ayuda a lograr la inmutabilidad de la infraestructura.

Patrones de despliegue

Cuando es necesario realizar un cambio en la carga de trabajo, la estrategia inmutable de despliegue de la infraestructura requiere el despliegue de un nuevo conjunto de recursos de infraestructura que incluya todos los cambios necesarios. Es importante que este nuevo conjunto de recursos siga un patrón de despliegue que minimice la repercusión en los usuarios. Hay dos estrategias principales para este despliegue:

Despliegue de valores controlados: es la práctica de dirigir a una pequeña cantidad de los clientes a la nueva versión, que normalmente se ejecuta en una instancia de servicio único (la de valor controlado). A continuación, puede analizar en profundidad los errores o los cambios en el comportamiento que se hayan generado. Puede eliminar el tráfico del valor controlado si encuentra problemas críticos y enviar a los usuarios de vuelta a la versión anterior. Si el despliegue se realiza correctamente, puede seguir desplegando a la velocidad que desee, mientras supervisa los cambios en busca de errores, hasta completar el despliegue. AWS CodeDeploy puede configurarse con unos ajustes de despliegue que permitan un despliegue de valores controlados.

Despliegues azul-verde: son similares a los despliegues de valores controlados, excepto que una flota completa de la aplicación se despliega en paralelo. Alterne sus implementaciones en las dos pilas (azul y verde). Una vez más, puede enviar tráfico a la nueva versión y volver a la versión anterior si observa problemas con el despliegue. Normalmente, todo el tráfico se conmuta a la vez. Sin embargo, también puede usar fracciones de tráfico para cada versión para acelerar la adopción de la nueva versión utilizando las capacidades de enrutamiento ponderado por DNS de Amazon Route 53. AWS CodeDeploy y AWS Elastic Beanstalk se pueden configurar con unos ajustes de despliegue que permitan un despliegue azul-verde.

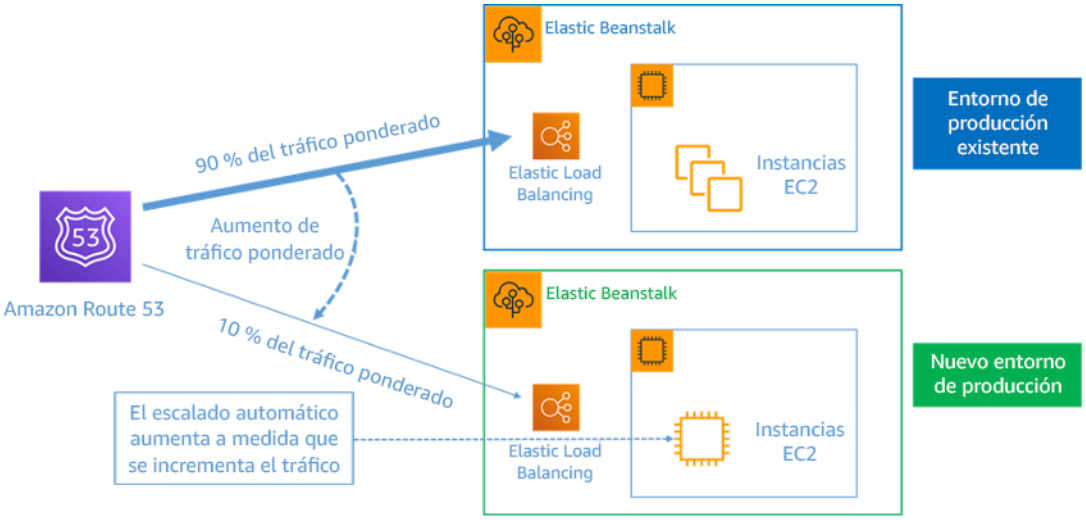



Figura 8: Despliegue azul-verde con AWS Elastic Beanstalk y Amazon Route 53

Detección de desviaciones

Una desviación es cualquier cambio que provoca que un recurso de la infraestructura tenga un estado o una configuración diferente a la esperada. Cualquier tipo de cambio de configuración no administrado va en contra de la noción de infraestructura inmutable y debe detectarse y remediarse para que la infraestructura inmutable se implemente correctamente.

Pasos para la implementación

- No permita que se realicen modificaciones in situ de los recursos de la infraestructura en ejecución.
  - Puede usar AWS Identity and Access Management (IAM) para especificar quién o qué puede acceder a los servicios y recursos de AWS, administrar de forma centralizada los permisos detallados y analizar el acceso para refinar los permisos en AWS.
- Automatice el despliegue de los recursos de la infraestructura para aumentar la reproducibilidad y minimizar la posibilidad de que se cometan errores humanos.
  - Como se describe en el documento técnico «Introduction to DevOps on AWS», la automatización es una piedra angular de los servicios de AWS y es compatible internamente con todos los servicios, características y ofertas.
  - La preparación previade su imagen de máquina de Amazon (AMI) puede acelerar el tiempo de lanzamiento. EC2 Image Builder es un servicio de AWS totalmente administrado que le ayuda a automatizar la creación, el mantenimiento, la validación, el uso compartido y el despliegue de AMI para Linux o Windows personalizadas, seguras y actualizadas.

- Esto son algunos de los servicios que permiten la automatización:
  - AWS Elastic Beanstalk es un servicio para desplegar y escalar rápidamente aplicaciones web desarrolladas con Java, .NET, PHP, Node.js, Python, Ruby, Go y Docker en servidores conocidos como Apache, NGINX, Passenger e IIS.
  - AWS Proton ayuda a los equipos de plataforma a conectar y coordinar todas las herramientas que sus equipos de desarrollo necesitan para el aprovisionamiento de la infraestructura, los despliegues de código, la supervisión y las actualizaciones. AWS Proton permite una infraestructura automatizada, como el aprovisionamiento de código y el despliegue de aplicaciones basadas en contenedores y sin servidor.
- El uso de la infraestructura como código facilita la automatización del despliegue de la infraestructura y ayuda a lograr la inmutabilidad de la misma. AWS proporciona servicios que permiten la creación, el despliegue y el mantenimiento de la infraestructura de forma programática, descriptiva y declarativa.
  - AWS CloudFormation ayuda a los desarrolladores a crear recursos de AWS de una manera ordenada y predecible. Los recursos se escriben en archivos de texto en formato JSON o YAML. Las plantillas requieren una sintaxis y una estructura específicas que dependen de los tipos de recursos que se crean y administran. Los recursos se crean en JSON o YAML con cualquier editor de código, como AWS Cloud9, se registra en un sistema de control de versiones y, a continuación, CloudFormation crea los servicios especificados de una forma segura y repetible.
  - AWS Serverless Application Model (AWS SAM) es un marco de código abierto que puede utilizar para crear aplicaciones sin servidor. enAWS. AWS SAM se integra con otros servicios de AWS y es una extensión de AWS CloudFormation.
  - AWS Cloud Development Kit (AWS CDK) es un marco de desarrollo de software de código abierto para modelar y aprovisionar los recursos de sus aplicaciones en la nube mediante lenguajes de programación conocidos. Puede usar AWS CDK para modelar la infraestructura de las aplicaciones mediante TypeScript, Python, Java y .NET. AWS CDK utiliza AWS CloudFormation en segundo plano para aprovisionar recursos de una forma segura y repetible.
  - AWS Cloud Control API presenta un conjunto común de API de creación, lectura, actualización, eliminación y enumeración (CRUDL) para ayudar a los desarrolladores a administrar su infraestructura en la nube de una forma sencilla y coherente. Las API comunes de Cloud Control API permiten a los desarrolladores administrar de manera uniforme el ciclo de vida de los servicios de AWS y de terceros.
- Implemente patrones de despliegue que tengan la mínima repercusión en los usuarios.

  - Despliegue de valores controlados:
    - «Configuración de un despliegue de un lanzamiento canary de API Gateway»
    - «Create a pipeline with canary deployments for Amazon ECS using AWS App Mesh»
  - Despliegues azul-verde: en el documento técnico «Blue/Green Deployments on AWS», se describen ejemplos de técnicas para implementar estrategias de despliegue azul-verde.
- Detecte desviaciones de la configuración o el estado. Para obtener más información, consulte «Detección de cambios de configuración no administrados en pilas y recursos».

### Recursos

Prácticas recomendadas relacionadas:

- «REL08-BP05 Desplegar cambios con automatización»

Documentos relacionados:

- Automatización de implementaciones seguras y sin intervención
- «Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank»
- Infraestructura como código
- «Implementing an alarm to automatically detect drift in AWS CloudFormation stacks»

Vídeos relacionados:

- «AWS re:Invent 2020: Reliability, consistency, and confidence through immutability»

## REL08-BP05 Desplegar cambios con automatización

Las implementaciones y la aplicación de parches se automatizan para eliminar su impacto negativo.

Los cambios en los sistemas de producción son una de las mayores áreas de riesgo para muchas organizaciones. Consideramos que los despliegues son un problema de primera clase que se debe resolver junto con los problemas comerciales que nuestro software aborda. Hoy en día, significa el uso de la automatización siempre que sea práctico en las operaciones, incluidas las pruebas y la implementación de cambios, la adición o eliminación de capacidad y la migración de datos.

Resultado deseado: incorpore seguridad automatizada de despliegue en el proceso de lanzamiento con extensas pruebas de preproducción, reversiones automáticas y despliegues de producción escalonados. Esta automatización minimiza la posible repercusión en producción causada por despliegues fallidos. Además, los desarrolladores ya no tienen que vigilar activamente los despliegues en producción.

Antipatrones usuales:

- Los cambios se realizan de forma manual.
- Se salta los pasos de la automatización mediante flujos de trabajo de emergencia manuales.
- No sigue sus planes y procesos establecidos a favor de plazos más rápidos.
- Realiza despliegues de seguimiento rápidos sin dejar tiempo de incorporación.

Beneficios de establecer esta práctica recomendada:  cuando se utiliza la automatización para desplegar todos los cambios, se elimina la posibilidad de que se introduzcan errores humanos y se ofrece la posibilidad de realizar pruebas antes de cambiar la producción. Al realizar este proceso antes de la fase de producción, se verifica que los planes estén completos. Además, con la reversión automática al proceso de lanzamiento se pueden identificar los problemas de producción y devolver la carga de trabajo a su estado operativo anterior.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

Automatice su canalización de despliegue. Las canalizaciones de implementación le permiten invocar pruebas automatizadas, detectar anomalías y detener la canalización en un paso determinado antes de la implementación en producción o revertir automáticamente un cambio. Una parte integral de esto es la adopción de la cultura de integración continua y entrega/despliegue continuos (CI/CD), en la que una confirmación o un cambio de código pasan por varias etapas automatizadas, desde las etapas de creación y prueba hasta el despliegue en entornos de producción.

Aunque la sabiduría convencional sugiere que mantenga a las personas informadas sobre los procedimientos operativos más difíciles, le recomendamos que automatice los procedimientos más difíciles por esa misma razón.

Pasos para la implementación

Siga estos pasos de automatización de los despliegues para eliminar las operaciones manuales:

- Configure un repositorio de código para almacenar su código de forma segura: utilice AWS CodeCommit para crear un repositorio seguro basado en Git.
- Configure un servicio de integración continua para compilar el código fuente, ejecutar pruebas y crear artefactos de despliegue: para configurar un proyecto de compilación, consulte «Getting started with AWS CodeBuild using the console».
- Configure un servicio de despliegue que automatice los despliegues de aplicaciones y gestione la complejidad de las actualizaciones de las aplicaciones sin depender de despliegues manuales propensos a errores: AWS CodeDeploy automatiza los despliegues de software en una variedad de servicios de computación, como Amazon EC2, AWS Fargate, AWS Lambda y sus servidores locales. Para configurar estos pasos, consulte «Getting started with CodeDeploy».
- Configure un servicio de entrega continua que automatice sus canalizaciones de lanzamiento para obtener actualizaciones de aplicaciones e infraestructuras más rápidas y fiables: considere la posibilidad de usar AWS CodePipeline para ayudarlo a automatizar sus canalizaciones de lanzamiento. Para obtener más información, consulte los tutoriales de CodePipeline.

## Recursos

Prácticas recomendadas relacionadas:

- OPS05-BP04 Utilizar sistemas de administración de compilación y despliegue
- OPS05-BP10 Automatizar completamente la integración y el despliegue
- OPS06-BP02 Despliegues de prueba
- OPS06-BP04 Automatizar las pruebas y la reversión

Documentos relacionados:

- «Continuous Delivery of Nested AWS CloudFormation Stacks Using AWS CodePipeline»
- «Complete CI/CD with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline»
- Socio de APN: socios que pueden ayudarle a crear soluciones de implementación automatizadas
- «AWS Marketplace: products that can be used to automate your deployments»
- Automatice los mensajes de chat con webhooks.
- La Amazon Builders' Library: Garantizar la seguridad de las reversiones durante las implementaciones

- La Amazon Builders' Library: Agilizar el proceso con la entrega continua
- «¿Qué es AWS CodePipeline?»
- «¿Qué es CodeDeploy?»
- «AWS Systems Manager Patch Manager»
- «What is Amazon SES?»
- ¿Qué es Amazon Simple Notification Service?

Vídeos relacionados:

- «AWS Summit 2019: CI/CD on AWS»

# Administración de errores

> Los errores son algo que se da por descontado y, con el tiempo, todo acabará fallando: desde routers hasta discos duros, desde sistemas operativos hasta unidades de memoria que corrompen paquetes TCP, desde errores transitorios hasta errores permanentes. Este es un hecho innegable, ya se use el hardware de más alta calidad o los componentes más baratos. Werner Vogels, CTO - Amazon.com

Los errores en componentes de hardware de bajo nivel son algo con lo que hay que lidiar cada día en un centro de datos local. Sin embargo, en la nube debe estar protegido ante la mayoría de este tipo de errores. Por ejemplo, los volúmenes de Amazon EBS se colocan en una zona de disponibilidad específica, donde se replican automáticamente para protegerle de los errores en un único componente. Todos los volúmenes de EBS se han diseñado para tener un 99,999 % de disponibilidad. Los objetos de Amazon S3 se almacenan en un mínimo de tres zonas de disponibilidad, lo que ofrece una durabilidad del 99,999999999 % de los objetos a lo largo de un año determinado. Independientemente de cuál sea su proveedor en la nube, existe cierto potencial de que los errores afecten su carga de trabajo. Por tanto, debe dar los pasos necesarios para implementar la resiliencia si necesita que su carga de trabajo sea fiable.

Un requisito previo para la aplicación de las prácticas recomendadas abordadas aquí es que debe asegurarse de que las personas que diseñan, implementan y operan sus cargas de trabajo sean conscientes de los objetivos empresariales y los objetivos de fiabilidad para conseguirlos. Estas personas deben conocer estos requisitos de fiabilidad y estar entrenadas para cumplirlos.

En las siguientes secciones se explican las prácticas recomendadas para administrar los errores de modo que se eviten los impactos sobre su carga de trabajo.

Temas

# Copia de seguridad de los datos

Realice una copia de seguridad de los datos, las aplicaciones y la configuración para satisfacer los requisitos de objetivos de tiempo de recuperación (RTO) y objetivos de punto de recuperación (RPO).

Prácticas recomendadas

- REL09-BP01 Identificar todos los datos de los que se debe hacer una copia de seguridad y crearla o reproducir los datos a partir de los orígenes
- REL09-BP02 Proteger y cifrar copias de seguridad
- REL09-BP03 Realizar copias de seguridad de los datos automáticamente
- REL09-BP04 Realizar una recuperación periódica de los datos para verificar la integridad de la copia de seguridad y los procesos

## REL09-BP01 Identificar todos los datos de los que se debe hacer una copia de seguridad y crearla o reproducir los datos a partir de los orígenes

Conozca y use las funciones de copia de seguridad de los servicios y recursos de datos usados por su carga de trabajo. La mayoría de los servicios ofrecen capacidades para realizar copias de seguridad de los datos de la carga de trabajo.

Resultado deseado: los orígenes de datos se han identificado y clasificado en función del nivel de criticidad. A continuación, establece una estrategia de recuperación de datos basada en el RPO. Esta estrategia supone crear una copia de seguridad de estos orígenes de datos o tener la capacidad de reproducir datos desde otros orígenes. En el caso de la pérdida de datos, la estrategia implementada permite la recuperación o reproducción de datos dentro de los RPO y RTO definidos.

Fase de madurez de la nube: básica

Antipatrones usuales:

- No ser consciente de todos los orígenes de datos para la carga de trabajo y su nivel de criticidad.
- No realizar copias de seguridad de orígenes de datos críticos.
- Realizar copias de seguridad solamente de algunos orígenes de datos sin usar la criticidad como criterio.
- RPO sin definir, o una frecuencia de copias de seguridad que no puede ajustarse al RPO.

- No evaluar si una copia de seguridad es necesaria o si se pueden reproducir datos desde otros orígenes.

Beneficios de establecer esta práctica recomendada: identificar los lugares en los que las copias de seguridad son necesarias e implementar un mecanismo para crear copias de seguridad, o ser capaz de reproducir los datos desde una fuente externa mejora la capacidad de restaurar y recuperar datos durante una interrupción.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

Todos los almacenes de datos de AWS ofrecen capacidades de copia de seguridad. En los servicios como Amazon RDS y Amazon DynamoDB también se pueden hacer copias de seguridad automatizadas, lo que facilita la recuperación a un momento dado (PITR). De este modo, podrá restaurar una copia de seguridad a cualquier momento hasta cinco minutos (o menos) antes del momento actual. Muchos servicios de AWS ofrecen la capacidad de copiar copias de seguridad en otra Región de AWS. AWS Backup es una herramienta que permite centralizar y automatizar la protección de datos entre servicios de AWS. AWS Elastic Disaster Recovery le permite copiar cargas de trabajo de servidor completas y mantener una protección de datos continua localmente o entre zonas de disponibilidad o regiones, con un objetivo de punto de recuperación (RPO) medido en segundos.

Amazon S3 puede usarse como destino de copias de seguridad para los orígenes de datos autoadministrados y administrados por AWS. Los servicios de AWS como Amazon EBS, Amazon RDS y Amazon DynamoDB tienen capacidades integradas para crear copias de seguridad. También se puede usar software de copias de seguridad de terceros.

Se pueden realizar copias de seguridad de los datos locales en Nube de AWS con AWS Storage Gateway o AWS DataSync. Los buckets de Amazon S3 se pueden usar para almacenar estos datos en AWS. Amazon S3 ofrece varios niveles de almacenamiento, como Amazon S3 Glacier o S3 Glacier Deep Archive para reducir el coste del almacenamiento de datos.

Es posible que pueda satisfacer las necesidades de recuperación de datos reproduciendo los datos desde otros orígenes. Por ejemplo, los nodos de réplicas de Amazon ElastiCache o bien las réplicas de lectura de Amazon RDS podrían usarse para reproducir datos si se pierde la principal. En casos en los que orígenes como este puedan usarse para cumplir su objetivo de punto de recuperación (RPO) y su objetivo de tiempo de recuperación (RTO), puede que no necesite una copia de seguridad. Otro ejemplo: si trabaja con Amazon EMR, puede que no sea necesario crear

copias de seguridad de sus almacenes de datos HDFS, en la medida en que puede reproducir los datos enAmazon EMR desde Amazon S3.

Al seleccionar una estrategia de copia de seguridad, piense en el tiempo que se necesita para recuperar los datos. El tiempo necesario para recuperar datos depende del tipo de copia de seguridad (en el caso de una estrategia de copia de seguridad) o de la complejidad del mecanismo de reproducción de datos. Este tiempo debería ajustarse al RTO de la carga de trabajo.

Pasos para la implementación

1. Identifique todos los orígenes de datos para la carga de trabajo. Los datos se pueden almacenar en diversos recursos, como bases de datos, volúmenes, sistemas de archivos, sistemas de registro y almacenamiento de objetos. Consulte la sección Recursos para encontrar Documentos relacionados sobre distintos servicios deAWS en los que se almacenan los datos y la capacidad de copia de seguridad que proporcionan estos servicios.
2. Clasifique los orígenes de datos en función de su criticidad. Los distintos conjuntos de datos tendrán diferentes niveles de criticidad para una carga de trabajo y, por tanto, distintos requisitos de resiliencia. Por ejemplo, algunos datos podrían ser críticos y requerir un RPO cercano a cero, mientras que otros datos podrían ser menos críticos y tolerar un RPO más alto y cierta pérdida de datos. Del mismo modo, los distintos conjuntos de datos podrían tener también diferentes requisitos en cuanto al RTO.
3. Utilice AWS o servicios de terceros para crear copias de seguridad de los datos. AWS Backup es un servicio administrado que permite la creación de copias de seguridad de diferentes orígenes de datos en AWS. AWS Elastic Disaster Recovery administra la replicación automatizada de datos en menos de un segundo a una Región de AWS. La mayoría de los servicios de AWS también disponen de capacidades nativas para crear copias de seguridad. AWS Marketplace tiene muchas soluciones que ofrecen también estas capacidades. Consulte la sección Recursos que aparece a continuación para ver información sobre cómo crear copias de seguridad de datos desde distintos servicios de AWS.
4. En el caso de los datos que no tengan copia de seguridad, establezca un mecanismo de reproducción de datos. Puede decidir no crear una copia de seguridad de datos que puedan reproducirse desde otros orígenes y por distintos motivos. Podría darse una situación en la que sea más barato reproducir datos de orígenes cuando sea necesario en lugar de crear una copia de seguridad, ya que podría existir un coste asociado con el almacenamiento de copias de seguridad. Otro ejemplo es cuando la restauración desde una copia de seguridad tarda más tiempo que la reproducción de los datos desde el origen, lo que implica un incumplimiento del RTO. En tales situaciones, sopese los pros y los contras y establezca un proceso bien

definido sobre cómo se pueden reproducir los datos desde estos orígenes cuando sea necesaria una recuperación de los datos. Por ejemplo, si ha cargado datos desde Amazon S3 en un almacenamiento de datos (como Amazon Redshift) o un clúster de MapReduce (como Amazon EMR) para analizar dichos datos, esto podría ser un ejemplo de datos que se pueden reproducir desde otros orígenes. Siempre y cuando los resultados de estos análisis se almacenen en algún lugar o sean reproducibles, no sufriría una pérdida de datos por un error en el almacenamiento de datos o el clúster de MapReduce. Otros ejemplos que se pueden reproducir desde el origen son las cachés (como Amazon ElastiCache) o las réplicas de lectura de RDS.

5. Establezca una cadencia de copia de seguridad de los datos. La creación de copias de seguridad de orígenes de datos es un proceso periódico y la frecuencia debería depender del RPO.

Nivel de esfuerzo para el plan de implementación: moderado.

## Recursos

Prácticas recomendadas relacionadas:

REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos

REL13-BP02 Usar estrategias de recuperación definidas para cumplir los objetivos de recuperación

Documentos relacionados:

- What Is AWS Backup? (¿Qué es AWS Backup?)
- What is AWS DataSync? (¿Qué es AWS DataSync?)
- What is Volume Gateway? (¿Qué es una puerta de enlace de volumen?)
- Socio de APN: socios que pueden ayudar con la copia de seguridad
- AWS Marketplace: products that can be used for backup (AWS Marketplace: productos que pueden usarse para la copia de seguridad)
- Instantáneas de Amazon EBS
- Backing Up Amazon EFS (Copia de seguridad de Amazon EFS)
- Backing up Amazon FSx for Windows File Server (Copia de seguridad de Amazon FSx para Windows File Server)
- Copia de seguridad y restauración de ElastiCache for Redis
- Creating a DB Cluster Snapshot in Neptune (Creación de una instantánea de base de datos en Neptune)

- Crear una instantánea de base de datos
- Creating an EventBridge Rule That Triggers on a Schedule (Creación de una regla de EventBridge que se ejecuta según una programación)
- Replicación entre regiones con Amazon S3
- AWS Backup de EFS a EFS
- Exportación de datos de registro a Amazon S3
- Administración del ciclo de vida de los objetos
- On-Demand Backup and Restore for DynamoDB (Copia de seguridad y restauración bajo demanda para DynamoDB)
- Recuperación a un momento dado en DynamoDB
- Trabajo con instantáneas de índice en Amazon OpenSearch Service
- What is AWS Elastic Disaster Recovery? (¿Qué es AWS Elastic Disaster Recovery?

Vídeos relacionados:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS (AWS re:Invent 2021: Copia de seguridad, recuperación de desastres y protección contra ransomware con AWS)
- AWS Backup Demo: Cross-Account and Cross-Region Backup (Demostración de AWS Backup: copia de seguridad entre cuentas y entre regiones)
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341) (AWS re:Invent 2019: Análisis en profundidad en AWS Backup, ft. Rackspace)

Ejemplos relacionados:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3 (Laboratorio de Well-Architected: Implementación de la replicación bidireccional entre regiones (CRR) para Amazon S3)
- Well-Architected Lab - Testing Backup and Restore of Data (Laboratorio de Well-Architected: Probar la copia de seguridad y restauración de los datos)
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload (Laboratorio de Well-Architected: Copia de seguridad y restauración con conmutación por recuperación para cargas de trabajo de análisis)

- Well-Architected Lab - Disaster Recovery - Backup and Restore (Laboratorio de Well-Architected: Recuperación de desastres, copia de seguridad y restauración)

## REL09-BP02 Proteger y cifrar copias de seguridad

Controle y detecte el acceso a las copias de seguridad con autenticación y autorización. Evite que la integridad de los datos de las copias de seguridad se vea comprometida (y detecte los casos en los que así sea) mediante el cifrado.

Antipatrones usuales:

- Tener el mismo acceso a las automatizaciones de las copias de seguridad y restauración que a los datos
- No cifrar las copias de seguridad

Beneficios de establecer esta práctica recomendada: proteger las copias de seguridad impide que se manipulen los datos y el cifrado de los datos impide el acceso a esos datos si se exponen por error.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

### Guía para la implementación

Controle y detecte el acceso a las copias de seguridad con autenticación y autorización, como AWS Identity and Access Management (IAM). Evite que la integridad de los datos de las copias de seguridad se vea comprometida (y detecte los casos en los que así sea) mediante el cifrado.

Amazon S3 admite varios métodos de cifrado de los datos en reposo. Con el cifrado del lado del servidor, Amazon S3 acepta sus objetos como datos sin cifrar y después los cifra a medida que se almacenan. Utilizando el cifrado del cliente, su aplicación de carga de trabajo es la responsable de cifrar los datos antes de que se envíen a Amazon S3. Ambos métodos le permiten utilizar AWS Key Management Service (AWS KMS) para crear y almacenar la clave de los datos, o puede facilitar la suya propia, de la que será responsable. Con AWS KMS, puede establecer políticas utilizando IAM sobre quién puede acceder a sus claves de datos y datos descifrados y quién no.

Para Amazon RDS, si ha decidido cifrar las bases de datos, sus copias de seguridad estarán cifradas también. Las copias de seguridad de DynamoDB siempre están cifradas. Al usar AWS Elastic Disaster Recovery, todos los datos en tránsito y en reposo están cifrados. Con Elastic Disaster Recovery, los datos en reposo se pueden cifrar utilizando la clave de cifrado de volumen predeterminada de Amazon EBS o una clave personalizada administrada por el cliente.

Pasos para la implementación

1. Usar el cifrado en cada uno de los almacenes de datos. Si los datos de origen están cifrados, la copia de seguridad también estará cifrada.
   - Utilice el cifrado en Amazon RDS. Puede configurar el cifrado en reposo mediante AWS Key Management Service al crear una instancia de RDS.
   - Utilice el cifrado en volúmenes de Amazon EBS. Puede configurar el cifrado predeterminado o especificar una clave única al crear los volúmenes.
   - Utilice el cifrado de Amazon DynamoDB requerido. DynamoDB cifra todos los datos en reposo. Puede utilizar una clave AWS propiedad de AWS KMS o una clave KMS administrada por AWS, especificando una clave que esté almacenada en su cuenta.
   - Cifre los datos almacenados en Amazon EFS. Configure el cifrado cuando cree el sistema de archivos.
   - Configure el cifrado en las regiones de origen y destino. Puede configurar el cifrado en reposo en Amazon S3 con las claves almacenadas en KMS, pero las claves son específicas de la región. Puede especificar las claves de destino cuando configure la replicación.
   - Elija si desea utilizar el cifrado de Amazon EBS para Elastic Disaster Recovery predeterminado o personalizado. Esta opción cifrará sus datos replicados en reposo en los discos de la subred de la zona de preparación y en los discos replicados.
2. Implemente permisos de privilegios mínimos para acceder a las copias de seguridad. Siga las prácticas recomendadas para limitar el acceso a las copias de seguridad, instantáneas y réplicas de acuerdo con las prácticas recomendadas de seguridad.

## Recursos

Documentos relacionados:

- AWS Marketplace: products that can be used for backup (AWS Marketplace: productos que pueden usarse para la copia de seguridad)
- Cifrado de Amazon EBS
- Amazon S3: Protección de datos mediante cifrado
- Configuración adicional de CRR: replicación de objetos creados con el cifrado del servidor (SSE) usando las claves de cifrado almacenadas en AWS KMS
- Cifrado en reposo en DynamoDB
- Cifrado de recursos de Amazon RDS

- Cifrado de datos y metadatos en Amazon EFS
- Cifrado para copias de seguridad en AWS
- Administrar las tablas cifradas
- Pilar de seguridad: AWS Well-Architected Framework
- What is AWS Elastic Disaster Recovery? (¿Qué es AWS Elastic Disaster Recovery?

Ejemplos relacionados:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3 (Laboratorio de Well-Architected: Implementación de la replicación bidireccional entre regiones (CRR) para Amazon S3)

## REL09-BP03 Realizar copias de seguridad de los datos automáticamente

Configure las copias de seguridad para que se realicen automáticamente mediante el uso de un calendario periódico determinado por el objetivo de punto de recuperación (RPO) o cuando se produzcan cambios en el conjunto de datos. En el caso de los conjuntos de datos críticos con requisitos de pérdida de datos bajos, es necesario realizar una copia de seguridad automática con frecuencia, mientras que en el de los datos menos críticos para los que resultan aceptables ciertas pérdidas, las copias de seguridad pueden ser menos frecuentes.

Resultado deseado: un proceso automatizado que crea copias de seguridad de los orígenes de datos a un ritmo establecido.

Antipatrones usuales:

- Realizar las copias de seguridad manualmente
- Usar recursos que tengan la función de copia de seguridad, pero no incluir la copia de seguridad en la automatización

Beneficios de establecer esta práctica recomendada: la automatización de las copias de seguridad verifica que se realicen con regularidad en función del RPO, y emite una alerta en caso contrario.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

AWS Backup se puede usar para crear copias de seguridad de los datos automatizadas para varios orígenes de datos de AWS. Es posible realizar copias de seguridad de las instancias de Amazon RDS casi de forma continua cada cinco minutos, y de los objetos de Amazon S3 cada quince minutos, lo que facilita una recuperación a un momento dado (PITR) en un punto específico del historial de copias de seguridad. Para otros orígenes de datos de AWS, como los volúmenes Amazon EBS, las tablas de Amazon DynamoDB o los sistemas de archivos de Amazon FSx, AWS Backup puede ejecutar una copia de seguridad automatizada con una frecuencia que puede llegar a ser de una hora. Estos servicios ofrecen también capacidades de copia de seguridad nativas. Entre los servicios de AWS que ofrecen copia de seguridad automatizada con recuperación a un momento dado se incluyen Amazon DynamoDB, Amazon RDS y Amazon Keyspaces (para Apache Cassandra) —la restauración se puede realizar a un punto temporal específico dentro del historial de copias de seguridad—. La mayoría del resto de servicios de almacenamiento de datos de AWS ofrecen la capacidad de programar copias de seguridad periódicas, con una frecuencia que puede llegar a ser de una hora.

Amazon RDS y Amazon DynamoDB ofrecen copias de seguridad continuas con recuperación a un momento dado. El control de versiones de Amazon S3, una vez activado, es automático. Amazon Data Lifecycle Manager se puede utilizar para automatizar la creación, copia y eliminación de instantáneas de Amazon EBS. También puede automatizar la creación, copia, desuso y anulación de registro de imágenes de máquina de Amazon (AMI) basadas en Amazon EBS y sus instantáneas de Amazon EBS subyacentes.

AWS Elastic Disaster Recovery proporciona replicación continua a nivel de bloque desde el entorno de origen (local o AWS) a la región de recuperación de destino. El servicio crea y administra automáticamente instantáneas de Amazon EBS de un momento dado.

Para obtener una vista centralizada de la automatización y el historial de sus copias de seguridad, AWS Backup proporciona una solución de copia de seguridad totalmente administrada y basada en políticas. Centraliza y automatiza la copia de seguridad de datos entre varios servicios de AWS en la nube y en el entorno local utilizando AWS Storage Gateway.

De forma adicional al control de versiones, Amazon S3 incluye también replicación. Todo el bucket de S3 se puede replicar automáticamente en otro bucket de la misma Región de AWS o una diferente.

Pasos para la implementación

1. Identifique los orígenes de datos de los que, actualmente, se están haciendo copias de seguridad de forma manual. Para obtener más detalles, consulte REL09-BP01 Identificar todos los datos de los que se debe hacer una copia de seguridad y crearla o reproducir los datos a partir de los orígenes.
2. Determine el RPO de la carga de trabajo. Para obtener más detalles, consulte REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos.
3. Utilice una solución de copia de seguridad o un servicio administrado automatizados. AWS Backup es un servicio totalmente administrado que facilita la centralización y automatización de la protección de datos entre servicios de AWS, en la nube y en el entorno local. Mediante el uso de planes de copia de seguridad en AWS Backup, cree reglas que definan de qué recursos se debe hacer copia de seguridad y con qué frecuencia deben crearse. Esta frecuencia debe determinarla el RPO establecido en el paso 2. Para obtener orientación práctica sobre cómo crear copias de seguridad automatizadas con AWS Backup, consulte Testing Backup and Restore of Data (Pruebas de copia de seguridad y restauración de datos). La mayoría de servicios de AWS que almacenan datos ofrecen capacidades de copia de seguridad nativas. Por ejemplo, se puede utilizar RDS para realizar copias de seguridad automatizadas con recuperación a un momento dado (PITR).
4. Para los orígenes de datos no compatibles con un servicio administrado o solución de copia de seguridad automatizada, como los orígenes de datos o las colas de mensajes locales, considere el uso de una solución de terceros de confianza para crear copias de seguridad automatizadas. Como alternativa, puede crear una automatización que se encargue de esto con la AWS CLI o algún SDK. Puede usar AWS Lambda Functions o AWS Step Functions para definir la lógica implicada en la creación de una copia de seguridad de datos y utilizar Amazon EventBridge para ejecutarla a una frecuencia basada en su RPO.

Nivel de esfuerzo para el plan de implementación: bajo.

## Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudar con la copia de seguridad
- AWS Marketplace: products that can be used for backup (AWS Marketplace: productos que pueden usarse para la copia de seguridad)
- Creating an EventBridge Rule That Triggers on a Schedule (Creación de una regla de EventBridge que se ejecuta según una programación)

- What Is AWS Backup? (¿Qué es AWS Backup?)
- ¿Qué es AWS Step Functions?
- What is AWS Elastic Disaster Recovery? (¿Qué es AWS Elastic Disaster Recovery?

Vídeos relacionados:

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341) (AWS re:Invent 2019: Análisis en profundidad en AWS Backup, ft. Rackspace)

Ejemplos relacionados:

- Well-Architected Lab - Testing Backup and Restore of Data (Laboratorio de Well-Architected: Probar la copia de seguridad y restauración de los datos)

## REL09-BP04 Realizar una recuperación periódica de los datos para verificar la integridad de la copia de seguridad y los procesos

Valide que su implementación del proceso de copia de seguridad cumpla con los objetivos de tiempo de recuperación (RTO) y los objetivos de punto de recuperación (RPO) mediante una prueba de recuperación.

Resultado deseado: los datos de las copias de seguridad se recuperan periódicamente utilizando mecanismos bien definidos para verificar que la recuperación sea posible dentro del objetivo de tiempo de recuperación (RTO) determinado para la carga de trabajo. Verifique que la restauración a partir de una copia de seguridad dé como resultado un recurso que contenga los datos originales sin que ninguno de ellos resulte dañado o inaccesible, y una pérdida de datos coherente con el objetivo de punto de recuperación (RPO).

