Detección y mitigación de errores grises

Patrones de resiliencia de Multi-AZ avanzados

Patrones de resiliencia de Multi-AZ avanzados: Detección y mitigación de errores grises

Table of Contents

Patrones de resiliencia de Multi-AZ avanzados

Fecha de publicación: 11 de julio de 2023 (Revisiones del documento)

Muchos clientes ejecutan sus cargas de trabajo en configuraciones Multi-AZ de alta disponibilidad. Estas arquitecturas funcionan bien durante los eventos de error binarios, pero suelen tener problemas con los errores grises. Las manifestaciones de este tipo de error pueden ser sutiles y no ser detectadas de forma rápida y definitiva. En este documento técnico, se proporciona información sobre cómo instrumentar las cargas de trabajo para detectar el impacto de los errores grises que se aíslan en una misma zona de disponibilidad y, a continuación, cómo tomar medidas para mitigar ese impacto en la zona de disponibilidad.

Introducción

El objetivo de este documento es ayudarle a implementar de manera más eficaz arquitecturas Multi-AZ resilientes. Una de las prácticas recomendadas para crear sistemas resilientes en las redes de Amazon Virtual Private Cloud (VPC) es implementar cada carga de trabajo en varias zonas de disponibilidad.

Una Zona de disponibilidad consiste en uno o varios centros de datos discretos con alimentación, redes y conectividad redundantes. El uso de varias zonas de disponibilidad permite operar cargas de trabajo de mayor disponibilidad, tolerancia a errores y escalabilidad de lo que sería posible desde un único centro de datos.

Muchos servicios de AWS, como Amazon Elastic Compute Cloud (EC2) Auto Scaling o Amazon Relational Database Service (Amazon RDS), ofrecen una configuración Multi-AZ. Estos servicios no requieren que cree ninguna herramienta adicional de observabilidad o conmutación por error. Permiten que las cargas de trabajo sean resilientes a los modos de error binarios fácilmente detectables dentro de una Región de AWS que afectan a una única zona de disponibilidad. Puede ser un error físico total de hardware, una pérdida de alimentación o un error de software latente que afecte a la mayoría de los recursos.

Sin embargo, existe otra categoría de errores, denominados errores grises, cuyas manifestaciones son sutiles y no se detectan de forma rápida y definitiva. Como resultado, se tarda más en mitigar el impacto causado por el error. Este documento se centra en los impactos que los errores grises pueden tener en las arquitecturas Multi-AZ, cómo detectarlos y, por último, cómo mitigarlos.

Las recomendaciones que se proporcionan en este documento técnico se aplican principalmente a clases específicas de cargas de trabajo:

- Que utilizan principalmente servicios zonales de AWS
- Que deben mejorar la resiliencia de una sola región
- Para las que se puede realizar una inversión significativa para crear los patrones de observabilidad y resiliencia necesarios

En estas cargas de trabajo, es posible que no esté dispuesto a hacer algunas o todas las concesiones que se describen en ???, o que no tenga la opción de usar varias regiones. Es probable que estos tipos de cargas de trabajo representen un pequeño subconjunto de su cartera general y, por lo tanto, estas recomendaciones deben considerarse a nivel de carga de trabajo y no a nivel de plataforma.

Errores grises

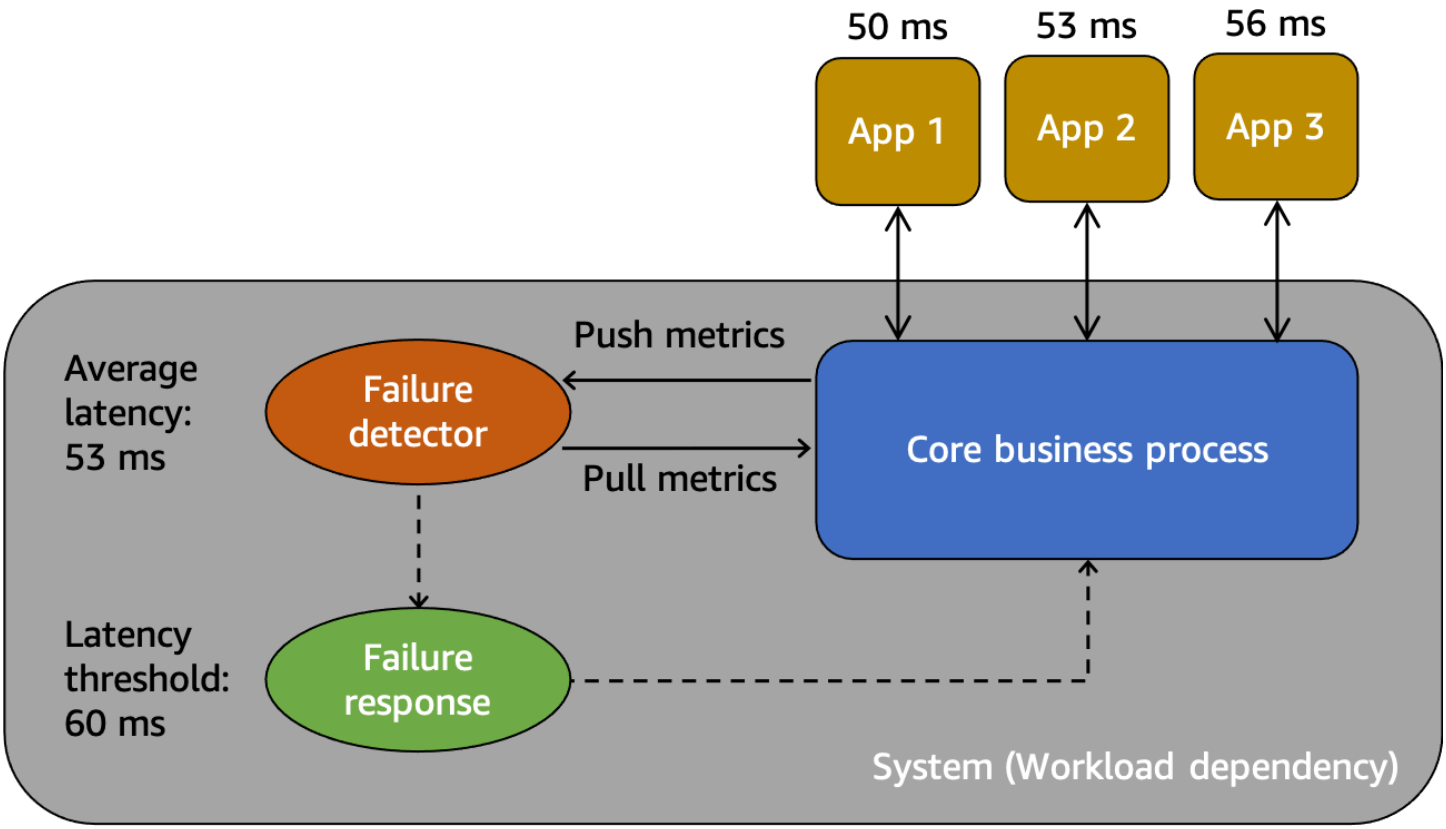
Los errores grises se definen por la característica de observabilidad diferencial, lo que significa que diferentes entidades observan el error de manera diferente. Definamos lo que esto significa.

Observabilidad diferencial

Las cargas de trabajo que utiliza suelen tener dependencias. Por ejemplo, estas pueden ser los servicios de nube de AWS que se utilizan para crear la carga de trabajo o un proveedor de identidades (IdP) externo que se utiliza para la federación. Esas dependencias casi siempre implementan su propia observabilidad, y registran métricas sobre los errores, la disponibilidad y la latencia, entre otras cosas, que se generan por el uso de los clientes. Cuando se supera un umbral para alguna de estas métricas, la dependencia suele realizar alguna acción para corregirlo.

Estas dependencias suelen tener varios consumidores de sus servicios. Los consumidores también implementan su propia observabilidad y registran métricas y registros sobre sus interacciones con las dependencias, por ejemplo, cuánta latencia hay en las lecturas del disco, cuántas solicitudes de API han fallado o cuánto ha tardado una consulta de base de datos.

Estas interacciones y mediciones se muestran en un modelo abstracto en la siguiente figura.

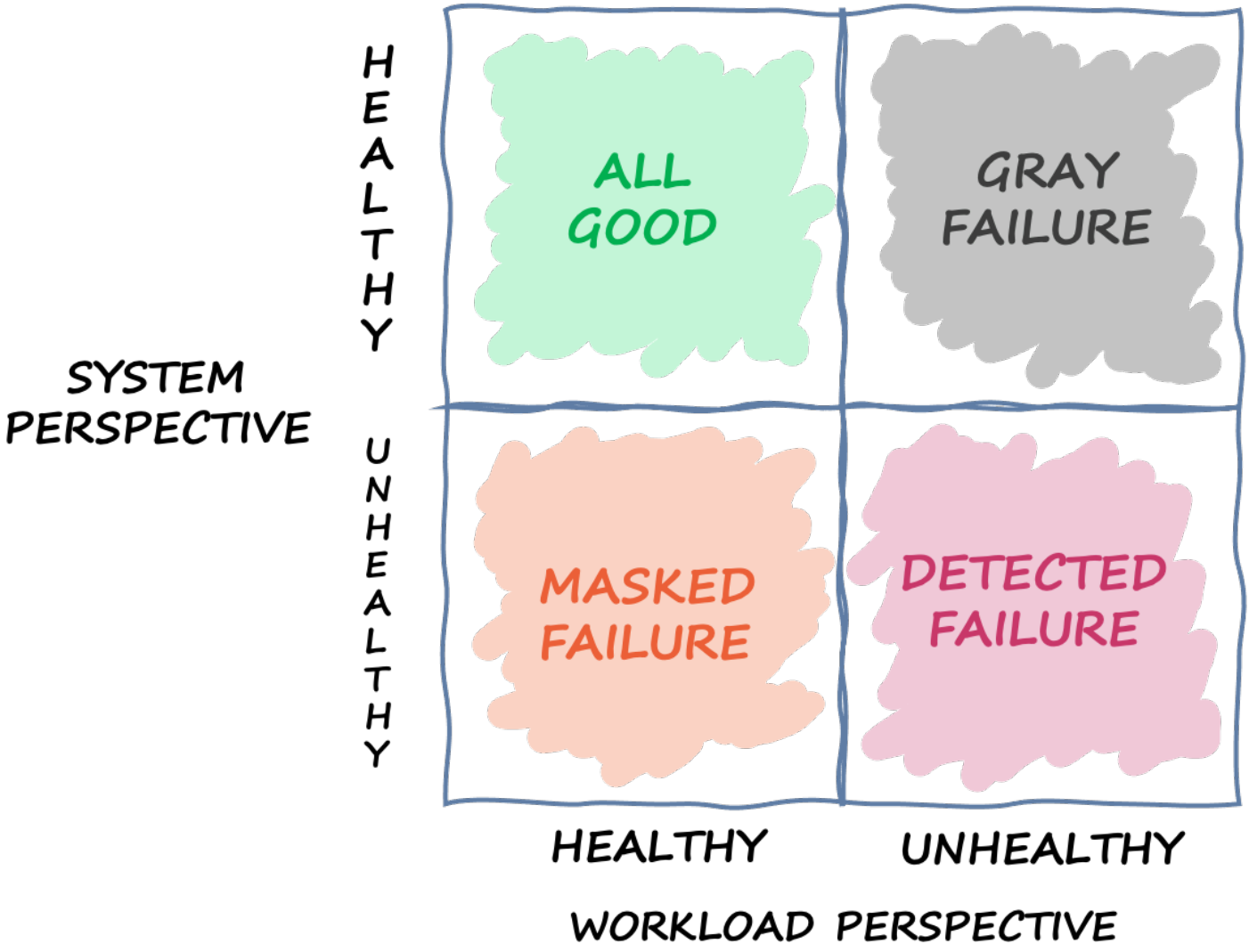


Un modelo abstracto para entender los errores grises

En primer lugar, tenemos el sistema, que es una dependencia de los consumidores Aplicación 1, Aplicación 2 y Aplicación 3 en este escenario. El sistema tiene un detector de errores que examina las métricas creadas a partir del proceso empresarial principal. También cuenta con un mecanismo de respuesta a los errores para mitigar o corregir los problemas observados por el detector de errores. El sistema detecta una latencia media general de 53 ms y ha establecido un umbral para invocar el mecanismo de respuesta a errores cuando la latencia media supera los 60 ms. La Aplicación 1, la Aplicación 2 y la Aplicación 3 también están realizando sus propias observaciones sobre su interacción con el sistema, y registran una latencia media de 50 ms, 53 ms y 56 ms, respectivamente.

La observabilidad diferencial es la situación en la que uno de los usuarios del sistema detecta que el sistema no está funcionando correctamente, pero la propia supervisión del sistema no detecta el problema o el impacto no supera el umbral de alarma. Supongamos que la Aplicación 1 comienza a experimentar una latencia media de 70 ms en lugar de 50 ms. La Aplicación 2 y la Aplicación 3 no ven ningún cambio en sus latencias medias. Esto aumenta la latencia media del sistema subyacente a 59,66 ms, pero no supera el umbral de latencia para activar el mecanismo de respuesta a errores. Sin embargo, la latencia de la Aplicación 1 aumenta un 40 %. Esto podría afectar a su disponibilidad

al superar el tiempo de espera del cliente configurado para la Aplicación 1, o puede provocar impactos en cascada en una cadena de interacciones más larga. Desde el punto de vista de la Aplicación 1, el sistema subyacente del que depende no funciona correctamente, pero desde el punto de vista del propio sistema, así como de la Aplicación 2 y la Aplicación 3, el sistema funciona correctamente. En la siguiente figura, se resumen estas diferentes perspectivas.



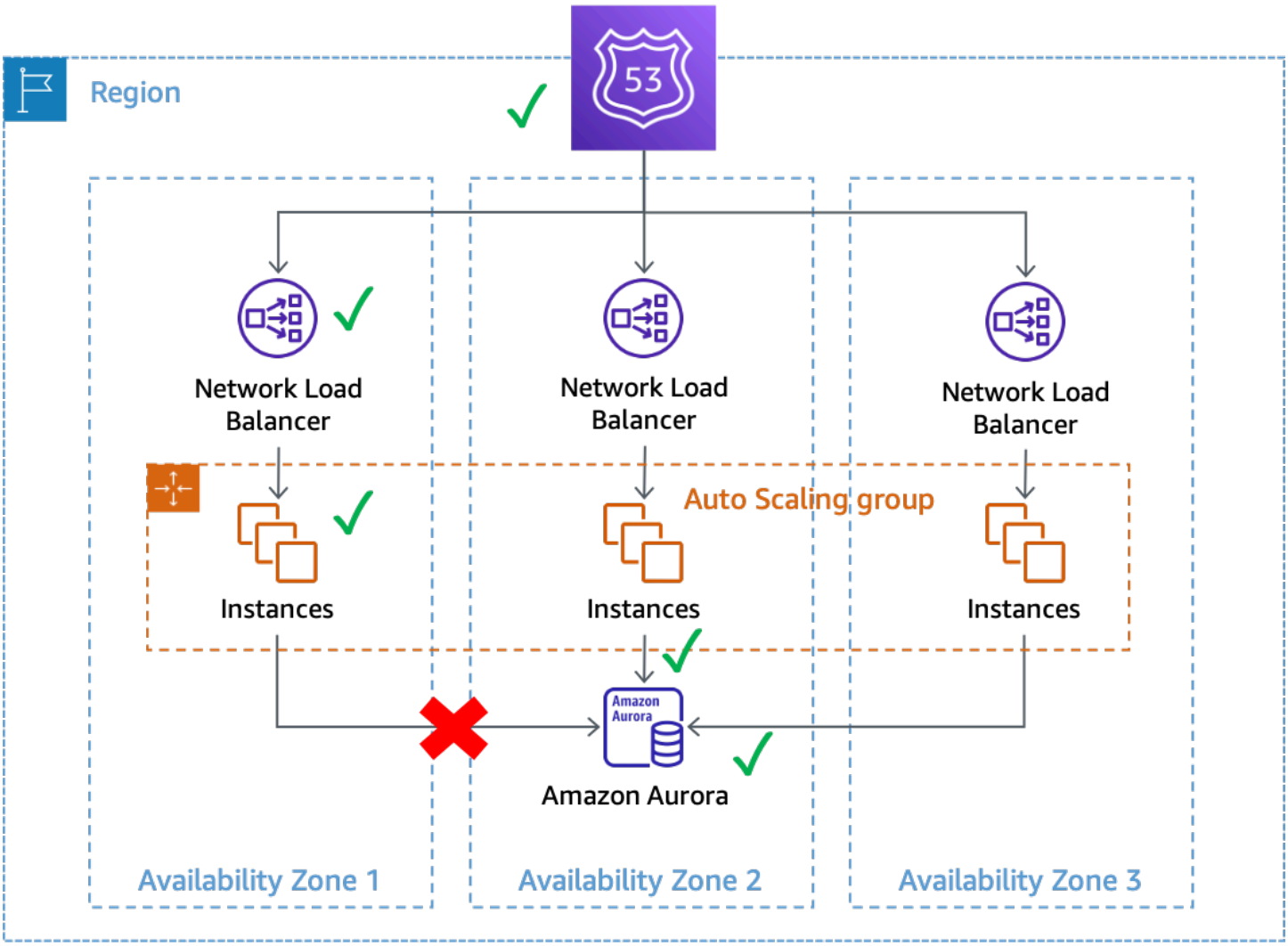
Un cuadrante que define los diferentes estados en los que puede encontrarse un sistema en función de diferentes perspectivas

Un error también puede atravesar este cuadrante. Un evento puede comenzar como un error gris y convertirse después en un error detectado, cambiar a un error enmascarado y, quizás, volver a un error gris. No hay un ciclo definido y casi siempre existe la posibilidad de que se repita el error hasta que se aborde su causa raíz.

La conclusión que sacamos es que las cargas de trabajo no siempre pueden confiar en el sistema subyacente para detectar y mitigar el error. Por muy sofisticado y resiliente que sea el sistema subyacente, siempre existe la posibilidad de que un error pase desapercibido o se quede por debajo del umbral de reacción. Los usuarios de ese sistema, como la Aplicación 1, deben estar preparados para detectar y mitigar rápidamente el impacto que provoca un error gris. Esto requiere crear mecanismos de observabilidad y recuperación para estas situaciones.

Ejemplo de errores grises

Los errores grises pueden afectar a los sistemas Multi-AZ en AWS. Por ejemplo, supongamos una flota de instancias de Amazon EC2 en un grupo de escalado automático implementado en tres zonas de disponibilidad. Todos se conectan a una base de datos de Amazon Aurora en una zona de disponibilidad. A continuación, se produce un error gris que afecta a las redes entre la Zona de disponibilidad 1 y la Zona de disponibilidad 2. El resultado de este deterioro es que un porcentaje de las conexiones de bases de datos nuevas y existentes desde las instancias de la Zona de disponibilidad 1 fallan. Esta situación se muestra en la siguiente figura.



Un error gris que afecta a las conexiones de bases de datos desde las instancias de la Zona de disponibilidad 1

En este ejemplo, Amazon EC2 considera que las instancias de la Zona de disponibilidad 1 tienen un estado correcto porque siguen pasando las comprobaciones de estado del sistema y de las instancias. Amazon EC2 Auto Scaling tampoco detecta un impacto directo en ninguna zona de disponibilidad y continúa lanzando capacidad en las zonas de disponibilidad configuradas. El Equilibrador de carga de red (NLB) también considera que las instancias subyacentes tienen un estado correcto, al igual que las comprobaciones de estado de Route 53 que se realizan en el punto de conexión de NLB. Del mismo modo, Amazon Relational Database Service (Amazon RDS) considera que el clúster de base de datos tiene un estado correcto y no desencadena una conmutación por error automática. Tenemos muchos servicios diferentes y todos consideran que sus servicios y recursos están en buen estado, pero la carga de trabajo detecta un error que afecta a su disponibilidad. Se trata de un error gris.

Respuesta a los errores grises

Cuando se produce un error gris en su entorno de AWS, por lo general tiene tres opciones:

- No haga nada y espere a que el deterioro desaparezca.
- Si el deterioro está aislado en una sola zona de disponibilidad, evacue esa zona de disponibilidad.
- Realice una conmutación por error a otra Región de AWS y aproveche las ventajas del aislamiento regional de AWS para mitigar el impacto.

Muchos clientes de AWS están de acuerdo con la primera opción para la mayoría de sus cargas de trabajo. Aceptan tener un objetivo de tiempo de recuperación (RTO) posiblemente ampliado con la compensación de no haber tenido que desarrollar soluciones adicionales de observabilidad o resiliencia. Otros clientes optan por implementar la tercera opción, la Recuperación de desastres (DR) multirregional, como plan de mitigación para varios modos de error. Las arquitecturas multirregionales pueden funcionar bien en estos escenarios. Sin embargo, hay algunas contrapartidas cuando se utiliza este enfoque (consulte los Aspectos básicos de la multirregión de AWS para ver un análisis completo sobre las consideraciones de la multirregión).

En primer lugar, crear y operar una arquitectura multirregional puede ser una tarea difícil, compleja y potencialmente costosa. Las arquitecturas multirregionales requieren una cuidadosa consideración de la estrategia de recuperación de desastres que se elija. Puede que no sea viable desde el punto de vista fiscal implementar una solución de DR activa-activa multirregional solo para manejar deterioros zonales, mientras que una estrategia de respaldo y restauración puede que no cumpla sus requisitos de resiliencia. Además, las conmutaciones por error multirregionales deben practicarse de forma continua en la producción para tener la seguridad de que funcionarán cuando sea necesario. Todo esto requiere una gran cantidad de tiempo y recursos dedicados a la creación, el funcionamiento y las pruebas.

En segundo lugar, la replicación de datos en distintas Regiones de AWS utilizando servicios de AWS en la actualidad se realiza de forma asíncrona. La replicación asíncrona puede provocar la pérdida de datos. Esto significa que, durante una conmutación por error regional, existe la posibilidad de que se pierdan algunos datos o se produzcan incoherencias. Su tolerancia a la cantidad de pérdida de datos se define como su Objetivo de punto de recuperación (RPO). Los clientes, para los que se exige una sólida coherencia de datos, deberán crear sistemas de conciliación para solucionar estos problemas de coherencia cuando la región principal vuelva a estar disponible. O bien, deberán crear sus propios sistemas de replicación sincrónica o de escritura doble, lo que puede tener un impacto significativo en la latencia, el costo y la complejidad de la respuesta. También hacen que la región

secundaria sea una fuerte dependencia para cada transacción, lo que puede reducir la disponibilidad del sistema en general.

Por último, para muchas cargas de trabajo que utilizan un enfoque activo/en espera, se necesita un tiempo distinto de cero para realizar la conmutación por error a otra región. Es posible que su cartera de cargas de trabajo deba reducirse en la región principal en un orden específico, agotar las conexiones o detener procesos específicos. Posteriormente, puede que sea necesario volver a activar los servicios en un orden específico. Es posible que también haya que aprovisionar nuevos recursos o que necesiten tiempo para pasar las comprobaciones de estado pertinentes antes de entrar en servicio. Este proceso de conmutación por error se puede experimentar como un período de completa falta de disponibilidad. Esto es lo que preocupa a los RTO.

Dentro de una región, muchos servicios de AWS ofrecen una persistencia de datos muy uniforme. Las implementaciones de Amazon RDS Multi-AZ utilizan la replicación sincrónica. Amazon Simple Storage Service (Amazon S3) ofrece una sólida coherencia de lectura después de escritura. Amazon Elastic Block Storage (Amazon EBS) ofrece instantáneas coherentes ante bloqueos de varios volúmenes. Amazon DynamoDB puede realizar lecturas muy coherentes. Estas características pueden ayudarle a conseguir un RPO más bajo (en la mayoría de los casos, igual a cero) en una sola región que en arquitecturas multirregionales.

La evacuación de una zona de disponibilidad puede tener un RTO más bajo que una estrategia multirregional, ya que la infraestructura y los recursos ya están aprovisionados en todas las zonas de disponibilidad. Las arquitecturas Multi-AZ pueden seguir funcionando de forma estática cuando una zona de disponibilidad se ve afectada, en lugar de tener que ordenar cuidadosamente los servicios que se activan y desactivan, o bien agotar las conexiones. En lugar de pasar por un período de total falta de disponibilidad como el que puede producirse durante una conmutación por error regional, durante la evacuación de una zona de disponibilidad, es posible que muchos sistemas solo sufran una ligera degradación, ya que el trabajo se traslada a las zonas de disponibilidad restantes. Si el sistema se ha diseñado para mantener una estabilidad estática ante un error en una zona de disponibilidad (en este caso, eso implicaría aprovisionar previamente la capacidad en las demás zonas de disponibilidad para absorber la carga), es posible que los clientes de la carga de trabajo no se vean afectados en absoluto.

> Es posible que el deterioro de una sola zona de disponibilidad afecte a uno o varios servicios regionales de AWS, además de a la carga de trabajo. Si observa un impacto regional, debe tratar el evento como un deterioro del servicio regional, aunque el origen de ese impacto provenga de una sola zona de disponibilidad. La evacuación de una zona de disponibilidad

no mitigará este tipo de problema. Utilice los planes de respuesta que tenga establecidos para responder a un deterioro del servicio regional cuando esto ocurra.

El resto de este documento se centra en la segunda opción, la evacuación de la zona de disponibilidad, como forma de reducir los RTO y los RPO en caso de errores grises en zonas de disponibilidad individuales. Estos patrones pueden ayudar a mejorar el rendimiento de las arquitecturas Multi-AZ y, para la mayoría de las clases de cargas de trabajo, pueden reducir la necesidad de crear arquitecturas multirregionales para manejar este tipo de eventos.

Observabilidad de Multi-AZ

Para poder evacuar una zona de disponibilidad durante un evento que está aislado en una sola zona de disponibilidad, primero debe poder detectar que el error está, de hecho, aislada en una sola zona de disponibilidad. Esto requiere una visibilidad de alta fidelidad del comportamiento del sistema en cada zona de disponibilidad. Muchos AWS servicios proporcionan out-of-the-box métricas que proporcionan información operativa sobre sus recursos. Por ejemplo, Amazon EC2 proporciona numerosas métricas, como la CPU utilización, las lecturas y escrituras del disco y el tráfico de entrada y salida de la red.

Sin embargo, a medida que crea cargas de trabajo que utilizan estos servicios, necesita más visibilidad que solo esas métricas estándar. Desea tener visibilidad de la experiencia del cliente que proporciona su carga de trabajo. Además, necesita que sus métricas estén alineadas con las zonas de disponibilidad en las que se producen. Esta es la información que necesita para detectar los errores grises que se pueden observar de forma diferencial. Este nivel de visibilidad requiere una instrumentación.

La instrumentación requiere escribir código explícito. Este código debería realizar acciones como registrar la duración de las tareas, contar cuántos elementos se han ejecutado correctamente o no, recopilar metadatos sobre las solicitudes, etc. También es necesario definir los umbrales con antelación para definir qué se considera normal y qué no. Debe describir los objetivos y los diferentes umbrales de gravedad para la latencia, la disponibilidad y el recuento de errores de la carga de trabajo. El artículo de Amazon Builders' Library sobre la Instrumentación de sistemas distribuidos para la visibilidad operativa proporciona una serie de prácticas recomendadas.

Las métricas deben generarse tanto en el lado del servidor como en el lado del cliente. Una práctica recomendada para generar métricas del lado del cliente y comprender su experiencia es utilizar valores controlados, un software que analiza periódicamente la carga de trabajo y registra las métricas.