Antipatrones usuales:

- Restaurar una copia de seguridad, pero no consultar ni recuperar ningún dato para comprobar que la restauración sea posible.
- Suponer que existe una copia de seguridad.
- Suponer que la copia de seguridad de un sistema está plenamente operativa y que es posible recuperar datos de ella.

- Suponer que el tiempo de restauración o recuperación de datos de una copia de seguridad entra dentro del RTO para la carga de trabajo.
- Suponer que los datos que contiene la copia de seguridad están dentro del RPO para la carga de trabajo.
- Restaurar cuando sea necesario, sin usar un runbook, o fuera de un procedimiento automatizado.

Beneficios de establecer esta práctica recomendada: comprobar la recuperación de las copias de seguridad verifica que los datos puedan restaurarse cuando sea necesario sin tener que preocuparse por si los datos faltan o están dañados, por si la restauración y la recuperación son o no posibles dentro del RTO para la carga de trabajo y por si la pérdida de datos se ajusta al RPO de la carga de trabajo.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

## Guía para la implementación

La comprobación de la capacidad de copia de seguridad y restauración aumenta la confianza en la capacidad de llevar a cabo estas acciones durante una interrupción. Restaure periódicamente las copias de seguridad en una nueva ubicación y lleve a cabo pruebas para verificar la integridad de los datos. Algunas de las pruebas habituales que deberían realizarse son la comprobación de si todos los datos están disponibles, no están dañados, son accesibles y si la pérdida de datos (si la hay) se ajusta al RPO de la carga de trabajo. Estas pruebas también pueden ayudar a determinar si los mecanismos de recuperación son lo suficientemente rápidos como para tener capacidad para el RTO de la carga de trabajo.

Con AWS, puede crear un entorno de prueba y restaurar sus copias de seguridad para evaluar a las capacidades en cuanto al RTO y al RPO y llevar a cabo pruebas sobre el contenido y la integridad de los datos.

Además, Amazon RDS y Amazon DynamoDB permiten la recuperación a un momento dado (PITR). Mediante la copia de seguridad continua, puede restaurar su conjunto de datos al estado en el que se encontrara en una fecha y hora específicas.

Si todos los datos están disponibles, no están dañados, son accesibles y si la pérdida de datos (si la hay) se ajusta al RPO de la carga de trabajo. Estas pruebas también pueden ayudar a determinar si los mecanismos de recuperación son lo suficientemente rápidos como para tener capacidad para el RTO de la carga de trabajo.

AWS Elastic Disaster Recovery ofrece instantáneas de recuperación a un momento dado continuas de volúmenes de Amazon EBS. A medida que se replican los servidores de origen, los estados de un momento dado se cronifican en el tiempo en función de la política configurada. Elastic Disaster Recovery ayuda a verificar la integridad de estas instantáneas lanzando instancias con fines de prueba y simulacro sin redirigir el tráfico.

Pasos para la implementación

1. Identifique los orígenes de datos de los que se estén haciendo copias de seguridad y dónde se están almacenando dichas copias. Guía para la implementación REL09-BP01 Identificar todos los datos de los que se debe hacer una copia de seguridad y crearla o reproducir los datos a partir de los orígenes.
2. Establezca criterios de validación de datos para cada origen de datos. Los diferentes tipos de datos tendrán distintas propiedades, lo que podría requerir diferentes mecanismos de validación. Considere cómo se podrían validar estos datos antes de contar con la confianza suficiente para usarlos en producción. Algunas formas habituales de validar los datos son usar las propiedades de datos y copias de seguridad como el tipo de datos, el formato, la suma de comprobación, el tamaño o una combinación de ellas con lógica de validación personalizada. Por ejemplo, podría tratarse de una comparación de los valores de las sumas de comprobación entre el recurso restaurado y el origen de datos en el momento en que se creó la copia de seguridad.
3. Establezca RTO y RPO para restaurar los datos sobre la base de la importancia crítica de los datos. Guía para la implementación REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos.
4. Evalúe su capacidad de recuperación. Revise su estrategia de copia de seguridad y restauración para comprender si se ajusta a su RTO y RPO, y ajuste la estrategia según sea necesario. Con AWSResilience Hub, puede llevar a cabo una evaluación de su carga de trabajo. La evaluación compara la configuración de su aplicación con la política de resiliencia y notifica si se pueden cumplir los objetivos de RTO y RPO.
5. Realice una restauración de prueba con los procesos establecidos actualmente utilizados en producción para la restauración de datos. Estos procesos dependen de cómo se haya realizado la copia de seguridad del origen de datos, el formato y la ubicación del almacenamiento de la copia de seguridad, o de si los datos se reproducen desde otros orígenes. Por ejemplo, si utiliza un servicio administrado como AWS Backup, podría ser tan sencillo como restaurar la copia de seguridad en un nuevo recurso. Si utilizó AWS Elastic Disaster Recovery puede lanzar un simulacro de recuperación.

6. Valide la recuperación de datos desde el recurso restaurado en función de los criterios que estableciera anteriormente para la validación de datos. ¿Los datos restaurados y recuperados contienen el registro o elemento más reciente en el momento de la copia de seguridad? ¿Estos datos se ajustan al RPO de la carga de trabajo?
7. Mida el tiempo necesario para la restauración y la recuperación y compárelo con el RTO establecido. ¿Este proceso se ajusta al RTO para la carga de trabajo? Por ejemplo, compare las marcas de tiempo del momento en que se inició el proceso de restauración y de cuando se completó la validación de la recuperación para calcular cuánto tarda este proceso. Todas las llamadas a la API de AWS llevan una marca de tiempo y esta información está disponible en AWS CloudTrail. Aunque esta información puede proporcionar detalles sobre cuándo se inició el proceso de restauración, la marca de tiempo final para el momento de finalización de la validación debería quedar registrada mediante su lógica de validación. Si se utiliza un proceso automatizado, se pueden usar servicios como Amazon DynamoDB para almacenar esta información. Además, muchos servicios de AWS proporcionan un historial de eventos que facilita información con marcas de tiempo cuando ocurren determinadas acciones. En AWS Backup, las acciones de copia de seguridad y restauración se denominan trabajos y estos trabajos contienen información con marca de tiempo como parte de estos metadatos, que se pueden utilizar para medir el tiempo necesario para la restauración y la recuperación.
8. Notifique a las partes interesadas si falla la validación de datos o si el tiempo necesario para la restauración y la recuperación supera el RTO establecido para la carga de trabajo. Al implementar la automatización para que haga esto, como en este laboratorio, se pueden usar servicios como Amazon Simple Notification Service (Amazon SNS) para enviar notificaciones push, por ejemplo, por correo electrónico o SMS, a los interesados. Estos mensajes también se pueden publicar en aplicaciones de mensajería, como Amazon Chime, Slack o Microsoft Teams, o usarse para crear tareas como OpsItems con AWS Systems Manager OpsCenter.
9. Automatice este proceso para que se ejecute periódicamente. Por ejemplo, servicios como AWS Lambda o una máquina de estados en AWS Step Functions se pueden usar para automatizar los procesos de restauración y recuperación, y Amazon EventBridge se puede usar para desencadenar este flujo de trabajo de automatización periódicamente como se muestra en el siguiente diagrama de arquitectura. Descubra cómo automatizar la validación de recuperación de datos con AWS Backup. Además, este laboratorio de Well-Architected contiene una experiencia práctica sobre una forma de llevar a cabo la automatización de varios de los pasos que aparecen aquí.

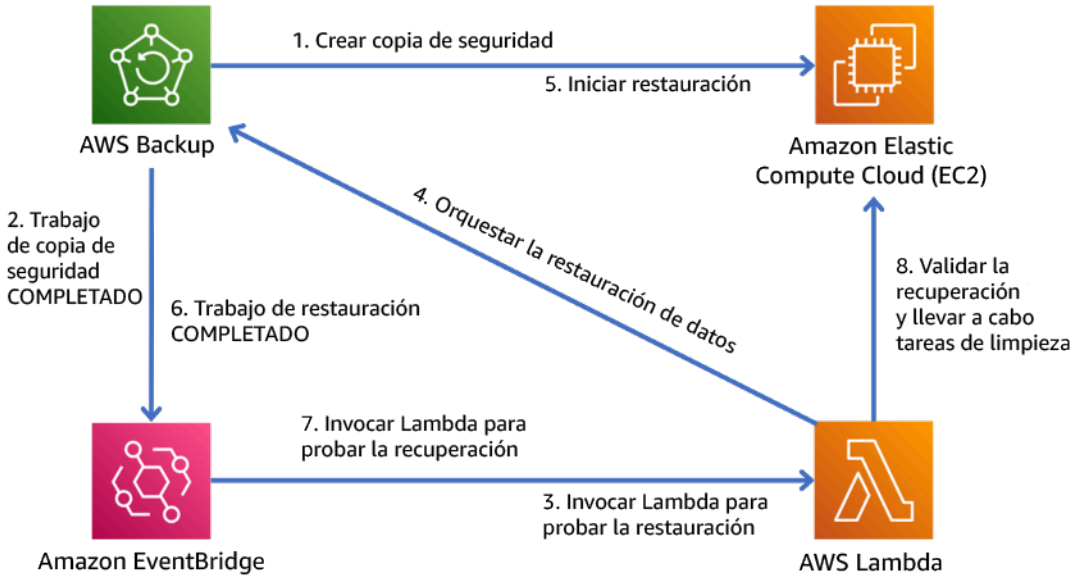



Figura 9. Un proceso de copia de seguridad y restauración automatizado

Nivel de esfuerzo para el plan de implementación: de moderado a alto, en función de la complejidad de los criterios de validación.

## Recursos

Documentos relacionados:

- Automate data recovery validation with AWS Backup (Automatizar la validación de recuperación de datos con AWS Backup)
- Socio de APN: socios que pueden ayudar con la copia de seguridad
- AWS Marketplace: products that can be used for backup (AWS Marketplace: productos que pueden usarse para la copia de seguridad)
- Creating an EventBridge Rule That Triggers on a Schedule (Creación de una regla de EventBridge que se ejecuta según una programación)
- On-demand backup and restore for DynamoDB (Copia de seguridad y restauración bajo demanda para DynamoDB)
- What Is AWS Backup? (¿Qué es AWS Backup?)
- ¿Qué es AWS Step Functions?
- What is AWS Elastic Disaster Recovery (¿Qué es AWS Elastic Disaster Recovery?)
- AWS Elastic Disaster Recovery

Ejemplos relacionados:

- Laboratorio de Well-Architected: probar la copia de seguridad y restauración de los datos

# Usar el aislamiento de errores para proteger su carga de trabajo

Los límites aislados de los errores acotan el efecto de un error en una carga de trabajo a un número limitado de componentes. Los componentes fuera del límite no resultan afectados por el error. Mediante el uso de varios límites aislados de error, puede acotar el impacto en su carga de trabajo.

Prácticas recomendadas

- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones
- REL10-BP02 Seleccionar las ubicaciones adecuadas para el despliegue en varias ubicaciones
- REL10-BP03 Automatizar la recuperación de los componentes restringidos a una sola ubicación
- REL10-BP04 Usar arquitecturas herméticas para limitar el alcance del impacto

## REL10-BP01 Implementar la carga de trabajo en varias ubicaciones

Distribuya los datos y los recursos de la carga de trabajo entre varias zonas de disponibilidad o, si es necesario, entre varias Regiones de AWS. Estas ubicaciones pueden ser tan diversas como sea necesario.

Uno de los principios fundamentales para el diseño de servicios en AWS es evitar puntos únicos de error en la infraestructura física subyacente. Esto nos motiva a desarrollar software y sistemas que utilizan múltiples zonas de disponibilidad y son resistentes a errores de una sola zona. Del mismo modo, los sistemas están diseñados para resistir los errores de un solo nodo de informática, un solo volumen de almacenamiento o una sola instancia de una base de datos. Cuando se desarrolla un sistema que depende de componentes redundantes, es importante asegurarse de que estos componentes funcionen de forma independiente y, en el caso de las Regiones de AWS, de forma autónoma. Los beneficios que se obtienen de los cálculos teóricos de disponibilidad con componentes redundantes solo son válidos si esto se cumple.

Zonas de disponibilidad (AZ)

Las Regiones de AWS tienen varias zonas de disponibilidad diseñadas para ser independientes entre sí. Cada zona de disponibilidad está separada por una distancia física significativa de

otras zonas para evitar situaciones de error correlacionadas debido a peligros ambientales como incendios, inundaciones o tornados. Cada zona de disponibilidad también tiene una infraestructura física independiente: conexiones exclusivas para el suministro eléctrico, fuentes de energía de reserva independientes, servicios mecánicos independientes y conectividad de red independiente dentro y fuera de la zona de disponibilidad. Este diseño limita los errores en cualquiera de estos sistemas a la única AZ afectada. Pese a estar separadas geográficamente, las zonas de disponibilidad están situadas en la misma zona regional, lo que permite las redes de alto rendimiento y baja latencia. La totalidad de la Región de AWS (a través de todas las zonas de disponibilidad, que se componen de múltiples centros de datos físicamente independientes) se puede tratar como un único objetivo de despliegue lógico para su carga de trabajo, incluida la capacidad de replicar datos de forma síncrona (por ejemplo, entre bases de datos). De este modo, puede utilizar las zonas de disponibilidad en una configuración activa/activa o activa/en espera.

Las zonas de disponibilidad son independientes y, por lo tanto, la disponibilidad de la carga de trabajo se incrementa cuando esta se diseña para utilizar varias zonas. Algunos servicios de AWS (incluido el plano de datos de instancia Amazon EC2) se despliegan como servicios estrictamente zonales en los que tienen un destino compartido con la zona de disponibilidad en la que se encuentran. Las instancias Amazon EC2 de las demás AZ no se verán afectadas y seguirán funcionando. Del mismo modo, si un error en una zona de disponibilidad provoca el error de una base de datos de Amazon Aurora, es posible que una instancia de Aurora de réplica de lectura en una zona de disponibilidad no afectada se promueva automáticamente a principal. Los servicios de AWS regionales, como Amazon DynamoDB, utilizan internamente varias zonas de disponibilidad en una configuración activa/activa para alcanzar los objetivos de diseño de disponibilidad para ese servicio, sin que sea necesario configurar la ubicación AZ.

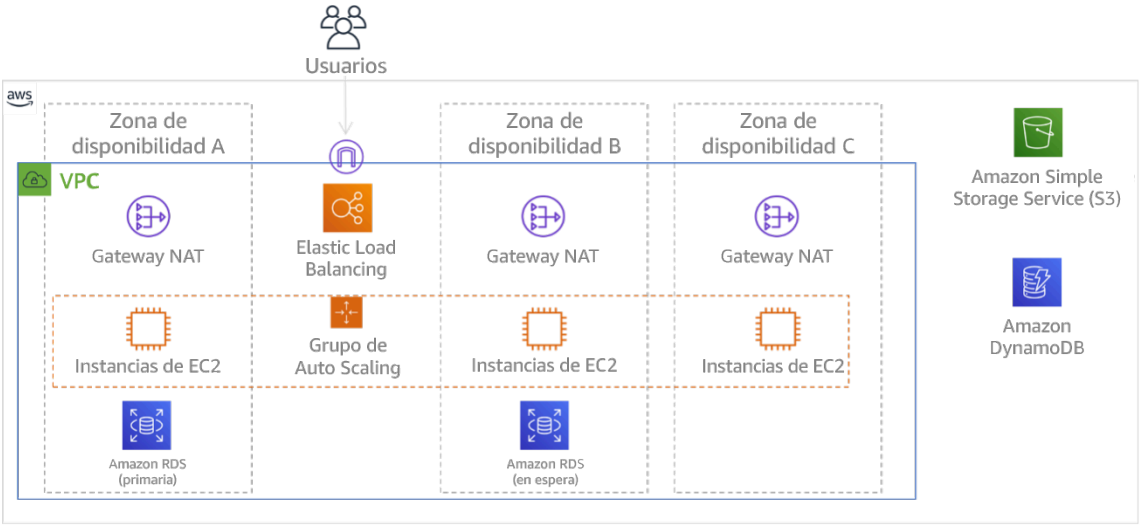


Figura 9: Diagrama que muestra la arquitectura de varios niveles desplegada en tres zonas de disponibilidad. Tenga en cuenta que Amazon S3 y Amazon DynamoDB son siempre Multi-AZ automáticamente. El ELB también se despliega en las tres zonas.

Si bien los planos de control de AWS generalmente ofrecen la capacidad de administrar recursos en toda la región (múltiples zonas de disponibilidad), ciertos planos de control (incluidos Amazon EC2 y Amazon EBS) pueden filtrar los resultados en una única zona de disponibilidad. Al hacerlo, la solicitud se procesa solo en la zona de disponibilidad indicada, lo que reduce la exposición a interrupciones en otras zonas de disponibilidad. Este ejemplo de AWS CLI ilustra la obtención de información de instancias Amazon EC2 solo de la zona de disponibilidad us-east-2c:

```
AWS ec2 describe-instances --filters Name=availability-zone,Values=us-east-2c
```

Zonas locales de AWS

Las zonas locales de AWS actúan de forma similar a las zonas de disponibilidad en su Región de AWS correspondiente en el sentido de que pueden seleccionarse como ubicación de recursos de AWS zonales, como subredes e instancias EC2. Lo que las hace especiales es que no están situadas en la Región de AWS asociada, sino cerca de los grandes centros de población, de la industria y de TI, donde no hay ninguna Región de AWS en la actualidad. Sin embargo, siguen reteniendo un gran ancho de banda y una conexión segura entre las cargas de trabajo de la zona local y las que se ejecutan en la Región de AWS. Debe utilizar las zonas locales de AWS para desplegar las cargas de trabajo más cerca de sus usuarios para cumplir los requisitos de baja latencia.

Red periférica global de Amazon

La red periférica global de Amazon consta de ubicaciones periféricas en ciudades de todo el mundo. Amazon CloudFront utiliza esta red para entregar contenido a los usuarios finales con una latencia menor. AWS Global Accelerator le permite crear sus puntos de conexión de la carga de trabajo en estas ubicaciones periféricas para proporcionar la incorporación a la red global de AWS cerca de sus usuarios. Amazon API Gateway permite que los puntos de conexión de la API optimizados para la periferia utilicen una distribución de CloudFront para facilitar el acceso de los clientes a través de la ubicación periférica más cercana.

Regiones de AWS

Las Regiones de AWS se han diseñado para ser autónomas, por lo que, para utilizar un enfoque multirregión, habría que desplegar copias dedicadas de los servicios en cada región.

Un enfoque multirregión es habitual en las estrategias de recuperación de desastres para cumplir los objetivos de recuperación cuando se producen eventos puntuales a gran escala. Consulte Planificar para la recuperación de desastres (DR) para obtener más información sobre estas estrategias. Sin embargo, aquí nos centramos en la disponibilidad, cuya finalidad es entregar un objetivo de tiempo de actividad medio a lo largo del tiempo. En el caso de los objetivos de alta disponibilidad, una arquitectura multirregión se diseñará generalmente para ser activa/activa, donde cada copia de servicio (en sus respectivas regiones) está activa (sirviendo solicitudes).

> **Recomendación**
>
> Los objetivos de disponibilidad para la mayoría de las cargas de trabajo pueden satisfacerse mediante una estrategia Multi-AZ en una sola Región de AWS. Considere la posibilidad de usar arquitecturas multirregión solo cuando las cargas de trabajo tengan requisitos extremos de disponibilidad, u otros objetivos empresariales, que requieran una arquitectura multirregión.

AWS le proporciona las capacidades para utilizar los servicios entre regiones. Por ejemplo, AWS proporciona una replicación continua y asíncrona de datos mediante la replicación de Amazon Simple Storage Service (Amazon S3), réplicas de lectura de Amazon RDS (incluidas las réplicas de lectura de Aurora) y las tablas globales de Amazon DynamoDB. Con la replicación continua, las versiones de sus datos están disponibles para usarse casi inmediatamente en cada una de sus regiones activas.

Con AWS CloudFormation, puede definir su infraestructura y desplegarla de forma coherente en varias Cuentas de AWS y Regiones de AWS. Y AWS CloudFormation StackSets amplía esta funcionalidad al permitirle crear, actualizar o eliminar pilas de AWS CloudFormation en varias cuentas y regiones con una sola operación. En el caso de los despliegues de instancias Amazon EC2, se utiliza una AMI (imagen de máquina de Amazon) para suministrar información como la configuración del hardware y el software instalado. Puede implementar una canalización del generador de imágenes de Amazon EC2 que cree las ANU que necesita y copiarlas en sus regiones activas. Esto garantiza que estas AMI doradas tiene todo lo que necesita para desplegar y escalar su carga de trabajo en cada nueva región.

Para enrutar el tráfico, tanto Amazon Route 53 como AWS Global Accelerator permiten la definición de políticas que determinen qué usuarios van a cada punto de conexión regional activo. Con Global Accelerator se establece un regulador de tráfico para controlar el porcentaje de tráfico que se dirige a cada punto de conexión de la aplicación. Route 53 es compatible con este enfoque porcentual y

también con varias políticas disponibles, incluidas las basadas en la geoproximidad y la latencia. Global Accelerator aprovecha automáticamente la amplia red de servidores periféricos de AWS, para integrar el tráfico en la estructura de red de AWS de lo antes posible, lo que se traduce en menores latencias de solicitudes.

El funcionamiento de todas estas capacidades permite preservar la autonomía de cada región. Existen muy pocas excepciones a este enfoque, incluidos nuestros servicios que proporcionan entrega periférica global (como Amazon CloudFront y Amazon Route 53), junto con el plano de control para el servicio AWS Identity and Access Management (IAM). La gran mayoría de los servicios funcionan completamente en una sola región.

Centro de datos local

En el caso de las cargas de trabajo que se ejecutan en un centro de datos local, diseñe una experiencia híbrida cuando sea posible. AWS Direct Connect proporciona una conexión de red dedicada desde su entorno local a AWS, lo que le permite la ejecución en ambos.

Otra opción es ejecutar la infraestructura y los servicios de AWS localmente mediante AWS Outposts. AWS Outposts es un servicio completamente administrado que extiende la infraestructura de AWS, los servicios de AWS, las API y las herramientas a su centro de datos. La misma infraestructura de hardware que se utiliza en la Nube de AWS se instala en su centro de datos. AWS Outposts se conectan a las Región de AWS. A continuación, puede usar AWS Outposts para respaldar las cargas de trabajo que tengan requisitos de baja latencia o de procesamiento local de datos.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

- Use varias zonas de disponibilidad y Regiones de AWS. Distribuya los datos y los recursos de la carga de trabajo entre varias zonas de disponibilidad o, si es necesario, entre varias Regiones de AWS. Estas ubicaciones pueden ser tan diversas como sea necesario.
  - Los servicios regionales se implementan en las zonas de disponibilidad.
    - Esto incluye Amazon S3, Amazon DynamoDB y AWS Lambda (cuando no está conectado a una VPC)
  - Implemente su contenedor, instancia y cargas de trabajo basadas en funciones en múltiples zonas de disponibilidad. Use los almacenes de datos multizona, incluidas las memorias caché. Use las características de EC2 Auto Scaling, ubicación de tareas de ECS, configuración de la función AWS Lambda cuando se ejecute en su VPC y clústeres ElastiCache.

    - Utilice subredes en zonas de disponibilidad distintas cuando implemente grupos de Auto Scaling.
        - Ejemplo: distribución de instancias en zonas de disponibilidad
        - Estrategias de asignación de tareas de Amazon ECS
        - Configurar una función AWS Lambda para obtener acceso a los recursos en una Amazon VPC
        - Elección de regiones y zonas de disponibilidad
    - Utilice subredes en zonas de disponibilidad distintas cuando implemente grupos de Auto Scaling.
        - Ejemplo: distribución de instancias en zonas de disponibilidad
    - Utilice los parámetros de colocación de tareas de ECS, especificando grupos de subred de base de datos
        - Estrategias de asignación de tareas de Amazon ECS
    - Utilice subredes en múltiples zonas de disponibilidad cuando configure una función para que se ejecute en su VPC.
        - Configurar una función AWS Lambda para obtener acceso a los recursos en una Amazon VPC
    - Utilice múltiples zonas de disponibilidad con clústeres ElastiCache.
        - Elección de regiones y zonas de disponibilidad
- Si la carga de trabajo se debe desplegar en varias regiones, elija una estrategia multirregión. La mayoría de los requisitos de fiabilidad se pueden satisfacer con una sola Región de AWS que use una estrategia de varias zonas de disponibilidad. Use una estrategia multirregión cuando sea necesario para satisfacer las necesidades del negocio.
    - AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (Patrones de arquitectura para aplicaciones activas-activas en varias regiones) (ARC209-R2)
        - Contar con otra Región de AWS puede añadir otra capa de seguridad en cuanto a la disponibilidad de los datos.
        - Algunas cargas de trabajo tienen requisitos normativos que exigen una estrategia multirregión.
- Evalúe AWS Outposts para su carga de trabajo. Si su carga de trabajo requiere baja latencia en el centro de datos local o si tiene requisitos de procesamiento de datos locales, A continuación, ejecute la infraestructura de AWS y los servicios locales con AWS Outposts.
    - ¿Qué es AWS Outposts?

- Determine si las zonas locales de AWS le ayudan a prestar servicio a sus usuarios. Si tiene requisitos de baja latencia, compruebe si las zonas locales de AWS están cerca de sus usuarios. En caso afirmativo, úselas para implementar las cargas de trabajo más cerca de esos usuarios.
  - Preguntas frecuentes sobre las zonas locales de AWS

### Recursos

Documentos relacionados:

- Infraestructura global de AWS
- Preguntas frecuentes sobre las zonas locales de AWS
- Estrategias de asignación de tareas de Amazon ECS
- Elección de regiones y zonas de disponibilidad
- Ejemplo: distribución de instancias en zonas de disponibilidad
- Tablas globales: replicación multirregión con DynamoDB
- Uso de las bases de datos globales de Amazon Aurora
- Serie de blog Creating a Multi-Region Application with AWS Services blog series (Creación de una aplicación multirregión con servicios de AWS)
- ¿Qué es AWS Outposts?

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (Patrones de arquitectura para aplicaciones activas-activas en varias regiones) (ARC209-R2)
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure (Innovación y funcionamiento de la infraestructura de red global de AWS) (NET339)

## REL10-BP02 Seleccionar las ubicaciones adecuadas para el despliegue en varias ubicaciones

### Resultado deseado

Para obtener una alta disponibilidad, despliegue siempre (cuando sea posible) sus componentes de carga de trabajo en varias zonas de disponibilidad (AZ), como se muestra en la figura 10. En el caso

de cargas de trabajo con requisitos de resiliencia extremos, evalúe cuidadosamente las opciones de una arquitectura multirregión.

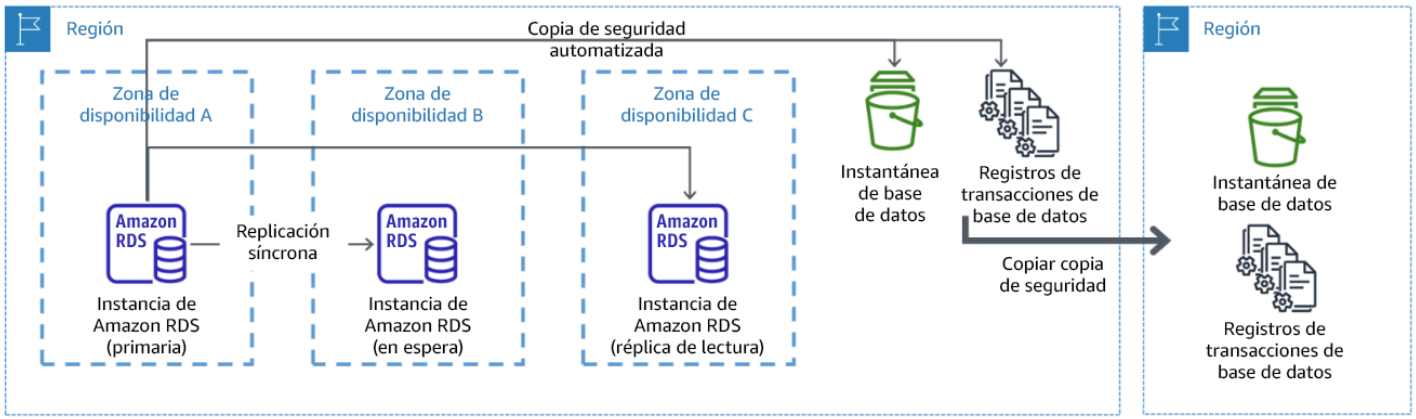



Figura 10: Un despliegue de base de datos multi-AZ resiliente con copia de seguridad en otra región de AWS

## Patrones de uso no recomendados comunes

- Elegir el diseño de una arquitectura multirregión cuando una arquitectura multi-AZ satisfaría los requisitos.
- No tener en cuenta las dependencias entre los componentes de la aplicación si los requisitos de resiliencia y multiubicación difieren entre esos componentes.

## Beneficios de establecer esta práctica recomendada

Para obtener resiliencia, debe utilizar un enfoque que cree capas de defensa. Una capa protege de las interrupciones más pequeñas y comunes mediante la creación de una arquitectura de alta disponibilidad con múltiples AZ. Otra capa de defensa está pensada para proteger de eventos poco frecuentes como los desastres naturales generalizados y las interrupciones en el nivel de la región. Esta segunda capa implica la arquitectura de su aplicación para que abarque múltiples Regiones de AWS.

- La diferencia entre una disponibilidad del 99,5 % y del 99,99 % es de más de 3,5 horas al mes. La disponibilidad prevista de una carga de trabajo solo puede alcanzar los «cuatro nueves» si se encuentra en varias AZ.
- Al ejecutar su carga de trabajo en varias AZ, puede aislar las interrupciones de energía eléctrica, refrigeración y redes, y la mayoría de los desastres naturales como incendios e inundaciones.

- La implementación de una estrategia multirregión para su carga de trabajo le ayuda a protegerla de desastres naturales generalizados que afecten a una región geográfica amplia de un país o de errores técnicos de alcance regional. Tenga en cuenta que implementar una arquitectura multirregión puede ser significativamente complejo y no suele ser necesario para la mayoría de las cargas de trabajo.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

En el caso de un evento de desastre provocado por la interrupción o pérdida parcial de una zona de disponibilidad, la implementación de una carga de trabajo con alta disponibilidad en varias zonas de disponibilidad en una sola Región de AWS contribuye a mitigar los desastres naturales o técnicos. Cada Región de AWS consta de varias zonas de disponibilidad, cada una aislada de los errores de las demás zonas y separadas por una distancia significativa. Sin embargo, en el caso de un evento de desastre que implique el riesgo de perder varios componentes de zona de disponibilidad que están alejados entre sí, debe implementar opciones de recuperación de desastres para mitigar los errores de alcance regional. Para las cargas de trabajo que requieren una resiliencia extrema (infraestructuras críticas, aplicaciones relacionadas con la sanidad, infraestructuras de sistemas financieros, etc.), puede ser necesaria una estrategia multirregión.

## Pasos de implementación

1. Evalúe su carga de trabajo y determine si las necesidades de resiliencia se pueden satisface con un enfoque multi-AZ (una sola Región de AWS) o si requieren un enfoque multirregión. La implementación de una arquitectura de multirregión para satisfacer estos requisitos supondrá una complejidad adicional, por lo que deberá considerar detenidamente su caso de uso y sus requisitos. Los requisitos de resiliencia se pueden cumplir casi siempre con una sola Región de AWS. Tenga en cuenta los siguientes requisitos posibles a la hora de determinar si necesita utilizar varias regiones:
    a. Recuperación de desastres (DR): en el caso de un evento de desastre provocado por la interrupción o pérdida parcial de una zona de disponibilidad, la implementación de una carga de trabajo con alta disponibilidad en varias zonas de disponibilidad en una sola Región de AWS contribuye a mitigar los desastres naturales o técnicos. En el caso de un evento de desastre que implique el riesgo de perder varios componentes de zona de disponibilidad que están alejados entre sí, debe implementar opciones de recuperación de desastres en varias regiones para mitigar los desastres naturales o los errores técnicos de alcance regional.

b. Alta disponibilidad: se puede utilizar una arquitectura de multirregión (mediante varias AZ en cada región) para lograr una disponibilidad superior a cuatro nueves (> 99,99 %).

c. Localización de pilas: al desplegar una carga de trabajo para una audiencia global, puede desplegar pilas localizadas en diferentes Regiones de AWS para atender a las audiencias de esas regiones. La localización puede incluir el idioma, la moneda y los tipos de datos almacenados.

d. Proximidad a los usuarios: al desplegar una carga de trabajo para una audiencia global, puede reducir la latencia si despliega las pilas en Regiones de AWS cerca de donde están los usuarios finales.

e. Residencia de los datos: algunas cargas de trabajo están sujetas a requisitos de residencia de datos, en los que los datos de ciertos usuarios deben permanecer dentro de las fronteras de un país específico. En función de la normativa en cuestión, puede optar por desplegar una pila completa, o solo los datos, en la Región de AWS en esas fronteras.

2. A continuación, se presentan algunos ejemplos de la funcionalidad multi-AZ proporcionada por los servicios de AWS:

   a. Para proteger las cargas de trabajo que utilizan EC2 o ECS, despliegue un equilibrador de carga elástico ante los recursos de computación. Elastic Load Balancing proporciona la solución para detectar las instancias en zonas con estado incorrecto y enrutar el tráfico a las que lo tienen correcto.

      i. Introducción a Application Load Balancers

      ii. Introducción a los equilibradores de carga de red

   b. En el caso de las instancias EC2 que ejecutan software estándar comercial y que no admiten el equilibrio de carga, puede conseguir una forma de tolerancia a errores mediante la implementación de una metodología de recuperación de desastres multi-AZ.

      i. the section called “REL13-BP02 Usar estrategias de recuperación definidas para cumplir los objetivos de recuperación”

   c. Para las tareas de Amazon ECS, despliegue su servicio de forma homogénea en tres zonas de disponibilidad para lograr un equilibrio entre la disponibilidad y el coste.

      i. Amazon ECS availability best practices | Containers (Prácticas recomendadas de disponibilidad de Amazon ECS | Contenedores

   d. En el caso de Aurora Amazon RDS, puede elegir Multi-AZ como una opción de configuración. En caso de error de la instancia de la base de datos principal, Amazon RDS promociona automáticamente una base de datos en espera para recibir el tráfico en otra zona de

disponibilidad. También se pueden crear réplicas de lectura multirregión para mejorar la resiliencia.

   i. Despliegues multi-AZ de Amazon RDS

   ii. Creación de una réplica de lectura en una Región de AWS diferente

3. A continuación, se presentan algunos ejemplos de la funcionalidad multirregión proporcionada por los servicios de AWS:

   a. Para las cargas de trabajo de Amazon S3, en las que la disponibilidad multi-AZ la proporciona automáticamente el servicio, considere la posibilidad de utilizar puntos de acceso multirregión si se necesita un despliegue multirregión.

      i. Puntos de acceso multirregión en Amazon S3

   b. En el caso de las tablas de DynamoDB, en las que el servicio proporciona automáticamente la disponibilidad multi-AZ, puede convertir fácilmente las tablas existentes en tablas globales para aprovechar las ventajas de múltiples regiones.

      i. Convert Your Single-Region Amazon DynamoDB Tables to Global Tables (Convierta sus tablas de Amazon DynamoDB de una sola región en tablas globales)

   c. Si su carga de trabajo está encabezada por Application Load Balancers o equilibradores de carga de red, use AWS Global Accelerator para mejorar la disponibilidad de su aplicación mediante el direccionamiento del tráfico a varias regiones que contengan puntos de conexión con el estado correcto.

      i. Endpoints for standard accelerators in AWS Global Accelerator - AWS Global Accelerator (Puntos de conexión para aceleradores estándar en AWS Global Accelerator - AWS Global Accelerator) (amazon.com)

   d. En el caso de las aplicaciones que utilizan AWS EventBridge, considere la posibilidad de utilizar buses entre regiones para reenviar los eventos a otras Regiones que seleccione.

      i. Sending and receiving Amazon EventBridge events between Regiones de AWS (Envío y recepción de eventos de Amazon EventBridge entre Regiones de AWS)

   e. En el caso de las bases de datos de Amazon Aurora, considere de usar bases de datos globales de Aurora, que abarcan varias regiones de AWS. Los clústeres existentes pueden modificarse para agregar también nuevas regiones.

      i. Introducción a las bases de datos globales de Amazon Aurora

   f. Si su carga de trabajo incluye claves de cifrado de AWS Key Management Service (AWS KMS), considere si las claves multirregión son adecuadas para su aplicación.

      i. Claves multirregión en AWS KMS

g. Para conocer otras características de servicios de AWS, consulte esta serie de blog en Serie Creating a Multi-Region Application with AWS Services blog series (Creación de una aplicación multirregión con servicios de AWS)

Nivel de esfuerzo para el plan de implementación: De moderado a alto

## Recursos

Documentos relacionados:

- Serie Creating a Multi-Region Application with AWS Services blog series (Creación de una aplicación multirregión con servicios de AWS)
- Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active (Arquitectura de recuperación de desastres (DR) en AWS, parte IV: activa-activa multisitio)
- Infraestructura global de AWS
- Preguntas frecuentes sobre las zonas locales de AWS
- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud (Arquitectura de recuperación de desastres (DR) en AWS, parte I: estrategias de recuperación en la nube)
- La recuperación de desastres es diferente en la nube
- Tablas globales: replicación multirregión con DynamoDB

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (Patrones de arquitectura para aplicaciones activas-activas en varias regiones) (ARC209-R2)
- Auth0: arquitectura de alta disponibilidad en varias regiones que se amplía a más de 1500 millones de inicios de sesión en un mes con conmutación por error automatizada

Ejemplos relacionados:

- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud (Arquitectura de recuperación de desastres (DR) en AWS, parte I: estrategias de recuperación en la nube)
- DTCC consigue un nivel de resiliencia mayor del que podría obtener localmente

- Expedia Group usa una arquitectura de varias regiones y varias zonas de disponibilidad con un servicio DNS propio para agregar resiliencia a las aplicaciones
- Uber: recuperación de desastres para Kafka en varias regiones
- Netflix: estrategia activa-activa para la resiliencia multirregional
- Cómo creamos Data Residency for Atlassian Cloud
- Intuit TurboTax se ejecuta en dos regiones

## REL10-BP03 Automatizar la recuperación de los componentes restringidos a una sola ubicación

Si los componentes de la carga de trabajo solo se pueden ejecutar en una zona de disponibilidad o en el centro de datos local, implemente la capacidad de volver a crear la carga de trabajo de acuerdo con los objetivos de recuperación definidos.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: medio

### Guía para la implementación

Si la práctica recomendada de desplegar la carga de trabajo en varias ubicaciones no es posible por limitaciones tecnológicas, debe implementar una ruta alternativa hacia la resiliencia. Debe automatizar la capacidad de recrear la infraestructura necesaria, reimplementar las aplicaciones y recrear los datos necesarios para estos casos.