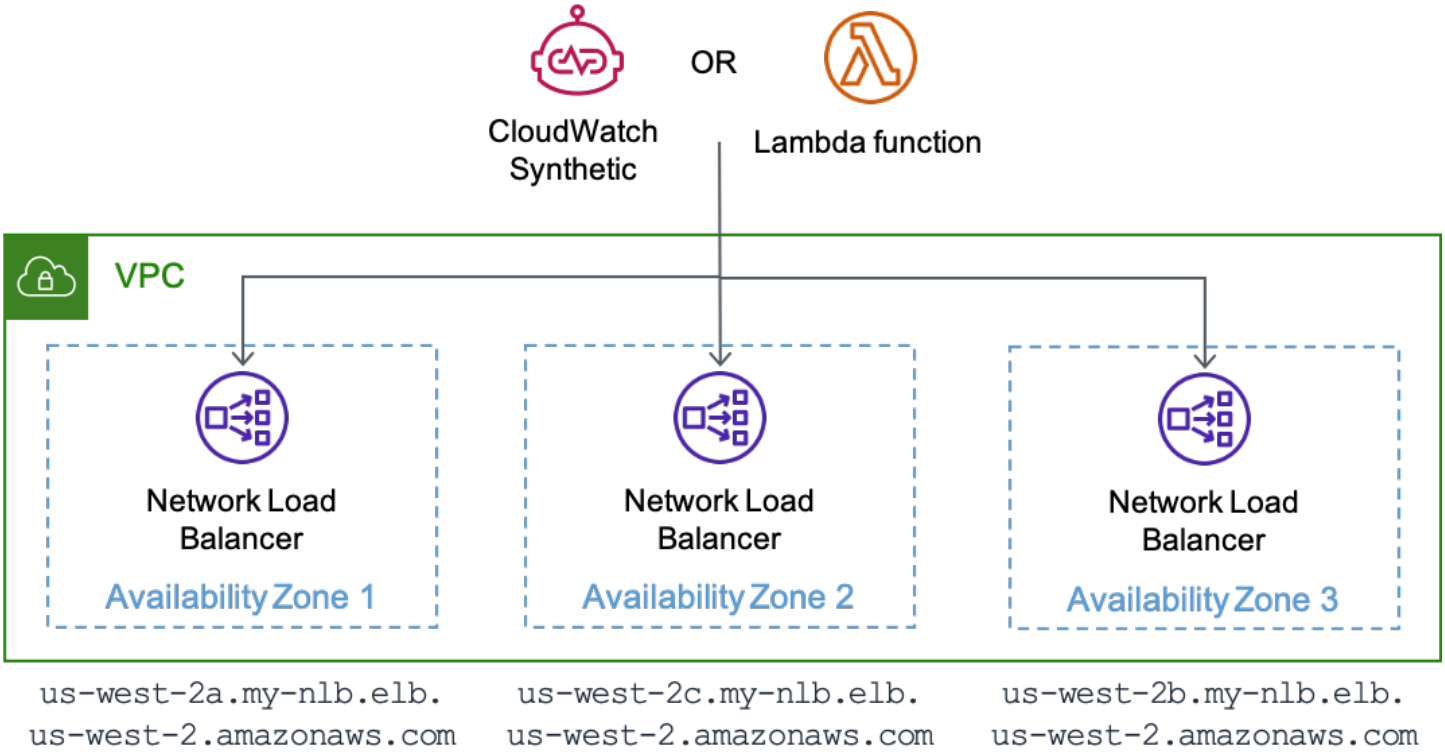
Además de generar estas métricas, también es necesario comprender su contexto. Una forma de hacerlo consiste en utilizar dimensiones. Las dimensiones dan a las métricas una identidad única y ayudan a explicar lo que indican las métricas. En el caso de las métricas que se utilizan para identificar errores en la carga de trabajo (por ejemplo, la latencia, la disponibilidad o el recuento de errores), debe utilizar dimensiones que se ajusten a los límites de aislamiento de errores.

Por ejemplo, si ejecuta un servicio web en una región, en varias zonas de disponibilidad y utiliza un marco web M odel-view-controller (MVC), debe utilizar, `Region` `Availability Zone`

IDControllerAction, y InstanceId como dimensiones para sus conjuntos de dimensiones (si utilizara microservicios, podría utilizar el nombre y el HTTP método del servicio en lugar de los nombres del controlador y de la acción). Esto es así porque se espera que estos límites aíslen los distintos tipos de errores. No se espera que un error en el código del servicio web que afecta a su capacidad de enumerar productos también afecte a la página de inicio. Del mismo modo, no esperaría que un EBS volumen completo en una sola EC2 instancia afectara a otras EC2 instancias a la hora de publicar su contenido web. La dimensión del ID de zona de disponibilidad es lo que le permite identificar los impactos relacionados con la zona de disponibilidad de forma coherente en todas las Cuentas de AWS. Puede encontrar el ID de zona de disponibilidad en sus cargas de trabajo de varias maneras. Consulte Apéndice A: Obtener el ID de la zona de disponibilidad para ver algunos ejemplos.

Si bien este documento utiliza principalmente Amazon EC2 como recurso informático en los ejemplos, InstanceId podría sustituirse por un ID de contenedor para los recursos informáticos de Amazon Elastic Container Service (AmazonECS) y Amazon Elastic Kubernetes Service (EKSAmazon) como componentes de sus dimensiones.

Sus valores controlados también pueden usar Controller, Action, AZ-ID y Region como dimensiones en sus métricas si tiene puntos de conexión zonales para su carga de trabajo. En este caso, alinee sus valores controlados para que se ejecuten en la zona de disponibilidad que estén probando. Esto garantiza que si un evento aislado de la zona de disponibilidad afecta a la zona de disponibilidad donde se ejecuta su valor controlado, no registra las métricas que hacen que una zona de disponibilidad diferente que esté probando parezca tener un estado incorrecto. Por ejemplo, Canary puede probar cada punto final zonal para detectar un servicio detrás de un Network Load Balancer NLB () o Application Load ALB Balancer () utilizando sus nombres zonales. DNS



Un canario que funciona con CloudWatch Synthetics o una AWS Lambda función que prueba cada punto final zonal de un NLB

Al generar métricas con estas dimensiones, puede establecer CloudWatch alarmas de Amazon que le notifiquen cuando se produzcan cambios en la disponibilidad o la latencia dentro de esos límites. También puede analizar rápidamente esos datos mediante paneles. Para utilizar las métricas y los registros de forma eficiente, Amazon CloudWatch ofrece el formato de métrica integrado (EMF) que te permite integrar métricas personalizadas en los datos de registro. CloudWatchextrae automáticamente las métricas personalizadas para que puedas visualizarlas y alarmarlas. AWS proporciona varias bibliotecas de clientes para diferentes lenguajes de programación que facilitan su inicioEMF. Se pueden usar con AmazonEC2, Amazon ECS EKS AWS Lambda, Amazon y entornos locales. Con las métricas integradas en sus registros, también puede usar Amazon CloudWatch Contributor Insights para crear gráficos de series temporales que muestren los datos de los colaboradores. En este escenario, podemos mostrar los datos agrupados por dimensiones como `AZ-ID`, `InstanceId` o `Controller`, así como cualquier otro campo del registro, como `SuccessLatency` o `HttpResponseCode`.

```
{
  "_aws": {
    "Timestamp": 1634319245221,
```

```
    "CloudWatchMetrics": [
      {
        "Namespace": "workloadname/frontend",
        "Metrics": [
          { "Name": "2xx", "Unit": "Count" },
          { "Name": "3xx", "Unit": "Count" },
          { "Name": "4xx", "Unit": "Count" },
          { "Name": "5xx", "Unit": "Count" },
          { "Name": "SuccessLatency", "Unit": "Milliseconds" }
        ],
        "Dimensions": [
          [ "Controller", "Action", "Region", "AZ-ID", "InstanceId"],
          [ "Controller", "Action", "Region", "AZ-ID"],
          [ "Controller", "Action", "Region"]
        ]
      }
    ],
    "LogGroupName": "/loggroupname"
  },
  "CacheRefresh": false,
  "Host": "use1-az2-name.example.com",
  "SourceIp": "34.230.82.196",
  "TraceId": "|e3628548-42e164ee4d1379bf.",
  "Path": "/home",
  "OneBox": false,
  "Controller": "Home",
  "Action": "Index",
  "Region": "us-east-1",
  "AZ-ID": "use1-az2",
  "InstanceId": "i-01ab0b7241214d494",
  "LogGroupName": "/loggroupname",
  "HttpResponseCode": 200,
  "2xx": 1,
  "3xx": 0,
  "4xx": 0,
  "5xx": 0,
  "SuccessLatency": 20
}
```

Este registro tiene tres conjuntos de dimensiones. Progresan en orden de granularidad, desde la Instancia a la Zona de disponibilidad y a la Región (`Controller` y `Action` siempre se incluyen en este ejemplo). Permiten crear alarmas en toda la carga de trabajo que indican cuándo hay un impacto en una determinada acción del controlador en una instancia individual, en una zona de

disponibilidad individual o en toda la Región de AWS. Estas dimensiones se utilizan para el recuento de las métricas de HTTP respuesta de 2xx, 3xx, 4xx y 5xx, así como para la latencia de las métricas de solicitudes correctas (si la solicitud fallara, también registraría una métrica para la latencia de las solicitudes fallidas). El registro también registra otra información, como la HTTP ruta, la IP de origen del solicitante y si esta solicitud requirió la actualización de la caché local. Luego, estos puntos de datos se pueden usar para calcular la disponibilidad y la latencia de cada una de las cargas de trabajo proporcionadasAPI.

> **Nota sobre el uso de códigos de HTTP respuesta para las métricas de disponibilidad**
>
> Por lo general, puede considerar que las respuestas 2xx y 3xx son correctas y las respuestas 5xx son fallidas. Los códigos de respuesta 4xx se encuentran en un punto intermedio. Por lo general, se producen debido a un error del cliente. Puede que un parámetro esté fuera del rango y provoque una respuesta 400, o que estén solicitando algo que no existe, lo que provoca una respuesta 404. Estas respuestas no deben contar para la disponibilidad de su carga de trabajo. Sin embargo, esto también puede deberse a un error en el software. Por ejemplo, si ha introducido una validación de entradas más estricta que rechaza una solicitud que anteriormente hubiera sido correcta, la respuesta 400 puede considerarse una disminución de la disponibilidad. O tal vez está limitando el número de peticiones del cliente, lo que devuelve una respuesta 429. Si bien limitar un cliente protege el servicio para mantener la disponibilidad, desde la perspectiva del cliente, el servicio no está disponible para procesar su solicitud. Deberá decidir si los códigos de respuesta 4xx forman parte o no de su cálculo de disponibilidad.

Si bien en esta sección se describe su uso CloudWatch como una forma de recopilar y analizar métricas, no es la única solución que puede utilizar. También puede enviar las métricas a Amazon Managed Service para Prometheus y Amazon Managed Grafana, a una tabla de Amazon DynamoDB, o utilizar una solución de supervisión de terceros. La clave es que las métricas que genere su carga de trabajo deben contener un contexto sobre los límites de aislamiento de errores de la carga de trabajo.

Con cargas de trabajo que generan métricas con dimensiones alineadas con los límites del aislamiento de errores, puede crear una observabilidad que detecte los errores aislados de las zonas de disponibilidad. En las siguientes secciones, se describen tres enfoques complementarios para detectar los errores que se derivan del deterioro de una única zona de disponibilidad.

Temas

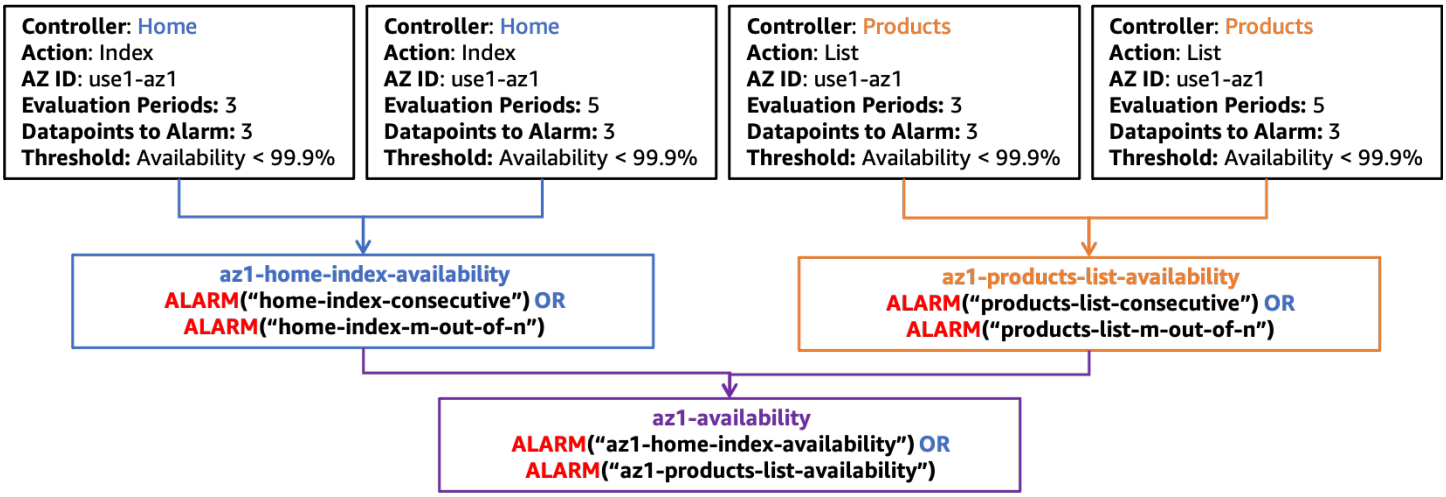
Detección de fallos con alarmas CloudWatch compuestas

En CloudWatch las métricas, cada conjunto de dimensiones es una métrica única y puede crear una CloudWatch alarma en cada una de ellas. A continuación, puede crear alarmas CloudWatch compuestas de Amazon para agregar estas métricas.

Para detectar con precisión el impacto, los ejemplos de este documento utilizarán dos estructuras de CloudWatch alarma diferentes para cada dimensión activada en la que se active la alarma. Cada alarma utilizará un período de un minuto, lo que significa que la métrica se evalúa una vez por minuto. El primer enfoque consiste en utilizar tres puntos de datos de infracción consecutivos estableciendo los Períodos de evaluación y los Puntos de datos para la alarma en tres, lo que supone un impacto de tres minutos en total. El segundo enfoque consiste en utilizar “M de N” cuando se produzca una infracción de tres puntos de datos en un intervalo de cinco minutos, estableciendo los Períodos de evaluación en cinco y los Puntos de datos para la alarma en tres. Esto permite detectar una señal constante, así como una que fluctúe en un breve período de tiempo. Las duraciones de tiempo y los números de puntos de datos que se incluyen aquí son una sugerencia. Utilice valores que se adapten a sus cargas de trabajo.