Por ejemplo, Amazon EMR lanza todos los nodos para un clúster determinado en la misma zona de disponibilidad, porque la ejecución de un clúster en la misma zona mejora el rendimiento de los flujos de trabajo, ya que ofrece una velocidad de acceso a los datos más alta. Si este componente resulta necesario para la resiliencia de la carga de trabajo, debe tener una forma de volver a desplegar el clúster y sus datos. Además, para Amazon EMR, debería aprovisionar la redundancia de formas diferentes al uso de multi-AZ. Puede aprovisionar varios nodos. Con el sistema de archivos EMR (EMRFS), los datos en EMR se pueden almacenar en Amazon S3, lo que a su vez puede replicarse entre varias zonas de disponibilidad o Regiones de AWS.

De modo similar, en el caso de Amazon Redshift, aprovisiona de forma predeterminada el clúster en una zona de disponibilidad seleccionada al azar dentro de la Región de AWS que haya seleccionado. Todos los nodos del clúster se aprovisionan en la misma zona.

Para cargas de trabajo basadas en servidores con estado implementadas en un centro de datos local, puede utilizar AWS Elastic Disaster Recovery para proteger sus cargas de trabajo en AWS. Si

ya está alojado en AWS, puede utilizar Elastic Disaster Recovery para proteger su carga de trabajo en una zona o región de disponibilidad alternativa. Elastic Disaster Recovery utiliza la replicación continua a nivel de bloque en un área de preparación ligera para proporcionar una recuperación rápida y fiable de las aplicaciones locales y basadas en la nube.

Pasos para la implementación

1. Implemente la autorrecuperación. Implemente sus instancias o contenedores con escalado automático siempre que sea posible. Si no puede usar el escalado automático, utilice la recuperación automática para instancias EC2 o implemente la automatización de autorrecuperación basada en eventos de ciclo de vida del contenedor de Amazon EC2 o ECS.
   - Utilice los grupos de Amazon EC2 Auto Scaling para instancias y cargas de trabajo de contenedor que no tienen requisitos para una sola dirección IP de instancia, dirección IP privada, dirección IP elástica y metadatos de instancia.
     - Los datos de usuario de la plantilla de lanzamiento se pueden usar para implementar una automatización que pueda solucionar la mayoría de las cargas de trabajo.
   - Utilice la recuperación de instancias Amazon EC2 automática para cargas de trabajo que requieren una única dirección ID de instancia, dirección IP privada, dirección IP elástica y metadatos de instancia.
     - La recuperación automática enviará alertas de estado de recuperación a un tema de SNS cuando se detecte un error en la instancia.
   - Utilice los eventos del ciclo de vida de la instancia Amazon EC2 o los eventos de Amazon ECS para automatizar la autorrecuperación cuando no se pueda utilizar el escalado automático ni la recuperación EC2.
     - Utilice los eventos para invocar la automatización que reparará su componente de acuerdo con la lógica de proceso que necesita.
   - Proteja las cargas de trabajo con estado que se limitan a una única ubicación con AWS Elastic Disaster Recovery.

## Recursos

Documentos relacionados:

- Amazon ECS events (Eventos de Amazon ECS)
- Amazon EC2 Auto Scaling lifecycle hooks (Enlaces de ciclo de vida de Amazon EC2 Auto Scaling)
- Recupere la instancia.

- Escalado automático del servicio
- What Is Amazon EC2 Auto Scaling? (¿Qué es Amazon EC2 Auto Scaling?)
- AWS Elastic Disaster Recovery

## REL10-BP04 Usar arquitecturas herméticas para limitar el alcance del impacto

La implementación de arquitecturas herméticas (también conocidas como arquitecturas basadas en celdas) restringe el efecto del fallo dentro de una carga de trabajo a un número limitado de componentes.

Resultado deseado: una arquitectura basada en celdas utiliza numerosas instancias aisladas de una carga de trabajo, donde cada instancia se conoce como celda. Cada celda es independiente, no comparte estado con otras celdas y gestiona un subconjunto de las solicitudes de la carga de trabajo global. Esto reduce la posible repercusión de un error, como una actualización de software incorrecta, en una celda individual y en las solicitudes que está procesando. Si una carga de trabajo utiliza 10 celdas para atender 100 peticiones cuando se produce un error, el 90 % del total de las solicitudes no se verá afectado por el error.

Antipatrones usuales:

- Permitir que las celdas crezcan sin límites.
- Aplicar actualizaciones o despliegues de código a todas las celdas al mismo tiempo.
- Compartir estado o componentes entre celdas (a excepción de la capa de enrutador).
- Añadir lógica compleja de negocio o de enrutamiento a la capa de enrutador.
- No minimizar las interacciones entre celdas.

Beneficios de establecer esta práctica recomendada: con las arquitecturas basadas en celdas, muchos tipos habituales de errores están contenidos dentro de la propia celda, lo que proporciona un aislamiento adicional de los errores. Estos límites de errores pueden proporcionar resiliencia contra tipos de errores que, de otro modo, serían difíciles de contener, como despliegues de código infructuosos o solicitudes que se corrompen o activan un modo de error concreto (también conocidas como solicitudes de píldora envenenada).

## Guía para la implementación

En un barco, los mamparos garantizan que una brecha en el casco quede contenida en una sola sección del casco. En los sistemas complejos, este modelo de contención suele imitarse para facilitar el aislamiento de errores. Los límites aislados de los errores restringen el efecto de un error en una carga de trabajo a un número limitado de componentes. Los componentes fuera del límite no resultan afectados por el error. Mediante el uso de varios límites aislados de error, puede acotar el impacto en su carga de trabajo. En AWS, los clientes pueden utilizar varias zonas y regiones de disponibilidad para proporcionar aislamiento de errores, pero el concepto de aislamiento de errores también puede extenderse a la arquitectura de su carga de trabajo.

La carga de trabajo global se divide en celdas mediante una clave de partición. Esta clave tiene que alinearse con la corriente del servicio, o la forma natural en que la carga de trabajo de un servicio puede subdividirse con mínimas interacciones entre celdas. Algunos ejemplos de claves de partición son el ID de cliente, el ID de recurso o cualquier otro parámetro fácilmente accesible en la mayoría de las llamadas a la API. Una capa de enrutador de celdas distribuye las solicitudes a celdas individuales en función de la clave de partición, y presenta un único punto de conexión a los clientes.

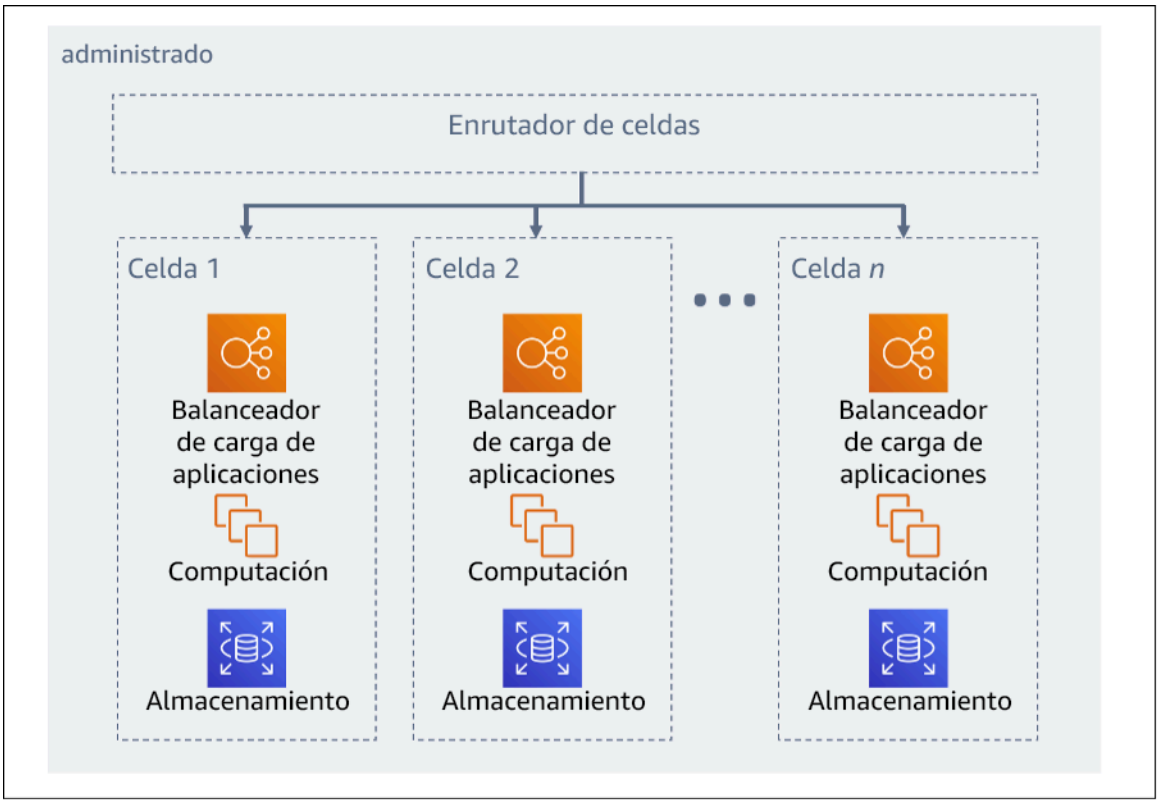



Figura 11: Arquitectura basada en celdas

### Pasos para la implementación

Al diseñar una arquitectura basada en celdas, hay que tener en cuenta varias consideraciones de diseño.

1. Clave de partición: debe prestarse especial atención a la hora de elegir la clave de partición.
   - Debe alinearse con la corriente del servicio o con la forma natural en que la carga de trabajo de un servicio puede subdividirse con mínimas interacciones entre celdas. Algunos ejemplos son: `ID de cliente` o bien `ID de recurso`.
   - La clave de partición debe estar disponible en todas las solicitudes, ya sea de modo directo o de una manera que se pueda inferir con facilidad de forma determinista por otros parámetros.
2. Asignación persistente de celdas: los servicios ascendentes solo deben interactuar con una única celda durante el ciclo de vida de sus recursos.
   - Según la carga de trabajo, puede ser necesaria una estrategia de migración de celda para migrar datos de una celda a otra. Un posible escenario en el que puede ser precisa una migración de celda es si un usuario o recurso concreto de la carga de trabajo crece demasiado y requiere una celda dedicada.
   - Las celdas no deben compartir estados ni componentes entre ellas.
   - En consecuencia, las interacciones entre celdas deben evitarse o mantenerse al mínimo, ya que dichas interacciones crean dependencias entre las celdas y, por lo tanto, disminuyen las ventajas en el aislamiento de errores.
3. Capa de enrutador: la capa de enrutador es un componente compartido entre celdas, lo que significa que no puede seguir la misma estrategia de compartimentación que las celdas.
   - Se recomienda que la capa de enrutador distribuya las solicitudes a las celdas individuales mediante un algoritmo de asignación de particiones de una manera eficiente a nivel computacional, como la combinación de funciones hash criptográficas y aritmética modular para asignar claves de partición a las celdas.
   - Para evitar impactos multicelda, la capa de enrutador debe ser lo más simple y escalable horizontalmente posible, lo que requiere evitar una lógica de negocio compleja dentro de esta capa. Esto tiene la ventaja añadida de facilitar la comprensión de su comportamiento esperado en todo momento, lo que permite una comprobabilidad exhaustiva. Como explica Colm MacCárthaigh en Reliability, constant work, and a good cup of coffee (Fiabilidad, trabajo constante y una buena taza de café), los diseños sencillos y los patrones de trabajo constantes producen sistemas fiables y reducen la antifragilidad.
4. Tamaño de la celda: las celdas deben tener un tamaño máximo y no debe permitirse que lo superen.

- Para determinar el tamaño máximo, se deben llevar a cabo pruebas exhaustivas hasta que se alcancen puntos de ruptura y se establezcan márgenes de funcionamiento seguros. Para obtener más detalles sobre cómo implementar prácticas de prueba, consulte REL07-BP04 Realizar pruebas de su carga de trabajo
- La carga de trabajo global crecerá a medida que se añadan celdas adicionales, lo que permite escalar la carga de trabajo con los aumentos de la demanda.

5. Estrategias multi-AZ o en varias regiones: se deben aprovechar numerosas capas de resiliencia para ofrecer protección contra diferentes dominios de error.
   - Para obtener resiliencia, debe utilizar un enfoque que cree capas de defensa. Una capa protege de las interrupciones más pequeñas y frecuentes mediante la creación de una arquitectura de alta disponibilidad con múltiples AZ. Otra capa de defensa está pensada para proteger de eventos poco frecuentes, como las catástrofes naturales generalizadas y las interrupciones a nivel regional. Esta segunda capa implica la arquitectura de su aplicación para que abarque múltiples Regiones de AWS. La implementación de una estrategia multirregión para su carga de trabajo le ayuda a protegerla de catástrofes naturales generalizadas que afecten a una región geográfica amplia de un país o de errores técnicos de alcance regional. Tenga en cuenta que implementar una arquitectura multirregión puede ser significativamente complejo y no suele ser necesario para la mayoría de las cargas de trabajo. Para obtener más detalles, consulte REL10-BP02 Seleccionar las ubicaciones adecuadas para el despliegue en varias ubicaciones.
6. Despliegue de código: se prefiere una estrategia de despliegue de código escalonado en lugar de desplegar los cambios de código en todas las celdas al mismo tiempo.
   - Esto ayudará a minimizar posibles errores en numerosas celdas provocados por un despliegue incorrecto o a un error humano. Para obtener más detalles, consulte Automatización de implementaciones seguras y sin intervención.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Recursos

Prácticas recomendadas relacionadas:

- REL07-BP04 Realizar pruebas de su carga de trabajo
- REL10-BP02 Seleccionar las ubicaciones adecuadas para el despliegue en varias ubicaciones

Documentos relacionados:

- Reliability, constant work, and a good cup of coffee (Fiabilidad, trabajo constante y una buena taza de café)
- AWS and Compartmentalization (AWS y compartimentalización)
- Aislamiento de las cargas de trabajo a través de la fragmentación aleatoria
- Automatización de implementaciones seguras y sin intervención

Vídeos relacionados:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: Cerrar los bucles y abrir las mentes: cómo asumir el control de los sistemas grandes y pequeños)
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338) (AWS re:Invent 2018: Cómo AWS minimiza el radio de efecto de los errores)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328) (Fragmentación aleatoria: AWS re:Invent 2019: Presentación de Amazon Builders' Library)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience (AWS Summit ANZ 2021: Todo falla todo el tiempo: diseñar para la resiliencia)

Ejemplos relacionados:

- Well-Architected Lab - Fault isolation with shuffle sharding (Laboratorio de Well-Architected: Aislamiento de errores con fragmentación aleatoria)

## Diseñar su carga de trabajo para que soporte los errores de los componentes

Las cargas de trabajo con un requisito de alta disponibilidad y un tiempo de recuperación (MTTR) bajo deben diseñarse para que sean resilientes.

Prácticas recomendadas

- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores
- REL11-BP02 Conmutación por error a recursos en buen estado
- REL11-BP03 Automatizar la reparación en todas las capas
- REL11-BP04 Confiar en el plano de datos y no en el plano de control durante la recuperación

- REL11-BP05 Usar la estabilidad estática para evitar el comportamiento bimodal
- REL11-BP06 Enviar notificaciones cuando los eventos afecten a la disponibilidad
- REL11-BP07 Diseñar su producto para cumplir objetivos de disponibilidad y acuerdos de nivel de servicio (SLA) de tiempo de actividad

## REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores

Supervise continuamente el estado de las cargas de trabajo para que usted y los sistemas automatizados sepan cuándo se produce degradaciones o errores en cuanto ocurran. Supervise los indicadores clave de rendimiento (KPI) en función del valor empresarial.

Todos los mecanismos de recuperación y corrección deben comenzar por la capacidad de detectar problemas rápidamente. Los fallos técnicos deberían detectarse en primer lugar para poder resolverse. Sin embargo, la disponibilidad se basa en la capacidad de su carga de trabajo para ofrecer valor empresarial, de modo que los indicadores clave de rendimiento (KPI) que midan esto tengan que formar parte de su estrategia de detección y corrección.

Resultado deseado: los componentes esenciales de una carga de trabajo se supervisan de forma independiente para detectar y alertar sobre los errores en el momento y el lugar en que se producen.

Patrones comunes de uso no recomendados:

- No se han configurado alarmas, por lo que las interrupciones se producen sin notificación.
- Existen alarmas, pero en umbrales que no proporcionan el tiempo necesario para reaccionar.
- No se recopilan métricas con la suficiente regularidad para satisfacer el objetivo de tiempo de recuperación (RTO).
- Solo se supervisan activamente las interfaces de la carga de trabajo orientadas a los clientes.
- Solo se recopilan métricas técnicas, no métricas de funciones empresariales.
- No hay métricas que midan la experiencia del usuario con la carga de trabajo.
- Se crean demasiadas supervisiones.

Beneficios de establecer esta práctica recomendada: Una supervisión adecuada de todas las capas le permite reducir el tiempo de recuperación al reducirse el tiempo de detección.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Identifique todas las cargas de trabajo que se revisarán para su supervisión. Una vez que haya identificado todos los componentes de la carga de trabajo que deberán supervisarse, tendrá que determinar el intervalo de supervisión. El intervalo de supervisión tendrá un impacto directo en la rapidez con la que se puede iniciar la recuperación en función del tiempo que se tarde en detectar un error. El tiempo medio de detección (MTTD) es el tiempo transcurrido entre la aparición de un error y el inicio de las operaciones de reparación. La lista de servicios debe ser amplia y completa.

La supervisión debe cubrir todas las capas de la pila de aplicaciones, incluidas la aplicación, la plataforma, la infraestructura y la red.

Su estrategia de supervisión debe considerar el impacto de los errores grises. Para obtener más información sobre los errores grises, consulte la sección de errores grises en el documento técnico Advanced Multi-AZ Resilience Patterns.

Pasos para la implementación

- El intervalo de supervisión depende de la rapidez con la que deba recuperarse. El tiempo de recuperación depende del tiempo que tarde la recuperación, por lo que debe determinar la frecuencia de recopilación teniendo en cuenta este tiempo y el objetivo de tiempo de recuperación (RTO).
- Configure la supervisión detallada de los componentes y los servicios administrados.
  - Determine si supervisión detallada de instancias de EC2 y Auto Scaling es necesaria. La supervisión detallada proporciona métricas en intervalos de un minuto y la supervisión predeterminada proporciona métricas en intervalos de cinco minutos.
  - Determine si supervisión mejorada para RDS es necesaria. La supervisión mejorada usa un agente en las instancias de RDS para obtener información útil sobre los diferentes procesos o subprocesos.
  - Determine los requisitos de supervisión de los componentes sin servidor cruciales para Lambda, API Gateway, Amazon EKS, Amazon ECSy todos los tipos de equilibradores de carga.
  - Determine los requisitos de supervisión de los componentes de almacenamiento para Amazon S3, Amazon FSx, Amazon EFSy Amazon EBS.
- Cree métricas personalizadas para medir los indicadores clave de rendimiento (KPI) de la empresa. Las cargas de trabajo implementan funciones empresariales clave, que deben usarse como KPI para ayudar a identificar cuándo se produce un problema indirecto.

- Supervise la experiencia del usuario para detectar errores mediante valores controlados del usuario. Las pruebas de transacciones sintéticas (también denominadas pruebas de valores controlados, que no deben confundirse con los despliegues de valores controlados) que puedan ejecutar y simular el comportamiento de los clientes son uno de los procesos de prueba más importantes. Ejecute estas pruebas constantemente en los puntos de conexión de las cargas de trabajo desde distintas ubicaciones remotas.
- Cree métricas personalizadas que siguen la experiencia del usuario. Si puede instrumentar la experiencia del cliente, puede determinar cuándo se degrada la experiencia del cliente.
- Configure alarmas para detectar cuándo alguna parte de la carga de trabajo no funciona correctamente y para indicar cuándo escalar automáticamente los recursos. Las alarmas pueden mostrarse visualmente en paneles, enviar alertas a través de Amazon SNS o por correo electrónico y trabajar con Auto Scaling para escalar o desescalar verticalmente los recursos de la carga de trabajo.
- Cree paneles para visualizar las métricas. Se pueden usar paneles para visualizar las tendencias, los valores atípicos y otros indicadores de problemas potenciales, o para proporcionar una indicación de problemas que tal vez le convenga investigar.
- Cree supervisión de rastreo distribuido para sus servicios. Con la supervisión distribuida, podrá saber cómo se comporta su aplicación y sus servicios subyacentes para identificar y resolver la causa raíz de los problemas y errores de rendimiento.
- Cree paneles de sistemas de supervisión (mediante CloudWatch o bien X-Ray) y recopilación de datos en una región y una cuenta independientes.
- Cree una integración para la supervisión de Amazon Health Aware para poder supervisar la visibilidad de los recursos de AWS que podrían estar degradados. Para las cargas de trabajo empresariales esenciales, esta solución proporciona acceso a alertas proactivas y en tiempo real para los servicios de AWS.

## Recursos

Prácticas recomendadas relacionadas:

- Definición de disponibilidad
- REL11-BP06 Enviar notificaciones cuando los eventos afecten a la disponibilidad

Documentos relacionados:

- Amazon CloudWatch Synthetics enables you to create user canaries
- Habilitar o deshabilitar la supervisión detallada de su instancia
- Monitoreo mejorado
- Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch
- Publicar métricas personalizadas
- Using Amazon CloudWatch Alarms
- Using CloudWatch Dashboards
- Using Cross Region Cross Account CloudWatch Dashboards
- Using Cross Region Cross Account X-Ray Tracing
- Understanding availability
- Implementing Amazon Health Aware (AHA)

Vídeos relacionados:

- Mitigating gray failures

Ejemplos relacionados:

- Laboratorio de Well-Architected: Nivel 300: Implementación de comprobaciones de estado y administración de dependencias para mejorar la fiabilidad
- One Observability Workshop: Explore X-Ray

Herramientas relacionadas:

- CloudWatch
- CloudWatch X-Ray

## REL11-BP02 Conmutación por error a recursos en buen estado

Si un recurso fallara, los recursos en buen estado deberían seguir atendiendo las solicitudes. Para problemas de ubicación (como zonas de disponibilidad o Región de AWS), asegúrese de que dispone de sistemas para conmutar por error a recursos en buen estado en ubicaciones sin problemas.

Al diseñar un servicio, distribuya la carga entre los recursos, las zonas de disponibilidad o las regiones. De esta manera, el error de un recurso individual o el deterioro puede mitigarse al desplazar el tráfico a los recursos restantes en buen estado. Tenga en cuenta cómo se descubren los servicios y cómo se enruta a ellos en caso de que se produzca un error.

Tenga en cuenta la recuperación de errores al diseñar sus servicios. En AWS, diseñamos servicios para minimizar el tiempo de recuperación de los errores y el impacto en los datos. Nuestros servicios utilizan principalmente almacenes de datos que confirman las solicitudes solo después de que se almacenan de forma duradera en varias réplicas en una región. Se han diseñado para utilizar el aislamiento basado en celdas y el aislamiento de errores que proporcionan las zonas de disponibilidad. Utilizamos ampliamente la automatización en nuestros procedimientos operativos. También optimizamos nuestra funcionalidad de reemplazo y reinicio para recuperarnos rápidamente de las interrupciones.

Los patrones y diseños que permiten la conmutación por error varían para cada servicio de plataforma de AWS. Muchos servicios administrados nativos de AWS son zonas de disponibilidad múltiples de forma nativa (como Lambda o API Gateway). Otros servicios de AWS (como EC2 y EKS) requieren diseños específicos de las prácticas recomendadas para admitir la conmutación por error de los recursos o el almacenamiento de datos en las AZ.

La supervisión debe configurarse para que compruebe que el recurso de conmutación por error esté en buen estado, realizar un seguimiento del progreso de los recursos de conmutación por error y supervisar la recuperación de los procesos empresariales.

Resultado deseado: los sistemas son capaces de utilizar nuevos recursos de forma automática o manual para recuperarse de la degradación.

Patrones comunes de uso no recomendados:

- La planificación para errores no forma parte de la fase de planificación y diseño.
- No se establecen el RTO y el RPO.
- Supervisión insuficiente para detectar recursos defectuosos.
- Aislamiento adecuado de los dominios de error.
- No se considera la conmutación por error multirregional.
- La detección de errores es demasiado sensible o agresiva a la hora de decidir realizar una conmutación por error.
- No se prueba ni se valida el diseño de la conmutación por error.

- Realizar la automatización de la autorreparación, pero no notificar que se necesita una reparación
- No hay un período de amortiguación para evitar que la conmutación por error se lleve a cabo demasiado pronto.

Beneficios de establecer esta práctica recomendada: con una degradación uniforme y una recuperación rápida, puede crear sistemas más resilientes que mantengan la fiabilidad cuando se producen errores.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Los servicios de AWS, como Elastic Load Balancing y Amazon EC2 Auto Scaling, ayudan a distribuir la carga entre los recursos y las zonas de disponibilidad. Por lo tanto, el error de un recurso individual (como una instancia de EC2) o el deterioro de una zona de disponibilidad puede mitigarse si se desplaza el tráfico a los recursos restantes en buen estado.

Para las cargas de trabajo multirregión, los diseños son más complicados. Por ejemplo, las réplicas de lectura entre regiones le permiten desplegar sus datos en varias Regiones de AWS. Sin embargo, la conmutación por error sigue siendo necesaria para convertir la réplica de lectura en principal y, a continuación, dirigir el tráfico al nuevo punto de conexión. Amazon Route 53, Route 53 Route 53 ARC, CloudFront y AWS Global Accelerator pueden ayudar a dirigir el tráfico a través de las Regiones de AWS.

Los servicios de AWS, como Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge o Amazon DynamoDB, se despliegan automáticamente en varias zonas de disponibilidad mediante AWS. En caso de error, estos servicios de AWS dirigen automáticamente el tráfico a ubicaciones en buen estado. Los datos se almacenan de forma redundante en varias zonas de disponibilidad y siguen estando disponibles.

Para Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS o Amazon ECS, Multi-AZ es una opción de configuración. AWS puede dirigir el tráfico a la instancia en buen estado si se inicia la conmutación por error. Esta acción de conmutación por error puede ser realizada por AWS o según lo requiera el cliente.

Para las instancias de Amazon EC2, Amazon Redshift, tareas de Amazon ECS o pods de Amazon EKS, usted elige en qué zonas de disponibilidad desea realizar el despliegue. En algunos diseños,

Elastic Load Balancing proporciona la solución para detectar las instancias en las zonas que no tienen un estado correcto y enrutar el tráfico a las que sí lo tienen. Elastic Load Balancing también puede enrutar el tráfico a los componentes de su centro de datos local.

En cuanto a la conmutación por error del tráfico multirregional, el reenrutamiento puede utilizar Amazon Route 53, Route 53 ARC, AWS Global Accelerator, Route 53 Private DNS for VPCs o CloudFront para proporcionar una forma de definir dominios de Internet y asignar políticas de enrutamiento, incluidas comprobaciones de estado, para enrutar el tráfico a regiones en buen estado. AWS Global Accelerator proporciona direcciones IP estáticas que actúan como punto de entrada fijo a su aplicación; a continuación, se enrutan a los puntos de conexión de las Regiones de AWS que elija, mediante la red global de AWS en lugar de Internet para mejorar el rendimiento y la fiabilidad.

Pasos para la implementación

- Cree diseños de conmutación por error para todas las aplicaciones y servicios pertinentes. Aísle cada componente de la arquitectura y cree diseños de conmutación por error que satisfagan el RTO y el RPO de cada componente.
- Configure entornos inferiores (como los de desarrollo o prueba) con todos los servicios que sean necesarios para tener un plan de conmutación por error. Despliegue las soluciones mediante la infraestructura como código (IaC) para garantizar la repetibilidad.
- Configure un sitio de recuperación, como una segunda región, para implementar y probar los diseños de conmutación por error. Si fuera necesario, los recursos para las pruebas se pueden configurar temporalmente para limitar los costes adicionales.
- Determine qué planes de conmutación por error se automatizan mediante AWS, cuáles pueden automatizarse mediante un proceso de DevOps y cuáles pueden ser manuales. Documente y mida el RTO y el RPO de cada servicio.
- Cree una guía de estrategias de conmutación por error e incluya todos los pasos de la conmutación por error de cada recurso, aplicación y servicio.
- Cree una guía de estrategias de conmutación por recuperación e incluya todos los pasos de la conmutación por recuperación (con plazos) de cada recurso, aplicación y servicio.
- Cree un plan para iniciar y ensayar la guía de estrategias. Utilice simulaciones y pruebas de caos para poner a prueba los pasos de la guía de estrategias y la automatización.
- Para problemas de ubicación (como zonas de disponibilidad o Región de AWS), asegúrese de que dispone de sistemas para conmutar por error a recursos en buen estado en ubicaciones sin problemas. Compruebe la cuota, los niveles de escalado automático y los recursos en ejecución antes de realizar la prueba de conmutación por error.

## Recursos

Prácticas recomendadas por Well-Architected:

- REL13: Plan para DR
- REL10: Uso del aislamiento de errores para proteger la carga de trabajo

Documentos relacionados:

- Setting RTO and RPO Targets
- Set up Route 53 ARC with application loadbalancers
- Failover using Route 53 Weighted routing
- DR with Route 53 ARC
- EC2 with autoscaling
- EC2 Deployments - Multi-AZ
- ECS Deployments - Multi-AZ
- Switch traffic using Route 53 ARC
- Lambda with an Application Load Balancer and Failover
- ACM Replication and Failover
- Parameter Store Replication and Failover
- ECR cross region replication and Failover
- Secrets manager cross region replication configuration
- Enable cross region replication for EFS and Failover
- EFS Cross Region Replication and Failover
- Networking Failover
- S3 Endpoint failover using MRAP
- Create cross region replication for S3
- Failover Regional API Gateway with Route 53 ARC
- Failover using multi-region global accelerator
- Failover with DRS
- Creating Disaster Recovery Mechanisms Using Amazon Route 53

Ejemplos relacionados:

- Disaster Recovery on AWS
- Elastic Disaster Recovery on AWS

## REL11-BP03 Automatizar la reparación en todas las capas

Cuando se detecte un error, utilice las funciones automatizadas para tomar medidas correctivas. Las degradaciones pueden repararse automáticamente a través de mecanismos de servicio interno o requerir que los recursos se reinicien o eliminen a través de medidas de corrección.

Para las aplicaciones autoadministradas y la reparación entre regiones, los diseños de recuperación y los procesos de reparación automatizados se pueden extraer de las prácticas recomendadas existentes.

La capacidad de reiniciar o eliminar un recurso es una herramienta importante para corregir los errores. Una práctica recomendada es convertir los servicios en servicios sin estado siempre que sea posible. Esto evita la pérdida de datos o disponibilidad tras el reinicio del recurso. En la nube, puede (y generalmente debería) sustituir todo el recurso (por ejemplo, la instancia de computación o la función sin servidor) como parte del reinicio. El reinicio en sí es una forma sencilla y fiable de recuperarse de un error. En las cargas de trabajo ocurren muchos tipos de errores diferentes. Los errores pueden ocurrir en el hardware, el software, las comunicaciones y el funcionamiento.

El reinicio o el reintento también se aplican a las solicitudes de red. Se aplica el mismo enfoque de recuperación tanto a un tiempo de espera de la red como a un error en la dependencia, en el que la dependencia devuelve un error. Ambos eventos tienen un efecto similar en el sistema, por lo que en lugar de intentar convertir cada uno en un caso especial, se aplicaría una estrategia similar de reintento con retroceso exponencial y fluctuación. La capacidad de reiniciar es un mecanismo de recuperación que aparece en la computación orientada a la recuperación y en las arquitecturas de clústeres de alta disponibilidad.

Resultado deseado: se llevan a cabo medidas automatizadas para corregir la detección de un error.

Patrones comunes de uso no recomendados:

- Aprovisionar recursos sin escalado automático.
- Desplegar las aplicaciones en instancias o contenedores individualmente.
- Implementar aplicaciones que no se pueden implementar en varias ubicaciones sin usar la recuperación automática

- Reparar manualmente las aplicaciones que el escalamiento automático y la recuperación automática no pueden reparar.
- No hay automatización de las bases de datos de conmutación por error.
- Carencia de métodos automatizados para redirigir el tráfico a nuevos puntos de conexión.
- No hay replicación del almacenamiento.

Beneficios de establecer esta práctica recomendada: la reparación automática puede reducir el tiempo medio de recuperación y mejorar la disponibilidad.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Los diseños de Amazon EKS u otros servicios de Kubernetes deben incluir conjuntos de réplicas o con estado mínimo y máximo y el tamaño mínimo del clúster y los grupos de nodos. Estos mecanismos proporcionan una cantidad mínima de recursos de procesamiento disponibles de forma continua y, al mismo tiempo, corrigen automáticamente cualquier error mediante el plano de control de Kubernetes.

Los patrones de diseño a los que se accede a través de un equilibrador de carga mediante clústeres de computación deben utilizar los grupos de Auto Scaling. Elastic Load Balancing (ELB) distribuye automáticamente el tráfico de aplicaciones entrante entre varios destinos y dispositivos virtuales en una o más zonas de disponibilidad (AZ).

Los diseños basados en computación en clúster que no utilizan el equilibrio de carga deben tener un diseño de tamaño que de cabida a la pérdida de al menos un nodo. Esto permitirá que el servicio siga funcionando con una capacidad potencialmente reducida mientras recupera un nuevo nodo. Algunos servicios son Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK y Amazon OpenSearch Service. Muchos de estos servicios se pueden diseñar con características adicionales de autorreparación. Algunas tecnologías de clústeres deben generar una alerta ante la pérdida de un nodo, lo que desencadena un flujo de trabajo automático o manual para recrear un nuevo nodo. Este flujo de trabajo se puede automatizar con AWS Systems Manager para corregir los problemas rápidamente.

Se puede usar Amazon EventBridge para supervisar y filtrar los eventos, como las alarmas de Amazon EC2 Auto Scaling o cambios en el estado en otros servicios de AWS. En función de la información del evento, se puede invocar a AWS Lambda, la automatización de Systems Manager

u otros destinos para ejecutar una lógica de corrección personalizada en su carga de trabajo. Amazon EC2 Auto Scaling se puede configurar para comprobar el estado de la instancia de EC2. Si la instancia está en un estado que no sea el de ejecución, o si el estado del sistema se ve deteriorado, Amazon EC2 Auto Scaling considera que la instancia no está en buen estado y lanza una instancia de sustitución. Para sustituciones a gran escala (como la pérdida de toda una zona de disponibilidad), se prefiere la estabilidad estática para la alta disponibilidad.

Pasos para la implementación

- Use grupos de Auto Scaling para desplegar niveles en una carga de trabajo. Auto Scaling puede realizar una autorreparación de aplicaciones sin estado, y añadir y eliminar capacidad.
- Para las instancias de computación indicadas anteriormente, utilice el equilibrio de carga y elija el tipo de equilibrador de carga adecuado.
- Considere la posibilidad de reparación para Amazon RDS. Con las instancias en espera, configure la conmutación por error automática a la instancia en espera. Para la réplica de lectura de Amazon RDS, se requiere un flujo de trabajo automatizado para convertir una réplica de lectura en principal.
- Implemente la recuperación automática en instancias de EC2 que tengan aplicaciones desplegadas que no se puedan desplegar en varias ubicaciones y puedan tolerar el reinicio tras un error. La recuperación automática se puede usar para reemplazar hardware defectuoso y reiniciar la instancia cuando la aplicación no se puede implementar en varias ubicaciones. Se conservan los metadatos de la instancia y las direcciones IP asociadas, así como los volúmenes de EBS y los puntos de montaje en Amazon Elastic File System o bien sistemas de archivos para Lustre y Windows. Con AWS OpsWorks, puede configurar la autorreparación de las instancias de EC2 en el nivel de capa.
- Implemente la recuperación automatizada mediante AWS Step Functions y AWS Lambda cuando no pueda usar el escalamiento automático ni la recuperación automática, o cuando la recuperación automática produzca un error. Cuando no pueda usar el escalamiento automático ni la recuperación automática, o esta produzca un error, puede automatizar la reparación con AWS Step Functions y AWS Lambda.
- Amazon EventBridge se puede usar para supervisar y filtrar los eventos, como las alarmas de CloudWatch o los cambios en el estado en otros servicios de AWS. En función de la información del evento, se puede invocar a AWS Lambda (u otros destinos) para ejecutar una lógica de corrección personalizada en su carga de trabajo.

## Recursos

Prácticas recomendadas relacionadas:

- Definición de disponibilidad
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores

Documentos relacionados:

- How AWS Auto Scaling Works
- Amazon EC2 Automatic Recovery
- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- What is Amazon FSx for Lustre?
- What is Amazon FSx for Windows File Server?
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances
- What isAWS Step Functions?
- What is AWS Lambda?
- What is Amazon EventBridge?
- Using Amazon CloudWatch Alarms
- Amazon RDS Failover
- SSM - Systems Manager Automation
- Resilient Architecture Best Practices

Vídeos relacionados:

- Automatically Provision and Scale OpenSearch Service
- Amazon RDS Failover Automatically

Ejemplos relacionados:

- Workshop on Auto Scaling
- Amazon RDS Failover Workshop

Herramientas relacionadas:

- CloudWatch
- CloudWatch X-Ray

## REL11-BP04 Confiar en el plano de datos y no en el plano de control durante la recuperación

Los planos de control proporcionan las API administrativas que se utilizan para crear, leer y describir, actualizar, eliminar y enumerar los recursos (CRUDL), mientras que los planos de datos gestionan el tráfico de servicio diario. Al implementar respuestas de recuperación o mitigación a eventos que puedan afectar a la resiliencia, concéntrese en utilizar un número mínimo de operaciones del plano de control para recuperar, reescalar, restaurar, reparar o conmutar por error el servicio. La acción del plano de datos debe reemplazar cualquier actividad durante estos eventos de degradación.

Por ejemplo, las siguientes son todas acciones del plano de control: lanzar una nueva instancia de computación, crear almacenamiento en bloques y describir los servicios de colas. Al lanzar instancias de computación, el plano de control debe realizar varias tareas, como encontrar un host físico con capacidad, asignar interfaces de red, preparar los volúmenes de almacenamiento en bloques locales, generar credenciales y añadir reglas de seguridad. Los planos de control suelen tener una orquestación complicada.

Resultado deseado: cuando un recurso entra en un estado deteriorado, el sistema es capaz de recuperarse automática o manualmente al cambiar el tráfico de recursos deteriorados a recursos en buen estado.

Patrones comunes de uso no recomendados:

- Dependencia de cambiar los registros de DNS para redirigir el tráfico.
- Dependencia de las operaciones de escalado del plano de control para reemplazar los componentes dañados debido a que no se han aprovisionado suficientes recursos.
- Confiar en amplias acciones del plano de control, multiservicio y multiAPI para corregir cualquier categoría de deterioro.

Beneficios de establecer esta práctica recomendada: el aumento de la tasa de éxito de la corrección automatizada puede reducir el tiempo medio de recuperación y mejorar la disponibilidad de la carga de trabajo.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio. En ciertos tipos de degradaciones del servicio, los planos de control se ven afectados. La dependencia del uso extensivo del plano de control para la corrección puede aumentar el tiempo de recuperación (RTO) y el tiempo medio de recuperación (MTTR).

## Guía para la implementación

Para limitar las acciones del plano de datos, evalúe cada servicio para determinar qué acciones son necesarias para restablecer el servicio.

Utilice Amazon Route 53 Application Recovery Controller para cambiar el tráfico de DNS. Estas características supervisan continuamente la capacidad de la aplicación de recuperarse de los errores y le permiten controlar la recuperación de la aplicación en las distintas Regiones de AWS, zonas de disponibilidad y localmente.

Las políticas de enrutamiento de Route 53 utilizan el plano de control, por lo que no debe confiar en él para la recuperación. Los planos de datos de Route 53 responden a consultas de DNS y llevan a cabo y evalúan comprobaciones de estado. Se distribuyen a nivel mundial y están diseñados para un acuerdo de nivel de servicio (SLA) del 100 % de disponibilidad..

Las API de administración de Route 53 y las consolas en las que se crean, actualizan y eliminan recursos de Route 53 se ejecutan en planos de control diseñados para dar prioridad a la sólida coherencia y durabilidad que necesita al administrar DNS. Para conseguirlo, los planos de control se encuentran en una única región: Este de EE. UU. (Norte de Virginia). Aunque ambos sistemas se han diseñado para ser muy fiables, los planos de control no están incluidos en el SLA. Podría haber eventos poco frecuentes en los que el diseño resiliente del plano de datos permita mantener la disponibilidad mientras que los planos de control no lo permitan. Con los mecanismos de recuperación de desastres y conmutación por error, utilice las funciones del plano de datos para proporcionar la mejor fiabilidad posible.

Para Amazon EC2, utilice diseños de estabilidad estática para limitar las acciones del plano de control. Las acciones del plano de control incluyen la ampliación de los recursos de forma individual o mediante grupos de Auto Scaling (ASG). Para obtener los niveles más altos de resiliencia, aprovisione suficiente capacidad en el clúster utilizado para la conmutación por error. Si este umbral de capacidad debe limitarse, establezca reguladores en todo el sistema de principio a fin para limitar de forma segura el tráfico total que llega al conjunto limitado de recursos.

Para servicios como Amazon DynamoDB, Amazon API Gateway, los equilibradores de carga y los servicios de AWS Lambda sin servidor, el uso de esos servicios utiliza el plano de datos. Sin

embargo, la creación de nuevas funciones, equilibradores de carga, puertas de enlace de API o tablas de DynamoDB es una acción del plano de control y debe completarse antes de la degradación como preparación para un evento y ensayo de las acciones de conmutación por error. En el caso de Amazon RDS, las acciones del plano de datos permiten el acceso a los datos.

Para obtener más información sobre los planos de datos, los planos de control y cómo AWS crea servicios para cumplir los objetivos de alta disponibilidad, consulte Estabilidad estática con zonas de disponibilidad.

Comprenda qué operaciones están en el plano de datos y cuáles están en el plano de control.

Pasos para la implementación

Para cada carga de trabajo que deba restaurarse después de un evento de degradación, evalúe el runbook de conmutación por error, el diseño de alta disponibilidad, el diseño de reparación automática o el plan de restauración de recursos de alta disponibilidad. Identifique cada acción que pueda considerarse una acción del plano de control.

Considere cambiar la acción de control por una acción del plano de datos:

- Auto Scaling (plano de control) en comparación con los recursos de Amazon EC2 preescalados (plano de datos).
- Migre a Lambda y sus métodos de escalado (plano de datos) o a Amazon EC2 y ASG (plano de control).
- Evalúe cualquier diseño con Kubernetes y la naturaleza de las acciones del plano de control. Añadir pods es una acción del plano de datos en Kubernetes. Las acciones deben limitarse a añadir pods y no a añadir nodos. Con nodos sobreaprovisionados es el método preferido para limitar las acciones del plano de control.

Considere enfoques alternativos que permitan que las acciones del plano de datos afecten a la misma corrección.

- Cambio de registro de Route 53 (plano de control) o Route 53 ARC (plano de datos).
- Comprobaciones de estado de Route 53 para obtener actualizaciones más automatizadas.

Considere algunos servicios en una región secundaria, si el servicio es crucial para la misión, para permitir más acciones del plano de control y el plano de datos en una región no afectada.

- Amazon EC2 Auto Scaling o Amazon EKS en una región principal en comparación con Amazon EC2 Auto Scaling o Amazon EKS en una región secundaria y enrutamiento del tráfico a la región secundaria (acción del plano de control).
- Hacer réplicas de lectura en la región principal secundaria o intentar la misma acción en la región principal (acción del plano de control).

## Recursos

Prácticas recomendadas relacionadas:

- Definición de disponibilidad
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores

Documentos relacionados:

- APN Partner: socios que pueden ayudar con la automatización de su tolerancia a errores
- AWS Marketplace: productos que pueden usarse para tolerancia a errores
- La Amazon Builders' Library: Evitar la sobrecarga de los sistemas distribuidos asumiendo el control del servicio más pequeño
- API de Amazon DynamoDB (plano de control y plano de datos)
- AWS Lambda Executions (divididas entre el plano de control y el plano de datos)
- AWS Elemental MediaStore Data Plane
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack
- Creating Disaster Recovery Mechanisms Using Amazon Route 53
- What is Route 53 Application Recovery Controller
- Kubernetes Control Plane and data plane

Vídeos relacionados:

- Back to Basics - Using Static Stability

- Building resilient multi-site workloads using AWS global services

Ejemplos relacionados:

- Introducing Amazon Route 53 Application Recovery Controller
- La Amazon Builders' Library: Evitar la sobrecarga de los sistemas distribuidos asumiendo el control del servicio más pequeño
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 2: Multi-Region stack
- Estabilidad estática con zonas de disponibilidad

Herramientas relacionadas:

- Amazon CloudWatch
- AWS X-Ray

## REL11-BP05 Usar la estabilidad estática para evitar el comportamiento bimodal

Las cargas de trabajo deben ser estáticamente estables y funcionar solo en un único modo normal. El comportamiento bimodal se produce cuando la carga de trabajo presenta un comportamiento diferente en los modos normal y de error.

Por ejemplo, puede intentar recuperarse de un error en una zona de disponibilidad lanzando nuevas instancias en una zona de disponibilidad diferente. Esto puede dar como resultado una respuesta bimodal durante un modo de error. En lugar de ello, debe crear cargas de trabajo que sean estables estáticamente y operen dentro de un solo modo. En este ejemplo, esas instancias deberían haberse aprovisionado en la segunda zona de disponibilidad antes del error. Este diseño de estabilidad estática verifica que la carga de trabajo solo funcione en un solo modo.

Resultado deseado: las cargas de trabajo no muestran un comportamiento bimodal durante los modos normal y de error.

Patrones comunes de uso no recomendados:

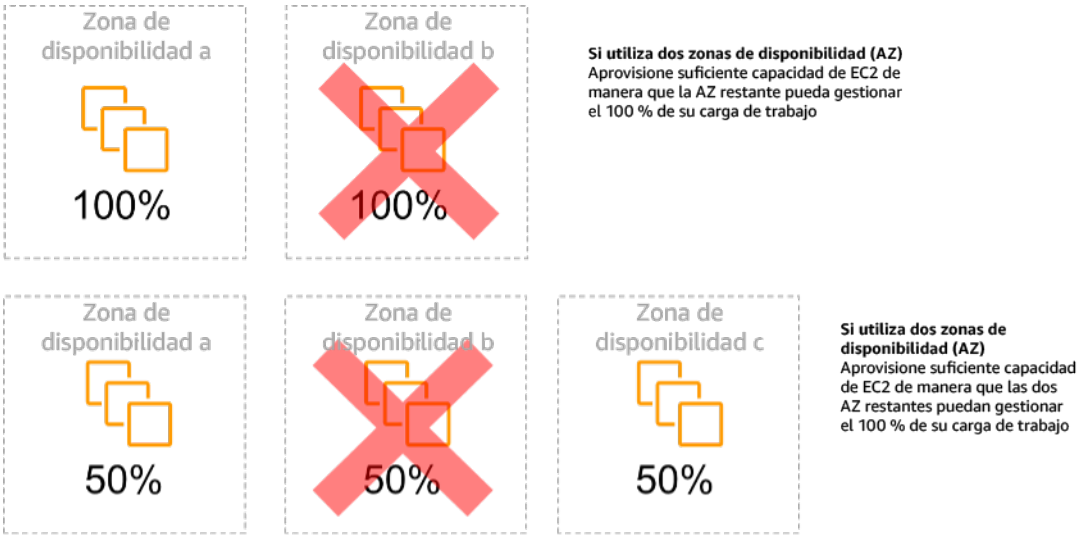
- Suponer que los recursos siempre se pueden aprovisionar independientemente del alcance del error.
- Intentar adquirir recursos de forma dinámica durante un error.
- No aprovisionar los recursos adecuados en todas las zonas o regiones hasta que se produzca un error.
- Considerar diseños estáticos estables solo para recursos de computación.

Beneficios de establecer esta práctica recomendada: las cargas de trabajo que se ejecutan con diseños estáticamente estables pueden tener resultados predecibles durante eventos normales y de error.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio

## Guía para la implementación

El comportamiento bimodal ocurre cuando la carga de trabajo exhibe diferentes comportamientos en los modos normal y de error (como confiar en el lanzamiento de nuevas instancias si se produce un error en una zona de disponibilidad). Un ejemplo de comportamiento bimodal ocurre cuando los diseños de Amazon EC2 estables aprovisionan suficientes instancias en cada zona de disponibilidad para gestionar la carga de trabajo si se eliminara una de estas zonas. Se comprobaría el estado de Elastic Load Balancing o Amazon Route 53 para retirar una carga de las instancias dañadas. Una vez que el tráfico ha cambiado, use AWS Auto Scaling para sustituir de forma asíncrona las instancias de la zona con errores y lanzarlas en las zonas en buen estado. La estabilidad estática para despliegues de computación (como instancias EC2 o contenedores) da como resultado la máxima fiabilidad.



Estabilidad estática de las instancias EC2 entre zonas de disponibilidad

Esto debe sopesarse en relación al coste de este modelo y el valor empresarial de mantener la carga de trabajo en todos los casos de resiliencia. Es menos caro aprovisionar menos capacidad de computación y confiar en el lanzamiento de nuevas instancias en caso de error, pero en el caso de errores a gran escala (como una deterioro regional o de zona de disponibilidad), este enfoque es menos eficaz porque se basa tanto en un plano operativo como en la disponibilidad de recursos suficientes en las zonas o regiones no afectadas.

Su solución también debe sopesar la fiabilidad en comparación con los costes necesarios para su carga de trabajo. Las arquitecturas de estabilidad estática se aplican a una variedad de arquitecturas, incluidas las instancias de computación distribuidas en las zonas de disponibilidad, los diseños de réplicas de lectura de bases de datos, los diseños de clústeres de Kubernetes (Amazon EKS) y las arquitecturas de conmutación por error multirregional.

También es posible implementar un diseño más estable desde el punto de vista estático mediante el uso de más recursos en cada zona. Al agregar más zonas, reduce la cantidad de procesamiento adicional que necesita para la estabilidad estática.

Un ejemplo de comportamiento bimodal sería un tiempo de espera de la red que podría provocar que un sistema intente actualizar el estado de configuración de todo el sistema. Se añadiría una carga inesperada a otro componente, lo que podría hacer que falle y desencadene otras consecuencias inesperadas. Este bucle de retroalimentación negativa afecta a la disponibilidad de su carga de trabajo. En lugar de ello, puede crear cargas de trabajo que sean estables estáticamente y operen en un solo modo. Un diseño estáticamente estable haría un trabajo constante y actualizaría continuamente el estado de configuración a una cadencia establecida. Cuando una llamada genera un error, la carga de trabajo utiliza el valor previamente almacenado en caché e inicia una alarma.

Otro ejemplo de comportamiento bimodal es permitir que los clientes eludan la caché de la carga de trabajo si se produce un error. Esto podría parecer una solución para satisfacer las necesidades del cliente, pero puede cambiar notablemente la demanda de la carga de trabajo y es probable que produzca un error.

Evalúe las cargas de trabajo críticas para determinar cuáles requieren este tipo de diseño de resiliencia. Se debe revisar cada componente de la aplicación en las cargas que se consideren cruciales. Algunos tipos de servicios que requieren evaluaciones de estabilidad estática son:

- Computación: Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- Bases de datos: Amazon Redshift, Amazon RDS, Amazon Aurora

- Storage (Almacenamiento): Amazon S3 (zona única), Amazon EFS (montajes), Amazon FSx (montajes)
- Equilibradores de carga: en ciertos diseños

Pasos para la implementación

- Cree cargas de trabajo que sean estables estáticamente y operen en un solo modo. En este caso, aprovisione suficientes instancias en cada región o zona de disponibilidad para gestionar la capacidad de la carga de trabajo si se eliminara una región o zona de disponibilidad. Puede usar una variedad de servicios para el enrutamiento a recursos en buen estado, como:
  - Enrutamiento de DNS entre regiones
  - Enrutamiento de punto de acceso de varias regiones de Amazon S3
  - AWS Global Accelerator
  - Amazon Route 53 Application Recovery Controller
- Configure las réplicas de lectura de base de datos de modo que tengan en cuenta la pérdida de una única instancia principal o una réplica de lectura. Si las réplicas de lectura atienden el tráfico, la cantidad en cada zona de disponibilidad y cada región debe ser igual a la necesidad general en caso de que se produzca un error en la zona o región.
- Configure los datos esenciales en el almacenamiento Amazon S3 que está diseñado para ser estáticamente estable para los datos almacenados en caso de que se produzca un error en la zona de disponibilidad. Si se usa la clase de almacenamiento de acceso poco frecuente en una única zona de Amazon S3, no debe considerarse estable desde el punto de vista estático, ya que la pérdida de esa zona minimiza el acceso a los datos almacenados.
- Los equilibradores de carga a veces están configurados incorrectamente o por diseño para prestar servicio a una zona de disponibilidad específica. En este caso, el diseño estáticamente estable podría consistir en distribuir una carga de trabajo entre varias zonas de disponibilidad en un diseño más complejo. El diseño original se puede utilizar para reducir el tráfico entre zonas por motivos de seguridad, latencia o coste.

## Recursos

Prácticas recomendadas por Well-Architected:

- Definición de disponibilidad
- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores

- REL11-BP04 Confiar en el plano de datos y no en el plano de control durante la recuperación

Documentos relacionados:

- Minimizar las dependencias en un plan de recuperación de desastres
- La Amazon Builders' Library: Estabilidad estática con zonas de disponibilidad
- Límites de aislamiento de errores
- Estabilidad estática con zonas de disponibilidad
- RDS multizona
- Minimizar las dependencias en un plan de recuperación de desastres
- Enrutamiento de DNS entre regiones
- Enrutamiento de punto de acceso de varias regiones de Amazon S3
- AWS Global Accelerator
- Route 53 ARC
- Zona única de Amazon S3
- Equilibrio de carga entre zonas

Vídeos relacionados:

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

Ejemplos relacionados:

- La Amazon Builders' Library: Estabilidad estática con zonas de disponibilidad

## REL11-BP06 Enviar notificaciones cuando los eventos afecten a la disponibilidad

Se envían notificaciones cuando se detecta que se han superado los umbrales, incluso si el evento que causó el problema se ha resuelto automáticamente.

La corrección automática permite que la carga de trabajo sea fiable. Sin embargo, también puede ocultar problemas subyacentes que deberían abordarse. Implemente una supervisión y unos

eventos apropiados para poder detectar patrones de problemas, incluidos los que pueden abordarse mediante corrección automática, para que pueda resolver los problemas de la causa principal.

Los sistemas resilientes están diseñados para que los eventos de degradación se comuniquen inmediatamente a los equipos correspondientes. Estas notificaciones deben enviarse a través de uno o varios canales de comunicación.

Resultado deseado: las alertas se envían inmediatamente a los equipos de operaciones cuando se superan los umbrales, como las tasas de error, la latencia u otras métricas cruciales de los indicadores clave de rendimiento (KPI), para que estos problemas se resuelvan lo antes posible y se evite o minimice el impacto en los usuarios.

Patrones comunes de uso no recomendados:

- Enviar demasiadas alarmas.
- Enviar alarmas que no son procesables.
- Establecer umbrales de alarma demasiado altos (muy sensibles) o demasiado bajos (poco sensibles).
- No enviar alarmas para dependencias externas.
- No tener en cuenta los errores grises al diseñar la supervisión y las alarmas.
- Realizar la automatización de la reparación, pero sin notificar al equipo adecuado que se necesita una reparación.

Beneficios de establecer esta práctica recomendada: las notificaciones de recuperación permiten que los equipos operativos y empresariales estén al tanto de las degradaciones del servicio para que puedan reaccionar de inmediato y minimizar tanto el tiempo medio de detección (MTTD) como el tiempo medio de reparación (MTTR). Las notificaciones de los eventos de recuperación también garantizan que no se ignoren problemas que ocurren con poca frecuencia.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio. Si no se implementan los mecanismos adecuados de supervisión y notificación de eventos, es posible que no se detecten patrones de problemas, incluidos los que se pueden abordarse mediante la corrección automática. El equipo solo se descubrirá la degradación del sistema cuando los usuarios contacten con el servicio de atención al cliente o por casualidad.

## Guía para la implementación

Al definir una estrategia de supervisión, la activación de una alarma es un evento frecuente. Es probable que este evento contenga un identificador de la alarma, el estado de la alarma (como `En alarma` o bien `Aceptar`) y detalles sobre qué la desencadenó. En muchos casos, se debe detectar el evento de alarma y enviar una notificación por correo electrónico. Este es un ejemplo de una acción en una alarma. La notificación de alarmas es fundamental en la observabilidad, ya que informa a las personas adecuadas de que existe un problema. Sin embargo, cuando la acción sobre los eventos madura en su solución de observabilidad, puede solucionar el problema automáticamente sin necesidad de intervención humana.

Una vez que se hayan establecido las alarmas de supervisión de los KPI, se deben enviar alertas a los equipos correspondientes cuando se superen los umbrales. Esas alertas también se pueden usar para activar procesos automatizados que intentarán corregir la degradación.

Para una supervisión de umbrales más compleja, se deben considerar las alarmas compuestas. Las alarmas compuestas utilizan una serie de alarmas de supervisión de KPI para crear una alerta basada en la lógica empresarial operativa. Las alarmas de CloudWatch se pueden configurar para enviar correos electrónicos o para registrar incidentes en sistemas de seguimiento de incidentes de terceros mediante la integración con Amazon SNS o Amazon EventBridge.

### Pasos para la implementación

Cree varios tipos de alarmas en función de la forma en que se supervisan las cargas de trabajo, como por ejemplo:

- Las alarmas de las aplicaciones se utilizan para detectar cuando alguna parte de la carga de trabajo no funciona correctamente.
- Las alarmas de infraestructura indican cuándo escalar los recursos. Las alarmas se pueden mostrar visualmente en paneles, enviar alertas a través de Amazon SNS o por correo electrónico y trabajar con Auto Scaling para aumentar o reducir los recursos de la carga de trabajo.
- Se pueden crear alarmas estáticas simples para supervisar cuando una métrica supera un umbral estático durante un número específico de períodos de evaluación.
- Las alarmas compuestas pueden abarcar alarmas complejas de numerosas fuentes.
- Una vez creada la alarma, cree los eventos de notificación adecuados. Puede invocar directamente una API de Amazon SNS para enviar notificaciones y vincular cualquier automatización para su corrección o comunicación.

- Integrar Amazon Health Aware para poder supervisar la visibilidad de los recursos de AWS que podrían estar degradados. Para las cargas de trabajo empresariales esenciales, esta solución proporciona acceso a alertas proactivas y en tiempo real para los servicios de AWS.

### Recursos

Prácticas recomendadas por Well-Architected:

- Definición de disponibilidad

Documentos relacionados:

- Cree una alarma de CloudWatch basada en un umbral estático
- What is Amazon EventBridge?
- ¿Qué es Amazon Simple Notification Service?
- Publicar métricas personalizadas
- Using Amazon CloudWatch Alarms
- Amazon Health Aware (AHA)
- Setup CloudWatch Composite alarms
- What's new in AWS Observability at re:Invent 2022

Herramientas relacionadas:

- CloudWatch
- CloudWatch X-Ray

## REL11-BP07 Diseñar su producto para cumplir objetivos de disponibilidad y acuerdos de nivel de servicio (SLA) de tiempo de actividad

Diseñe su producto para que cumpla objetivos de disponibilidad y acuerdos de nivel de servicio (SLA) de tiempo de actividad. Si publica o acuerda en privado objetivos de disponibilidad o SLA de tiempo de actividad, verifique que su arquitectura y procesos operativos están diseñados para darles cabida.

Resultado deseado: cada aplicación tiene un objetivo definido de disponibilidad y un SLA para las métricas de rendimiento, que pueden supervisarse y mantenerse para alcanzar los resultados empresariales.

Antipatrones usuales:

- Diseño y despliegue de cargas de trabajo sin establecer acuerdos de nivel de servicio.
- Las métricas de los SLA se fijan en niveles altos sin justificación ni requisitos empresariales.
- Establecimiento de SLA sin tener en cuenta las dependencias y sus SLA subyacentes.
- Los diseños de aplicaciones se crean sin tener en cuenta el modelo de responsabilidad compartida para la resiliencia.

Beneficios de establecer esta práctica recomendada: el diseño de aplicaciones basado en objetivos clave de resistencia ayuda a cumplir los objetivos empresariales y las expectativas de los clientes. Estos objetivos contribuyen a impulsar el proceso de diseño de aplicaciones que evalúa diferentes tecnologías y tiene en cuenta varios compromisos.

## Guía para la implementación

El diseño de aplicaciones debe tener en cuenta una serie de requisitos derivados de objetivos empresariales, operativos y financieros. Dentro de los requisitos operativos, las cargas de trabajo deben tener objetivos concretos de métricas de resistencia para que se puedan supervisar y respaldar adecuadamente. Las métricas de resistencia no deben establecerse ni derivarse después de desplegar la carga de trabajo. En cambio, deben definirse durante la fase de diseño y ayudar a orientar diversas decisiones y compromisos.

- Cada carga de trabajo debe tener su propio conjunto de métricas de resistencia. Esas métricas pueden ser diferentes de las de otras aplicaciones empresariales.
- Reducir las dependencias puede tener un efecto positivo en la disponibilidad. Cada carga de trabajo debe considerar sus dependencias y sus SLA. En general, seleccione dependencias con objetivos de disponibilidad iguales o superiores a los objetivos de su carga de trabajo.
- Siempre que sea posible, examine diseños de acoplamiento flexible para que la carga de trabajo pueda funcionar correctamente a pesar del deterioro de las dependencias.
- Reduzca las dependencias del plano de control, especialmente durante la recuperación o una degradación. Evalúe diseños que sean estáticamente estables para las cargas de trabajo cruciales para la misión. Utilice el ahorro de recursos para aumentar la disponibilidad de esas dependencias en una carga de trabajo.

- La observabilidad y la instrumentación son fundamentales para alcanzar los SLA al reducir el tiempo medio de detección (MTTD) y el tiempo medio de reparación (MTTR).
- Los tres factores que se utilizan para mejorar la disponibilidad en los sistemas distribuidos son menos errores frecuentes (MTBF más largo), tiempos de detección de errores más cortos (MTTD más corto) y tiempos de reparación más cortos (MTTR más corto).
- Establecer y cumplir las métricas de resistencia para una carga de trabajo es un elemento imprescindible en todo diseño eficaz. Estos diseños deben tener en cuenta los compromisos de la complejidad del diseño, las dependencias de los servicios, el rendimiento, la escalabilidad y los costes.

Pasos para la implementación

- Revise y documente el diseño de la carga de trabajo teniendo presente las siguientes preguntas:
  - ¿Dónde se utilizan los planos de control en la carga de trabajo?
  - ¿Cómo implementa la carga de trabajo la tolerancia a errores?
  - ¿Cuáles son los patrones de diseño para el escalado, el escalado automático, la redundancia y los componentes de alta disponibilidad?
  - ¿Cuáles son los requisitos de coherencia y disponibilidad de los datos?
  - ¿Se tiene en cuenta el ahorro de recursos o la estabilidad estática de los recursos?
  - ¿Cuáles son las dependencias de los servicios?
- Defina las métricas de los SLA basándose en la arquitectura de la carga de trabajo mientras trabaja con las partes interesadas. Considere los SLA de todas las dependencias utilizadas por la carga de trabajo.
- Una vez establecido el objetivo del SLA, optimice la arquitectura para que cumpla el SLA.
- Una vez establecido el diseño que cumplirá el SLA, implemente cambios operativos, automatización de procesos y runbooks que también se centren en reducir el MTTD y el MTTR.
- Una vez desplegado, supervise y cree informes del SLA.

## Recursos

Prácticas recomendadas relacionadas:

- REL03-BP01 Elegir cómo segmentar su carga de trabajo
- REL10-BP01 Implementar la carga de trabajo en varias ubicaciones

- REL11-BP01 Supervisar todos los componentes de la carga de trabajo para detectar errores
- REL11-BP03 Automatizar la reparación en todas las capas
- REL12-BP05 Probar la resiliencia mediante la ingeniería del caos
- REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos
- Comprender el estado de las cargas de trabajo

Documentos relacionados:

- Disponibilidad con redundancia
- Pilar de fiabilidad: disponibilidad
- Measuring availability (Medición de la disponibilidad)
- Límites de aislamiento de errores de AWS
- Shared Responsibility Model for Resiliency (Modelo de responsabilidad compartida para la resiliencia)
- Estabilidad estática con zonas de disponibilidad
- Acuerdos de nivel de servicios (SLA) de AWS
- Guidance for Cell-based Architecture on AWS (Guía para la arquitectura basada en celdas en AWS)
- AWS infrastructure (Infraestructura de AWS)
- Advanced Multi-AZ Resiliance Patterns whitepaper (Documento técnico sobre patrones avanzados de resistencia multi-AZ)

Servicios relacionados:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

## Comprobar la fiabilidad

Una vez diseñada la carga de trabajo para que sea resiliente al estrés de producción, las pruebas son la única forma de garantizar que funcionará según lo previsto y proporcionará la resiliencia esperada.

Realice pruebas para validar que su carga de trabajo se ajuste a los requisitos funcionales y no funcionales, ya que los errores o los embudos de rendimiento pueden afectar la fiabilidad de su carga de trabajo. Pruebe la resiliencia de su carga de trabajo para poder encontrar errores latentes que solo aparecen en la producción. Realice estas pruebas regularmente.

Prácticas recomendadas

- REL12-BP01 Usar guías de estrategias para investigar los errores
- REL12-BP02 Realizar un análisis después del incidente
- REL12-BP03 Comprobar los requisitos funcionales
- REL12-BP04 Requisitos de escalado y rendimiento de las pruebas
- REL12-BP05 Probar la resiliencia mediante la ingeniería del caos
- REL12-BP06 Planificación regular de días de juego

## REL12-BP01 Usar guías de estrategias para investigar los errores

Puede obtener respuestas sistemáticas e inmediatas a escenarios de error que no se entiendan bien documentando el proceso de investigación en guías de estrategias. Las guías de estrategias son pasos predefinidos realizados para identificar los factores que contribuyen a un escenario de error. Los resultados de cualquier paso del proceso se utilizan para determinar los siguientes pasos, hasta que el problema se haya identificado o deba derivarse.

Las guías de estrategias implican una planificación proactiva que debe llevar a cabo para poder emprender acciones reactivas de forma eficaz. Cuando se encuentran en producción casos de error que no están contemplados en la guía de estrategias, primero debe solucionar el problema (apagar el fuego). Luego, deberá volver y analizar los pasos que ha seguido para abordar el problema y, sobre ellos, añadir una nueva entrada en la guía.

Tenga en cuenta que las guías de estrategias se usan en respuesta a incidentes específicos y los runbooks se usan para conseguir resultados determinados. A menudo, los runbooks se usan para actividades rutinarias, mientras que las guías de estrategias se utilizan para responder a eventos no rutinarios.

Antipatrones usuales:

- Planificar la implementación de una carga de trabajo sin conocer los procesos para diagnosticar los problemas o responder a los incidentes

- Decisiones no planificadas sobre de qué sistemas se recopilan registros y métricas cuando se investiga un evento
- No conservar las métricas y los eventos el tiempo suficiente para poder recuperar los datos

Beneficios de establecer esta práctica recomendada: La captura de esta información en guías de estrategias garantiza que el proceso pueda seguirse sistemáticamente. La creación de guías de estrategias limita la introducción de errores de la actividad manual. La automatización de guías de estrategias reduce el tiempo para responder a un evento al eliminar el requisito de intervención de un miembro del equipo o al disponer de información adicional al inicio de su intervención.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

- Use guías de estrategias para identificar problemas. Las guías de estrategias son procesos documentados para investigar problemas. Permita las respuestas sistemáticas e inmediatas a escenarios de error documentando los procesos en guías de estrategias. Las guías de estrategias deben contener la información y las instrucciones necesarias para que alguien con la formación adecuada reúna la información correspondiente, identifique las posibles fuentes de error, aísle los errores y determine los factores que han contribuido al problema (realizar un análisis después del incidente).
  - Implemente en código las guías de estrategias. Realice sus operaciones como código creando scripts de sus guías de estrategias para garantizar la sistematicidad y reducir los errores causados por los procesos manuales. Las guías de estrategias pueden constar de varios scripts que representen los diferentes pasos que podrían ser necesarios para identificar los factores que contribuyen a un problema. Se pueden programar o realizar actividades de runbook como parte de las actividades de una guía de estrategias, o se puede solicitar la ejecución de una guía de estrategias en respuesta a eventos identificados.
    - Automatizar las guías de estrategias operativas con AWS Systems Manager
    - AWS Systems Manager Run Command
    - AWS Systems Manager Automation
    - ¿Qué es AWS Lambda?
    - ¿Qué es Amazon EventBridge?
    - Uso de alarmas de Amazon CloudWatch

### Recursos

Documentos relacionados:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- Automatizar las guías de estrategias operativas con AWS Systems Manager
- Uso de alarmas de Amazon CloudWatch
- Uso de valores controlados (Amazon CloudWatch Synthetics)
- ¿Qué es Amazon EventBridge?
- ¿Qué es AWS Lambda?