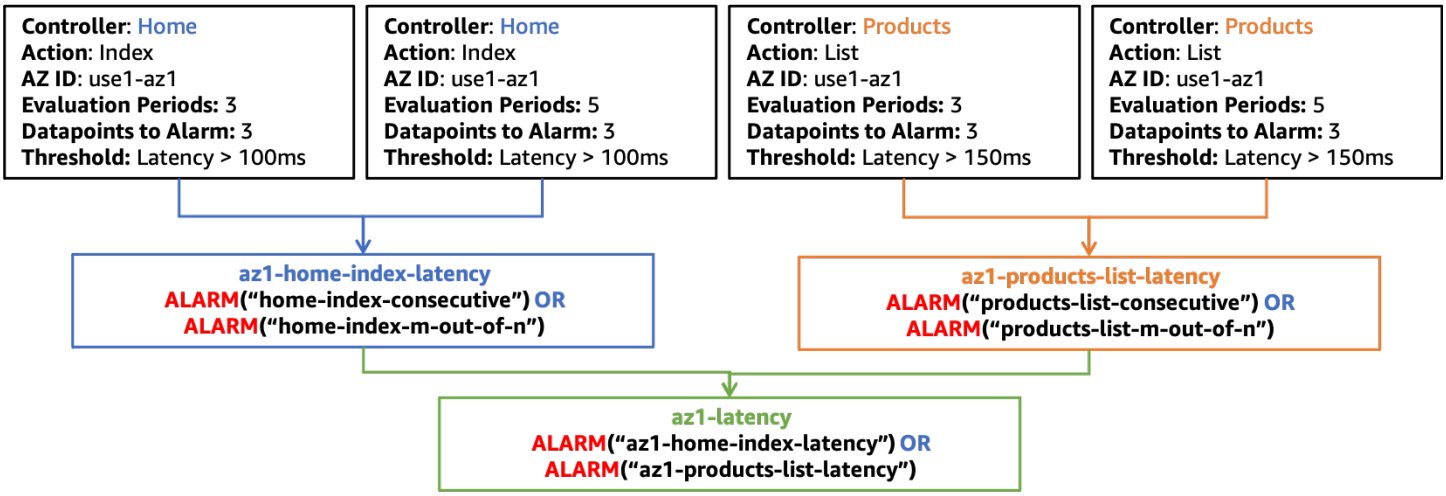
Detectar el impacto en una sola zona de disponibilidad

Con este constructo, supongamos una carga de trabajo que utiliza `Controller`, `Action`, `InstanceId`, `AZ-ID` y `Region` como dimensiones. La carga de trabajo tiene dos controladores, Products y Home, y una acción por controlador, List e Index, respectivamente. Opera en tres zonas de disponibilidad de la región `us-east-1`. Deberá crear dos alarmas de disponibilidad para cada combinación de `Controller` y `Action` en cada zona de disponibilidad, así como dos alarmas de latencia para cada una. A continuación, si lo desea, puede crear una alarma compuesta de disponibilidad para cada combinación de `Controller` y `Action`. Por último, debe crear una alarma compuesta que agrupe todas las alarmas de disponibilidad de la zona de disponibilidad. Esto se muestra en la siguiente figura para una sola zona de disponibilidad, `use1-az1`, utilizando la alarma compuesta opcional para cada combinación de `Controller` y `Action` (también hay alarmas similares para las zonas de disponibilidad `use1-az2` y `use1-az3`, pero no se muestran por motivos de simplicidad).



Estructura de alarma compuesta de disponibilidad en *use1-az1*

También debe crear una estructura de alarma similar para la latencia, como se muestra en la siguiente figura.



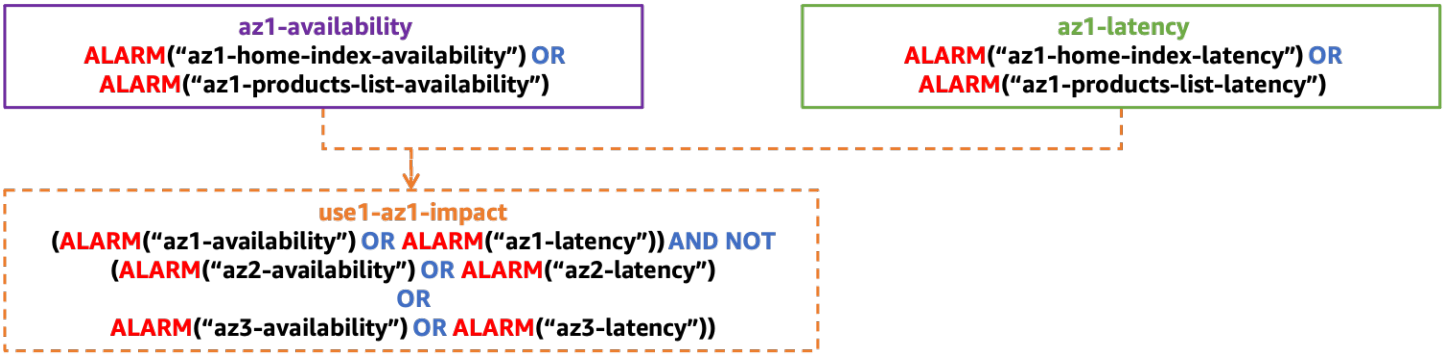
Estructura de alarma compuesta de latencia en *use1-az1*

Para el resto de las figuras de esta sección, solo se mostrarán las alarmas compuestas `az1-availability` y `az1-latency` en el nivel superior. Estas alarmas compuestas `az1-availability` y `az1-latency` indicarán si la disponibilidad cae por debajo de los umbrales definidos o si la latencia supera los umbrales definidos en una determinada zona de disponibilidad para cualquier parte de su carga de trabajo. También puede considerar la posibilidad de medir el rendimiento para detectar un impacto que impida que su carga de trabajo en una sola zona de disponibilidad reciba trabajo. También puede integrar las alarmas generadas a partir de las métricas emitidas por los valores controlados en estas alarmas compuestas. De esta forma, si el lado del

servidor o el lado del cliente ve un impacto en la disponibilidad o la latencia, la alarma generará una alerta.

Asegurarse de que el impacto no sea regional

Se puede usar otro conjunto de alarmas compuestas para garantizar que solo un evento aislado de la zona de disponibilidad provoque la activación de la alarma. Para ello, debe asegurarse de que la alarma compuesta de la zona de disponibilidad tenga un estado `ALARM`, mientras que las alarmas compuestas de las demás zonas de disponibilidad tienen el estado `OK`. Esto generará una alarma compuesta por cada zona de disponibilidad que utilice. En la siguiente figura, se muestra un ejemplo (recuerde que hay alarmas de latencia y disponibilidad en `use1-az2` y `use1-az3`, `az2-latency`, `az2-availability`, `az3-latency` y `az3-availability`, que no se muestran por motivos de simplicidad).



Estructura de alarma compuesta para detectar un impacto aislado en una única zona de disponibilidad

Asegurarse de que el impacto no se deba a una sola instancia

Una sola instancia (o un pequeño porcentaje de su flota total) puede tener un impacto desproporcionado en las métricas de disponibilidad y latencia, lo que puede hacer que toda la zona de disponibilidad parezca estar afectada, cuando en realidad no es así. Eliminar una sola instancia problemática es más rápido e igual de eficaz que evacuar una zona de disponibilidad.

Por lo general, las instancias y los contenedores se tratan como recursos efímeros y, con frecuencia, se sustituyen por servicios como AWS Auto Scaling. Es difícil crear una nueva CloudWatch alarma cada vez que se crea una nueva instancia (pero sin duda es posible con los ganchos de ciclo de vida de Amazon EventBridge o Amazon EC2 Auto Scaling). En su lugar, puedes usar CloudWatch Contributor Insights para identificar la cantidad de contribuyentes a las métricas de disponibilidad y latencia.

Por ejemplo, en el caso de una aplicación HTTP web, puede crear una regla para identificar a los principales contribuyentes de cinco veces HTTP las respuestas en cada zona de disponibilidad. Esto identificará qué instancias están contribuyendo a una disminución de la disponibilidad (nuestra métrica de disponibilidad definida anteriormente se basa en la presencia de errores 5xx). Con el ejemplo del EMF registro, cree una regla con una clave de`InstanceId`. A continuación, filtre el registro por el campo `HttpResponseCode`. Este ejemplo es una regla para la zona de disponibilidad de `use1-az1`.

```
{
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.InstanceId",
                "IsPresent": true
            },
            {
                "Match": "$.HttpStatusCode",
                "IsPresent": true
            },
            {
                "Match": "$.HttpStatusCode",
                "GreaterThan": 499
            },
            {
                "Match": "$.HttpStatusCode",
                "LessThan": 600
            },
            {
                "Match": "$.AZ-ID",
                "In": ["use1-az1"]
            },
        ],
        "Keys": [
            "$.InstanceId"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/loggroupname"
    ],
    "Schema": {
```

```
        "Name": "CloudWatchLogRule",
        "Version": 1
    }
}
```

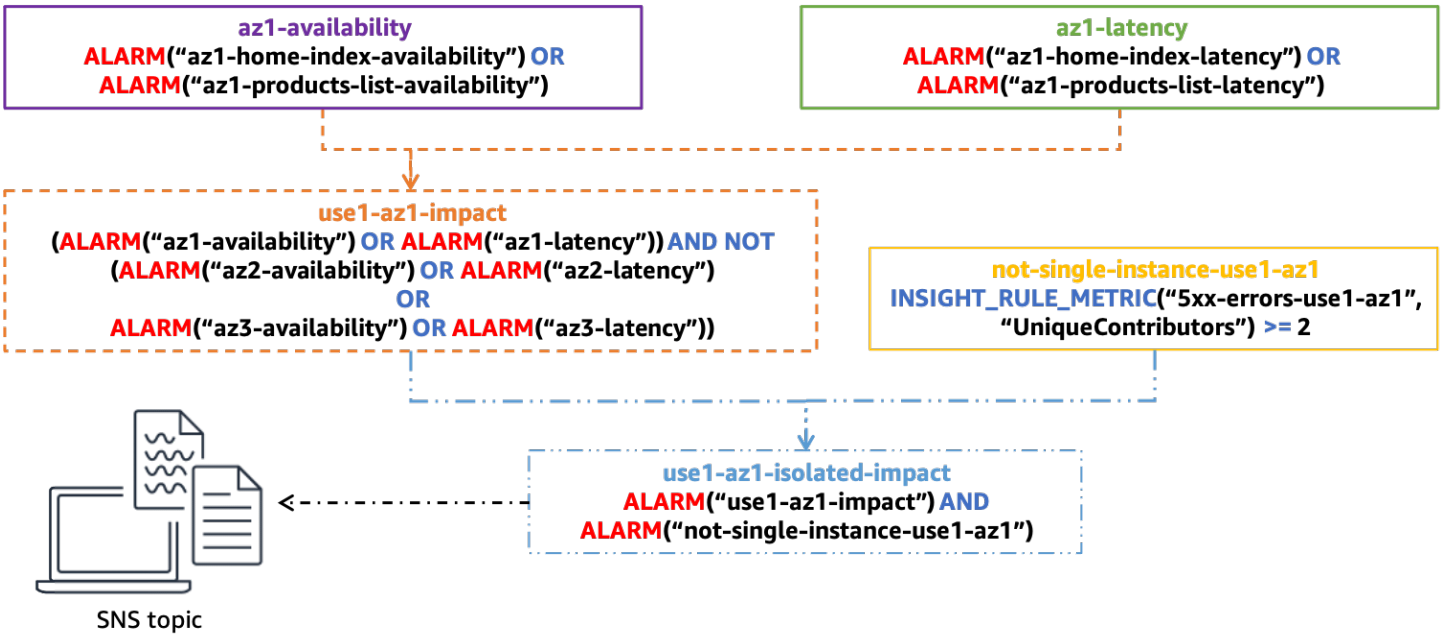
CloudWatch También se pueden crear alarmas en función de estas reglas. Puede crear alarmas en función de las reglas de Contributor Insights utilizando la matemática de métricas y la función `INSIGHT_RULE_METRIC` con la métrica `UniqueContributors`. También puedes crear reglas adicionales de Contributor Insights con CloudWatch alarmas para métricas como la latencia o el recuento de errores, además de otras relacionadas con la disponibilidad. Estas alarmas se pueden usar con las alarmas compuestas de impacto en la zona de disponibilidad aislada para garantizar que las instancias individuales no activen la alarma. La métrica de la regla de Insights para `use1-az1` será parecida a la siguiente:

```
INSIGHT_RULE_METRIC("5xx-errors-use1-az1", "UniqueContributors")
```

Puede definir una alarma cuando esta métrica supere un umbral; en este ejemplo, dos. Se activa cuando los colaboradores únicos a las respuestas 5xx superan ese umbral, lo que indica que el impacto se origina en más de dos instancias. La razón por la que esta alarma utiliza una comparación “mayor que” en lugar de “menor que” es garantizar que un valor cero para los colaboradores únicos no active la alarma. Esto indica que el impacto no proviene de una sola instancia. Ajuste este umbral para su carga de trabajo individual. Una regla general es que este número sea un 5 % o más del total de recursos de la zona de disponibilidad. Si el tamaño de la muestra es suficiente, más del 5 % de los recursos afectados tiene relevancia estadística.

Resumen global

La siguiente figura muestra la estructura de alarma compuesta completa para una única zona de disponibilidad:



Estructura de alarma compuesta completa para determinar el impacto en una zona de disponibilidad única

La última alarma compuesta, `use1-az1-isolated-impact`, se activa cuando la alarma compuesta que indica un impacto aislado en la zona de disponibilidad debido a la latencia o la disponibilidad, `use1-az1-aggregate-alarm`, tiene un estado `ALARM`, y cuando la alarma basada en la regla de Contributor Insights para la misma zona de disponibilidad, `not-single-instance-use1-az1`, también tiene un estado `ALARM` (lo que significa que el impacto se produce en más de una sola instancia). Debe crear esta pila de alarmas para cada zona de disponibilidad que utilice la carga de trabajo.

Puedes adjuntar una alerta de Amazon Simple Notification Service (AmazonSNS) a esta última alarma. Todas las alarmas anteriores se configuran sin necesidad de realizar ninguna acción. La alerta puede notificar a un operador por correo electrónico que inicie una investigación manual. También puede iniciar la automatización para evacuar la zona de disponibilidad. No obstante, hay que tener cuidado al crear la automatización para responder a estas alertas. Tras la evacuación de una zona de disponibilidad, el resultado debería ser que se mitigue el aumento de las tasas de error y que la alarma vuelva a su estado `OK`. Si el impacto se produce en otra zona de disponibilidad, es posible que la automatización pueda evacuar una segunda o tercera zona de disponibilidad, lo que podría eliminar toda la capacidad disponible de la carga de trabajo. La automatización debe comprobar si ya se ha realizado una evacuación antes de tomar ninguna medida. Es posible que también deba escalar los recursos en otras zonas de disponibilidad para que la evacuación se lleve a cabo correctamente.

Cuando agrega nuevos controladores o acciones a su aplicación MVC web, o un nuevo microservicio o, en general, cualquier funcionalidad adicional que desee monitorear por separado, solo necesita modificar algunas alarmas en esta configuración. Deberá crear nuevas alarmas de disponibilidad y latencia para la nueva funcionalidad y, a continuación, las añadirá a las alarmas compuestas de disponibilidad y latencia alineadas con la zona de disponibilidad correspondiente, `az1-latency` y `az1-availability` en el ejemplo que hemos estado usando aquí. Las alarmas compuestas restantes permanecen estáticas una vez configuradas. Esto permite que la incorporación de nuevas funcionalidades con este enfoque sea un proceso muy sencillo.