Ejemplos relacionados:

- Automatización de operaciones con guías de estrategias y runbooks

## REL12-BP02 Realizar un análisis después del incidente

Revise los eventos que afectan a los clientes e identifique los factores que contribuyen al evento y las medidas preventivas. Use esta información para desarrollar un plan de mitigación que limite o evite la reaparición del problema. Desarrolle procedimientos para proporcionar respuestas rápidas y eficaces. Comunique los factores que han contribuido al problema y las medidas correctivas según corresponda, adaptados al público de destino. Disponga de un método para comunicar estas causas a otros usuarios según sea necesario.

Evalúe por qué las pruebas existentes no han detectado el problema. Añada pruebas para este caso si no hay pruebas ya establecidas.

Resultado deseado: sus equipos tienen un enfoque uniforme y consensuado para gestionar el análisis posterior a los incidentes. Uno de los mecanismos es el proceso de corrección de errores (COE). El proceso COE ayuda a sus equipos a identificar, comprender y abordar las causas fundamentales de los incidentes, a la vez que crea mecanismos y barreras de protección para limitar la probabilidad de que se repita el mismo incidente.

Antipatrones usuales:

- Buscar los factores que han contribuido al problema, pero no seguir investigando si existen otros problemas potenciales o enfoques que mitigar
- Identificar solo los errores humanos y no proporcionar ninguna formación o automatización que pueda evitar estos errores
- Concentrarse en determinar la culpa en lugar de en conocer la causa raíz, lo que da lugar a una cultura de miedo y obstaculiza la comunicación abierta
- Falta de intercambio de ideas, lo que hace que los resultados del análisis de incidentes los conozca solo un grupo pequeño e impide que otros se beneficien de las lecciones aprendidas
- No tener ningún mecanismo para capturar el conocimiento institucional, por lo que se pierde información valiosa al no preservar las lecciones aprendidas en forma de actualizaciones de las prácticas recomendadas y, por lo tanto, se repiten incidentes con la misma causa raíz o una similar

Ventajas de establecer esta práctica recomendada: realizar análisis después de un incidente y compartir los resultados permite que el riesgo se mitigue en otras cargas de trabajo si estas tienen implementados los mismos factores que han contribuido al problema, y permite también implementar la mitigación o la recuperación automatizada antes de que se produzca un incidente.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

Un buen análisis posterior a un incidente ofrece oportunidades de proponer soluciones comunes para problemas con patrones arquitectónicos que se utilizan en otros lugares de los sistemas.

Una piedra angular del proceso COE es documentar y abordar los problemas. Es recomendable definir una forma estandarizada de documentar las causas raíz críticas y asegurarse de que estas se revisan y solucionan. Asigne una propiedad clara al proceso de análisis posterior al incidente. Designe a un equipo o persona responsable que supervise las investigaciones y el seguimiento de los incidentes.

Fomente una cultura que se centre en el aprendizaje y la mejora en lugar de en la culpa. Haga hincapié en que el objetivo es prevenir futuros incidentes, no penalizar a las personas.

Desarrolle procedimientos bien definidos para realizar análisis posteriores a los incidentes. En estos procedimientos, se deben describir los pasos que se deben seguir, la información que se va a recopilar y las preguntas clave que se abordarán durante el análisis. Investigue los incidentes a fondo y vaya más allá de las causas inmediatas para identificar las causas raíz y los factores

que contribuyen a ellos. Use técnicas como los cinco porqués para profundizar en los problemas subyacentes.

Mantenga un repositorio de las lecciones aprendidas de los análisis de incidentes. Este conocimiento institucional puede servir como referencia para futuros incidentes y esfuerzos de prevención. Comparta las conclusiones y los conocimientos de los análisis posteriores a los incidentes y considere la posibilidad de celebrar reuniones de revisión de invitación abierta después de los incidentes para analizar las lecciones aprendidas.

Pasos para la implementación

- Al realizar un análisis posterior al incidente, asegúrese de que en el proceso no se culpe a nadie. Esto permite que las personas involucradas en el incidente se muestren imparciales con respecto a las medidas correctivas propuestas, además de fomentar una autoevaluación honesta y la colaboración entre los equipos.
- Defina una forma estandarizada de documentar los problemas críticos. Un ejemplo de estructura para dicho documento es el siguiente:
  - ¿Qué ha ocurrido?
  - ¿Cómo ha afectado a los clientes y a la empresa?
  - ¿Cuál ha sido la causa raíz?
  - ¿Qué datos tiene para corroborarlo?
    - Por ejemplo, métricas y gráficos
  - ¿Qué pilares básicos estuvieron implicados, con especial atención a la seguridad?
    - Al diseñar cargas de trabajo, se hacen concesiones entre pilares según el contexto del negocio. Estas decisiones de negocios pueden impulsar sus prioridades de ingeniería. Podría dar preferencia a reducir el costo a expensas de la fiabilidad en el desarrollo de entornos o, para soluciones de misión crítica, podría optimizar la fiabilidad con costos aumentados. La seguridad siempre es la tarea primordial, ya que sus clientes deben estar protegidos.
  - ¿Qué lecciones aprendió?
  - ¿Qué medidas correctivas está tomando?
    - Medidas
    - Artículos relacionados
- Cree procedimientos operativos estándar bien definidos para realizar análisis posteriores a los incidentes.

- Configure un proceso estandarizado de notificación de incidentes. Documente todos los incidentes de manera exhaustiva, incluido el informe inicial del incidente, los registros, las comunicaciones y las medidas tomadas durante el incidente.
- Recuerde que un incidente no requiere una interrupción. Podría tratarse de un cuasi incidente o de un sistema que funciona de una forma inesperada, pero que cumple su función.
- Mejore continuamente su proceso de análisis posterior a un incidente en función de los comentarios y las lecciones aprendidas.
- Registre los resultados clave en un sistema de administración del conocimiento y considere cualquier patrón que deba añadirse a las guías para desarrolladores o a las listas de verificación previas al despliegue.

### Recursos

Documentos relacionados:

- «Why you should develop a correction of error (COE)»

Vídeos relacionados:

- «Amazon's approach to failing successfully»
- «AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon»

## REL12-BP03 Comprobar los requisitos funcionales

Use técnicas como pruebas unitarias y pruebas de integración que validen la funcionalidad necesaria.

Conseguirá los mejores resultados cuando estas pruebas se lleven a cabo automáticamente como parte de las acciones de compilación y despliegue. Por ejemplo, al utilizar AWS CodePipeline, los desarrolladores confirman los cambios en un repositorio de origen en el que CodePipeline detecta los cambios automáticamente. Esos cambios se incorporan y se realizan pruebas. Una vez completadas las pruebas, el código compilado se implementa en los servidores provisionales para comprobarlo. Desde el servidor provisional, CodePipeline ejecuta más pruebas, como pruebas de integración o carga. Una vez completadas correctamente esas pruebas, CodePipeline implementa el código comprobado y aprobado en instancias de producción.

Además, la experiencia demuestra que las pruebas de transacciones sintéticas (denominadas también pruebas de valores controlados, que no deben confundirse con las implementaciones de valores controlados) que puedan ejecutar y simular el comportamiento de los clientes son uno de los procesos de prueba más importantes. Ejecute estas pruebas constantemente en los puntos de conexión de las cargas de trabajo desde distintas ubicaciones remotas. Amazon CloudWatch Synthetics le permite crear valores controlados para supervisar sus puntos de conexión y API.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

- Compruebe los requisitos funcionales. Entre estas se incluyen las pruebas unitarias y las pruebas de integración que validan la funcionalidad necesaria.
  - Use CodePipeline y AWS CodeBuild para probar el código y ejecutar compilaciones
  - AWS CodePipeline añade compatibilidad a las pruebas unitarias y de integración personalizadas con AWS CodeBuild
  - Entrega continua e integración continua
  - Uso de valores controlados (Amazon CloudWatch Synthetics)
  - Automatización de pruebas de software

## Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudar con la implementación de una canalización de integración continua
- AWS CodePipeline añade compatibilidad a las pruebas unitarias y de integración personalizadas con AWS CodeBuild
- AWS Marketplace: productos que pueden usarse para la integración continua
- Entrega continua e integración continua
- Automatización de pruebas de software
- Use CodePipeline y AWS CodeBuild para probar el código y ejecutar compilaciones
- Uso de valores controlados (Amazon CloudWatch Synthetics)

# REL12-BP04 Requisitos de escalado y rendimiento de las pruebas

Use técnicas como las pruebas de carga para validar que la carga de trabajo satisface los requisitos de escalado y rendimiento.

En la nube, puede crear un entorno de pruebas a escala de producción bajo demanda para su carga de trabajo. Si ejecuta estas pruebas en una infraestructura desescalada verticalmente, debe escalar los resultados observados a lo que cree que ocurrirá en producción. Las pruebas de carga y rendimiento también pueden realizarse en producción si se tiene cuidado de no afectar a los usuarios reales y se etiquetan los datos de prueba para que no se mezclen con los datos de usuarios reales y alteren las estadísticas de uso o los informes de producción.

Con las pruebas, asegúrese de que sus recursos base, la configuración de escalado, las cuotas de servicio y el diseño de resiliencia funcionan del modo esperado bajo carga.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

- Pruebe los requisitos de escalado y de rendimiento. Realice pruebas de carga para validar que la carga de trabajo satisface los requisitos de escalado y rendimiento.
  - Pruebas de carga distribuida en AWS: simular miles de usuarios conectados
  - Apache JMeter
    - Implemente la aplicación en un entorno idéntico al de producción y ejecute una prueba de carga.
      - Utilice conceptos de infraestructura como código para crear un entorno tan similar al entorno de producción como sea posible.

## Recursos

Documentos relacionados:

- Pruebas de carga distribuida en AWS: simular miles de usuarios conectados
- Apache JMeter

# REL12-BP05 Probar la resiliencia mediante la ingeniería del caos

Realice experimentos de caos con regularidad en entornos que estén en producción o lo más cerca posible de ella para entender cómo responde su sistema a condiciones adversas.

Resultado deseado:

La resiliencia de la carga de trabajo se verifica regularmente aplicando la ingeniería del caos en forma de experimentos de inyección de errores o inyección de carga inesperada, además de las pruebas de resiliencia que validan el comportamiento esperado conocido de su carga de trabajo durante un evento. Combine la ingeniería del caos y las pruebas de resiliencia para tener la seguridad de que su carga de trabajo puede sobrevivir a los errores de los componentes y puede recuperarse de las interrupciones inesperadas con un impacto mínimo o nulo.

Patrones comunes de uso no recomendados:

- Diseñar para lograr la resiliencia, pero no verificar cómo funciona la carga de trabajo en su conjunto cuando se producen errores.
- No experimentar nunca en condiciones reales y con la carga prevista.
- No tratar los experimentos como código ni mantenerlos durante el ciclo de desarrollo.
- No ejecutar experimentos de caos tanto como parte de su canalización de CI/CD, así como fuera de los despliegues.
- No utilizar los análisis posteriores a los incidentes a la hora de determinar los errores con los que experimentar.

Beneficios de establecer esta práctica recomendada: Inyectar errores para verificar la resiliencia de la carga de trabajo permite ganar confianza sobre el hecho de que los procedimientos de recuperación de su diseño resiliente funcionarán en caso de un error real.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Medio

## Guía para la implementación

La ingeniería del caos proporciona a sus equipos capacidades para inyectar continuamente interrupciones del mundo real (simulaciones) de forma controlada a nivel de proveedor de servicios, infraestructura, carga de trabajo y componentes, con un impacto mínimo o nulo para sus clientes. Permite que sus equipos aprendan de los fallos y observen, midan y mejoren la resiliencia de

sus cargas de trabajo, además de validar que las alertas se disparen y que los equipos reciban notificaciones en caso de algún evento.

Cuando se realiza de forma continua, la ingeniería del caos puede poner de manifiesto deficiencias en sus cargas de trabajo que, si no se abordan, podrían afectar negativamente la disponibilidad y al funcionamiento.

> **Note**
>
> La ingeniería del caos es la disciplina que consiste en experimentar en un sistema para generar confianza en la capacidad del sistema de resistir condiciones adversas en producción. Principios de la ingeniería del caos

Si un sistema es capaz de soportar estas interrupciones, el experimento del caos debería mantenerse como una prueba de regresión automatizada. De este modo, los experimentos de caos deben realizarse como parte de su ciclo de vida de desarrollo de sistemas (SDLC) y como parte de su canalización de CI/CD.

Para asegurarse de que su carga de trabajo puede sobrevivir a los errores de los componentes, inyecte eventos del mundo real como parte de sus experimentos. Por ejemplo, experimente con la pérdida de instancias de Amazon EC2 o la conmutación por error de la instancia primaria de la base de datos de Amazon RDS y verifique que su carga de trabajo no se ve afectada (o solo mínimamente). Utilice una combinación de errores de componentes para simular los eventos que puede causar una interrupción en una zona de disponibilidad.

Para los errores a nivel de aplicación (como las caídas), se puede empezar con factores de estrés como el agotamiento de la memoria y la CPU.

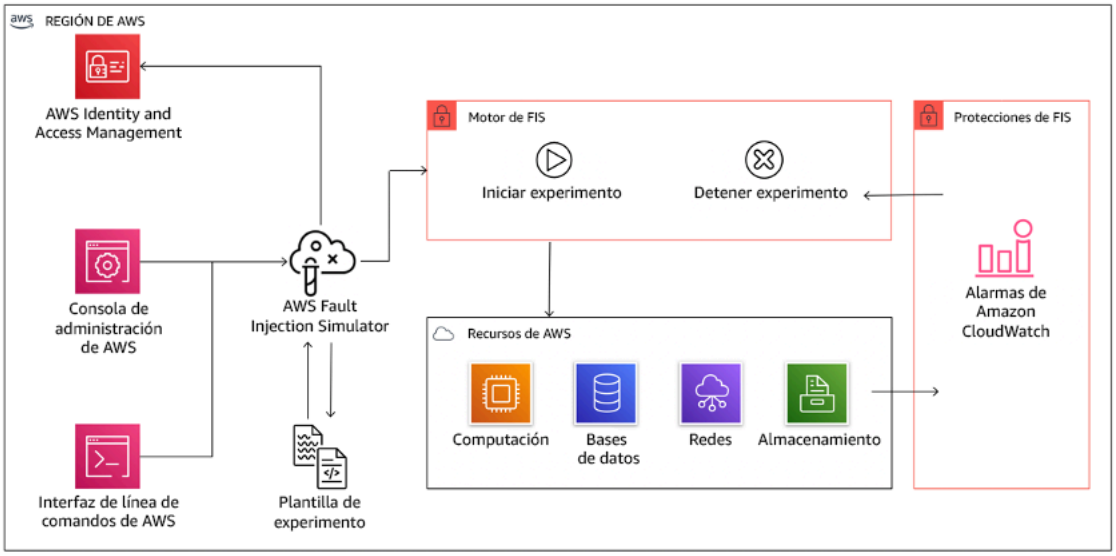
Para validar los mecanismos de recuperación o de conmutación por error para las dependencias externas debido a interrupciones intermitentes de la red, sus componentes deben simular un evento de este tipo bloqueando el acceso a los proveedores de terceros durante una duración especificada que puede durar desde segundos hasta horas.

Otros modos de degradación podrían provocar una funcionalidad reducida y respuestas lentas, lo que a menudo da como resultado una interrupción de sus servicios. Las fuentes comunes de esta degradación son una mayor latencia en los servicios críticos y una comunicación de red poco fiable (paquetes omitidos). Los experimentos con estos errores, que incluyen efectos de red como la

latencia, los mensajes perdidos y los errores de DNS, podrían incluir la incapacidad de resolver un nombre, alcanzar el servicio DNS o establecer conexiones con servicios dependientes.

Herramientas de ingeniería del caos:

AWS Fault Injection Service (AWS FIS ) es un servicio completamente administrado para realizar experimentos de inserción de errores que puede utilizarse como parte de su canalización de CD. AWS FIS es una buena opción para usar durante los días de juego de ingeniería del caos. Admite la introducción simultánea de errores en diferentes tipos de recursos, como Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS) y Amazon RDS. Estos errores incluyen la terminación de los recursos, forzado de conmutación por error, estrés de CPU o memoria, limitación, latencia y pérdida de paquetes. Al estar integrado con Amazon CloudWatch Alarms, puede configurar las condiciones de parada como barreras de protección para revertir un experimento si provoca un impacto inesperado.




AWS Fault Injection Service se integra con recursos de AWS para permitirle ejecutar experimentos de inserción de errores para sus cargas de trabajo.

También hay varias opciones de terceros para los experimentos de inserción de errores. Incluyen herramientas de código abierto como Chaos Toolkit, Chaos Mesh y Litmus Chaos, además de opciones comerciales como Gremlin. Para ampliar el alcance de los errores que se pueden inyectar en AWS, AWS FIS se integra con Chaos Mesh y Litmus Chaos, lo que le permite coordinar los flujos de trabajo de inyección de errores entre varias herramientas. Por ejemplo, puede ejecutar una prueba de estrés en la CPU de un pod utilizando errores de Chaos Mesh o Litmus mientras termina un porcentaje seleccionado al azar de nodos del clúster utilizando acciones de error de AWS FIS.

## Pasos para la aplicación

- Determine qué errores se van a utilizar en los experimentos.

  Evalúe el diseño de su carga de trabajo para la resiliencia. Estos diseños (creados utilizando las prácticas recomendadas del Marco de buena arquitectura) tienen en cuenta los riesgos basados en las dependencias críticas, los eventos pasados, los problemas conocidos y los requisitos de cumplimiento. Enumere cada elemento del diseño destinado a mantener la resiliencia y los errores que pretende mitigar. Para obtener más información sobre la creación de estas listas, consulte el documento técnico Revisión de la preparación operativa que le orienta sobre cómo crear un proceso para evitar que se repitan incidentes anteriores. El proceso de análisis de modos de error y efectos (AMFE) le proporciona un marco para realizar un análisis de los fallos a nivel de componente y cómo afectan a su carga de trabajo. Adrian Cockcroft describe con más detalle el AMFE en Failure Modes and Continuous Resilience (Modos de error y resiliencia continua).

- Asigne una prioridad a cada error.

  Comience con una categorización gruesa como alta, media o baja. Para evaluar la prioridad, hay que tener en cuenta la frecuencia del error y el impacto del mismo en la carga de trabajo global.

  Al considerar la frecuencia de un determinado error, analice los datos anteriores de esta carga de trabajo cuando estén disponibles. Si no están disponibles, utilice datos de otras cargas de trabajo que se ejecuten en un entorno similar.

  Cuando se considera el impacto de un error determinado, cuanto mayor sea el alcance del error, generalmente mayor será el impacto. También hay que tener en cuenta el diseño y la finalidad de la carga de trabajo. Por ejemplo, la capacidad de acceder a los almacenes de datos de origen es fundamental para una carga de trabajo que realice transformaciones y análisis de datos. En este caso, se daría prioridad a los experimentos de errores de acceso, así como al acceso limitado y a la inserción de latencia.

  Los análisis posteriores a los incidentes son un buena fuente de datos para comprender tanto la frecuencia como el impacto de los modos de error.

  Utilice la prioridad asignada para determinar con qué fallos experimentar primero y el orden con el que desarrollar nuevos experimentos de inyección de errores.

- En cada experimento que realice, siga la ingeniería del caos y el volante de resiliencia continua.


Ingeniería del caos y volante de resiliencia continua, utilizando el método científico de Adrian Hornsby.

- Defina el estado estable como un resultado medible de una carga de trabajo que indica un comportamiento normal.

  Su carga de trabajo exhibe un estado estable si está operando de manera fiable y como se espera. Por tanto, valide que su carga de trabajo tenga un buen estado antes de definir el estado estable. El estado estable no significa necesariamente que no haya impacto en la carga de trabajo cuando se produce un error, ya que un cierto porcentaje en los errores podría estar dentro de los límites aceptables. El estado estable es la línea de base que observará durante el experimento, que pondrá de manifiesto las anomalías si su hipótesis definida en el siguiente paso no resulta de la forma esperada.

  Por ejemplo, un estado estable de un sistema de pagos puede definirse como el procesamiento de 300 TPS con una tasa de éxito del 99 % y un tiempo de ida y vuelta de 500 ms.

- Formule una hipótesis sobre cómo reaccionará la carga de trabajo ante el error.

  Una buena hipótesis se basa en cómo se espera que la carga de trabajo mitigue el error para mantener el estado estable. La hipótesis establece que dado el error de un tipo específico, el sistema o la carga de trabajo continuará en estado estable, porque la carga de trabajo fue diseñada con mitigaciones específicas. En la hipótesis deben especificarse el tipo específico de error y las mitigaciones.

  Se puede utilizar la siguiente plantilla para la hipótesis (pero también se acepta otra redacción):

  > **Note**
  >
  > Si se produce el *error específico* , la carga de trabajo *nombre de carga de trabajo* , *describirá los controles mitigantes* para mantener el *impacto de las métricas empresariales o técnicas*.

  Por ejemplo:

  - Si el 20 % de los nodos del grupo de nodos de Amazon EKS se caen, la API de creación de transacciones sigue sirviendo el percentil 99 de peticiones en menos de 100 ms (estado estable). Los nodos de Amazon EKS se recuperarán en cinco minutos, y los pods se programarán y procesarán el tráfico en ocho minutos tras el inicio del experimento. Las alertas se disparan en tres minutos.
  - Si se produce un error de instancia de Amazon EC2, la comprobación de estado de Elastic Load Balancing del sistema de pedidos hará que Elastic Load Balancing solo envíe solicitudes a las instancias en buen estado restantes mientras Amazon EC2 Auto Scaling sustituye la instancia con error, manteniendo un aumento inferior al 0,01 % en los errores del lado del servidor (5xx) (estado estable).
  - Si la instancia de la base de datos primaria de Amazon RDS falla, la carga de trabajo de recopilación de datos de la cadena de suministro se conmutará por error y se conectará a la instancia de la base de datos de Amazon RDS en espera para mantener menos de 1 minuto de errores de lectura o escritura en la base de datos (estado estable).
- Realiza el experimento inyectando el error.

  Un experimento debería ser por defecto a prueba de errores y tolerado por la carga de trabajo. Si sabe que la carga de trabajo va a fallar, no realice el experimento. Debe utilizarse la ingeniería del caos para encontrar conocidos-desconocidos o desconocidos-desconocidos.

Conocidos-desconocidos son cosas de las que es consciente pero no comprende del todo, y desconocidos-desconocidos son cosas de las que no es consciente ni comprende del todo. Experimentar con una carga de trabajo que sabe que está rota no proporcionará nuevas ideas. Su experimento debe estar cuidadosamente planificado, tener un alcance claro de impacto y proporcionar un mecanismo de retroceso que pueda aplicarse en caso de turbulencias inesperadas. Si su diligencia demuestra que su carga de trabajo debería sobrevivir al experimento, siga adelante con el mismo. Hay varias opciones para inyectar los errores. Para cargas de trabajo en AWS, AWS FIS proporciona muchas simulaciones de errores predefinidas llamadas acciones. También puede definir acciones personalizadas que se ejecuten en AWS FIS utilizando documentos de AWS Systems Manager.

Desaconsejamos el uso de scripts personalizados para los experimentos de caos, a menos que los scripts tengan la capacidad de entender el estado actual de la carga de trabajo, sean capaces de emitir registros y proporcionen mecanismos para retrocesos y condiciones de parada cuando sea posible.

Un marco o conjunto de herramientas eficaz que apoye la ingeniería del caos debe hacer el seguimiento del estado actual de un experimento, emitir registros y proporcionar mecanismos de reversión para apoyar la ejecución controlada de un experimento. Comience con un servicio establecido como AWS FIS que permite realizar experimentos con un alcance claramente definido y mecanismos de seguridad que reviertan el experimento en el caso de que introduzca turbulencias inesperadas. Para conocer una mayor variedad de experimentos con AWS FIS, consulte también el Laboratorio de Aplicaciones resilientes y bien diseñadas con ingeniería del caos. Además, AWS Resilience Hub analizará su carga de trabajo y creará experimentos que puede elegir para implementar y ejecutar en AWS FIS.

> **Note**
>
> Para cada experimento, comprenda claramente el alcance y su impacto. Recomendamos que los fallos se simulen primero en un entorno no productivo antes de ejecutarlos en producción.

Los experimentos deben realizarse en producción bajo carga real utilizando despliegue de valores controlados que acelera tanto el despliegue de un sistema de control como el experimental, cuando es factible. Ejecutar los experimentos durante las horas de menor actividad es una buena práctica para mitigar el impacto potencial cuando se experimenta por

primera vez en producción. Además, si utilizar el tráfico real del cliente supone demasiado riesgo, puede realizar experimentos utilizando tráfico sintético en la infraestructura de producción contra los despliegues de control y experimentales. Cuando no sea posible utilizar la producción, ejecute los experimentos en entornos de preproducción que sean lo más parecidos posible a la producción.

Debe establecer y supervisar las barreras de seguridad para garantizar que el experimento no afecte al tráfico de producción o a otros sistemas más allá de los límites aceptables. Establezca condiciones de parada para detener un experimento si alcanza un umbral en una métrica de barrera que defina. Esto debería incluir las métricas para el estado estable de la carga de trabajo, así como la métrica contra los componentes en los que está inyectando el error. Una monitorización sintética (también conocida como valor controlado) es una métrica que normalmente debería incluir como proxy de usuario. Las condiciones de parada para AWS FIS se admiten como parte de la plantilla del experimento, permitiendo hasta cinco condiciones de parada por plantilla.

Uno de los principios del caos es minimizar el alcance del experimento y su impacto:

Aunque hay que tener en cuenta algún impacto negativo a corto plazo, es responsabilidad y obligación del ingeniero del caos garantizar que las consecuencias de los experimentos se minimicen y contengan.

Un método para verificar el alcance y el impacto potencial es realizar el experimento primero en un entorno de no producción, verificando que los umbrales para las condiciones de parada se activan como se espera durante un experimento y la observabilidad está implantada para detectar una excepción, en lugar de experimentar directamente en la producción.

Cuando se realicen experimentos de inyección de errores, verifique que todas las partes responsables estén bien informadas. Comuníquese con los equipos adecuados, como los equipos de operaciones, los equipos de fiabilidad del servicio y el servicio de atención al cliente, para informarles de cuándo se llevarán a cabo los experimentos y qué pueden esperar. Proporcione a estos equipos herramientas de comunicación para que informen a los que dirigen el experimento si observan algún efecto adverso.

Debe restablecer la carga de trabajo y sus sistemas subyacentes al estado bueno conocido original. A menudo, el diseño resistente de la carga de trabajo se autorrepara. Pero algunos diseños de errores o experimentos fallidos pueden dejar su carga de trabajo en un estado de error inesperado. Al final del experimento, debe ser consciente de ello y restablecer la carga de

trabajo y los sistemas. Con AWS FIS puede establecer una configuración de reversión (también llamada acción posterior) dentro de los parámetros de la acción. Una acción posterior devuelve el objetivo al estado en el que se encontraba antes de ejecutar la acción. Ya sean automatizadas (como el uso de AWS FIS) o manuales, estas acciones posteriores deben formar parte de una guía de estrategias que describa cómo detectar y gestionar los errores.

- Verifique la hipótesis.

  Principios de la ingeniería del caos ofrece estas directrices sobre cómo verificar el estado estable de su carga de trabajo:

  Céntrese en los resultados medibles de un sistema, más que en los atributos internos del mismo. Las mediciones de esa producción durante un corto periodo de tiempo constituyen una aproximación al estado estable del sistema. El rendimiento global del sistema, las tasas de error y los percentiles de latencia podrían ser métricas de interés que representen el comportamiento en estado estable. Al centrarse en los patrones de comportamiento sistémico durante los experimentos, la ingeniería del caos verifica que el sistema funcione, en lugar de intentar validar cómo funciona.

  En nuestros dos ejemplos anteriores, incluimos las métricas de estado estable de menos del 0,01 % de aumento de errores del lado del servidor (5xx) y menos de un minuto de errores de lectura y escritura en la base de datos.

  Los errores 5xx son una buena métrica porque son una consecuencia del modo de error que un cliente de la carga de trabajo experimentará directamente. La medición de los errores de la base de datos es buena como consecuencia directa del error, pero también debe complementarse con una medición del impacto en el cliente, como las solicitudes fallidas de los clientes o los errores que aparecen en el cliente. Además, incluya una monitorización sintética (también conocida como valor controlado) en cualquier API o URI al que acceda directamente el cliente de su carga de trabajo.

- Mejore el diseño de la carga de trabajo para la resiliencia.

  Si el estado estable no se mantuvo, entonces investigue cómo se puede mejorar el diseño de la carga de trabajo para mitigar el error, aplicando las mejores prácticas del Pilar de fiabilidad de AWS Well-Architected. Se pueden encontrar orientaciones y recursos adicionales en la AWS Builder's Libraryque aloja artículos sobre cómo mejorar las comprobaciones de estado o bien emplear reintentos con retroceso en el código de su aplicación, entre otros.

Una vez aplicados estos cambios, vuelva a realizar el experimento (mostrado por la línea de puntos en el volante de ingeniería del caos) para determinar su eficacia. Si el paso de verificación indica que la hipótesis es cierta, entonces la carga de trabajo estará en estado estable, y el ciclo continúa.

- Realice experimentos con regularidad.

  Un experimento de caos es un ciclo, y los experimentos deben realizarse regularmente como parte de la ingeniería del caos. Después de que una carga de trabajo cumpla con la hipótesis del experimento, este debe automatizarse para ejecutarse continuamente como parte de la regresión de su canalización de CI/CD. Para saber cómo hacerlo, consulte este blog sobre cómo ejecutar experimentos de AWS FIS con AWS CodePipeline. Este laboratorio sobre experimentos de AWS FIS periódicos en una canalización de CI/CD le permite trabajar de forma práctica.

  Los experimentos de inyección de errores también forman parte de los días de juego (consulte REL12-BP06 Planificación regular de días de juego). En los días de juego se simula un error o un evento para verificar los sistemas, los procesos y las respuestas de los equipos. El objetivo es, de hecho, realizar las acciones que llevaría a cabo el equipo si se produjera un evento excepcional.

- Capture y almacene los resultados de los experimentos.

  Los resultados de los experimentos de inyección de errores deben capturarse y persistir. Incluya todos los datos necesarios (como el tiempo, la carga de trabajo y las condiciones) para poder analizar posteriormente los resultados y las tendencias del experimento. Algunos ejemplos de resultados pueden ser capturas de pantalla de paneles de control, volcados CSV de la base de datos de su métrica o un registro escrito a mano de los eventos y observaciones del experimento. Experimentar el registro con AWS FIS puede formar parte de esta captura de datos.

## Recursos

Prácticas recomendadas relacionadas:

- REL08-BP03 Integrar las pruebas de resiliencia como parte de su despliegue
- REL13-BP03 Probar la implementación de recuperación de desastres para validarla

Documentos relacionados:

- ¿Qué es AWS Fault Injection Service?

- ¿Qué es AWS Resilience Hub?
- Principios de la ingeniería del caos
- Ingeniería del caos: Planificar su primer experimento
- Ingeniería de resiliencia: aprender a asumir los errores
- Historias de ingeniería del caos
- Evitar los planes alternativos en los sistemas distribuidos
- Despliegue de valores controlados para experimentos de caos

Vídeos relacionados:

- AWS re:Invent 2020: Pruebas de resistencia mediante la ingeniería del caos (ARC316)
- AWS re:Invent 2019: Mejorar la resiliencia con la ingeniería del caos (DOP309-R1)
- AWS re:Invent 2019: Realizar ingeniería del caos en un mundo sin servidores (CMY301)

Ejemplos relacionados:

- Laboratorio de Well-Architected: Nivel 300: pruebas de resiliencia de Amazon EC2, Amazon RDS y Amazon S3
- Laboratorio de ingeniería del caos en AWS
- Laboratorio de aplicaciones resilientes y bien diseñadas con ingeniería del caos
- Laboratorio de caos sin servidor
- Medir y mejorar la resiliencia de sus aplicaciones con laboratorio de AWS Resilience Hub

Herramientas relacionadas:

- AWS Fault Injection Service
- AWS Marketplace: Gremlin Chaos Engineering Platform (Plataforma de ingeniería del caos de Gremlin)
- Chaos Toolkit
- Chaos Mesh
- Litmus

# REL12-BP06 Planificación regular de días de juego

Utilice días de juego para poner en práctica frecuentemente sus procedimientos para responder a los eventos y los errores lo más cerca de la fecha de lanzamiento a producción posible (incluidos los entornos de producción) con las personas que trabajarán en los escenarios de error reales. Los días de juego sirven para imponer medidas que garanticen que los eventos de producción no afecten a los usuarios.

En los días de juego se simula un error o un evento para probar los sistemas, los procesos y las respuestas de los equipos. El objetivo es, de hecho, realizar las acciones que llevaría a cabo el equipo si se produjera un evento excepcional. Esto ayudará a comprender dónde se pueden realizar mejoras y a desarrollar la experiencia organizacional en la gestión de eventos. Deberían hacerse habitualmente para que el equipo desarrolle “memoria muscular” sobre cómo responder.

Una vez que cuente con un diseño de resiliencia y que lo haya probado en entornos que no sean de producción, los días de juego son la fórmula ideal para garantizar que todo funcione según lo previsto en el entorno de producción. Un día de juego, especialmente el primero, es una actividad para todo el equipo en la que los ingenieros y el personal de operaciones están informados de cuándo ocurrirá y de qué pasará. Hay runbooks preparados. Se ejecutan eventos simulados, incluidos los posibles eventos de error, en los sistemas de producción y de la forma prescrita y, entonces, se evalúa el impacto. Si todos los sistemas funcionan según lo diseñado, la detección y la autocorrección se producirán con un impacto mínimo o inexistente. Sin embargo, si se observa algún impacto negativo, se da marcha atrás a la prueba y los problemas con la carga de trabajo se remedian, de forma manual si fuera necesario (utilizando el runbook). Dado que los días de juego suelen desarrollarse en el entorno de producción, deben tomarse todas las precauciones necesarias para garantizar que no haya ningún impacto sobre la disponibilidad para los clientes.

Antipatrones usuales:

- Documentar los procedimientos, pero no ponerlos nunca en práctica
- No incluir a los responsables de la toma de decisiones del negocio en los ejercicios de prueba

Beneficios de establecer esta práctica recomendada: Realizar días de juego periódicamente garantiza que todos los empleados sigan las políticas y los procedimientos cuando se produzca un incidente real y valida que esas políticas y procedimientos son apropiados.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

### Guía para la implementación

- Programe días de juego para practicar con las guías de estrategias y runbooks. Todo el mundo que pueda verse involucrado en un evento de producción debe participar en los días de juego: el propietario de la empresa, el personal de desarrollo, el personal de operaciones y los equipos de respuesta a incidentes.
  - Ejecute sus pruebas de carga o rendimiento y después su inyección de errores.
  - Busque anomalías en los runbooks y oportunidades para poner en práctica las guías de estrategias.
    - Si se desvía de los runbooks, mejórelos o corrija este comportamiento. Si utiliza la guía de estrategias, identifique el runbook que debería haberse usado o cree uno nuevo.

### Recursos

Documentos relacionados:

- ¿Qué es AWS GameDay?

Videos relacionados:

- AWS re:Invent 2019: Mejorar la resiliencia con la ingeniería del caos (DOP309-R1)

Ejemplos relacionados:

- Laboratorios de AWS Well-Architected: comprobación de resiliencia

## Planificar para la recuperación de desastres (DR)

Disponer de copias de seguridad y de componentes de cargas de trabajo redundantes es el principio de su estrategia de DR. El RTO y el RPO son sus objetivos para la restauración de su carga de trabajo. Estos se definen en función de las necesidades del negocio. Implemente una estrategia para satisfacer estos objetivos teniendo en cuenta las ubicaciones y la función de los recursos de las cargas de trabajo y los datos. La probabilidad de una interrupción y el coste de recuperación son también factores clave que ayudan a conocer el valor empresarial de proporcionar recuperación de desastres para una carga de trabajo.

Tanto la disponibilidad como la recuperación de desastres se basan sobre las mismas prácticas que la supervisión de errores, la implementación en varias ubicaciones y la conmutación por error automática. Sin embargo, la disponibilidad se centra en componentes de la carga de trabajo, mientras que la recuperación de desastres se centra en copias discretas de la totalidad de la carga de trabajo. La recuperación de desastres tiene objetivos diferentes a los de la disponibilidad, y se centran en el tiempo de recuperación tras un desastre.

Prácticas recomendadas

- REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos
- REL13-BP02 Usar estrategias de recuperación definidas para cumplir los objetivos de recuperación
- REL13-BP03 Probar la implementación de recuperación de desastres para validarla
- REL13-BP04 Administrar la desviación de la configuración en el sitio de o en la región de recuperación de desastres
- REL13-BP05 Automatizar la recuperación

## REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos

La carga de trabajo tiene un objetivo de tiempo de recuperación (RTO) y un objetivo de punto de recuperación (RPO).

Objetivo de tiempo de recuperación (RTO) es el retraso máximo aceptable entre la interrupción del servicio y su restablecimiento. Esto determina lo que se considera un intervalo de tiempo aceptable cuando el servicio no está disponible.

Objetivo de punto de recuperación (RPO)  es el periodo de tiempo máximo aceptable desde el último punto de recuperación de datos. Determina lo que se considera una pérdida aceptable de datos entre el último punto de recuperación y la interrupción del servicio.

Los valores de RTO y RPO son consideraciones importantes a la hora de seleccionar una estrategia de recuperación de desastres (DR) adecuada para su carga de trabajo. Estos objetivos los determina la empresa y los utilizan los equipos técnicos para seleccionar e implementar una estrategia de recuperación de desastres.

Resultado deseado:

Cada carga de trabajo tiene un RTO y un RPO asignados, definidos en función del impacto empresarial. La carga de trabajo se asigna en un nivel predefinido, lo que define la disponibilidad del servicio y la pérdida de datos aceptable, con un RTO y un RPO asociados. Si no es posible esta jerarquización, se puede asignar de forma personalizada por carga de trabajo, con la intención de crear niveles más adelante. RTO y RPO se utilizan como una de las principales consideraciones para la selección de la implementación de una estrategia de recuperación de desastres para la carga de trabajo. Otras consideraciones a la hora de elegir una estrategia de recuperación de desastres son las restricciones de costes, las dependencias de la carga de trabajo y los requisitos operativos.

Para RTO, entienda el impacto basado en la duración de una interrupción. ¿Es lineal o hay implicaciones no lineales? (por ejemplo, después de cuatro horas, se cierra una línea de fabricación hasta el comienzo del siguiente turno).

Una matriz de recuperación de desastres, como la siguiente, puede ayudarle a entender cómo se relaciona la criticidad de la carga de trabajo con los objetivos de recuperación. (Tenga en cuenta que los valores reales de los ejes X e Y deben adaptarse a las necesidades de su organización).

| Matriz de recuperación de desastres | | | | | | |
|---|---|---|---|---|---|---|
| | | Objetivo de punto de recuperación | | | | |
| | | < 1 minuto | < 1 hora | < 6 horas | < 1 día | Más de 1 día |
| Objetivo de tiempo de recuperación | < 10 minutos | Crítico | Crítico | Alto | Medio | Medio |
| | < 2 horas | Crítico | Alto | Medio | Medio | Bajo |
| | < 8 horas | Alto | Medio | Medio | Bajo | Bajo |
| | < 24 horas | Medio | Medio | Bajo | Bajo | Bajo |
| | Más de 24 horas | Medio | Bajo | Bajo | Bajo | Bajo |

Figura 16: Matriz de recuperación de desastres

Patrones de uso no recomendados comunes:

- No hay objetivos de recuperación definidos.
- Seleccionar objetivos de recuperación arbitrarios.
- Seleccionar objetivos de recuperación demasiado permisivos y no satisfacer los objetivos empresariales
- No entender el impacto del tiempo de inactividad y la pérdida de datos.

- Seleccionar objetivos de recuperación poco realistas, como el tiempo de recuperación cero y la pérdida de datos cero, que pueden no ser alcanzables para la configuración de su carga de trabajo.
- Seleccionar objetivos de recuperación más estrictos que los objetivos empresariales reales Esto obliga a realizar implementaciones de recuperación de desastres más costosas y complejas de lo que necesita la carga de trabajo.
- Seleccionar objetivos de recuperación incompatibles con los de una carga de trabajo dependiente.
- Sus objetivos de recuperación no tienen en cuenta los requisitos de cumplimiento normativo.
- RTO y RPO definidos para una carga de trabajo, pero nunca se han probado.

Beneficios de establecer esta práctica recomendada: Los objetivos de recuperación de tiempo y pérdida de datos son necesarios para guiar su implementación de DR.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Alto

## Guía para la implementación

Para la carga de trabajo dada, debe entender el impacto del tiempo de inactividad y la pérdida de datos en su empresa. Por lo general, el impacto aumenta con un mayor tiempo de inactividad o pérdida de datos, pero la forma de este crecimiento puede variar en función del tipo de carga de trabajo. Por ejemplo, puede ser capaz de tolerar un tiempo de inactividad de hasta una hora con poco impacto, pero después el impacto aumenta rápidamente. El impacto en la empresa se manifiesta de muchas formas, como el coste económico (por ejemplo, la pérdida de ingresos), la confianza de los clientes (y el impacto en la reputación), los problemas operativos (por ejemplo, la pérdida de nóminas o la disminución de la productividad) y el riesgo normativo. Siga estos pasos para entender estos impactos y establecer RTO y RPO para su carga de trabajo.

Pasos de implementación

1. Determine las partes interesadas de su empresa para esta carga de trabajo y colabore con ellas para implementar estos pasos. Los objetivos de recuperación de una carga de trabajo son una decisión empresarial. Después, los equipos técnicos trabajan con las partes interesadas de la empresa para utilizar estos objetivos para seleccionar una estrategia de recuperación de desastres.

> **Note**
>
> Para los pasos 2 y 3, puede usar la the section called “Hoja de trabajo de implementación”.

2. Responda a las preguntas siguientes para reunir la información necesaria a fin de tomar una decisión.
3. ¿Tiene categorías o niveles de criticidad para el impacto de la carga de trabajo en su organización?
   a. En caso afirmativo, asigne esta carga de trabajo a una categoría.
   b. En caso contrario, establezca estas categorías. Cree un máximo de cinco categorías y ajuste el intervalo de su objetivo de tiempo de recuperación para cada una. Algunos ejemplos de categorías son: crítica, alta, media, baja. Para entender cómo se asignan las cargas de trabajo a las categorías, considere si la carga de trabajo es de misión crítica, importante para la empresa o no lo es.
   c. Establezca el RTO y el RPO de la carga de trabajo en función de la categoría. Acceda a este paso para elegir siempre una categoría más estricta (RTO y RPO más bajos) que los valores sin procesar calculados. Si esto da lugar a un cambio de valor inadecuado, considere la posibilidad de crear una nueva categoría.
4. Según estas respuestas, asigne los valores de RTO y RPO a la carga de trabajo. Se puede hacer directamente o mediante la asignación de la carga de trabajo a un nivel de servicio predefinido.
5. Documente el plan de recuperación de desastres (DRP) de esta carga de trabajo, que forma parte del plan de continuidad del negocio (BCP)de su organización, en una ubicación accesible al equipo de la carga de trabajo y las partes interesadas.
   a. Registre el RTO y el RPO, así como la información utilizada para determinar estos valores. Incluya la estrategia que se utiliza para evaluar el impacto de la carga de trabajo en la empresa.
   b. Registre otras métricas, además de RTO y RPO, de las que hace un seguimiento o planifica hacerlo para los objetivos de recuperación de desastres.
   c. Agregará los detalles de su estrategia de recuperación de desastres y su runbook a este plan cuando los cree.
6. Si busca la criticidad de la carga de trabajo en una matriz como la de la figura 15, puede empezar a establecer los niveles de servicio predefinidos para su organización.
7. Después de haber implementado una estrategia de recuperación de desastres (o una prueba de concepto para una estrategia de este tipo) según the section called “REL13-BP02 Usar estrategias

de recuperación definidas para cumplir los objetivos de recuperación", pruebe esta estrategia para determinar la capacidad de tiempo de recuperación (RTC) y de punto de recuperación (RPC) reales de la carga de trabajo. Si no cumplen los objetivos de recuperación previstos, es posible colaborar con las partes interesadas de su empresa para ajustar dichos objetivos o realizar cambios en la estrategia de RD para cumplir los objetivos previstos.

Preguntas principales

1. ¿Cuál es el tiempo máximo que la carga de trabajo puede estar inactiva antes de que se produzca un impacto grave en la empresa?
   a. Determine el coste económico (impacto financiero directo) para la empresa por minuto si se interrumpe la carga de trabajo.
   b. Considere que el impacto no siempre es lineal. El impacto puede ser limitado al principio e ir aumentando rápidamente a partir de un punto crítico.
2. ¿Cuál es la cantidad máxima de datos que puede perderse antes de que se produzca un impacto grave en la empresa?
   a. Considere este valor para su almacén de datos más crítico. Identifique la criticidad correspondiente de otros almacenes de datos.
   b. ¿Se pueden recrear los datos de la carga de trabajo si se pierden? Si esto es operativamente más fácil que la copia de seguridad y la restauración, elija el RPO en función de la criticidad de los orígenes de los datos que se utilizan para recrear los datos de la carga de trabajo.
3. ¿Cuáles son los objetivos de recuperación y las expectativas de disponibilidad de las cargas de trabajo de las que depende esta (descendente), o de las cargas de trabajo que dependen de esta (ascendente)?
   a. Elija objetivos de recuperación que permitan a esta carga de trabajo cumplir los requisitos de las dependencias ascendentes.
   b. Elija objetivos de recuperación que sean alcanzables teniendo en cuenta las capacidades de recuperación de las dependencias descendentes. Se pueden excluir las dependencias descendentes no críticas (aquellas que puede «resolver»). O bien, trabaje con las dependencias críticas posteriores para mejorar sus capacidades de recuperación cuando sea necesario.

Preguntas adicionales

Considere estas preguntas y cómo pueden aplicarse a esta carga de trabajo:

4. ¿Tiene diferentes RTO y RPO en función del tipo de interrupción región con respecto a AZ, etc.)?
5. ¿Hay algún momento específico (estacionalidad, eventos de ventas, lanzamientos de productos) en el que pueda cambiar su RTO/RPO? Si es así, ¿cuál es el límite de medida y tiempo diferente?
6. ¿Cuántos clientes se verán afectados si se interrumpe la carga de trabajo?
7. ¿Cuál es el impacto en la reputación si se interrumpe la carga de trabajo?
8. ¿Qué otros impactos operativos pueden producirse si se interrumpe la carga de trabajo? Por ejemplo, el impacto en la productividad de los empleados si los sistemas de correo electrónico no están disponibles o si los sistemas de nómina no pueden enviar las transacciones.
9. ¿Cómo se alinean el RTO y el RPO de la carga de trabajo con la línea de negocio y la estrategia organizativa de recuperación de desastres?
10. ¿Existen obligaciones contractuales internas para la prestación de un servicio? ¿Existen sanciones por incumplirlas?
11. ¿Cuáles son las restricciones normativas o de cumplimiento con los datos?

## Hoja de trabajo de implementación

Puede utilizar esta hoja de trabajo para implementar los pasos 2 y 3. Puede ajustar esta hoja de trabajo para adaptarla a sus necesidades específicas, por ejemplo, puede agregar preguntas adicionales.

| Paso 2: Preguntas principales | ¿Se aplica a la carga de trabajo? | RTO de carga de trabajo | RPO de carga de trabajo | Ajuste de RTO | Ajuste de RPO | Instrucciones |
|---|---|---|---|---|---|---|
| [1] tiempo máximo que puede estar inoperativa la carga de trabajo | | | | | | medido en tiempo desde el inicio de la interrupción hasta la recuperación |
| [2] cantidad máxima de datos que se pueden perder | | | | | | medido en tiempo desde el último conjunto de datos restaurable correcto conocido |
| [3a] dependencias upstream | | | | | | introduzca los objetivos de recuperación upstream más estrictos |
| [3b] dependencias downstream | | | | | | introduzca los objetivos de recuperación downstream menos estrictos |
| [3a] dependencias upstream reconciliadas | | | | | | Si el valor upstream es menor que los valores actuales y el valor downstream es mayor, |
| [3b] dependencias downstream reconciliadas | | | | | | trabaje con las dependencias para reconciliarlas e introducir aquí los valores reconciliados |
| [3] dependencias | | | | | | reduzca los valores para ajustarse a las dependencias upstream o auméntelos en función de las capacidades de dependencias downstream |
| | | | | | | |
| **Paso 2: Preguntas adicionales** | | | | | | Indique si se aplica la pregunta. Si no se aplica, omítala |
| RTO/RPO base | | | | | | Traslade los valores de RTO y RPO de arriba aquí |
| [4] tipo de interrupción | [ ]S / [ ]N | | | | | Introduzca los objetivos de recuperación del tipo de evento con los requisitos más estrictos |
| [5] objetivos específicos basados en el tiempo | [ ]S / [ ]N | | | | | Introduzca los objetivos de recuperación de tiempo con los requisitos más estrictos |
| [6] clientes afectados | [ ]S / [ ]N | | | | | Realice un gráfico de los clientes afectados en función del tiempo de inactividad o de los datos perdidos. Utilícelo para introducir los RTO y RPO máximos permisibles en función del impacto sobre los clientes |
| [7] impacto reputacional | [ ]S / [ ]N | | | | | Trabaje con la empresa para determinar los RTO y RPO máximos en función del impacto sobre la reputación |
| [8] impacto operativo | [ ]S / [ ]N | | | | | Introduzca los RTO y RPO máximos en función del impacto operativo |
| [9] alineación organizativa | [ ]S / [ ]N | | | | | Introduzca los RTO y RPO máximos para cargas de trabajo de este tipo según los requisitos organizativos y de LOB |
| [10] obligaciones contractuales | [ ]S / [ ]N | | | | | Introduzca los RTO y RPO máximos en función de las obligaciones contractuales |
| [11] conformidad normativa | [ ]S / [ ]N | | | | | Introduzca los RTO y RPO máximos en función de la conformidad normativa pertinente |
| objetivo basado en preguntas adicionales | | | | | | Tome el valor mínimo (valor más estricto) de entre Q 4-11 e introdúzcalo aquí |
| objetivo ajustado | | | | | | Si no se puede dar cabida a los objetivos de la línea anterior, colabore con los interesados para transigir en los límites e introduzca aquí un nuevo valor mínimo |
| | | | | | | |
| RTO/RPO ajustados | | | | | | Introduzca los valores base de RPO/RTO o el objetivo ajustado, el menor de los valores |
| | | | | | | |
| **Paso 3** | | | | | | |
| Mapa a categoría o nivel predefinidos | | | | | | Ajuste ambos valores a la baja (lo más estricto) para alinearlos con el nivel definido más cercano |

Hoja de trabajo

Nivel de esfuerzo para el plan de implementación: Bajo

## Recursos

Prácticas recomendadas relacionadas:

- the section called “REL09-BP04 Realizar una recuperación periódica de los datos para verificar la integridad de la copia de seguridad y los procesos”
- the section called “REL13-BP02 Usar estrategias de recuperación definidas para cumplir los objetivos de recuperación”
- the section called “REL13-BP03 Probar la implementación de recuperación de desastres para validarla”

Documentos relacionados:

- Blog de arquitectura de AWS: serie de recuperación de desastres

- Recuperación de desastres de las cargas de trabajo en AWS: recuperación en la nube (documento técnico de AWS)
- Managing resiliency policies with AWS Resilience Hub (Administración de las políticas de resiliencia con AWS Resilience Hub)
- Socio de APN: socios que pueden ayudar con la recuperación de desastres
- AWS Marketplace: productos que pueden usarse para la recuperación de desastres

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (Patrones de arquitectura para aplicaciones activas-activas en varias regiones) (ARC209-R2)
- Disaster Recovery of Workloads on AWS (Recuperación de desastres de cargas de trabajo en AWS)

## REL13-BP02 Usar estrategias de recuperación definidas para cumplir los objetivos de recuperación

Defina una estrategia de recuperación de desastres (DR) que se ajuste a los objetivos de recuperación de su carga de trabajo. Elija una estrategia como copia de seguridad y restauración, estado de espera (activa/pasiva) o activa/activa.

Resultado deseado: para cada carga de trabajo, existe una estrategia de DR definida e implementada que permite que esa carga de trabajo alcance los objetivos de DR. Las estrategias de DR entre cargas de trabajo emplean patrones reutilizables (como las estrategias descritas anteriormente).

Antipatrones usuales:

- Implementar procedimientos de recuperación incoherentes para cargas de trabajo con objetivos de DR similares.
- Dejar la estrategia de DR para implementarla ad hoc cuando se produzca un desastre.
- No tener un plan de recuperación de desastres.
- Depender de las operaciones del plano de control durante la recuperación.

Beneficios de establecer esta práctica recomendada:

- El uso de estrategias de recuperación definidas le permite emplear herramientas y procedimientos de prueba comunes.
- El uso de estrategias de recuperación definidas mejora el intercambio de conocimiento entre equipos y la implementación de la DR en las cargas de trabajo que se encuentran bajo su responsabilidad.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto Sin una estrategia de DR planificada, implementada y probada, es poco probable que consiga sus objetivos de recuperación en caso de desastre.

## Guía para la implementación

Una estrategia de DR depende de la capacidad de poner en marcha su carga de trabajo en un sitio de recuperación si su ubicación principal deja de estar disponible para la ejecución de dicha carga de trabajo. Los objetivos de recuperación más comunes son el RTO y el RPO, como explicamos en REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos.

Una estrategia de DR en varias zonas de disponibilidad (AZ) dentro de una única Región de AWS puede ofrecer mitigación contra eventos de desastres como incendios, inundaciones y cortes de suministro eléctrico considerables. Si es necesario implementar medidas de protección contra un evento poco probable que evite que su carga de trabajo pueda ejecutarse en una Región de AWS determinada, puede seguir una estrategia de DR que abarque múltiples regiones.

A la hora de diseñar una estrategia de DR en varias regiones, debería elegir una de las siguientes estrategias. Se indican en orden ascendente de coste y complejidad y en orden descendente en cuanto al RTO y RPO. Región de recuperación se refiere a una Región de AWS diferente de la principal que se usa para su carga de trabajo.


Figura 17: Estrategias de recuperación de desastres (DR)

- Generación de copias de seguridad y restauración (RPO en horas, RTO en 24 horas o menos): cree una copia de seguridad de sus datos y aplicaciones en la región de recuperación. El uso de copias de seguridad automatizadas o continuas permitirá la recuperación a un momento dado, lo que puede reducir el RPO a hasta 5 minutos en algunos casos. En caso de desastre, desplegará su infraestructura (utilizando la infraestructura como código para reducir el RTO), desplegará su código y restaurará los datos desde la copia de seguridad para recuperarse del desastre en la región de recuperación.
- Luz piloto (RPO en minutos, RTO en decenas de minutos): aprovisione una copia de la infraestructura principal de su carga de trabajo en la región de recuperación. Replique sus datos en la región de recuperación y cree allí copias de seguridad de estos. Los recursos necesarios para permitir la replicación y copia de seguridad de los datos, como el almacenamiento de bases de datos y objetos, están siempre disponibles. Otros elementos, como los servidores de aplicaciones o la computación sin servidor, no se despliegan, pero pueden crearse cuando sea necesario con la configuración y el código de aplicación pertinentes.
- Espera semiactiva (RPO en segundos, RTO en minutos): mantenga una versión reducida pero totalmente funcional de su carga de trabajo que se ejecute continuamente en la región de recuperación. Los sistemas críticos se duplican en su totalidad y siempre están activos, pero con una flota reducida. Los datos se replican y están activos en la región de recuperación. Cuando llegue el momento de la recuperación, el sistema se amplía rápidamente para asumir la carga de producción. Cuanto mayor sea la escala de la espera semiactiva, menor será el RTO y la fiabilidad del plano de control. Cuando alcanza su plena escala, la espera pasa a denominarse espera activa.

- Activa-activa multirregión (multisitio) (RPO próximo a cero, RTO potencial de cero): la carga de trabajo está desplegada en varias Regiones de AWS y entrega tráfico de forma activa desde estas regiones. Esta estrategia requiere que sincronice datos entre regiones. Los posibles conflictos causados por escrituras en el mismo registro en dos diferentes réplicas regionales deben evitarse o gestionarse, lo que puede resultar complejo. La replicación de datos es útil para la sincronización de datos y le protegerá ante algunos tipos de desastres, pero no ante el daño o la destrucción de datos, a no ser que su solución incluya también opciones para una recuperación a un momento dado.

> **Note**
>
> La diferencia entre la luz piloto y la espera semiactiva a veces puede ser difícil de comprender. Ambos métodos incluyen un entorno en su región de recuperación con copias de los activos de su región principal. La distinción es que la luz piloto no puede procesar solicitudes sin tomar primero acciones adicionales, mientras que la espera semiactiva puede gestionar el tráfico (a niveles de capacidad reducidos) inmediatamente. La luz piloto exige que active servidores, posiblemente que despliegue infraestructura adicional (no principal) y que escale verticalmente, mientras que la espera semiactiva solo requiere que escale verticalmente (ya está todo desplegado y en ejecución). Elija una de estas opciones en función de sus necesidades de RTO y RPO.
>
> Cuando el coste sea una preocupación y desee alcanzar unos objetivos de RPO y RTO similares a los definidos en la estrategia de espera semiactiva, podría plantearse soluciones nativas de la nube, como AWS Elastic Disaster Recovery, que adoptan el enfoque de luz piloto y ofrecen objetivos de RPO y RTO mejorados.

Pasos para la implementación

1. Determine una estrategia de DR que satisfaga los requisitos de recuperación de esta carga de trabajo.

La selección de una estrategia de DR requiere alcanzar un punto de equilibrio entre la reducción del tiempo de inactividad y la pérdida de datos (RTO y RPO) y los costes y la complejidad de implementar la estrategia. Debería evitar implementar una estrategia que sea más exigente de lo necesario, ya que esto supone costes innecesarios.

Por ejemplo, en el siguiente diagrama, la empresa ha determinado su RTO máximo permisible y el límite de gasto en su estrategia de restauración del servicio. Dados los objetivos de la empresa, las estrategias de DR de luz piloto o espera semiactiva satisfarán tanto el RTO como los criterios de coste.

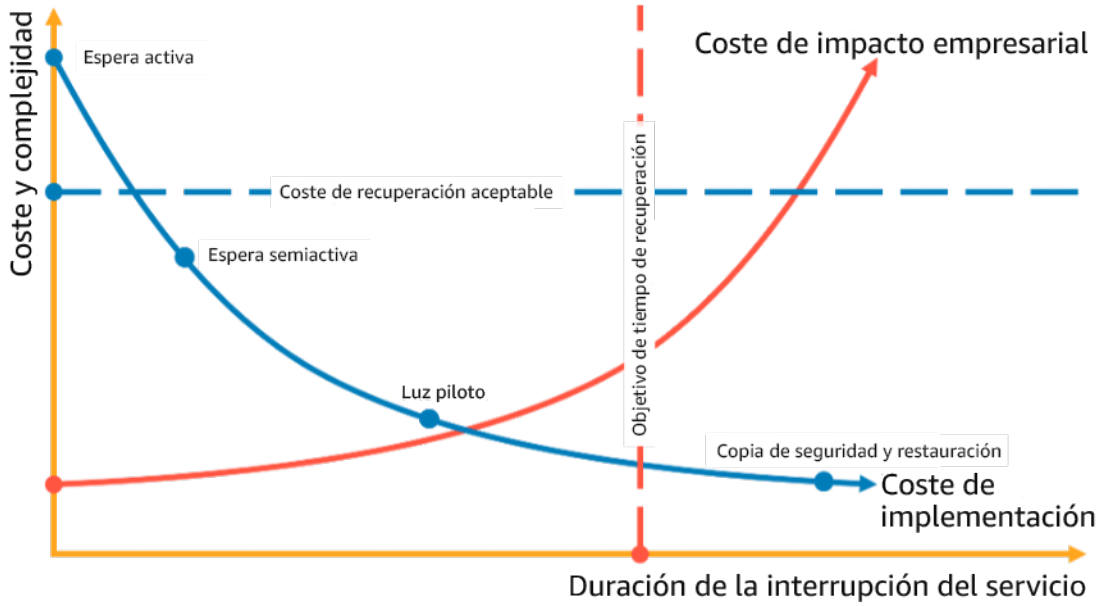



Figura 18: Selección de una estrategia de DR basada en el RTO y el coste

Para obtener más información, consulte Business Continuity Plan (BCP) (Plan de continuidad del negocio [BCP]).

2. Revise los patrones de implementación de la estrategia de DR seleccionada.

Este paso implica comprender cómo implementará la estrategia seleccionada. Las estrategias se explican utilizando Regiones de AWS para determinar un sitio principal y otro de recuperación. Sin embargo, también puede decidir utilizar zonas de disponibilidad en una única región como estrategia de DR, que utiliza los elementos de varias de estas estrategias.

En los siguientes pasos, puede aplicar la estrategia a su carga de trabajo específica.

Generación de copias de seguridad y restauración

Generación de copias de seguridad y restauración es la estrategia menos compleja de implementar, pero requiere más tiempo y esfuerzo para restaurar la carga de trabajo, lo que genera un mayor RTO y RPO. Se recomienda realizar siempre copias de seguridad de los datos y copiarlas en otro sitio (por ejemplo, otra Región de AWS).

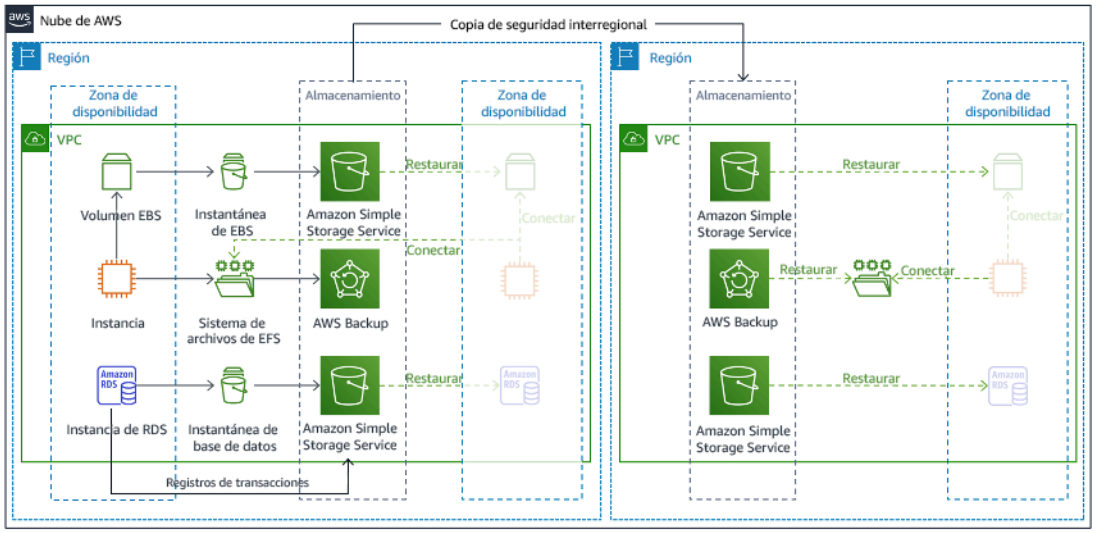



Figura 19: Arquitectura de copia de seguridad y restauración

Para obtener más detalles sobre esta estrategia, consulte Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery (Arquitectura de recuperación de desastres [DR] en AWS, parte II: copias de seguridad y restauración con Rapid Recovery).

Luz piloto

Con el enfoque luz piloto, replica sus datos de la región principal en la región de recuperación. Los recursos principales utilizados para la infraestructura de la carga de trabajo se despliegan en la región de recuperación; sin embargo, se siguen necesitando recursos adicionales y las dependencias pertinentes para que esta pila sea funcional. Por ejemplo, en la figura 20 no se despliegan instancias de computación.

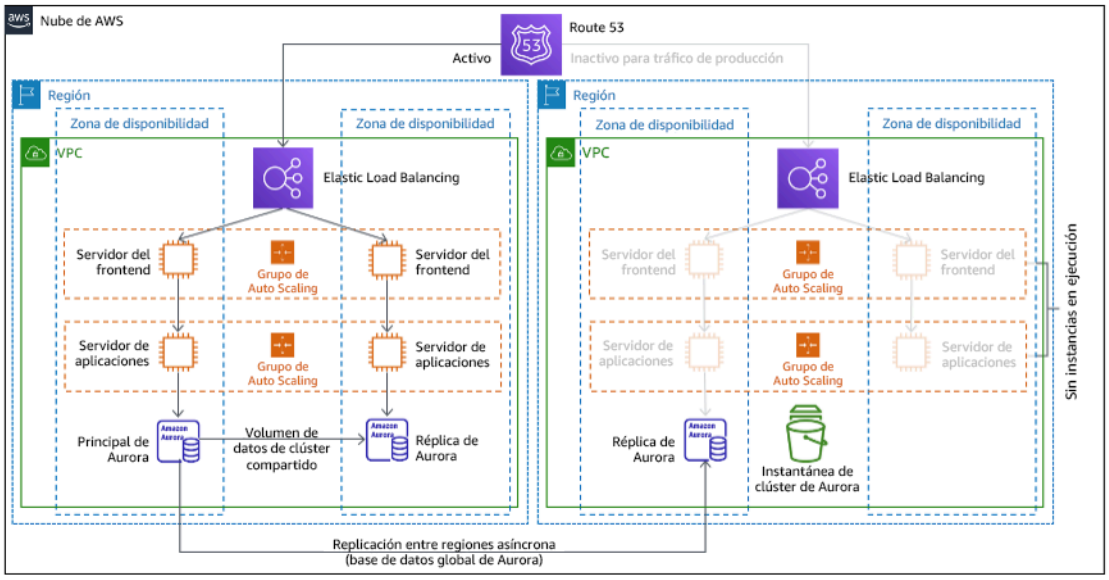



Figura 20: Arquitectura de luz piloto

Para obtener más detalles sobre esta estrategia, consulte Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby (Arquitectura de recuperación de desastres [DR] en AWS, parte III: luz piloto y espera semiactiva).

Espera semiactiva

El enfoqueespera semiactiva supone garantizar que exista una copia con desescalada vertical pero con plena funcionalidad de su entorno de producción en otra región. Este enfoque extiende el concepto de luz piloto y reduce el tiempo de recuperación, ya que su carga de trabajo tiene disponibilidad permanente en otra región. Si la región de recuperación se despliega a plena capacidad, esto se denomina espera activa.

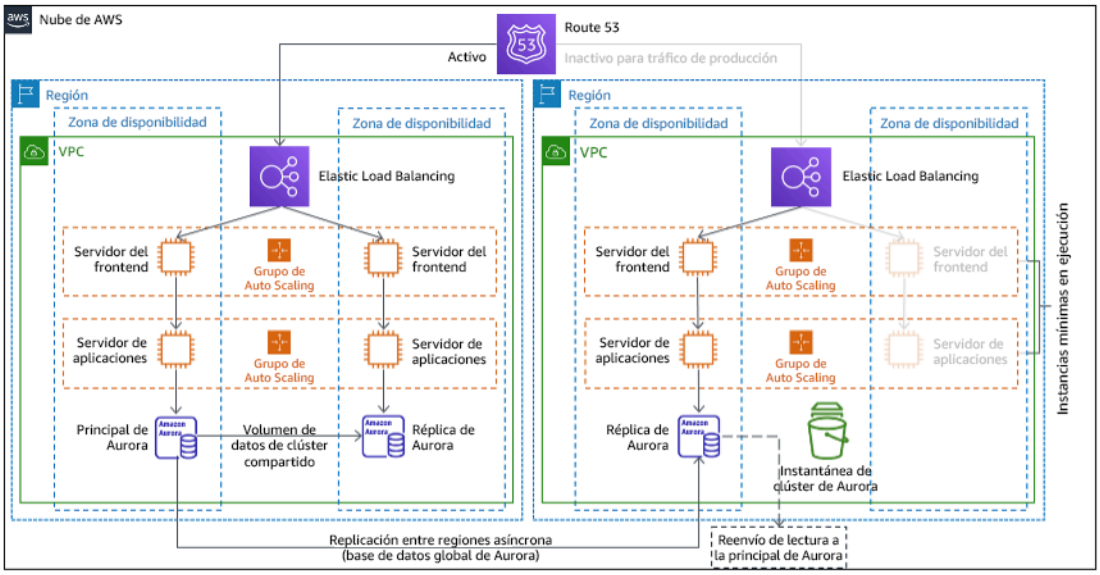



Figura 21: Arquitectura de espera semiactiva

El uso de las estrategias espera semiactiva o luz piloto requiere escalar verticalmente los recursos en la región de recuperación. Para verificar que la capacidad esté disponible cuando se necesita, considere el uso de reservas de capacidad para instancias EC2. Si usa AWS Lambda, entonces la simultaneidad aprovisionada puede proporcionar entornos de ejecución de forma que estén preparados para responder de inmediato a las invocaciones de la función.

Para obtener más detalles sobre esta estrategia, consulte Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby (Arquitectura de recuperación de desastres [DR] en AWS, parte III: luz piloto y espera semiactiva).

Activa-activa multisitio

Puede ejecutar su carga de trabajo de forma simultánea en varias regiones como parte de una estrategia activa-activa multisitio. La estrategia activa-activa multisitio suministra tráfico desde todas las regiones en las que se despliega. Los clientes podrían seleccionar esta estrategia por motivos ajenos a la DR. Se puede usar para aumentar la disponibilidad o cuando se despliega una carga de trabajo para una audiencia global (para colocar el punto de conexión más cerca de los usuarios o para desplegar pilas localizadas para la audiencia de esa región). Como estrategia de DR, si la carga de trabajo no es compatible en una de las Regiones de AWS en la que esté desplegada, esa región se evacúa y las regiones restantes se usan para mantener la disponibilidad. La estrategia activa-activa multisitio es la más compleja de las estrategias de DR a nivel operativo, y solo debería seleccionarse cuando los requisitos empresariales lo exijan.

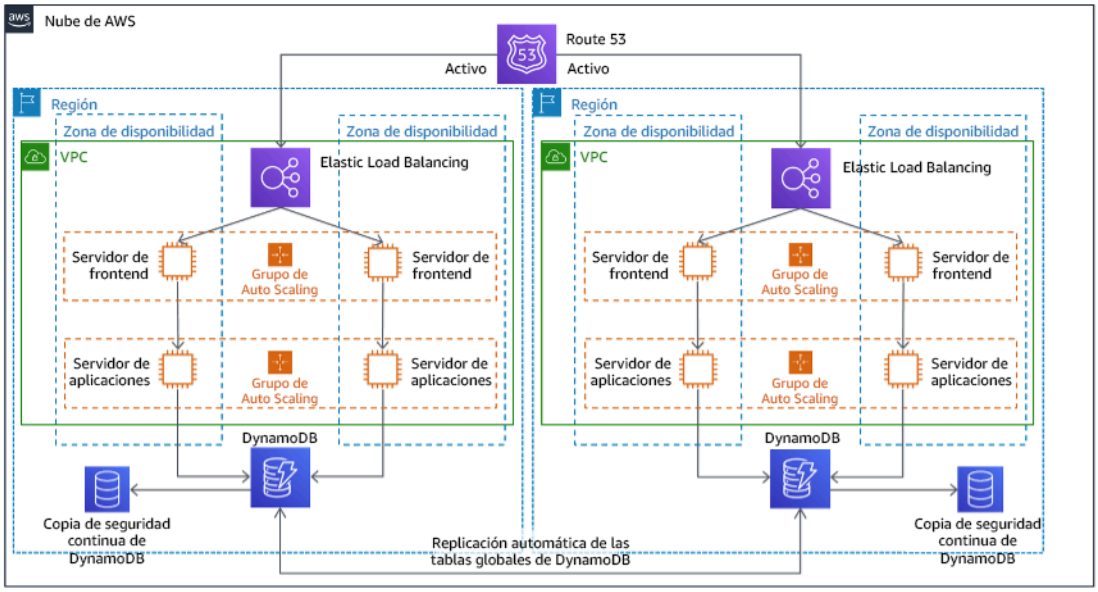



Figura 22: Arquitectura activa-activa multisitio

Para obtener más detalles sobre esta estrategia, consulte Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active (Arquitectura de recuperación de desastres [DR] en AWS, parte IV: activa-activa multisitio).