Detección de errores utilizando la detección de valores atípicos

Con el enfoque anterior, puede surgir una brecha si se observan tasas de errores elevadas en varias zonas de disponibilidad por un motivo no correlacionado. Imagine un escenario en el que tiene EC2 instancias implementadas en tres zonas de disponibilidad y su umbral de alarma de disponibilidad es del 99%. En él, se produce un deterioro en una sola zona de disponibilidad, lo que aísla muchas instancias y hace que la disponibilidad en esa zona disminuya al 55 %. Al mismo tiempo, pero en una zona de disponibilidad diferente, una sola EC2 instancia agota todo el almacenamiento de su EBS volumen y ya no puede escribir archivos de registro. Esto hace que comience a generar errores, pero aun así supera las comprobaciones de estado del equilibrador de carga, ya que estas no activan la escritura de un archivo de registro. Esto hace que la disponibilidad disminuya al 98 % en esa zona de disponibilidad. En este caso, la alarma de impacto en una única zona de disponibilidad no se activará, porque está viendo un impacto en la disponibilidad en varias zonas de disponibilidad. Sin embargo, aún podría mitigar casi todo el impacto evacuando la zona de disponibilidad deteriorada.

En algunos tipos de cargas de trabajo, es posible que se produzcan errores de forma uniforme en todas las zonas de disponibilidad, donde la métrica de disponibilidad anterior podría no ser útil. Tomemos, AWS Lambda por ejemplo. AWS permite a los clientes crear su propio código para ejecutarlo en la función Lambda. Para utilizar el servicio, debe cargar el código en un ZIP archivo, incluidas las dependencias, y definir el punto de entrada a la función. Sin embargo, a veces los clientes se equivocan en esta parte; por ejemplo, pueden olvidar una dependencia crítica en el ZIP archivo o escribir mal el nombre del método en la definición de la función Lambda. Esto provoca que no se pueda invocar la función y se produce un error. AWS Lambda ve este tipo de errores todo el tiempo, pero no son indicativos de que algo esté necesariamente en mal estado. Sin embargo, un deterioro de la zona de disponibilidad también puede provocar la aparición de estos errores.

Para detectar señales en este tipo de ruido, puede utilizar la detección de valores atípicos para determinar si existe un sesgo estadísticamente relevante en el número de errores entre las zonas de

disponibilidad. Aunque veamos errores en varias zonas de disponibilidad, si realmente se hubiera producido un error en una de ellas, esperaríamos ver una tasa de errores mucho mayor en esa zona de disponibilidad en comparación con las demás, o posiblemente mucho más baja. Pero, ¿cuánto mayor o cuánto menor?

Una forma de realizar este análisis consiste en utilizar una prueba de chi cuadrado (χ^2) para detectar diferencias estadísticamente relevantes en las tasas de errores entre las distintas zonas de disponibilidad (hay muchos algoritmos diferentes para detectar valores atípicos). Veamos cómo funciona la prueba de chi cuadrado.

La prueba de chi cuadrado evalúa la probabilidad de que se produzca alguna distribución de los resultados. En este caso, nos interesa la distribución de los errores en un conjunto definido deAZs. En este ejemplo, para facilitar las matemáticas, supongamos cuatro zonas de disponibilidad.

En primer lugar, establezca la hipótesis nula, que define cuál cree que es el resultado predeterminado. En esta prueba, la hipótesis nula es que se espera que los errores estén distribuidos uniformemente en cada zona de disponibilidad. A continuación, genere la hipótesis alternativa, según la cual los errores no se distribuyen uniformemente, lo que indica que la zona de disponibilidad se ha deteriorado. Ahora puede probar estas hipótesis con los datos de sus métricas. Para ello, muestreará sus métricas en un período de cinco minutos. Supongamos que obtiene 1000 puntos de datos publicados en esa ventana, en la que ve un total de 100 errores. Es de esperar que, con una distribución uniforme, los errores se produzcan el 25 % del tiempo en cada una de las cuatro zonas de disponibilidad. Supongamos que la siguiente tabla muestra lo que esperaba comparado con lo que observa realmente.

Tabla 1: Los errores esperados comparados con los errores reales observados

AZ	Expected	Real
`use1-az1`	25	20
`use1-az2`	25	20
`use1-az3`	25	25
`use1-az4`	25	35

Como puede ver, la distribución en realidad no es uniforme. Sin embargo, puede pensar que esto ha ocurrido debido a un cierto nivel de aleatoriedad en los puntos de datos que ha muestreado. Hay un

cierto nivel de probabilidad de que este tipo de distribución se produzca en el conjunto de la muestra y aun así suponer que la hipótesis nula es cierta. Esto nos lleva a la siguiente pregunta: ¿cuál es la probabilidad de obtener un resultado tan extremo como mínimo? Si esa probabilidad está por debajo de un umbral definido, debe rechazar la hipótesis nula. Para que sea estadísticamente relevante, esta probabilidad debe ser del 5 % o menos[1].

[1] Craparo, Robert M. (2007). “Significance level”. En Salkind, Neil J. Encyclopedia of Measurement and Statistics 3. Thousand Oaks, CA: SAGE Publicaciones. págs. 889—891. ISBN1-412-91611-9.

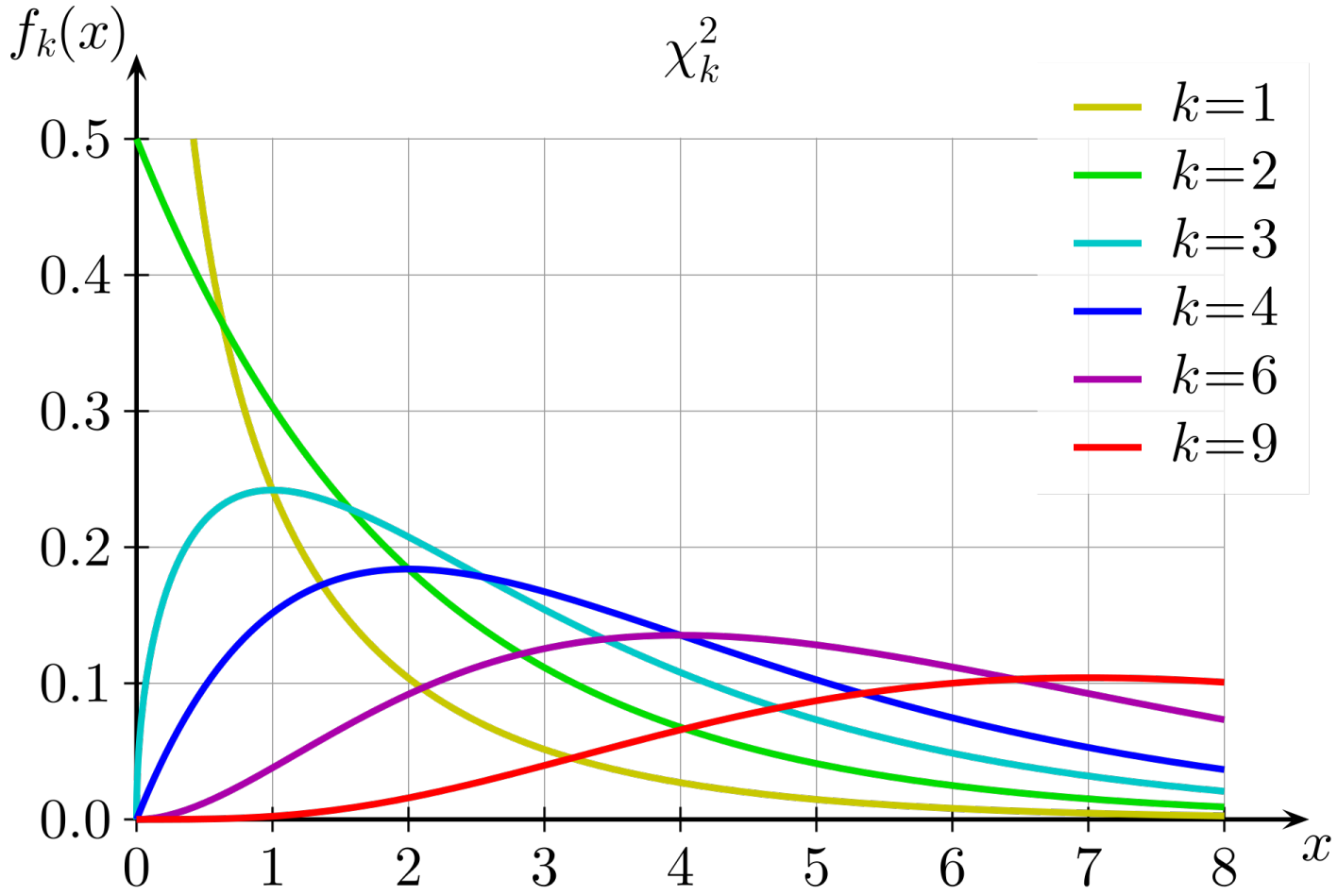
¿Cómo se calcula la probabilidad de este resultado? Utilice la estadística χ^2, que proporciona distribuciones muy estudiadas y permite determinar la probabilidad de obtener un resultado así de extremo o más con esta fórmula.

$$E_i = expected\ observations\ of\ type\ i$$
$$O_i = actual\ observations\ of\ type\ i \qquad (1)$$
$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

En nuestro ejemplo, esto da como resultado:

$$\chi^2 = \frac{(20-25)^2}{25} + \frac{(20-25)^2}{25} + \frac{(25-25)^2}{25} + \frac{(35-25)^2}{25}$$
$$\chi^2 = \frac{-5^2}{25} + \frac{-5^2}{25} + \frac{0^2}{25} + \frac{10^2}{25} \qquad (2)$$
$$\chi^2 = 1 + 1 + 0 + 4$$
$$\chi^2 = 6$$

Entonces, ¿qué significa 6 en términos de nuestra probabilidad? Es necesario observar una distribución de chi cuadrado con el grado de libertad adecuado. En la siguiente figura, se muestran varias distribuciones de chi cuadrado para distintos grados de libertad.



Distribuciones de chi cuadrado para diferentes grados de libertad

El grado de libertad se calcula como un valor menos que el número de opciones de la prueba. En este caso, dado que hay cuatro zonas de disponibilidad, el grado de libertad es tres. A continuación, querrá saber el área bajo la curva (la integral) de x ≥ 6 en el gráfico k = 3. También puede usar una tabla precalculada con valores de uso común para aproximar ese valor.

Tabla 2: Valores críticos de chi cuadrado

Grados de libertad	Probabilidad inferior al valor crítico				
	0.75	0.90	0,95	0,99	0,999
1	1.323	2.706	3.841	6.635	10,828
2	2.773	4,605	5.991	9.210	13,816

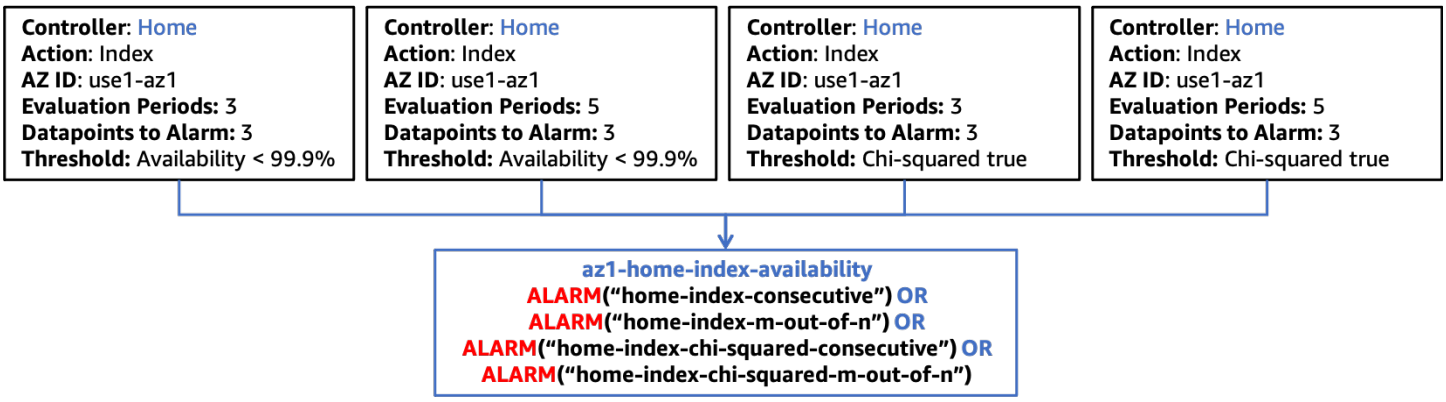
Grados de libertad	Probabilidad inferior al valor crítico				
	0.75	0.90	0,95	0,99	0,999
3	4.108	6.251	7,815	11,345	16.266
4	5.385	7,779	9,488	13.277	18.467

Para tres grados de libertad, el valor de chi cuadrado de seis se encuentra entre las columnas de probabilidad de 0,75 y 0,9. Esto significa que hay una probabilidad superior al 10 % de que se produzca esta distribución, lo que no es inferior al umbral del 5 %. Por lo tanto, acepta la hipótesis nula y determina que no hay una diferencia estadísticamente relevante en las tasas de errores entre las zonas de disponibilidad.

Las matemáticas CloudWatch métricas no admiten de forma nativa la realización de una prueba estadística con chi-cuadrado, por lo que tendrá que recopilar las métricas de error aplicables CloudWatch y ejecutar la prueba en un entorno informático como Lambda. Puedes decidir realizar esta prueba a nivel de MVC controlador/acción o microservicio individual, o bien a nivel de zona de disponibilidad. Deberá tener en cuenta si un deterioro de la zona de disponibilidad afectaría a cada controlador/acción o microservicio por igual, o si algo como un DNS fallo podría afectar a un servicio de bajo rendimiento y no a uno de alto rendimiento, lo que podría enmascarar el impacto al agregarse. En cualquier caso, seleccione las dimensiones adecuadas para crear la consulta. El nivel de granularidad también afectará a las alarmas resultantes que cree. CloudWatch

Recopile la métrica de recuento de errores para cada zona de disponibilidad y controlador/acción en un período de tiempo específico. Primero, calcule el resultado de la prueba de chi cuadrado como true (ha habido un sesgo estadísticamente relevante) o false (no lo ha habido, es decir, la hipótesis nula es válida). Si el resultado es false, publique un punto de datos 0 en su flujo de métricas para obtener resultados de chi cuadrado para cada zona de disponibilidad. Si el resultado es true, publique un punto de datos 1 para la zona de disponibilidad con los errores más alejados del valor esperado y 0 para los demás (consulte Apéndice B: Ejemplo de cálculo con chi cuadrado para ver el código de ejemplo que se puede utilizar en una función de Lambda). Puede seguir el mismo enfoque que las alarmas de disponibilidad anteriores mediante la creación de una alarma CloudWatch métrica de 3 filas y una alarma métrica de 3 de cada 5 CloudWatch en función de los puntos de datos que produce la función Lambda. Como en los ejemplos anteriores, este enfoque se puede modificar para utilizar más o menos puntos de datos en un intervalo más corto o más largo.