AWS Elastic Disaster Recovery

Si está considerando la estrategia luz piloto o espera semiactiva para la recuperación de desastres, AWS Elastic Disaster Recovery podría ofrecer un enfoque alternativo con mayores ventajas. Elastic Disaster Recovery puede ofrecer un objetivo de RPO y RTO similar al de la estrategia espera semiactiva, pero manteniendo el enfoque de bajo coste de la estrategia luz piloto. Elastic Disaster Recovery replica los datos desde la región primaria a la región de recuperación, utilizando la protección continua de datos para lograr un RPO medido en segundos y un RTO que puede medirse en minutos. En la región de recuperación solo se despliegan los recursos necesarios para replicar los datos, lo que mantiene los costes bajos, de forma similar a la estrategia luz piloto. Cuando se utiliza Elastic Disaster Recovery, el servicio coordina y organiza la recuperación de los recursos de computación cuando se inicia como parte de una conmutación por error o un simulacro.

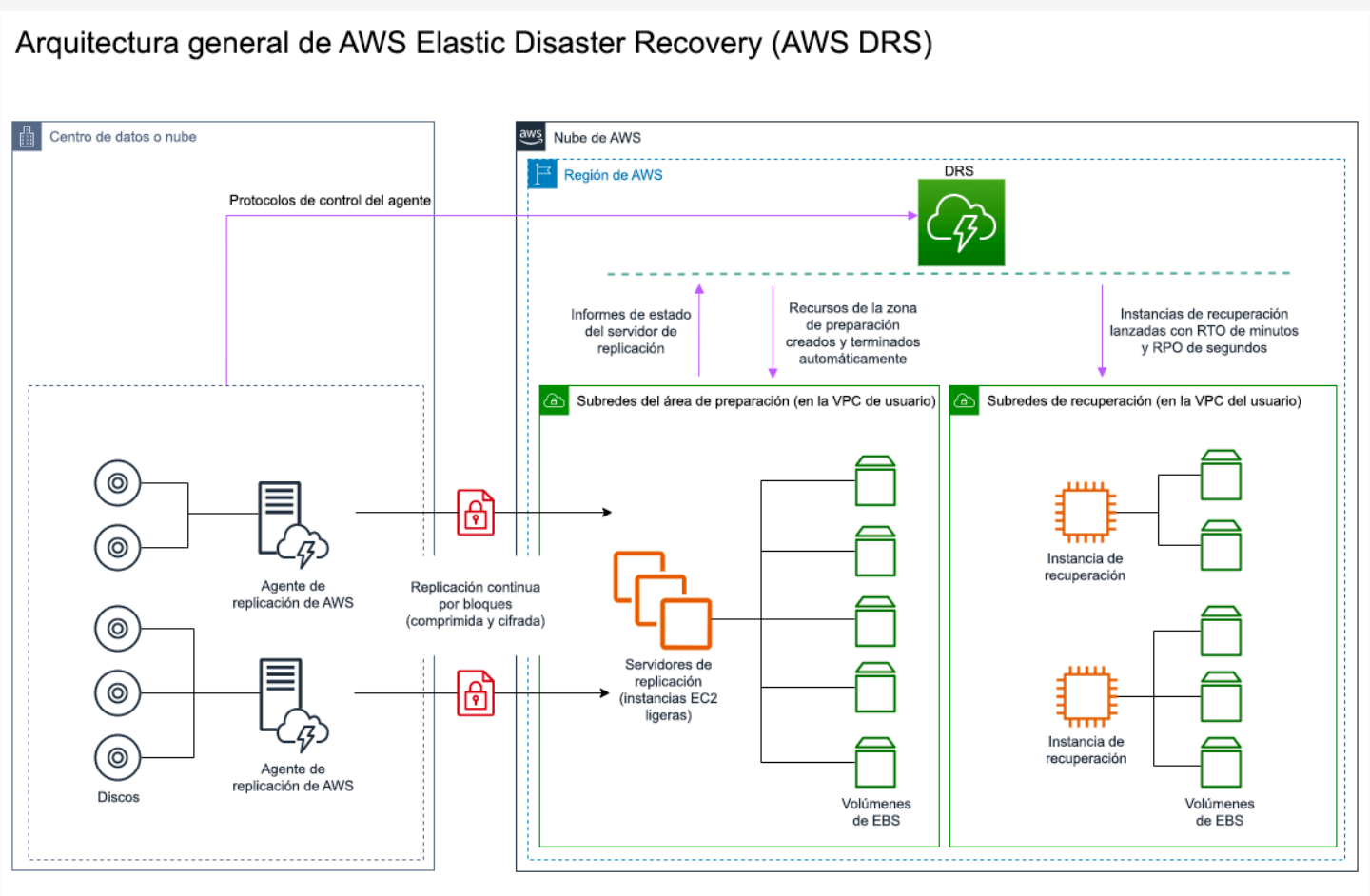



Figura 23: Arquitectura de AWS Elastic Disaster Recovery

Prácticas adicionales para la protección de datos

Con todas las estrategias, también debe mitigar los posibles desastres de datos. La replicación de datos continua le protegerá ante algunos tipos de desastres, pero no ante el daño o la destrucción de datos, a no ser que su estrategia incluya también control de versiones para una recuperación a un momento dado. También debe realizar una copia de seguridad de los datos replicados en el sitio de recuperación para crear copias de seguridad a un momento dado además de las réplicas.

Uso de varias zonas de disponibilidad (AZ) en una únicaRegión de AWS

Al usar varias AZ en una única región, su implementación de DR utiliza varios elementos de las estrategias anteriores. Primero, debe crear una arquitectura de alta disponibilidad utilizando varias AZ, como se muestra en la figura 23. Esta arquitectura hace uso de un enfoque activo-activo multisitio, ya que las instancias Amazon EC2 y el equilibrador de carga elástico tienen recursos

desplegados en varias AZ, que gestionan las solicitudes de forma activa. La arquitectura también dispone de espera activa, en la que si la instancia principal de Amazon RDS produce un error (o la propia AZ tiene un error), la instancia en espera pasa a ser la principal.

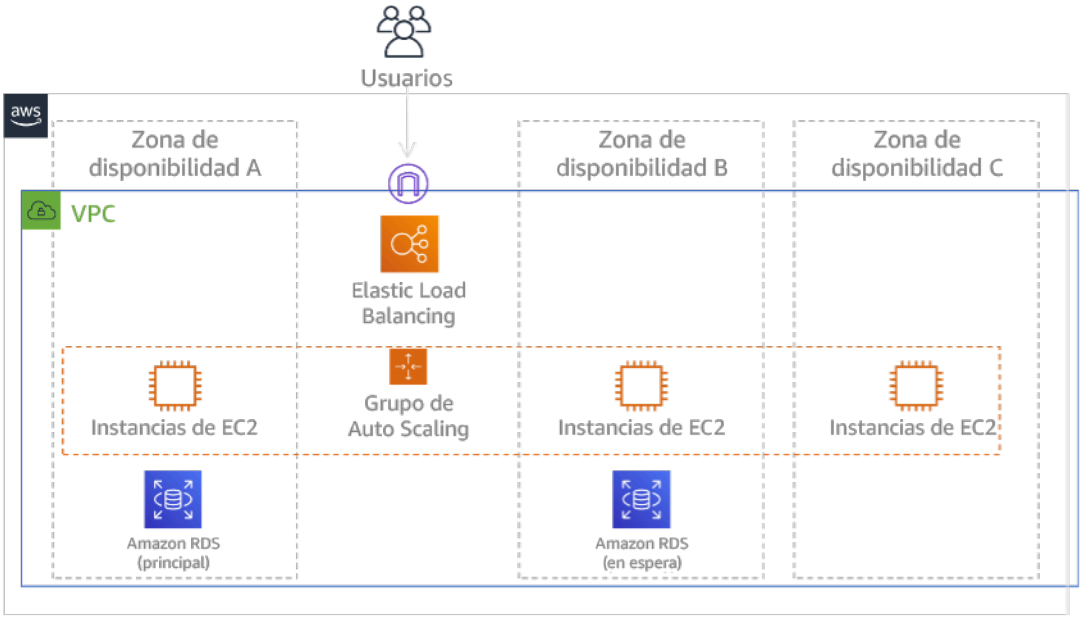



Figura 24: Arquitectura multi-AZ

Además de esta arquitectura de alta disponibilidad, necesita agregar copias de seguridad con todos los datos necesarios para ejecutar su carga de trabajo. Esto es especialmente importante para datos que estén limitados a una única zona, como los volúmenes de Amazon EBS o los clústeres de Amazon Redshift. Si falla una AZ, tendrá que restaurar estos datos en otra AZ. Siempre que sea posible, deberá copiar las copias de seguridad de los datos en otra Región de AWS como capa de protección adicional.

En la publicación de blog, Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack (Creación de aplicaciones de alta resiliencia con Amazon Route 53, parte 1: pila de una sola región) se ilustra un enfoque alternativo menos habitual a la RD multi-AZ de única región. Aquí, la estrategia es mantener la máxima cantidad posible de aislamiento entre las AZ, tal y como funcionan las regiones. Al utilizar esta estrategia alternativa, puede elegir un enfoque activo-activo o activo-pasivo.

> **Note**
>
> Algunas cargas de trabajo tienen requisitos normativos de residencia de los datos. Si esto se aplica a su carga de trabajo en una ubicación que actualmente tenga solo una Región

de AWS, el enfoque multirregión no se adaptará a sus necesidades empresariales. Las estrategias multi-AZ ofrecen una buena protección contra la mayor parte de los desastres.

3. Evalúe los recursos de su carga de trabajo y cuál será su configuración en la región de recuperación antes de la conmutación por error (durante la operación normal).

Para la infraestructura y los recursos de AWS, utilice infraestructura como código como AWS CloudFormation o herramientas de terceros como Hashicorp Terraform. Para un despliegue en varias cuentas y regiones con una única operación, puede utilizar AWS CloudFormation StackSets. En las estrategias activa-activa multisitio o espera activa, la infraestructura desplegada en su región de recuperación tiene los mismos recursos que su región principal. En las estrategias luz piloto y espera semiactiva, la infraestructura desplegada requerirá acciones adicionales para prepararse para la producción. Con parámetros y lógica condicional de CloudFormation, puede controlar si una pila desplegada está activa o en espera con una sola plantilla. Al utilizar Elastic Disaster Recovery, el servicio replicará y organizará la restauración de las configuraciones de las aplicaciones y los recursos de computación.

Todas las estrategias de DR requieren que se realicen copias de seguridad de los orígenes de datos en la Región de AWS, y que estas copias de seguridad se copien en la región de recuperación. AWS Backup proporciona una vista centralizada en la que puede configurar, programar y supervisar las copias de seguridad para estos recursos. En el caso de las opciones de luz piloto, espera semiactiva y activa-activa multisitio, también debería replicar los datos de la región principal en los recursos de datos de la región de recuperación, como las instancias de base de datos de Amazon Relational Database Service (Amazon RDS) o tablas de Amazon DynamoDB. De esta forma, estos recursos de datos estarán activos y preparados para responder a solicitudes en la región de recuperación.

Para obtener más información sobre cómo funcionan los servicios de AWS en todas las regiones, consulte esta serie de blogs sobre Creating a Multi-Region Application with AWS Services (Creación de una aplicación multirregión con servicios de AWS).

4. Determine e implemente cómo preparará su región de recuperación para la conmutación por error cuando sea necesario (durante un evento de desastre).

En el caso de la opción activa-activa multisitio, la conmutación por error implica evacuar una región y recurrir a las regiones activas restantes. En general, esas regiones están listas para aceptar tráfico. En las estrategias luz piloto y espera semiactiva, sus acciones de recuperación tendrán que

desplegar los recursos faltantes, como las instancias EC2 en la figura 20, además de otros recursos faltantes.

En todas las estrategias anteriores, es posible que tenga que promover instancias de solo lectura de bases de datos para que se conviertan en la instancia de lectura y escritura principal.

En copias de seguridad y restauración, la restauración de datos desde una copia de seguridad crea recursos para esos datos, como volúmenes EBS, instancias de bases de datos RDS y tablas de DynamoDB. También tiene que restaurar la infraestructura y desplegar el código. Puede usar AWS Backup para restaurar datos en la región de recuperación. Consulte REL09-BP01 Identificar todos los datos de los que se debe hacer una copia de seguridad y crearla o reproducir los datos a partir de los orígenes para obtener más información. La reconstrucción de la infraestructura incluye la creación de recursos como instancias EC2, además de Amazon Virtual Private Cloud (Amazon VPC), subredes y grupos de seguridad necesarios. Puede automatizar gran parte del proceso de restauración. Para saber cómo, consulte esta publicación de blog.

5. Determine e implemente cómo redirigirá su tráfico a la conmutación por error cuando sea necesario (durante un evento de desastre).

Esta operación de conmutación por error se puede iniciar automáticamente o manualmente. La conmutación por error iniciada automáticamente basada en comprobaciones de estado o alarmas se debe usar con cuidado, ya que una conmutación por error innecesaria (falsa alarma) supone ciertos inconvenientes, como la falta de disponibilidad y la pérdida de datos. Por tanto, la conmutación por error iniciada manualmente es la que se suele utilizar. En este caso, debe seguir automatizando los pasos de la conmutación por error, de modo que la iniciación manual sea como pulsar un botón.

Hay varias opciones de administración del tráfico que tener en cuenta al usar servicios de AWS. Una opción es utilizar Amazon Route 53. Al usar Amazon Route 53, puede asociar varios puntos de conexión de IP en una o varias Regiones de AWS con un nombre de dominio de Route 53. Para implementar la conmutación por error iniciada manualmente, puede utilizar el Controlador de recuperación de aplicaciones de Amazon Route 53, que proporciona una API de plano de datos de alta disponibilidad para redirigir el tráfico a la región de recuperación. Al implementar la conmutación por error, use las operaciones del plano de datos y evite las del plano de control, como se describe en REL11-BP04 Confiar en el plano de datos y no en el plano de control durante la recuperación.

Para obtener más información sobre esta y otras opciones, consulte esta sección del documento técnico de recuperación de desastres.

6. Diseñe un plan para la recuperación tras error de su carga de trabajo.

La conmutación por recuperación es cuando se devuelve la operación de una carga de trabajo a la región principal una vez que amaina un evento de desastre. El aprovisionamiento de la infraestructura y el código en la región principal generalmente sigue los mismos pasos que se utilizaron inicialmente, recurriendo a la infraestructura como código y las canalizaciones de despliegue del código. El reto que plantea la conmutación por recuperación es restaurar los almacenes de datos y asegurarse de que sean coherentes con la región de recuperación en funcionamiento.

En el estado de conmutación por error, las bases de datos en la región de recuperación están activas y tienen los datos actualizados. El objetivo es resincronizar desde la región de recuperación a la región principal, garantizando así que esté actualizada.

Algunos servicios de AWS harán esto automáticamente. Si utiliza las tablas globales de Amazon DynamoDB, incluso si la tabla en la región principal ha dejado de estar disponible, cuando vuelva a estar online, DynamoDB volverá a propagar las escrituras pendientes. Si utiliza la base de datos global de Amazon Aurora y utiliza la conmutación por error planificada administrada, se mantendrá la topología de replicación existente de la base de datos global de Aurora. Por tanto, la instancia de lectoescritura anterior en la región principal se convertirá en una réplica y recibirá actualizaciones desde la región de recuperación.

En los casos en los que esto no se haga automáticamente, tendrá que restablecer la base de datos en la región principal como una réplica de la base de datos en la región de recuperación. En muchos casos, esto supondrá eliminar la antigua base de datos principal y crear nuevas réplicas. Por ejemplo, para obtener instrucciones sobre cómo hacer esto con la base de datos global de Amazon Aurora en el caso de una conmutación por error no planificada, consulte este laboratorio: Fail Back a Global Database (Conmutación por recuperación a una base de datos global).

Tras una conmutación por error, si puede seguir operando en su región de recuperación, considere convertir esta región en la nueva región principal. Seguiría realizando los pasos anteriores para hacer que la antigua región principal fuera una región de recuperación. Algunas organizaciones llevan a cabo una rotación programada y cambian sus regiones principal y de recuperación periódicamente (por ejemplo, cada tres meses).

Todos los pasos necesarios para la conmutación por error y la restauración tras error deben mantenerse en una guía de estrategias que esté a disposición de todos los miembros del equipo y se revise periódicamente.

Si utiliza Elastic Disaster Recovery, el servicio ayudará a organizar y automatizar el proceso de conmutación por recuperación. Para obtener más información, consulte Performing a failback (Realizar una conmutación por recuperación).

Nivel de esfuerzo para el plan de implementación: alto

## Recursos

Prácticas recomendadas relacionadas:

- the section called “REL09-BP01 Identificar todos los datos de los que se debe hacer una copia de seguridad y crearla o reproducir los datos a partir de los orígenes”
- the section called “REL11-BP04 Confiar en el plano de datos y no en el plano de control durante la recuperación”
- the section called “REL13-BP01 Definir objetivos de recuperación para la inactividad y la pérdida de datos”

Documentos relacionados:

- AWS Architecture Blog: Disaster Recovery Series (Blog de arquitectura de AWS: serie de recuperación de desastres)
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (Recuperación de cargas de trabajo en caso de desastre en AWS: recuperación en la nube) (Documento técnico de AWS)
- Opciones de recuperación de desastres en la nube
- Crear una solución backend activa-activa sin servidor en varias regiones en una hora
- Backend sin servidor en varias regiones: actualizado
- RDS: replicación de una réplica de lectura entre regiones
- Route 53: Configuración de la recuperación ante errores a nivel de DNS
- S3: replicación entre regiones
- What Is AWS Backup? (¿Qué es AWS Backup?)
- What is Route 53 Application Recovery Controller? (¿Qué es el Controlador de recuperación de aplicaciones de Route 53?)
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Get Started - AWS (HashiCorp Terraform: primeros pasos: AWS)
- Socio de APN: socios que pueden ayudar con la recuperación de desastres

- AWS Marketplace: products that can be used for disaster recovery (AWS Marketplace: productos que pueden usarse para la recuperación de desastres)

Vídeos relacionados:

- Disaster Recovery of Workloads on AWS (Recuperación de desastres de cargas de trabajo en AWS)
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2) (AWS re:Invent 2018: Patrones de arquitectura para aplicaciones activas-activas en varias regiones)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services (Introducción a AWS Elastic Disaster Recovery | Amazon Web Services)

Ejemplos relacionados:

- Well-Architected Lab - Disaster Recovery - Series of workshops illustrating DR strategies (Laboratorio de Well-Architected: Recuperación de desastres: serie de talleres que ilustran las estrategias de DR)

## REL13-BP03 Probar la implementación de recuperación de desastres para validarla

Compruebe periódicamente la conmutación por error a su sitio de recuperación para verificar que funcione adecuadamente y que se cumplan el RTO y el RPO.

Antipatrones usuales:

- No llevar a cabo nunca conmutaciones por error en producción.

Beneficios de establecer esta práctica recomendada: las pruebas periódicas del plan de recuperación de desastres verifican que el plan funcione cuando llegue el momento y que su equipo sepa cómo ejecutar la estrategia.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: alto

## Guía para la implementación

Un patrón que debe evitarse es el desarrollo de rutas de recuperación que se pongan en práctica pocas veces. Por ejemplo, puede tener un almacén de datos secundario que se utilice para consultas de solo lectura. Cuando escribe en un almacén de datos y el almacén principal falla, es posible que quiera conmutar por error al almacén de datos secundario. Si no se prueba frecuentemente esta conmutación por error, es posible que sus suposiciones sobre las capacidades del almacén de datos secundario sean incorrectas. Es posible que la capacidad del almacén de datos secundario, que quizás fuera suficiente cuando se probó por última vez, ya no pueda tolerar la carga en esta situación. Nuestra experiencia ha demostrado que la única forma de recuperación de errores que funciona es aquella que prueba constantemente. Por ello, es mejor tener un número reducido de rutas de recuperación. Puede establecer patrones de recuperación y probarlos con frecuencia. Si tiene una ruta de recuperación compleja o crítica, todavía debe llevar a efecto ese error en producción periódicamente para asegurarse de que la ruta funcione. En el ejemplo que acabamos de comentar, se debe conmutar por error al modo de espera con regularidad, sin importar si es necesario.

Pasos para la implementación

1. Diseñe sus cargas de trabajo para que se puedan recuperar. Pruebe regularmente sus rutas de recuperación. La computación orientada a la recuperación identifica las características de los sistemas que mejoran la recuperación: aislamiento y redundancia, capacidad en todo el sistema para revertir los cambios, capacidad para supervisar y determinar el estado, capacidad para proporcionar diagnósticos, recuperación automatizada, diseño modular y capacidad para reiniciar. Ponga en práctica la ruta de recuperación para verificar que pueda cumplir la recuperación en el tiempo especificado para el estado especificado. Use sus runbooks durante esta recuperación para documentar los problemas y encontrar soluciones para ellos antes de la próxima prueba.
2. Para cargas de trabajo basadas en Amazon EC2, utilice AWS Elastic Disaster Recovery para implementar y lanzar instancias de simulacro para su estrategia de recuperación de desastres. AWS Elastic Disaster Recovery ofrece la posibilidad de ejecutar simulacros de manera eficiente, lo que le ayuda a prepararse para un evento de conmutación por error. También puede lanzar con frecuencia sus instancias mediante Elastic Disaster Recovery para realizar pruebas y simulacros sin redirigir el tráfico.

## Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudar con la recuperación de desastres
- AWS Architecture Blog: Disaster Recovery Series (Blog de arquitectura de AWS: serie de recuperación de desastres)
- AWS Marketplace: products that can be used for disaster recovery (AWS Marketplace: productos que pueden usarse para la recuperación de desastres)
- AWS Elastic Disaster Recovery
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (Recuperación de cargas de trabajo en caso de desastre en AWS: Recuperación en la nube) (documento técnico de AWS)
- AWS Elastic Disaster Recovery Preparing for Failover (AWS Elastic Disaster Recovery: Preparación para la conmutación por error)
- Proyecto de informática orientada a la recuperación de Berkeley/Stanford
- ¿Qué es AWS Fault Injection Simulator?

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (AWS re:Invent 2018: Patrones de arquitectura para aplicaciones activas-activas en varias regiones)
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS (AWS re:Invent 2019: Copia de seguridad y restauración y soluciones de recuperación de desastres con AWS)

Ejemplos relacionados:

- Well-Architected Lab - Testing for Resiliency (Laboratorio de Well-Architected: Prueba de resiliencia)

## REL13-BP04 Administrar la desviación de la configuración en el sitio de o en la región de recuperación de desastres

Asegúrese de que la infraestructura, los datos y la configuración estén cuando se necesiten en el sitio o región de DR. Por ejemplo, compruebe que las AMI y las cuotas de servicio están actualizadas.

AWS Config supervisa y registra continuamente las configuraciones de sus recursos de AWS. Puede detectar la desviación y desencadenar Automatización de AWS Systems Manager para solucionarlo

y generar alarmas. Además, AWS CloudFormation puede detectar la desviación en las pilas que ha desplegado.

Patrones de uso no recomendados comunes:

- No realizar actualizaciones en sus ubicaciones de recuperación, cuando realice cambios de configuración o de infraestructura en sus ubicaciones primarias.
- No considerar las posibles limitaciones (como las diferencias en los servicios) en las ubicaciones principales y de recuperación.

Beneficios de establecer esta práctica recomendada: Comprobar que su entorno de DR es coherente con el entorno existente garantiza una recuperación completa.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

## Guía para la implementación

- Asegúrese de que sus canalizaciones de entrega realizan la entrega tanto al sitio principal como al de copia de seguridad. Las canalizaciones de entrega para implementar aplicaciones en producción deben distribuir la entrega a todas las ubicaciones de la estrategia de recuperación de desastres especificadas, incluidos los entornos de desarrollo y pruebas.
- Habilite AWS Config para realizar un seguimiento de las posibles ubicaciones con desviaciones. Use reglas de AWS Config para crear sistemas que apliquen sus estrategias de recuperación de desastres y creen alertas si detectan divergencias.
  - Corrección de recursos de AWS disconformes con Reglas de AWS Config
  - Automatización de AWS Systems Manager
- Use AWS CloudFormation para desplegar su infraestructura. AWS CloudFormation puede detectar la desviación entre lo que especifican sus plantillas de CloudFormation y lo que realmente está desplegado.
  - AWS CloudFormation: detectar desviaciones en una pila completa de CloudFormation

## Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudar con la recuperación de desastres
- Blog de arquitectura de AWS: serie de recuperación de desastres

- AWS CloudFormation: detectar desviaciones en una pila completa de CloudFormation
- AWS Marketplace: productos que pueden usarse para la recuperación de desastres
- Automatización de AWS Systems Manager
- Recuperación de desastres de las cargas de trabajo en AWS: recuperación en la nube (documento técnico de AWS)
- ¿Cómo implemento una solución de administración de la configuración de la infraestructura en AWS?
- Corrección de recursos de AWS disconformes con Reglas de AWS Config

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (Patrones de arquitectura para aplicaciones activas-activas en varias regiones) (ARC209-R2)

## REL13-BP05 Automatizar la recuperación

Use AWS o herramientas de terceros para automatizar la recuperación del sistema y dirigir el tráfico al sitio o región de DR.

En función de las comprobaciones de estado configuradas, los servicios de AWS, como Elastic Load Balancing y AWS Auto Scaling, pueden distribuir la carga a zonas de disponibilidad en buen estado mientras que los servicios, como Amazon Route 53 y AWS Global Accelerator, pueden dirigir la carga a Regiones de AWS en buen estado. Amazon Route 53 Application Recovery Controller le ayuda a administrar y coordinar la conmutación por error mediante comprobaciones de idoneidad y funciones de control de enrutamiento. Estas características supervisan continuamente la capacidad de la aplicación de recuperarse de los errores, de modo que pueda controlar la recuperación de la aplicación en las distintas Regiones de AWS, zonas de disponibilidad y localmente.

Para cargas de trabajo en centros de datos físicos o virtuales existentes o nubes privadas, AWS Elastic Disaster Recovery, disponible en AWS Marketplace, permite a las organizaciones configurar una estrategia de recuperación de desastres automatizada en AWS. CloudEndure también admite la recuperación de desastres entre regiones o AZ en AWS.

Antipatrones usuales:

- La implementación de técnicas de conmutación por error y de conmutación por recuperación idénticas puede producir una alteración cuando surge un error.

Beneficios de establecer esta práctica recomendada: La recuperación automatizada reduce el tiempo de recuperación al eliminar la posibilidad de que se produzcan errores manuales.

Nivel de riesgo expuesto si no se establece esta práctica recomendada: Mediana

## Guía para la implementación

- Automatice las rutas de recuperación. Para tiempos de recuperación cortos, las decisiones y las acciones humanas no pueden usarse para escenarios de alta disponibilidad. El sistema debe recuperarse automáticamente en cada situación.
  - Use la recuperación de desastres de Cloudendure para la conmutación por error y la restauración tras error automatizadas. La recuperación de desastres de CloudEndure replica continuamente las máquinas (incluido el sistema operativo, la configuración de estado del sistema, las bases de datos, las aplicaciones y los archivos) en un área de ensayo de bajo costo en su Cuenta de AWS de destino y región preferida. En caso de desastre, puede indicar a CloudEndure Disaster Recovery que lance automáticamente miles de máquinas en su estado aprovisionado completo en solo unos minutos.
    - Realizar la conmutación por error y la conmutación por recuperación de recuperación de desastres
    - CloudEndure Disaster Recovery

## Recursos

Documentos relacionados:

- Socio de APN: socios que pueden ayudar con la recuperación de desastres
- Blog de arquitectura de AWS: serie de recuperación de desastres
- AWS Marketplace: productos que pueden usarse para la recuperación de desastres
- AWS Systems Manager Automation
- Recuperación de desastres de CloudEndure en AWS
- Recuperación de desastres de las cargas de trabajo en AWS: recuperación en la nube (documento técnico de AWS)

Videos relacionados:

- AWS re:Invent 2018: Patrones de arquitectura para aplicaciones activas-activas en varias regiones (ARC209-R2)

# Implementaciones de ejemplo para objetivos de disponibilidad

En esta sección, revisaremos diseños de cargas de trabajo con el despliegue de una aplicación web típica que consiste en un proxy inverso, contenido estático en Amazon S3, un servidor de aplicaciones y una base de datos SQL para el almacenamiento persistente de datos. Para cada objetivo de disponibilidad, proporcionamos un ejemplo de implementación. Esta carga de trabajo podría usar contenedores o AWS Lambda para la computación y NoSQL (como Amazon DynamoDB) para la base de datos, pero los enfoques son similares. En cada escenario, demostramos cómo cumplir los objetivos de disponibilidad mediante el diseño de cargas de trabajo para estos temas:

| Tema | Para obtener más información, consulte esta sección |
| --- | --- |
| Monitorear los recursos | Supervisar los recursos de la carga de trabajo |
| Adaptación a los cambios en la demanda | Diseñar su carga de trabajo para que se adapte a los cambios en la demanda |
| Implementar cambios | Implementar cambios |
| Copia de seguridad de los datos | Copia de seguridad de los datos |
| Diseñar para la resiliencia | Usar el aislamiento de errores para proteger su carga de trabajo<br><br>Diseñar su carga de trabajo para que soporte los errores de los componentes |
| Prueba de resiliencia | Comprobar la fiabilidad |
| Planificar para la recuperación de desastres (DR) | Planificar para la recuperación de desastres (DR) |

# Selección de dependencias

Hemos elegido usar Amazon EC2 para nuestras aplicaciones. Mostraremos cómo el uso de Amazon RDS y múltiples zonas de disponibilidad mejora la disponibilidad de nuestras aplicaciones. Usaremos Amazon Route 53 para el DNS. Cuando usemos varias zonas de disponibilidad, utilizaremos Elastic Load Balancing. Amazon S3 se usa para las copias de seguridad y el contenido estático. A medida que diseñamos para una mayor fiabilidad, debemos usar servicios con mayor disponibilidad.

# Situaciones de una sola región

Temas



## Situación de 2 nueves (99 %)

Estas cargas de trabajo son útiles para la empresa, pero solo suponen una inconveniencia si no están disponibles. Las cargas de trabajo de este tipo pueden ser herramientas internas, administración de conocimientos internos o seguimiento de proyectos. O pueden ser cargas de trabajo reales dirigidas al cliente pero atendidas desde un servicio experimental, con un conmutador de funciones que puede ocultar el servicio en caso necesario.

Estas cargas de trabajo se pueden desplegar con una región y una zona de disponibilidad.

### Monitorear los recursos

Tendremos un monitoreo simple, que indica si la página de inicio del servicio devuelve un estado HTTP 200 OK. Cuando se producen problemas, nuestra guía de estrategias indicará que el registro de la instancia se utilizará para establecer la causa raíz.

### Adaptación a los cambios en la demanda

Tendremos guías de estrategias para errores comunes de hardware, actualizaciones urgentes de software y otros cambios disruptivos.

## Implementar cambios

Usaremos AWS CloudFormation para definir nuestra infraestructura como código y, específicamente, para acelerar la reconstrucción en caso de error.

Las actualizaciones de software se realizan manualmente con un runbook, con un tiempo de inactividad requerido para la instalación y reinicio del servicio. Si ocurre un problema durante la implementación, el runbook describe cómo restaurar la versión anterior.

Las correcciones del error se llevan a cabo utilizando el análisis de registros por parte de los equipos de operaciones y desarrollo, y la corrección se despliega una vez que se prioriza y completa la solución.

## Copia de seguridad de los datos

Utilizaremos un proveedor o una solución de copia de seguridad especialmente diseñada para enviar datos de copia de seguridad encriptados a Amazon S3 usando un runbook. Probaremos que las copias de seguridad funcionen restaurando los datos y asegurando la capacidad de usarlos de manera regular con un runbook. Configuramos el control de versiones en nuestros objetos de Amazon S3 y eliminamos los permisos para eliminar las copias de seguridad. Utilizamos una política de ciclo de vida del bucket de Amazon S3 para archivar o eliminar permanentemente de acuerdo con nuestros requisitos.

## Diseñar para la resiliencia

Las cargas de trabajo se despliegan con una región y una zona de disponibilidad. Desplegaremos la aplicación, incluida la base de datos, en una sola instancia.

## Prueba de resiliencia

El despliegue del nuevo software está programado con algunas pruebas unitarias, pero principalmente pruebas de caja blanca o caja negra de la carga de trabajo ensamblada.

## Planificar para la recuperación de desastres (DR)

Durante los errores, esperamos a que finalicen y enrutamos opcionalmente las solicitudes a un sitio web estático con la modificación de DNS a través de un runbook. El tiempo de recuperación se determinará según la velocidad a la que se pueda desplegar la infraestructura y restaurar la base de datos a la copia de seguridad más reciente. Este despliegue puede realizarse en la misma zona de disponibilidad o en otra diferente en caso de error de la zona de disponibilidad con un runbook.

## Objetivo de diseño de disponibilidad

Nos tomamos 30 minutos para comprender y decidir ejecutar la recuperación, desplegar toda la pila en AWS CloudFormation en 10 minutos, asumir que desplegaremos en una nueva zona de disponibilidad y suponer que la base de datos se puede restaurar en 30 minutos. Esto implica que se tarda unos 70 minutos en recuperarse de un error. Suponiendo un error por trimestre, nuestro tiempo de impacto estimado para el año es de 280 minutos: es decir, cuatro horas y 40 minutos.

Esto significa que el límite superior de disponibilidad es del 99,9 %. La disponibilidad real también depende de la tasa real de error, la duración del error y la rapidez con que cada factor se recupera. En esta arquitectura, requerimos que la aplicación esté fuera de línea para aplicar las actualizaciones (estimando 24 horas al año: cuatro horas por cambio, seis veces al año), además de los eventos reales. Entonces, si consultamos la tabla de disponibilidad de aplicaciones anterior del documento técnico, vemos que nuestro objetivo de diseño de disponibilidad es del 99 %.

## Resumen

| Tema | Implementación |
| --- | --- |
| Monitorear los recursos | Comprobación solo del estado del sitio; sin alertas. |
| Adaptación a los cambios en la demanda | Ajuste de escalado vertical a través de la reimplementación. |
| Implementar cambios | Runbook para implementación y restauración. |
| Copia de seguridad de los datos | Runbook para copia de seguridad y restauración. |
| Diseñar para la resiliencia | Reconstrucción completa; restauración desde una copia de seguridad. |
| Prueba de resiliencia | Reconstrucción completa; restauración desde una copia de seguridad. |
| Planificar para la recuperación de desastres (DR) | Copias de seguridad encriptadas, restauración a una zona de disponibilidad diferente si es necesario. |

# Situación de 3 nueves (99,9 %)

El siguiente objetivo de disponibilidad es para aplicaciones para las cuales es importante tener alta disponibilidad, pero pueden tolerar periodos cortos de falta de disponibilidad. Este tipo de carga de trabajo generalmente se usa en operaciones internas que repercuten en los empleados cuando están inactivas. Este tipo de carga de trabajo también puede estar orientada al cliente, pero no genera ingresos elevados para el negocio y puede tolerar un tiempo de recuperación o un punto de recuperación más largos. Entre estas cargas de trabajo se incluyen aplicaciones administrativas para la gestión de cuentas o información.

Podemos mejorar la disponibilidad de las cargas de trabajo con dos zonas de disponibilidad para nuestro despliegue y separando las aplicaciones para separar los niveles.

## Monitorear los recursos

La supervisión se ampliará para alertar sobre la disponibilidad del sitio web en general verificando un estado HTTP 200 OK en la página de inicio. Además, habrá alertas en cada reemplazo de un servidor web y cuando la base de datos falle. También supervisaremos el contenido estático en Amazon S3 para verificar la disponibilidad y alertar si no está disponible. El registro se agregará para facilitar la administración y ayudar en el análisis de causa raíz.

## Adaptación a los cambios en la demanda

El escalado automático se configura para supervisar el uso de la CPU en instancias EC2 y para agregar o eliminar instancias para mantener el objetivo de la CPU en el 70 %, pero con no menos de una instancia EC2 por zona de disponibilidad. Si los patrones de carga en nuestra instancia del RDS indican que se necesita escalar verticalmente, cambiaremos el tipo de instancia durante un periodo de mantenimiento.

## Implementar cambios

Las tecnologías de despliegue de infraestructura siguen siendo las mismas que en el escenario anterior.

La entrega del nuevo software se realiza en un horario fijo de cada dos o cuatro semanas. Las actualizaciones de software se automatizarán no mediante patrones de despliegue de valor controlado o azul-verde, sino más bien mediante el reemplazo en su sitio. La decisión de restaurar se tomará con el runbook.

Tendremos guías de estrategias para establecer las causas raíz de los problemas. Una vez identificada la causa raíz, la corrección del error se identificará mediante una combinación de los equipos de operaciones y desarrollo. La corrección se desplegará después de que se desarrolle la solución.

## Copia de seguridad de los datos

La copia de seguridad y la restauración se pueden hacer con Amazon RDS. Se ejecutará regularmente con un runbook para garantizar que podamos cumplir con los requisitos de recuperación.

## Diseñar para la resiliencia

Podemos mejorar la disponibilidad de las aplicaciones con dos zonas de disponibilidad para nuestra implementación y separando las aplicaciones para separar los niveles. Utilizaremos servicios que funcionen en varias zonas de disponibilidad, como Elastic Load Balancing, Auto Scaling y Amazon RDS Multi-AZ con almacenamiento cifrado mediante AWS Key Management Service. De esta forma, se garantizará la tolerancia a errores en el nivel de recursos y de la zona de disponibilidad.

El balanceador de carga solo enrutará el tráfico a instancias de aplicaciones en buen estado. La comprobación de estado debe realizarse en el plano de datos o capa de aplicación que indica la capacidad de la aplicación en la instancia. Esta comprobación no debe ser contra el plano de control. Habrá una URL de comprobación de estado para la aplicación web y estará configurada que la use el equilibrador de carga y Auto Scaling, de modo que se eliminen y sustituyan las instancias que fallen. Amazon RDS administrará el motor de bases de datos activo para que esté disponible en la segunda zona de disponibilidad si la instancia falla en la AZ principal, y para que repare y restaure con la misma resiliencia.

Después de separar los niveles, podemos usar patrones de resiliencia de sistemas distribuidos para aumentar la fiabilidad de la aplicación, de modo que aún pueda estar disponible incluso cuando la base de datos no esté disponible temporalmente durante una conmutación por error de la zona de disponibilidad.

## Prueba de resiliencia

Llevamos a cabo pruebas funcionales, igual que en el escenario anterior. No comprobaremos las capacidades de autocorrección de ELB, el escalado automático ni la conmutación por error del RDS.

Tendremos guías de estrategias para problemas comunes de la base de datos, incidencias relacionadas con la seguridad y despliegues incorrectos.

## Planificar para la recuperación de desastres (DR)

Existen runbooks para la recuperación total de la carga de trabajo y la creación de informes habituales. La recuperación usa las copias de seguridad almacenadas en la misma región que la carga de trabajo.

## Objetivo de diseño de disponibilidad

Suponemos que al menos algunos errores requerirán una decisión manual para ejecutar la recuperación. Sin embargo, con una mayor automatización en esta situación, suponemos que solo dos eventos por año requerirán esta decisión. Tomamos 30 minutos para decidir ejecutar la recuperación y suponemos que la recuperación se completa en 30 minutos. Esto implica 60 minutos para recuperarse del error. Suponiendo dos incidencias al año, nuestro tiempo de impacto estimado para el año es de 120 minutos.

Esto significa que el límite superior de disponibilidad es del 99,95 %. La disponibilidad real también depende de la tasa real de error, la duración del error y la rapidez con que cada factor se recupera. En esta arquitectura, requerimos que la aplicación esté brevemente fuera de línea para aplicar actualizaciones, pero estas actualizaciones están automatizadas. Para esto, estimamos 150 minutos al año: 15 minutos por cambio, 10 veces al año. Esto hace un total de hasta 270 minutos al año cuando el servicio no está disponible, por lo que nuestro objetivo de diseño de disponibilidad es del 99,9 %.

## Resumen

| Tema | Implementación |
| --- | --- |
| Monitorear los recursos | Comprobación solo del estado del sitio; alertas enviadas cuando está inactivo. |
| Adaptación a los cambios en la demanda | ELB para nivel de aplicación con escalado automático y la web; redimensionamiento del RDS Multi-AZ. |
| Implementar cambios | Implementación automatizada en el lugar y runbook para restauración. |

| Tema | Implementación |
| --- | --- |
| Copia de seguridad de los datos | Copias de seguridad automatizadas a través de RDS para cumplir con RPO y runbook para la restauración. |
| Diseñar para la resiliencia | Escalado automático para proporcionar un nivel de aplicación y web con recuperación automática; el RDS es Multi-AZ. |
| Prueba de resiliencia | ELB y la aplicación se recuperan automátic amente; RDS es Multi-AZ; sin pruebas explícita s. |
| Planificar para la recuperación de desastres (DR) | Copias de seguridad cifradas a través del RDS en la misma región de AWS. |

## Situación de 4 nueves (99,99 %)

Este objetivo de disponibilidad para aplicaciones requiere que la aplicación esté altamente disponible y sea tolerante a errores de componentes. La aplicación debe ser capaz de absorber errores sin necesidad de obtener recursos adicionales. Este objetivo de disponibilidad es para aplicaciones críticas que son los principales o importantes generadores de ingresos para una corporación, como un sitio de comercio electrónico, un servicio web de empresa a empresa o un sitio de contenido o medios de alto tráfico.

Podemos mejorar aún más la disponibilidad mediante el uso de una arquitectura que será estáticamente estable dentro de la región. Este objetivo de disponibilidad no requiere ningún cambio de plano de control en el comportamiento de nuestra carga de trabajo para tolerar errores. Por ejemplo, debe haber suficiente capacidad para soportar la pérdida de una zona de disponibilidad. No deberíamos requerir actualizaciones para el DNS de Amazon Route 53. No deberíamos necesitar crear ninguna infraestructura nueva, ya sea crear o modificar un bucket de S3, crear nuevas políticas de IAM (o modificaciones de políticas) o modificar las configuraciones de tareas de Amazon ECS.

## Monitorear los recursos

El monitoreo incluirá métricas de éxito, así como alertas cuando ocurran problemas. Además, habrá alertas en cada reemplazo de un servidor web incorrecto, cuando la base de datos falle y cuando falle un AZ.

## Adaptación a los cambios en la demanda

Usaremos Amazon Aurora como RDS, lo que permitirá el escalado automático de las réplicas de lectura. En estas aplicaciones, diseñar priorizando la disponibilidad de lectura por encima de la disponibilidad de escritura del contenido principal también es una decisión clave de la arquitectura. Aurora también puede aumentar automáticamente el almacenamiento según sea necesario, en incrementos de 10 GB hasta 64 TB.

## Implementar cambios

Aplicaremos actualizaciones con despliegues de valor controlado o azul-verde en cada zona de aislamiento de forma separada. Las implementaciones están completamente automatizadas, incluida una restauración si los KPI indican un problema.

Los runbooks existirán para requisitos de informes rigurosos y seguimiento del rendimiento. Si las operaciones exitosas tienden a no cumplir con los objetivos de rendimiento o disponibilidad, se utilizará una guía de estrategias para establecer qué causa la tendencia. Las guías de estrategias existirán para modos de error no descubiertos e incidencias de seguridad. También existirán guías de estrategias para establecer la causa raíz de los errores. También nos comprometeremos con AWS Support para ofrecer gestión de eventos de infraestructura.

El equipo que cree y opere el sitio web identificará la corrección de cualquier error inesperado y priorizará la solución que se desplegará después.

## Copia de seguridad de los datos

La copia de seguridad y la restauración se pueden hacer con Amazon RDS. Se ejecutará regularmente con un runbook para garantizar que podamos cumplir con los requisitos de recuperación.

## Diseñar para la resiliencia

Recomendamos tres zonas de disponibilidad para este enfoque. Con un despliegue de tres zonas de disponibilidad, cada AZ tiene una capacidad estática del 50 % de pico. Se podrían utilizar dos zonas

de disponibilidad, pero el coste de la capacidad estáticamente estable sería mayor porque ambas zonas tendrían que tener el 100 % de la capacidad máxima. Agregaremos Amazon CloudFront para proporcionar almacenamiento en caché geográfico, así como la reducción de solicitudes en el plano de datos de nuestra aplicación.

Usaremos Amazon Aurora como RDS y desplegaremos réplicas de lectura en las tres zonas.

La aplicación se creará con los patrones de resiliencia de software o aplicación en todas las capas.

## Prueba de resiliencia

La canalización de despliegue tendrá un conjunto de pruebas completo, que incluye pruebas de inyección de error, carga y rendimiento..

Practicaremos nuestros procedimientos de recuperación de errores constantemente durante los “días de juego”, utilizando runbooks para asegurarnos de que podamos realizar las tareas y no desviarnos de los procedimientos. El equipo que compila el sitio web también lo opera.

## Planificar para la recuperación de desastres (DR)

Existen runbooks para la recuperación total de la carga de trabajo y la creación de informes habituales. La recuperación usa las copias de seguridad almacenadas en la misma región que la carga de trabajo. Los procedimientos de restauración se llevan a cabo de forma habitual como parte de los días de juego.

## Objetivo de diseño de disponibilidad

Suponemos que al menos algunos errores requerirán una decisión manual de ejecutar la recuperación. Sin embargo, con una mayor automatización en esta situación, suponemos que solo dos eventos al año requerirán esta decisión y las acciones de recuperación serán rápidas. Tomamos 10 minutos para decidir ejecutar la recuperación, y suponemos que la recuperación se completa en cinco minutos. Esto implica 15 minutos para recuperarse del error. Suponiendo dos fallos al año, nuestro tiempo de impacto estimado durante el año es de 30 minutos.

Esto significa que el límite superior de disponibilidad es del 99,99 %. La disponibilidad real también dependerá de la tasa real de error, la duración del error y la rapidez con que cada factor se recupera. En esta arquitectura, suponemos que la aplicación está en línea continuamente cuando recibe actualizaciones. En función de esto, nuestro objetivo de diseño de disponibilidad es del 99,99 %.

## Resumen

| Tema | Implementación |
| --- | --- |
| Monitorear los recursos | Comprobaciones de estado en todas las capas y KPI; alertas enviadas cuando se disparan las alarmas configuradas; que alertarán de todos los errores. Las reuniones operativas son rigurosas para detectar tendencias y lograr diseñar objetivos. |
| Adaptación a los cambios en la demanda | ELB para nivel de aplicación de escalado automático y la web; almacenamiento con escalado automático y réplicas de lectura en varias zonas para el RDS de Aurora. |
| Implementar cambios | Implementación automatizada a través del valor controlado o azul-verde y restauración automática cuando los KPI o alertas indican problemas no detectados en la aplicación. Las implementaciones se realizan por zona de aislamiento. |
| Copia de seguridad de los datos | Copias de seguridad automatizadas a través de RDS para cumplir con RPO y restauración automatizada que se practica regularmente en un “día de juego”. |
| Diseñar para la resiliencia | Zonas de aislamiento de errores implement ados para la aplicación; Auto Scaling para proporcionar un nivel de aplicación y web de recuperación automática; RDS es Multi-AZ. |
| Prueba de resiliencia | Las pruebas de errores de componentes y zonas de aislamiento están en proceso y se practican regularmente con el personal operativo en un “día de juego”; existen guías de estrategias para diagnosticar problemas |

| Tema | Implementación |
| --- | --- |
| | desconocidos y existe un proceso de análisis de causa raíz. |
| Planificar para la recuperación de desastres (DR) | Copias de seguridad cifradas a través del RDS en la misma región de AWS que se practica en un día de juego. |

# Situaciones de varias regiones

La implementación de nuestra aplicación en múltiples regiones de AWS aumentará el coste de operación, en parte porque aislamos regiones para mantener su autonomía. Seguir este camino debería ser una decisión muy reflexiva. Dicho esto, las regiones proporcionan un límite de aislamiento sólido y nos esforzamos para evitar errores correlacionados entre regiones. El uso de múltiples regiones le dará un mayor control sobre su tiempo de recuperación en caso de un error de dependencia estricta en un servicio de AWS regional. En esta sección, discutiremos varios patrones de implementación y su disponibilidad típica.

Temas



## Situación de 3 nueves y medio (99,95 %) con un tiempo de recuperación entre 5 y 30 minutos

Este objetivo de disponibilidad para aplicaciones requiere un tiempo de inactividad extremadamente corto y muy poca pérdida de datos durante tiempos específicos. Las aplicaciones con este objetivo de disponibilidad incluyen aplicaciones en las áreas de banca, inversión, servicios de emergencia y captura de datos. Estas aplicaciones tienen tiempos de recuperación y puntos de recuperación muy cortos.

Podemos mejorar aún más el tiempo de recuperación utilizando un enfoque de espera semiactiva en dos regiones de AWS. Desplegaremos la totalidad de la carga de trabajo en ambas regiones, con nuestro sitio pasivo desescalado verticalmente y todos los datos con coherencia posterior. Ambos despliegues serán estáticamente estable en relación con sus regiones respectivas. Las aplicaciones

deben compilarse con los patrones de resiliencia de sistemas distribuidos. Tendremos que crear un componente de enrutamiento ligero que supervise el estado de la carga de trabajo y pueda configurarse para enrutar el tráfico a la región pasiva si fuera necesario.

## Monitorear los recursos

Habrá alertas de cada reemplazo de un servidor web, cuando la base de datos y la región conmuten por error. También supervisaremos el contenido estático en Amazon S3 para verificar la disponibilidad y alertar si no está disponible. El registro se agregará para facilitar la administración y ayudar en el análisis de causa raíz en cada región.

El componente de enrutamiento supervisa tanto el estado de nuestra aplicación como las dependencias estrictas regionales que tengamos.

## Adaptación a los cambios en la demanda

Igual que la situación de 4 nueves.

## Implementar cambios

La entrega del nuevo software se realiza en un horario fijo de cada dos o cuatro semanas. Las actualizaciones de software se automatizarán con patrones de implementación de valor controlado o azul-verde.

Los runbooks existen para cuando se produce la conmutación por error de región, para problemas comunes de los clientes que ocurren durante esos eventos y para informes comunes.

Tendremos guías de estrategias para problemas comunes de la base de datos, incidencias relacionadas con la seguridad, implementaciones incorrectas, problemas inesperados de los clientes en la conmutación por error de la región y para establecer la causa raíz de los problemas. Una vez identificada la causa raíz, la corrección del error se identificará mediante una combinación de los equipos de operaciones y desarrollo y se implementará cuando se desarrolle la solución.

También tratamos con AWS Support para la gestión de eventos de infraestructura.

## Copia de seguridad de los datos

Al igual que en la situación de 4 nueves, usamos copias de seguridad del RDS automáticas y usamos el control de versiones de S3. Los datos se replican de forma automática y asíncrona desde el clúster del RDS de Aurora en la región activa a réplicas de lectura entre regiones en la región

pasiva. La replicación de S3 entre regiones se usa para trasladar datos de forma automática y asíncrona desde la región activa hasta la pasiva.

## Diseñar para la resiliencia

Igual que en la situación de 4 nueves, con posibilidad de conmutación por error regional. Esto se administra manualmente. Durante la conmutación por error, enrutaremos las solicitudes a un sitio web estático mediante la conmutación por error a nivel de DNS hasta la recuperación en la segunda región.

## Prueba de resiliencia

Igual que en la situación de 4 nueves, y además validaremos la arquitectura a través de los días de juego con runbooks. Asimismo, la corrección RCA se prioriza por encima de los lanzamientos de funciones para su despliegue inmediato.

## Planificar para la recuperación de desastres (DR)

La conmutación por error regional se administra manualmente. Todos los datos se replican de forma asíncrona. La infraestructura en la espera semiactiva se escala horizontalmente. Esto se puede automatizar mediante un flujo de trabajo ejecutado en AWS Step Functions. AWS Systems Manager (SSM) también puede ayudar con esta automatización, ya que puede crear documentos de SSM que actualicen los grupos de escalado automático y redimensione las instancias.

## Objetivo de diseño de disponibilidad

Asumimos que al menos algunos errores requerirán una decisión manual de ejecutar la recuperación. Sin embargo, con una buena automatización en esta situación, suponemos que solo dos eventos al año requerirán esta decisión. Tomamos 20 minutos para decidir ejecutar la recuperación y suponemos que la recuperación se completa en 10 minutos. Esto implica que se tarda unos 30 minutos en recuperarse de un error. Suponiendo dos fallos al año, nuestro tiempo de impacto estimado durante el año es de 60 minutos.

Esto significa que el límite superior de disponibilidad es del 99,95 %. La disponibilidad real también dependerá de la tasa real de error, la duración del error y la rapidez con que cada factor se recupera. En esta arquitectura, suponemos que la aplicación está en línea continuamente cuando recibe actualizaciones. En función de esto, nuestro objetivo de diseño de disponibilidad es del 99,95 %.

### Resumen

| Tema | Implementación |
| --- | --- |
| Monitorear los recursos | Comprobaciones de estado en todas las capas, incluido el estado de DNS en el nivel de región de AWS y KPI; se envían alertas cuando se disparan las alarmas configura das, que alertarán de todos los errores. Las reuniones operativas son rigurosas para detectar tendencias y lograr diseñar objetivos. |
| Adaptación a los cambios en la demanda | ELB para nivel de aplicación con escalado automático y la web; almacenamiento con escalado automático y réplicas de lectura en varias zonas de las regiones activa y pasiva del RDS de Aurora. Datos e infraestructura sincronizados entre regiones de AWS para estabilidad estática. |
| Implementar cambios | Despliegue automatizado a través de valores controlados azul-verde y restauración automática cuando los KPI o alertas indican problemas no detectados en la aplicación; los despliegues se realizan en una zona de aislamiento en una región de AWS a la vez. |
| Copia de seguridad de los datos | Copias de seguridad automatizadas en cada región de AWS a través del RDS para cumplir con el RPO y la restauración automatizada que se practica regularmente en un día de juego. Los datos del RDS de Aurora y de S3 se replican de forma automática y asíncrona desde la región activa hasta la pasiva. |
| Diseñar para la resiliencia | Escalado automático para proporcionar un nivel de aplicación y web con recuperación automática; el RDS es Multi-AZ; la conmutaci ón por error regional se administra manualmen |

| Tema | Implementación |
| --- | --- |
| | te con el sitio estático presentado durante la conmutación por error. |
| Prueba de resiliencia | Las pruebas de errores de componentes y zonas de aislamiento están en proceso y se practican regularmente con el personal operativo en un “día de juego”; existen guías de estrategias para diagnosticar problemas desconocidos y existe un proceso de análisis de causa raíz, con rutas de comunicación sobre cuál era el problema y cómo se corrigió o evitó. La corrección RCA se prioriza por encima de los lanzamientos de funciones para su implementación inmediata. |
| Planificar para la recuperación de desastres (DR) | Espera semiactiva desplegada en otra región. La infraestructura se escala horizontalmente utilizando flujos de trabajo ejecutados con AWS Step Functions o documentos de AWS Systems Manager. Copias de seguridad cifradas a través del RDS. Réplicas de lectura entre dos regiones de AWS. Replicación entre regiones de activos estáticos en Amazon S3. La restauración se realiza en la región de AWS activa actual, se practica en un día de juego y se coordina con AWS. |

## Situación de 5 nueves (99,999 %) o superior con un tiempo de recuperación inferior a un minuto

Este objetivo de disponibilidad para las aplicaciones casi no requiere tiempo de inactividad o pérdida de datos en momentos específicos. Las aplicaciones que podrían tener este objetivo de disponibilidad incluyen, por ejemplo, ciertas aplicaciones comerciales bancarias, de inversión, financieras, gubernamentales y críticas que son el negocio principal de un negocio que genera

ingresos extremadamente grandes. Lo deseable es tener almacenes de datos fuertemente consistentes y una completa redundancia en todas las capas. Seleccionamos un almacén de datos basado en SQL. Sin embargo, en algunas situaciones, nos resultará difícil lograr un RPO muy pequeño. Si puede particionar sus datos, es posible que no haya pérdida de datos. Esto podría requerir incorporar latencia y lógica de aplicación para asegurarse de tener datos consistentes entre ubicaciones geográficas, así como la capacidad de mover o copiar datos entre particiones. Realizar esta partición podría ser más fácil si usa una base de datos NoSQL.

Podemos mejorar aún más la disponibilidad mediante el uso de una estrategia activa-activa entre varias regiones de AWS. La carga de trabajo se desplegará en todas las regiones deseadas que sean estáticamente estable entre regiones (de modo que las regiones restantes puedan afrontar la carga con la pérdida de una región). A enrutamiento dirige el tráfico a ubicaciones geográficas que estén en buen estado y cambia automáticamente el destino cuando el estado de una ubicación no sea correcto. Además, detiene temporalmente las capas de replicación de datos. Amazon Route 53 ofrece comprobaciones de estado en intervalos de 10 segundos y también ofrece TTL en sus conjuntos de registros de hasta un segundo tan solo.

### Monitorear los recursos

Igual que la situación de 3 nueves y medio, y además alerta cuando se detecta una región en mal estado y se desvía el tráfico de ella.

### Adaptación a los cambios en la demanda

Igual que la situación de 3 nueves y medio.

### Implementar cambios

La canalización de implementación tendrá un conjunto de pruebas completo, que incluye pruebas de inyección de error, carga y rendimiento. Implementaremos actualizaciones mediante implementaciones de valor controlado o azul-verde en una zona de aislamiento a la vez, en una región antes de comenzar en la otra. Durante el despliegue, las versiones antiguas se seguirán ejecutando en instancias para facilitar una restauración más rápida. Están completamente automatizados y se incluye una restauración si los KPI indican un problema. El monitoreo incluirá métricas de éxito, así como alertas cuando ocurran problemas.

Los runbooks existirán para requisitos de informes rigurosos y seguimiento del rendimiento. Si las operaciones correctas tienden a no cumplir con los objetivos de rendimiento o disponibilidad, se utilizará una guía de estrategias para establecer qué causa la tendencia. Las guías de estrategias

existirán para modos de error no descubiertos e incidencias de seguridad. También existirán guías de estrategias para establecer la causa raíz de los errores.

El equipo que compila el sitio web también lo opera. Ese equipo identificará la corrección de cualquier error inesperado y priorizará la solución que se desplegará después. También tratamos con AWS Support para la gestión de eventos de infraestructura.

### Copia de seguridad de los datos

Igual que la situación de 3 nueves y medio.

### Diseñar para la resiliencia

Las aplicaciones deben compilarse con los patrones de resiliencia de software o aplicación. Es posible que se requieran muchas otras capas de enrutamiento para implementar la disponibilidad necesaria. La complejidad de esta implementación adicional no debe subestimarse. La aplicación se implementará en zonas de aislamiento de errores de implementación y se dividirá e implementará para que incluso un evento de toda la región no afecte a todos los clientes.

### Prueba de resiliencia

Validaremos la arquitectura a través de los “días de juego” con runbooks para asegurarnos de que podemos realizar las tareas y no nos desviamos de los procedimientos.

### Planificar para la recuperación de desastres (DR)

Despliegue multirregional activo-activo con infraestructura de carga de trabajo completa y datos en múltiples regiones. Se utiliza una estrategia de lectura local y escritura global, una región es la base de datos principal para todas las escrituras y los datos se replican para la lectura a otras regiones. Si la región de la base de datos principal falla, tendrá que promocionarse una nueva base de datos. El método de lectura local y escritura global tiene usuarios asignados a una región inicial donde se gestionan las escrituras en la base de datos. Esto permite que los usuarios lean o escriban desde cualquier región, pero requiere una lógica compleja para administrar los conflictos de datos potenciales entre escrituras en distintas regiones.

Cuando se detecta que una región está en mal estado, la capa de enrutamiento dirige el tráfico automáticamente a las regiones en buen estado restantes. No se requiere ninguna intervención manual.

Los almacenes de datos se deben replicar entre las regiones para poder resolver posibles conflictos. Será necesario crear herramientas y procesos automatizados para copiar o mover datos entre las

particiones por razones de latencia y para equilibrar las solicitudes o cantidades de datos en cada partición. La solución de la resolución de conflictos de datos también requerirá runbooks operativos adicionales.

## Objetivo de diseño de disponibilidad

Suponemos que se realizan grandes inversiones para automatizar toda la recuperación y que la recuperación se puede completar en un minuto. Suponemos que no hay recuperaciones desencadenadas manualmente, sino hasta una acción de recuperación automatizada por trimestre. Esto implica cuatro minutos al año para recuperarse. Asumimos que la aplicación está en línea continuamente a través de actualizaciones. En función de esto, nuestro objetivo de diseño de disponibilidad es del 99,999 %.

Resumen

| Tema | Implementación |
| --- | --- |
| Monitorear los recursos | Comprobaciones de estado en todas las capas, incluido el estado de DNS en el nivel de región de AWS y KPI; se envían alertas cuando se disparan las alarmas configura das, que alertarán de todos los errores. Las reuniones operativas son rigurosas para detectar tendencias y lograr diseñar objetivos. |
| Adaptación a los cambios en la demanda | ELB para nivel de aplicación con escalado automático y la web; almacenamiento con escalado automático y réplicas de lectura en varias zonas de las regiones activas y pasivas del RDS de Aurora. Datos e infraestructura sincronizados entre regiones de AWS para estabilidad estática. |
| Implementar cambios | Despliegue automatizado a través de valores controlados azul-verde y restauración automática cuando los KPI o alertas indican problemas no detectados en la aplicación; |

| Tema | Implementación |
| --- | --- |
| | los despliegues se realizan en una zona de aislamiento en una región de AWS a la vez. |
| Copia de seguridad de los datos | Copias de seguridad automatizadas en cada región de AWS a través del RDS para cumplir con el RPO y la restauración automatizada que se practica regularmente en un día de juego. Los datos del RDS de Aurora y de S3 se replican de forma automática y asíncrona desde la región activa hasta la pasiva. |
| Diseñar para la resiliencia | Zonas de aislamiento de errores implement ados para la aplicación; Auto Scaling para proporcionar un nivel de aplicación y web de recuperación automática; RDS es Multi-AZ; conmutación por error regional automatizada. |
| Prueba de resiliencia | Las pruebas de errores de componentes y zonas de aislamiento están en proceso y se practican regularmente con el personal operativo en un “día de juego”; existen guías de estrategias para diagnosticar problemas desconocidos y existe un proceso de análisis de causa raíz con rutas de comunicación sobre cuál era el problema y cómo se corrigió o evitó. La corrección RCA se prioriza por encima de los lanzamientos de funciones para su implementación inmediata. |

| Tema | Implementación |
| --- | --- |
| Planificar para la recuperación de desastres (DR) | Despliegue activo-activo en al menos dos regiones. Infraestructura a plena escala y con estabilidad estática entre regiones. Datos particionados y sincronizados entre regiones. Copias de seguridad cifradas a través del RDS. Fallos de regiones practicados mediante días de juego y coordinados con AWS. Durante la restauración, es posible que tenga que promocionarse una nueva base de datos principal. |

# Recursos

## Documentación

- Amazon Builders' Library : cómo desarrolla y opera Amazon el software
- Centro de arquitectura de AWS

## Laboratorios

- Laboratorios de fiabilidad de AWS Well-Architected

## Enlaces externos

- Patrón de cola adaptativa: errores a escala
- Disponibilidad y más allá: comprender y mejorar la resiliencia de sistemas distribuidos en AWS

## Libros

- Robert S. Hammer «Patrones para software tolerante a errores»
- Andrew Tanenbaum y Marten van Steen «Sistemas distribuidos: principios y paradigmas»

# Conclusión

Tanto si es nuevo en los temas de disponibilidad y fiabilidad como si es un veterano que busca información para maximizar la disponibilidad de su carga de trabajo esencial, esperamos que este documento técnico le haya hecho reflexionar, le haya ofrecido una nueva idea o haya abierto una nueva línea de preguntas. Esperamos que esto lleve a una comprensión más profunda del nivel adecuado de disponibilidad en función de las necesidades de su empresa y de cómo diseñar la fiabilidad para conseguirlo. Le recomendamos que aproveche las recomendaciones de diseño, operativas y orientadas a la recuperación que se ofrecen aquí, así como el conocimiento y la experiencia de nuestros arquitectos de soluciones de AWS. Nos encantaría conocer su opinión, sobre todo sus historias de éxito que logran altos niveles de disponibilidad en AWS. Contacte con su equipo de cuentas o utilice Contacte con nosotros en nuestro sitio web.

# Colaboradores

Entre los colaboradores de este documento, están las siguientes personas:

- Seth Eliot, Principal Developer Advocate, Amazon Web Services
- Mahanth Jayadeva, Solutions Architect, Well-Architected, Amazon Web Services
- Amulya Sharma, Principal Solutions Architect, Amazon Web Services
- Jason DiDomenico, Senior Solutions Architect, Cloud Foundations, Amazon Web Services
- Marcin Bednarz, Principal Solutions Architect, Amazon Web Services
- Tyler Applebaum, Senior Solutions Architect, Amazon Web Services
- Rodney Lester, Principal Solutions Architect, App Modernization, Amazon Web Services
- Joe Chapman, Senior Solutions Architect, Amazon Web Services
- Adrian Hornsby, Principal System Development Engineer, Amazon Web Services
- Kevin Miller, Vice President, S3, Amazon Web Services
- Shannon Richards, Principal Technical Program Manager, Amazon Web Services
- Laurent Domb, Chief Technologist, Fed Fin, Amazon Web Services
- Kevin Schwarz, Sr. Solutions Architect, Amazon Web Services
- Rob Martell, Principal Cloud Resilience Architect, Amazon Web Services
- Priyam Reddy, Senior Solutions Architect Manager DR, Amazon Web Services
- Jeff Ferris, Principal Technologist, Amazon Web Services
- Matias Battaglia, Senior Solutions Architect, Amazon Web Services

# Otra documentación

Para obtener información adicional, consulte la siguiente documentación:

- AWS Well-Architected Framework
- Centro de arquitectura de AWS

# Revisiones del documento

Para recibir notificaciones sobre las actualizaciones de este documento técnico, suscríbase al canal RSS.

| Cambio | Descripción | Fecha |
| --- | --- | --- |
| Actualizaciones de la guía sobre las prácticas recomendadas | Actualizaciones menores en todas las prácticas recomendadas. | June 27, 2024 |
| Actualizaciones de la guía sobre las prácticas recomendadas | Las prácticas recomendadas se han actualizado con nuevas directrices en las siguientes áreas: «Diseñe interacciones en un sistema distribuido para evitar errores», «Diseñar las interacciones en un sistema distribuido para mitigar o tolerar los errores», «Supervisar los recursos de la carga de trabajo», «Diseñar su carga de trabajo para que se adapte a los cambios en la demanda», «Implementar cambios» y «Comprobar la fiabilidad». | December 6, 2023 |
| Actualizaciones de la guía sobre las prácticas recomendadas | Las prácticas recomendadas se han actualizado con nuevas directrices en las siguientes áreas: «Supervisar los recursos de la carga de trabajo» y «Diseñar su carga de trabajo para que soporte los errores de los componentes». | October 3, 2023 |

| | | |
|---|---|---|
| Actualizaciones de la guía sobre las prácticas recomendadas | Las prácticas recomendadas se han actualizado con nuevas directrices en las siguientes áreas: «Diseñar la arquitectura de servicio de su carga de trabajo», «Diseñar las interacciones en un sistema distribuido para mitigar o tolerar los errores» y «Supervisar los recursos de la carga de trabajo». | July 13, 2023 |
| Actualización menor | Eliminación del lenguaje no inclusivo. | April 13, 2023 |
| Actualizaciones del nuevo marco | Prácticas recomendadas actualizadas con guía prescriptiva y prácticas recomendadas añadidas. | April 10, 2023 |
| Documento técnico actualizado | Prácticas recomendadas actualizadas con nueva guía de implementación. | December 15, 2022 |
| Actualizaciones menores | Corrección de cifras y cambios menores en todo el documento. | November 17, 2022 |
| Documento técnico actualizado | Se han ampliado las prácticas recomendadas y se han añadido planes de mejora. | October 20, 2022 |

| | | |
|---|---|---|
| Documento técnico actualizado | Se han añadido dos nuevas prácticas recomendadas al pilar de fiabilidad en las secciones Usar el aislamiento de errores para proteger su carga de trabajo y Diseñar su carga de trabajo para que soporte los errores de los componentes. | May 5, 2022 |
| Actualización menor | Se ha añadido Pilar de sostenibilidad a la introducción. | December 2, 2021 |
| Documento técnico actualizado | Se ha actualizado la guía de recuperación de desastres para incluir el Controlador de recuperación de aplicaciones Route 53. Se han añadido referencias a DevOps Guru. Se han actualizado varios enlaces de recursos y se han hecho otros cambios editoriales menores. | October 26, 2021 |
| Actualización menor | Se ha añadido información sobre AWS Fault Injection Service (AWS FIS). | March 15, 2021 |
| Actualización menor | Actualización de texto menor. | January 4, 2021 |

| | | |
|---|---|---|
| Documento técnico actualizado | Actualización del apéndice A en relación con el objetivo de diseño de la disponibilidad para Amazon SQS, Amazon SNS y Amazon MQ; reordenación de las filas de la tabla para facilitar la búsqueda; mejora de la explicación de las diferencias entre la disponibilidad y la recuperación de desastres y de cómo ambas contribuyen a la resiliencia; ampliación de la cobertura de las arquitecturas multirregión (para la disponibilidad) y de las estrategias multirregión (para la recuperación de desastres); actualización del libro de referencia a la última versión; ampliación de los cálculos de la disponibilidad para incluir el cálculo basado en las solicitudes y los cálculos abreviados; mejora de la descripción de los días de juego. | December 7, 2020 |
| Actualización menor | Se ha actualizado el Apéndice A para modificar el objetivo de diseño de disponibilidad para AWS Lambda | October 27, 2020 |
| Actualización menor | Se ha actualizado el Apéndice A para añadir el objetivo de diseño de disponibilidad para AWS Global Accelerator | July 24, 2020 |

| | | |
|---|---|---|
| Actualizaciones del nuevo marco | Actualizaciones sustancia les y contenido nuevo o revisado, incluido lo siguiente : se ha agregado la sección de prácticas recomendadas «Arquitectura de la carga de trabajo»; se han reorganizado las prácticas recomendadas en las secciones Administr ación de cambios y Administr ación de errores; se han actualizado los recursos para incluir los recursos y servicios de AWS más recientes, como AWS Global Accelerator, AWS Service Quotas y AWS Transit Gateway; se han agregado o actualizado las definiciones de fiabilidad, disponibilidad y resiliencia; se ha alineado mejor el documento técnico con AWS Well-Architected Tool (preguntas y prácticas recomendadas) que se usa para las revisiones de Well-Architected; se han reordenad o los principios de diseño; se ha trasladado Recuperación automática de errores antes de Prueba de los procedimi entos de recuperación; se han actualizado los diagramas y los formatos de las ecuacione s; se han eliminado las secciones Servicios clave y, en su lugar, se han integrado | July 8, 2020 |

| | | |
|---|---|---|
| | las referencias a los servicios de AWS clave en las prácticas recomendadas. | |
| Actualización menor | Se ha corregido el enlace que no funciona | October 1, 2019 |
| Documento técnico actualizado | Se ha actualizado el Apéndice A | April 1, 2019 |
| Documento técnico actualizado | Se han añadido recomendaciones específicas de redes de AWS Direct Connect y objetivos de diseño de servicio adicionales | September 1, 2018 |
| Documento técnico actualizado | Se agregaron las secciones «Principios de diseño» y «Administración de límites». Se actualizaron los enlaces, se eliminó la ambigüedad de la terminología ascendente/descendente y se agregaron referencias explícitas a los temas restantes del Pilar de fiabilidad en las situaciones de disponibilidad. | June 1, 2018 |
| Documento técnico actualizado | Se ha cambiado la solución entre regiones de DynamoDB por las tablas globales de DynamoDB. Se han añadido objetivos de diseño de servicios | March 1, 2018 |
| Actualizaciones menores | Corrección menor al cálculo de disponibilidad para incluir la disponibilidad de la aplicación | December 1, 2017 |

| | | |
|---|---|---|
| Documento técnico actualizado | Se ha actualizado para proporcionar orientación sobre diseños de alta disponibilidad, como conceptos, prácticas recomendadas e implement aciones de ejemplo. | November 1, 2017 |
| Publicación inicial | Publicación de Pilar de fiabilidad: AWS Well-Arch itected Framework. | November 1, 2016 |