A continuación, añada estas alarmas a la alarma de disponibilidad de la zona de disponibilidad existente para la combinación de controlador y acción, como se muestra en la siguiente figura.



Integración de la prueba estadística de chi cuadrado con alarmas compuestas

Como se mencionó anteriormente, al incorporar una nueva funcionalidad a su carga de trabajo, solo necesita crear las alarmas CloudWatch métricas adecuadas que sean específicas de esa nueva funcionalidad y actualizar el siguiente nivel de la jerarquía compuesta de alarmas para incluir esas alarmas. El resto de la estructura de alarmas permanece estático.

Detección de errores de recursos zonales de una sola instancia

En algunos casos, es posible que tenga una única instancia activa de un recurso zonal, generalmente sistemas que requieren un componente de escritura única, como una base de datos relacional (como AmazonRDS) o una caché distribuida (como Amazon ElastiCache (OSSRedis)). Si el deterioro de una sola zona de disponibilidad afecta a la zona de disponibilidad en la que se encuentra el recurso principal, puede afectar a todas las zonas de disponibilidad que acceden al recurso. Esto podría hacer que se superaran los umbrales de disponibilidad en todas las zonas de disponibilidad, lo que implicará que el primer enfoque no identifica correctamente la única fuente de impacto de la zona de disponibilidad. Además, es probable que vea tasas de errores similares en cada zona de disponibilidad, lo que significa que el análisis de valores atípicos tampoco detecta el problema. Esto significa que necesita implementar una observabilidad adicional para detectar específicamente este escenario.

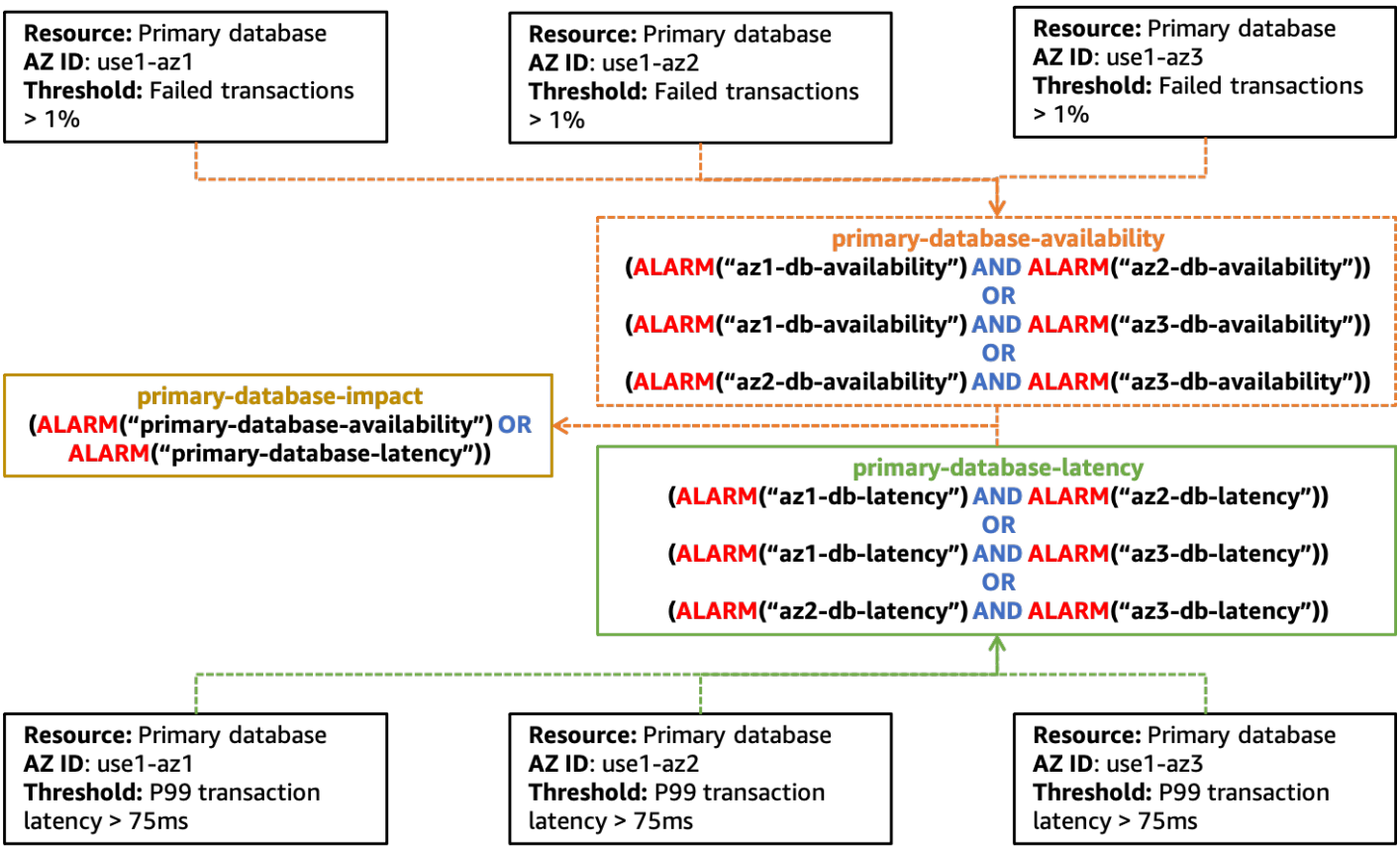
Es probable que el recurso que le preocupa genere sus propias métricas sobre su estado, pero si se produce un deterioro de la zona de disponibilidad, es posible que ese recurso no pueda ofrecer dichas métricas. En este caso, debe crear o actualizar alarmas para saber si está yendo a ciegas. Si hay métricas importantes que ya supervisa y para las que ha activado la alarma, puede configurar la alarma para que considere los datos que faltan como una infracción. Esto le permitirá saber

si el recurso deja de notificar datos, y se puede incluir en las mismas alarmas seguidas y m de n utilizadas anteriormente.

También es posible que en algunas de las métricas que indican el estado del recurso se publique un dato con un valor cero cuando no haya ninguna actividad. Si el deterioro impide las interacciones con el recurso, no puede utilizar el enfoque de los datos que faltan para este tipo de métricas. Probablemente tampoco desee crear una alarma cuando el valor sea cero, ya que podría haber escenarios legítimos en los que esto entre dentro de los umbrales normales. El mejor enfoque para detectar este tipo de problema consiste en generar métricas a partir de los recursos que utilizan esta dependencia. En este caso, queremos detectar el impacto en varias zonas de disponibilidad utilizando alarmas compuestas. Estas alarmas deben utilizar varias categorías de métricas críticas relacionadas con el recurso. A continuación, se muestran algunos ejemplos:

- Rendimiento: la tasa de unidades de trabajo entrantes. Pueden ser transacciones, lecturas, escrituras, etc.
- Disponibilidad: mide la cantidad de unidades de trabajo con éxito y erróneas.
- Latencia: mide varios percentiles de latencia para ver que el trabajo realizado con éxito en las operaciones críticas.

  Una vez más, puede crear alarmas de métricas seguidas y m de n para cada métrica de cada categoría de métrica que desee medir. Como antes, se pueden combinar en una alarma compuesta para determinar si este recurso compartido es la fuente del impacto en varias zonas de disponibilidad. Desea poder identificar el impacto en más de una zona de disponibilidad con las alarmas compuestas, pero no un requisito que el impacto se produzca necesariamente en todas las zonas de disponibilidad. La estructura de alarma compuesta de alto nivel para este tipo de enfoque se muestra en la siguiente figura.



Un ejemplo de creación de alarmas para detectar el impacto en varias zonas de disponibilidad debido a un solo recurso

Observará que este diagrama es menos prescriptivo sobre el tipo de alarmas de métricas que se deben utilizar y la jerarquía de las alarmas compuestas. Esto se debe a que descubrir este tipo de problema puede resultar difícil y requerirá una especial atención a las señales correctas del recurso compartido. Es posible que esas señales también deban evaluarse de manera específica.

Además, debe tener en cuenta que la alarma `primary-database-impact` no está asociada a una zona de disponibilidad específica. Esto se debe a que la instancia de base de datos principal puede estar ubicada en cualquier zona de disponibilidad que esté configurada para usar y no hay una CloudWatch métrica que especifique dónde se encuentra. Cuando vea que se activa esta alarma, debe utilizarla como una señal de que puede haber un problema con el recurso e iniciar una conmutación por error a otra zona de disponibilidad, si no se ha realizado automáticamente. Después de mover el recurso a otra zona de disponibilidad, puede esperar y comprobar si la alarma de zona de disponibilidad aislada está activada, o bien optar por invocar de forma preventiva su plan de evacuación de la zona de disponibilidad.

Resumen

En esta sección, se describen tres enfoques para identificar los deterioros en una sola zona de disponibilidad. Cada enfoque debe usarse en conjunto para proporcionar una visión holística del estado de su carga de trabajo.

El enfoque de alarma CloudWatch compuesta permite detectar problemas en los que el sesgo en la disponibilidad no es significativo desde el punto de vista estadístico, por ejemplo, con una disponibilidad del 98% (la zona de disponibilidad afectada), del 100% y del 99,99%, que no se debe a un único recurso compartido.

La detección de valores atípicos permitirá detectar los deterioros de una sola zona de disponibilidad donde se produzcan errores no correlacionados en varias zonas de disponibilidad que superen el umbral de alarma.

Por último, identificar la degradación de un recurso zonal de una sola instancia permite descubrir cuándo un deterioro de la zona de disponibilidad afecta a un recurso que se comparte entre varias zonas de disponibilidad.

Las alarmas resultantes de cada uno de estos patrones se pueden combinar en una jerarquía de alarmas CloudWatch compuesta para detectar cuándo se producen deficiencias en una sola zona de disponibilidad que afectan a la disponibilidad o la latencia de la carga de trabajo.

Patrones de evacuación de la zona de disponibilidad

Tras detectar el impacto en una única zona de disponibilidad, el siguiente paso es evacuar esa zona de disponibilidad. Hay dos resultados que debe lograr la evacuación.

En primer lugar, desea dejar de enviar trabajo a la zona de disponibilidad afectada. Esto puede significar cosas distintas en diferentes arquitecturas. En una carga de trabajo de solicitud/respuesta, significa impedir que objetos como las solicitudes HTTP o gRPC procedentes de sus clientes se envíen al equilibrador de carga o a otros recursos de la zona de disponibilidad. En un sistema de procesamiento por lotes o de procesamiento de colas, significa impedir que los recursos informáticos procesen ningún trabajo en la zona de disponibilidad afectada. También deberá evitar que los recursos de las zonas de disponibilidad no afectadas interactúen con los recursos de la zona de disponibilidad afectada, por ejemplo, una instancia EC2 que envíe tráfico a un punto de conexión de VPC de la interfaz en la zona de disponibilidad afectada o que se conecte a la instancia principal de una base de datos.

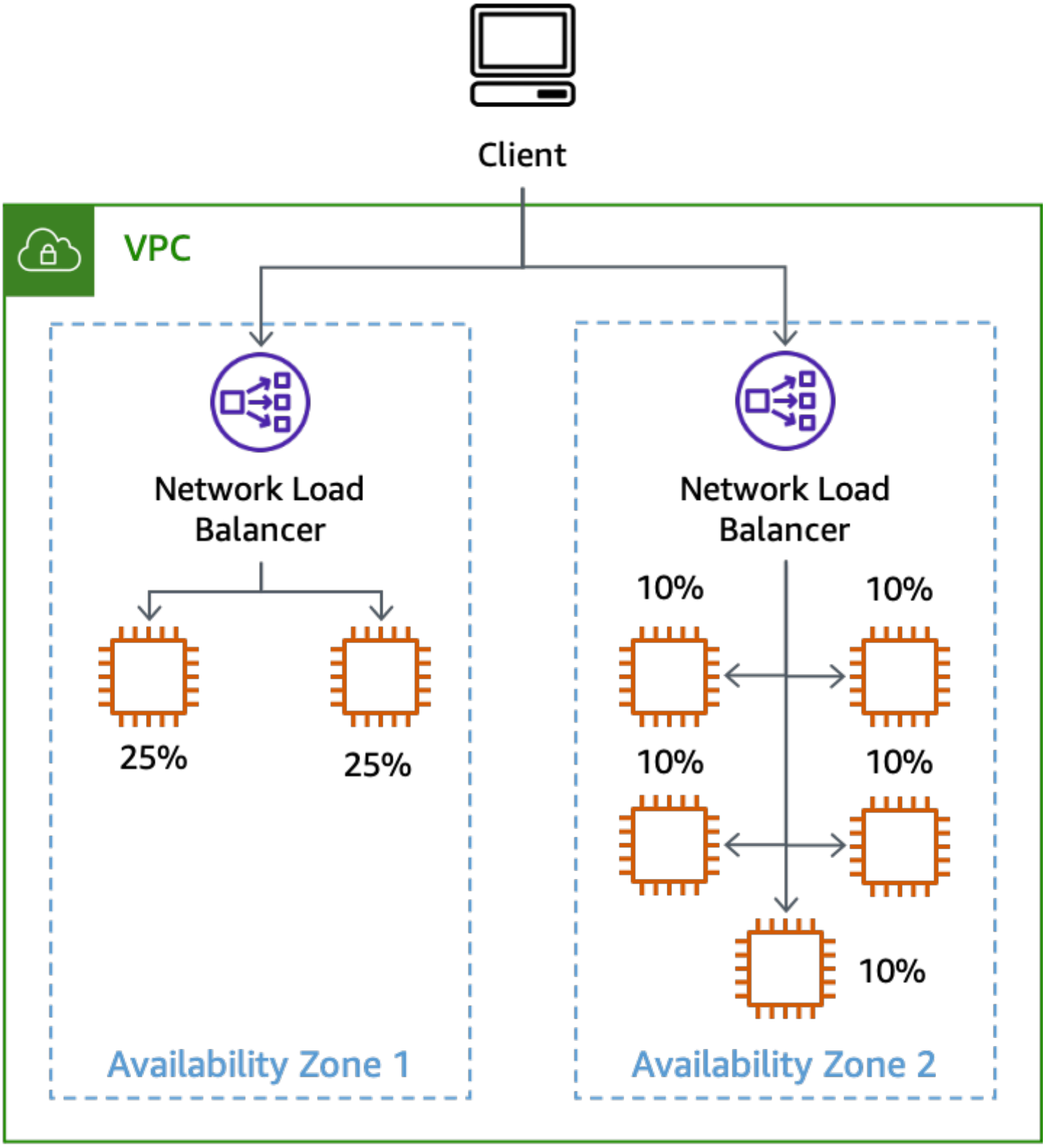
El segundo resultado es impedir que se aprovisione nueva capacidad en la zona de disponibilidad afectada. Esto es importante porque es probable que los nuevos recursos, como contenedores o instancias EC2, que se aprovisionen en la zona de disponibilidad afectada sufran el mismo impacto que los recursos existentes. Además, dado que el primer resultado impide que se les envíe el trabajo, no pueden absorber la carga que se les había aprovisionado. Esto provoca un aumento de la carga de los recursos existentes, lo que, en última instancia, puede provocar una degradación o una pérdida total de disponibilidad de la carga de trabajo. Hay varios servicios de escalado automático disponibles en AWS donde esto es aplicable: Amazon EC2 Auto Scaling, Application Auto Scaling y AWS Auto Scaling. Además, servicios como Amazon ECS, Amazon EKS y AWS Batch pueden programar el trabajo en los hosts de todas las zonas de disponibilidad de una VPC como parte de su funcionamiento normal.

Temas

Independencia de la zona de disponibilidad

Para lograr el primer resultado, dejar de enviar trabajo a la zona de disponibilidad afectada, la evacuación requiere la implementación de la Independencia de la zona de disponibilidad (AZI), también denominada a veces Afinidad de la zona de disponibilidad. Este patrón arquitectónico aísla los recursos dentro de una zona de disponibilidad e impide la interacción entre los recursos de distintas zonas de disponibilidad excepto cuando es absolutamente necesario como, por ejemplo, la conexión a una instancia de base de datos principal en una zona de disponibilidad diferente.

En una carga de trabajo de tipo solicitud/respuesta, la implementación de AZI requiere deshabilitar el equilibrio de carga entre zonas para los Equilibradores de carga de aplicación (ALB), los Equilibradores de carga clásicos (CLB) y los Equilibradores de carga de red (NLB) (el equilibrio de carga entre zonas está deshabilitado de forma predeterminada para los NLB). La deshabilitación del equilibrio de carga entre zonas tiene algunas desventajas. Al deshabilitar el equilibrio de carga entre zonas, el tráfico se divide equitativamente entre cada zona de disponibilidad, independientemente del número de instancias que haya en cada una. Si tiene recursos o grupos de escalado automático desequilibrados, esto puede suponer una carga adicional para los recursos de una zona de disponibilidad que tenga menos recursos que otras. Esto se ilustra en la siguiente figura, donde dos instancias de la zona de disponibilidad 1 reciben cada una el 25 % de la carga y las cinco instancias de la zona de disponibilidad 2 reciben cada una el 10 % de la carga.



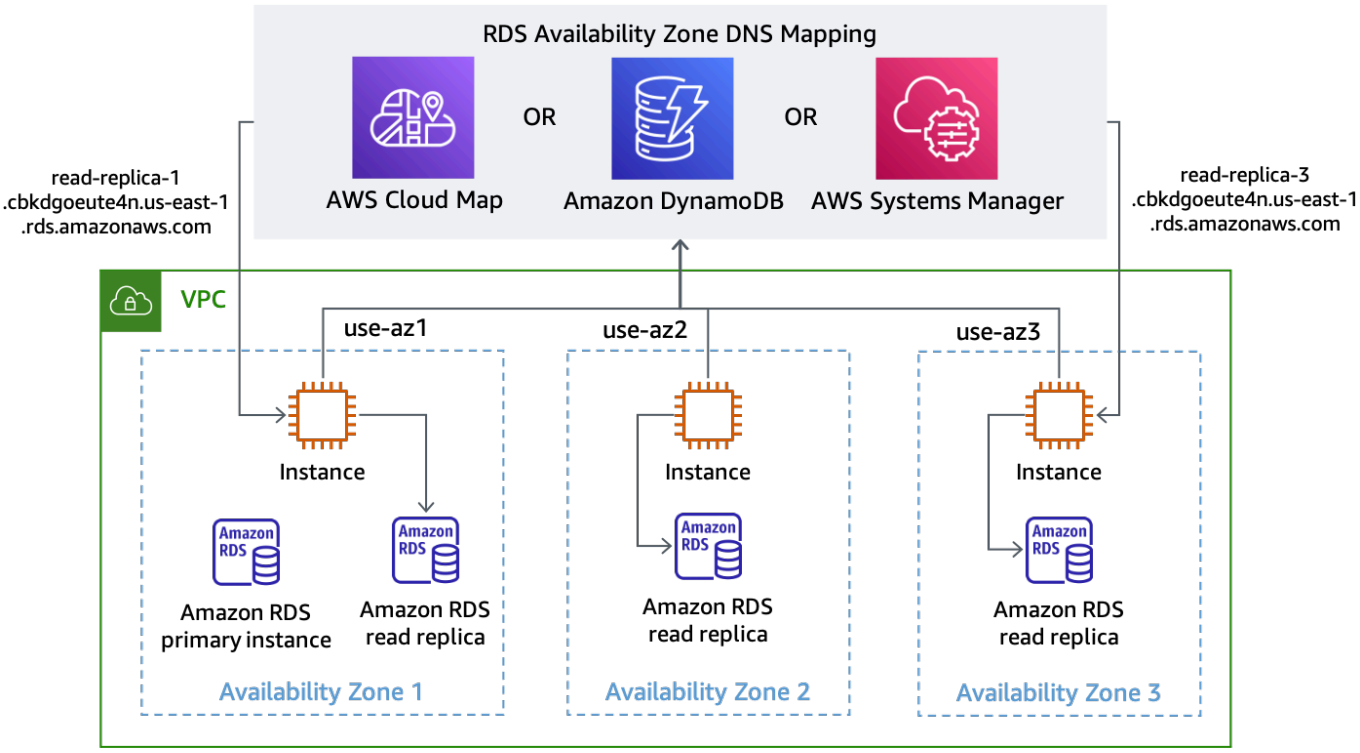
El efecto de deshabilitar el equilibrio de carga entre zonas con instancias desequilibradas

Otros servicios zonales que utilice también deberán implementarse utilizando patrones AZI para facilitar la evacuación efectiva de las zonas de disponibilidad. Por ejemplo, los puntos de conexión de VPC de la interfaz proporcionan nombres de DNS específicos para cada zona de disponibilidad en la que está disponible el punto de conexión de la interfaz.

Uno de los desafíos de la implementación de AZI es el de las bases de datos, especialmente porque la mayoría de las bases de datos relacionales solo admiten un único escritor principal en cualquier momento. Al comunicarse con la instancia principal, es posible que deba cruzar un límite de zona de disponibilidad. Muchos servicios de bases de datos de AWS admiten una configuración Multi-AZ definida por el usuario y tienen una característica de conmutación por error Multi-AZ integrada como, por ejemplo, Amazon RDS o Amazon Aurora. En muchos escenarios de error, el servicio puede detectar el impacto y conmutar por error automáticamente la base de datos a una zona de disponibilidad diferente cuando se produce un problema. Sin embargo, durante un error gris, es posible que el servicio no detecte el impacto que está afectando a la carga de trabajo o que el impacto no esté relacionado en absoluto con la base de datos. En estos casos, una vez que detecte el impacto en una zona de disponibilidad, puede invocar manualmente una conmutación por error para mover la base de datos principal. Esto permite reaccionar de forma eficaz a un deterioro de una sola zona de disponibilidad.

Si utiliza réplicas de lectura con esas bases de datos, puede que también desee implementar AZI para ellas, ya que no puede realizar la conmutación por error de una réplica de lectura a una zona de disponibilidad diferente, como ocurre con la base de datos principal. Si tiene una réplica de lectura única en la zona de disponibilidad 1 y las instancias de tres zonas de disponibilidad están configuradas para utilizarla, un deterioro que afecte a la zona de disponibilidad 1 también afectará a las operaciones en las otras dos zonas de disponibilidad. Este es el impacto que desea evitar.

En el caso de las instancias de RDS, recibirá un punto de conexión de DNS para acceder a la réplica en una zona de disponibilidad específica. Para lograr la AZI, necesitará una réplica de lectura por zona de disponibilidad y una forma de que su aplicación sepa qué punto de conexión de réplica debe utilizar en la zona de disponibilidad en la que se encuentra. Un enfoque que puede adoptar es usar el ID de la zona de disponibilidad como parte del identificador de la base de datos, por ejemplo, `use1-az1-read-replica.cbkdgoeute4n.us-east-1.rds.amazonaws.com`. También puede hacerlo utilizando la detección de servicios (por ejemplo, con AWS Cloud Map) o buscando un mapa sencillo almacenado en el Almacén de parámetros de AWS Systems Manager o en una tabla de DynamoDB. Este concepto se muestra en la siguiente figura.

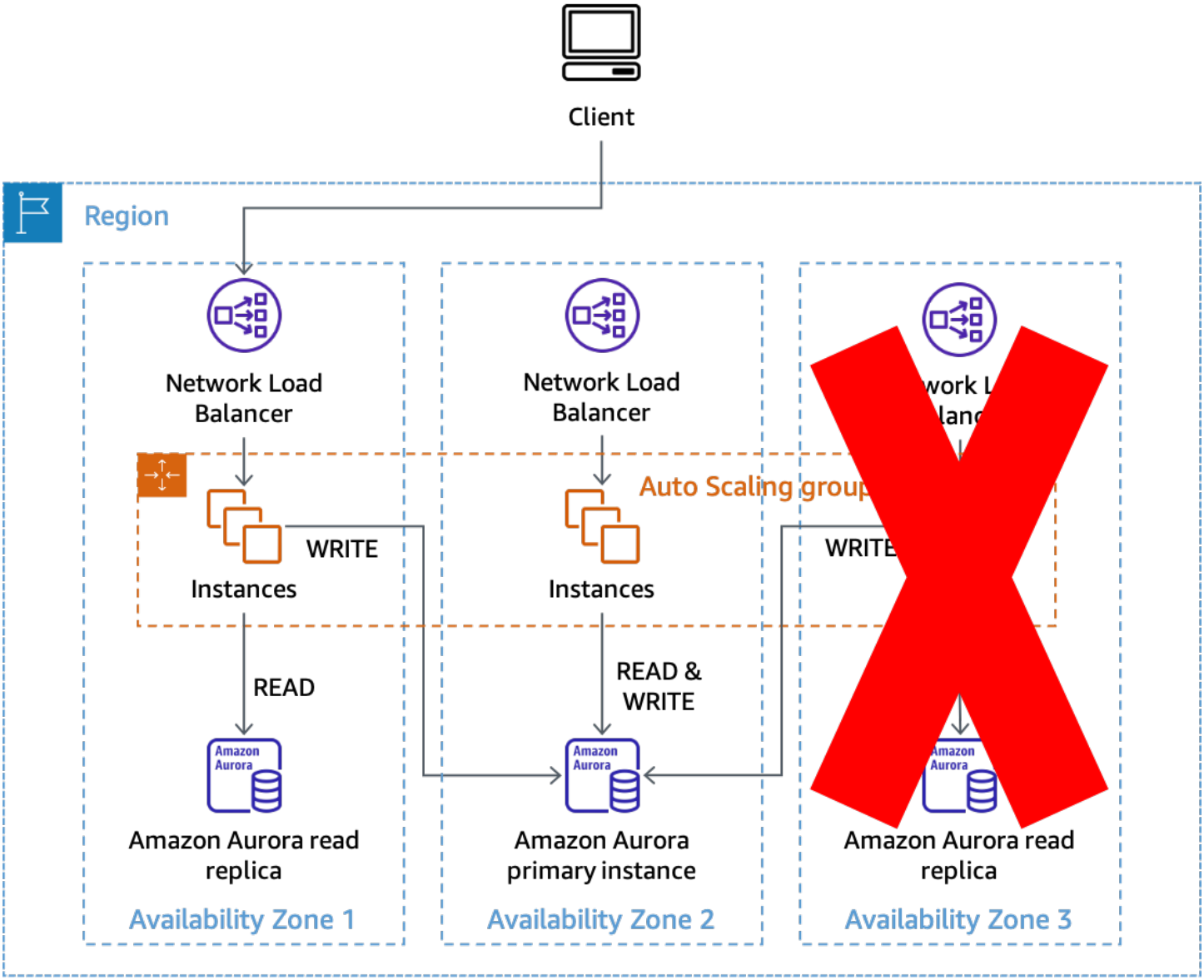


Descubrimiento de los nombres de DNS de los puntos de conexión de RDS para cada zona de disponibilidad

La configuración predeterminada de Amazon Aurora consiste en proporcionar un único punto de conexión de lectura que equilibre la carga de las solicitudes entre las réplicas de lectura disponibles. Para implementar AZI con Aurora, puede usar un punto de conexión personalizado para cada réplica de lectura que utilice el tipo ANY (para que pueda promover una réplica de lectura si es necesario). Asigne un nombre al punto de conexión personalizado en función del ID de la zona de disponibilidad donde se implementa la réplica. A continuación, puede usar el nombre de DNS proporcionado por el punto de conexión personalizado para conectarse a una réplica de lectura específica en una zona de disponibilidad específica, como se muestra en la siguiente figura.

Uso de un punto de conexión personalizado para una réplica de lectura de Aurora

Cuando el sistema está diseñado de esta manera, la evacuación de la zona de disponibilidad es una tarea mucho más sencilla. Por ejemplo, en la siguiente figura, cuando se produce un deterioro que afecta a la zona de disponibilidad 3, las operaciones de lectura y escritura en las zonas de disponibilidad 1 y 2 no se ven afectadas.



Uso de AZI para evitar impactos con réplicas de lectura de Amazon Aurora

Como alternativa, si la zona de disponibilidad 2 se viera afectada, las operaciones de lectura seguirían siendo satisfactorias en las zonas de disponibilidad 1 y 3. Posteriormente, si Amazon Aurora no ha realizado una conmutación por error automática en la base de datos principal, puede invocar manualmente una conmutación por error a una zona de disponibilidad diferente para restablecer la capacidad de procesar escrituras. Este enfoque evita tener que realizar cambios de configuración en las conexiones de la base de datos cuando necesite evacuar una zona de disponibilidad. Minimizar los cambios necesarios y mantener el proceso lo más simple posible hará que sea más fiable.

Planos de control y planos de datos

Antes de analizar los patrones reales que se pueden utilizar para realizar una evacuación de una zona de disponibilidad, debemos analizar los conceptos de planos de control y planos de datos. AWS hace una distinción entre planos de control y planos de datos en nuestros servicios. Los planos de control son la maquinaria necesaria para realizar cambios en un sistema (añadir recursos, eliminar recursos, modificar recursos) y hacer que esos cambios se propaguen donde sea necesario para que surtan efecto, por ejemplo, para actualizar la configuración de red de un ALB o crear una función de AWS Lambda.

Los planos de datos son la función principal de esos recursos, por ejemplo, la ejecución de la instancia EC2 o la obtención de elementos de una tabla de Amazon DynamoDB o su colocación en ella. Para ver un análisis más detallado de los planos de control y los planos de datos, consulte Estabilidad estática con zonas de disponibilidad y Límites de aislamiento de errores de AWS.

A efectos de este documento, tenga en cuenta que los planos de control suelen tener más partes móviles y dependencias que los planos de datos. Esto hace que sea estadísticamente más probable que el plano de control se deteriore en comparación con el plano de datos. Esto es especialmente importante en el caso de los servicios que proporcionan AZI como, por ejemplo, Amazon EC2 y EBS, ya que algunas partes de estos servicios tienen planos de control que también son independientes de la zona y pueden verse afectados durante un evento de zona de disponibilidad única.

Si bien las acciones del plano de control se pueden utilizar para realizar la evacuación de una zona de disponibilidad, según la información anterior, es posible que tengan menos probabilidades de éxito, especialmente en un evento de error. Para aumentar la probabilidad de mitigar el impacto con éxito, puede utilizar dos patrones diferentes. El primer patrón se basa únicamente en las acciones del plano de datos para mitigar inicialmente el impacto al evitar que el trabajo se enrute o impedir que el trabajo se ejecute en la zona de disponibilidad afectada. El segundo patrón consiste en actualizar la configuración de los recursos con acciones del plano de control para evitar que se aprovisione capacidad en la zona de disponibilidad afectada y detener la comunicación entre zonas de disponibilidad con esa zona de disponibilidad.

Los patrones de recuperación que se describen en esta sección son como interruptores generales. Son los mecanismos que se utilizan para tomar medidas a gran escala con rapidez, como tirar de un cable Andon en una cadena de montaje. Suponen que las cargas de trabajo han probado estrategias como volver a intentarlo con un retroceso exponencial con fluctuaciones en el código para superar los errores transitorios. Esto significa que cuando se detecta un impacto aislado en una zona de

disponibilidad, sus efectos en la disponibilidad o la latencia son lo suficientemente graves como para que sea necesario evacuar la zona de disponibilidad para mitigarlos de forma eficaz.

Evacuación controlada por un plano de datos

Hay varias soluciones que puede implementar para llevar a cabo una evacuación de una zona de disponibilidad utilizando acciones únicamente basadas en un plano de datos. En esta sección, se describen tres de ellas y los casos de uso donde deberá elegir una en lugar de otra.

Al utilizar cualquiera de estas soluciones, debe asegurarse de tener suficiente capacidad en las zonas de disponibilidad restantes para manejar la carga de la zona de disponibilidad de la que va a salir. La forma más resiliente de hacerlo es aprovisionando previamente la capacidad requerida en cada zona de disponibilidad. Si utiliza tres zonas de disponibilidad, tendrá implementada en cada una el 50 % de la capacidad necesaria para manejar los picos de carga, de modo que con la pérdida de una única zona de disponibilidad seguiría quedándole el 100 % de la capacidad necesaria sin tener que depender de un plano de control para aprovisionar más.

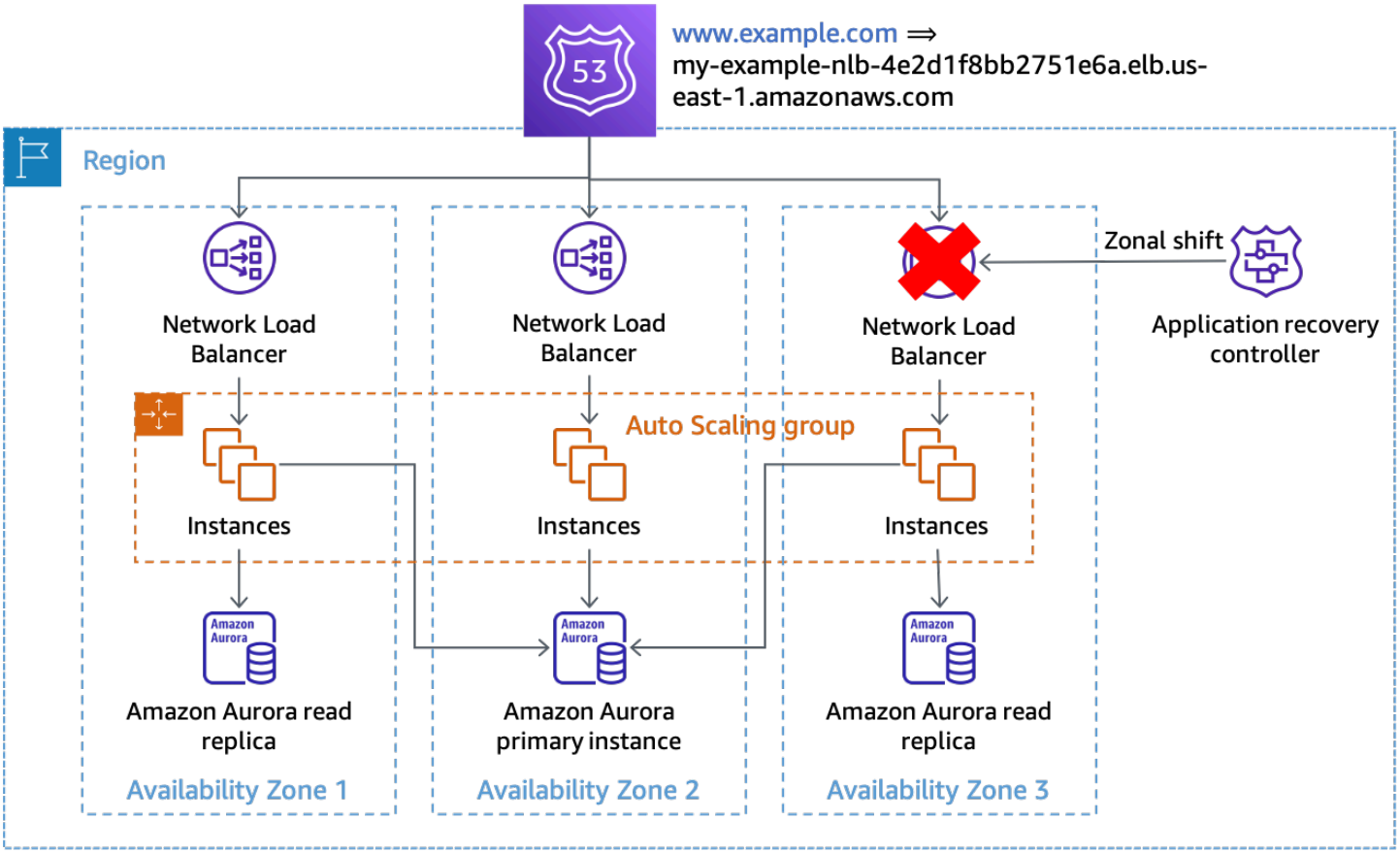
Además, si utiliza EC2 Auto Scaling, asegúrese de que su grupo de escalado automático (ASG) no se reduzca horizontalmente durante el cambio, para que cuando finalice el cambio todavía tenga suficiente capacidad en el grupo para manejar el tráfico de los clientes. Para ello, asegúrese de que la capacidad mínima deseada de su ASG pueda manejar su carga de clientes actual. También puede asegurarse de que su ASG no se reduzca horizontalmente de manera inadvertida utilizando promedios en las métricas en lugar de métricas de percentiles atípicos, como P90 o P99.

Durante un cambio, los recursos que ya no prestan servicio al tráfico deberían tener un uso muy bajo, pero los demás recursos aumentarán su utilización con el nuevo tráfico, manteniendo el promedio bastante uniforme, lo que impedirá adoptar medidas de escalamiento. Por último, también puede usar la configuración de estado de grupo de destino para ALB y NLB, para especificar la conmutación por error de DNS con un porcentaje o un recuento de hosts en buen estado. Esto evita que el tráfico se enrute a una zona de disponibilidad que no tiene suficientes hosts en buen estado.

Cambio de zona en Controlador de recuperación de aplicaciones (ARC) de Route 53

La primera solución para la evacuación de zonas de disponibilidad utiliza el cambio de zona en Route 53 ARC. Esta solución se puede utilizar para cargas de trabajo de solicitud/respuesta que utilizan un NLB o un ALB como punto de entrada para el tráfico de los clientes.

Cuando detecte que una zona de disponibilidad se ha deteriorado, puede iniciar un cambio de zona con el Route 53 ARC. Una vez que se complete esta operación y caduquen las respuestas de DNS almacenadas en caché, todas las solicitudes nuevas solo se enrutarán a los recursos de las zonas de disponibilidad restantes. En la figura siguiente, se muestra cómo funciona el cambio de zona. En la siguiente figura, tenemos un registro de alias de Route 53 para `www.example.com` que apunta a `my-example-nlb-4e2d1f8bb2751e6a.elb.us-east-1.amazonaws.com`. El cambio de zona se realiza para la zona de disponibilidad 3.



Cambio de zona

En el ejemplo, si la instancia de base de datos principal no se encuentra en la zona de disponibilidad 3, la única acción necesaria para lograr el primer resultado de evacuación es realizar el cambio de zona y evitar que el trabajo se procese en la zona de disponibilidad afectada. Si el nodo principal estaba en la zona de disponibilidad 3, puede realizar una conmutación por error iniciada manualmente (que depende del plano de control de Amazon RDS) en coordinación con el cambio de zona, si Amazon RDS no ha realizado ya la conmutación por error automática. Esto se aplicará a todas las soluciones controladas por el plano de datos de esta sección.

Debe iniciar el cambio de zona utilizando los comandos de CLI o la API para minimizar las dependencias necesarias para iniciar la evacuación. Cuanto más simple sea el proceso de evacuación, más fiable será. Los comandos específicos se pueden almacenar en un manual de procedimientos local al que los ingenieros de guardia pueden acceder fácilmente. El cambio de zona es la solución preferida y más sencilla para evacuar una zona de disponibilidad.

Route 53 ARC

La segunda solución utiliza las capacidades de Route 53 ARC para especificar manualmente el estado de determinados registros de DNS. Esta solución tiene la ventaja de utilizar el plano de datos del clúster de Route 53 ARC de alta disponibilidad, lo que la hace resiliente al deterioro de hasta dos Regiones de AWS diferentes. Tiene la desventaja de un costo adicional y requiere una configuración adicional de los registros de DNS. Para implementar este patrón, debe crear registros de alias para los nombres de DNS específicos de la zona de disponibilidad proporcionados por el equilibrador de carga (ALB o NLB). Esto se muestra en la siguiente tabla.

Tabla 3: Registros de alias de Route 53 configurados para los nombres de DNS zonales del equilibrador de carga

Política de enrutamiento: ponderada	Política de enrutamiento: ponderada	Política de enrutamiento: ponderada
Nombre: `www.example.com`	Nombre: `www.example.com`	Nombre: `www.example.com`
Tipo: `A` (alias)	Tipo: `A` (alias)	Tipo: `A` (alias)
Valor: `us-east-1b.load-balancer-name.elb.us-east-1.amazonaws.com`	Valor `us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com`	Valor `us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`
Peso: `100`	Peso: `100`	Peso: `100`
Evaluar el estado del destino: true	Evaluar el estado del destino: `true`	Evaluar el estado del destino: `true`

Para cada uno de estos registros de DNS, debe configurar una comprobación de estado de Route 53 asociada a un control de enrutamiento de Route 53 ARC. Cuando desee iniciar una evacuación de una zona de disponibilidad, establezca el estado del control de enrutamiento en `Off`. AWS recomienda hacerlo utilizando la CLI o la API para minimizar las dependencias necesarias para iniciar la evacuación de la zona de disponibilidad. Como práctica recomendada, debe conservar una copia local de los puntos de conexión del clúster de Route 53 ARC para no tener que recuperarlos del plano de control de ARC cuando necesite realizar una evacuación.

Para minimizar los costos al utilizar este enfoque, puede crear un único clúster de Route 53 ARC y realizar comprobaciones de estado en una sola Cuenta de AWS y compartir las comprobaciones de estado con otras Cuentas de AWS de su organización. Al adoptar este enfoque, debe utilizar el ID de zona de disponibilidad (AZ-ID) (por ejemplo, `use1-az1`) en lugar del nombre de la zona de disponibilidad (por ejemplo, `us-east-1a`) para sus controles de enrutamiento. Como AWS asigna la zona de disponibilidad física de forma aleatoria a los nombres de las zonas de disponibilidad de cada Cuenta de AWS, el uso del AZ-ID proporciona una forma coherente de hacer referencia a las mismas ubicaciones físicas. Al iniciar una evacuación de una zona de disponibilidad, por ejemplo, para `use1-az2`, los conjuntos de registros de Route 53 en cada Cuenta de AWS deberían asegurarse de utilizar la asignación de AZ-ID para configurar la comprobación de estado adecuada para cada registro de NLB.

Por ejemplo, supongamos que tenemos una comprobación de estado de Route 53 asociada a un control de enrutamiento de Route 53 ARC para `use1-az2`, con un ID de `0385ed2d-d65c-4f63-a19b-2412a31ef431`. Si en una Cuenta de AWS diferente que desea utilizar esta comprobación de estado, `us-east-1c` se ha asignado a `use1-az2`, deberá utilizar la comprobación de estado de `use1-az2` para el registro `us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`. Deberá utilizar el ID de comprobación de estado `0385ed2d-d65c-4f63-a19b-2412a31ef431` con ese conjunto de registros de recursos.

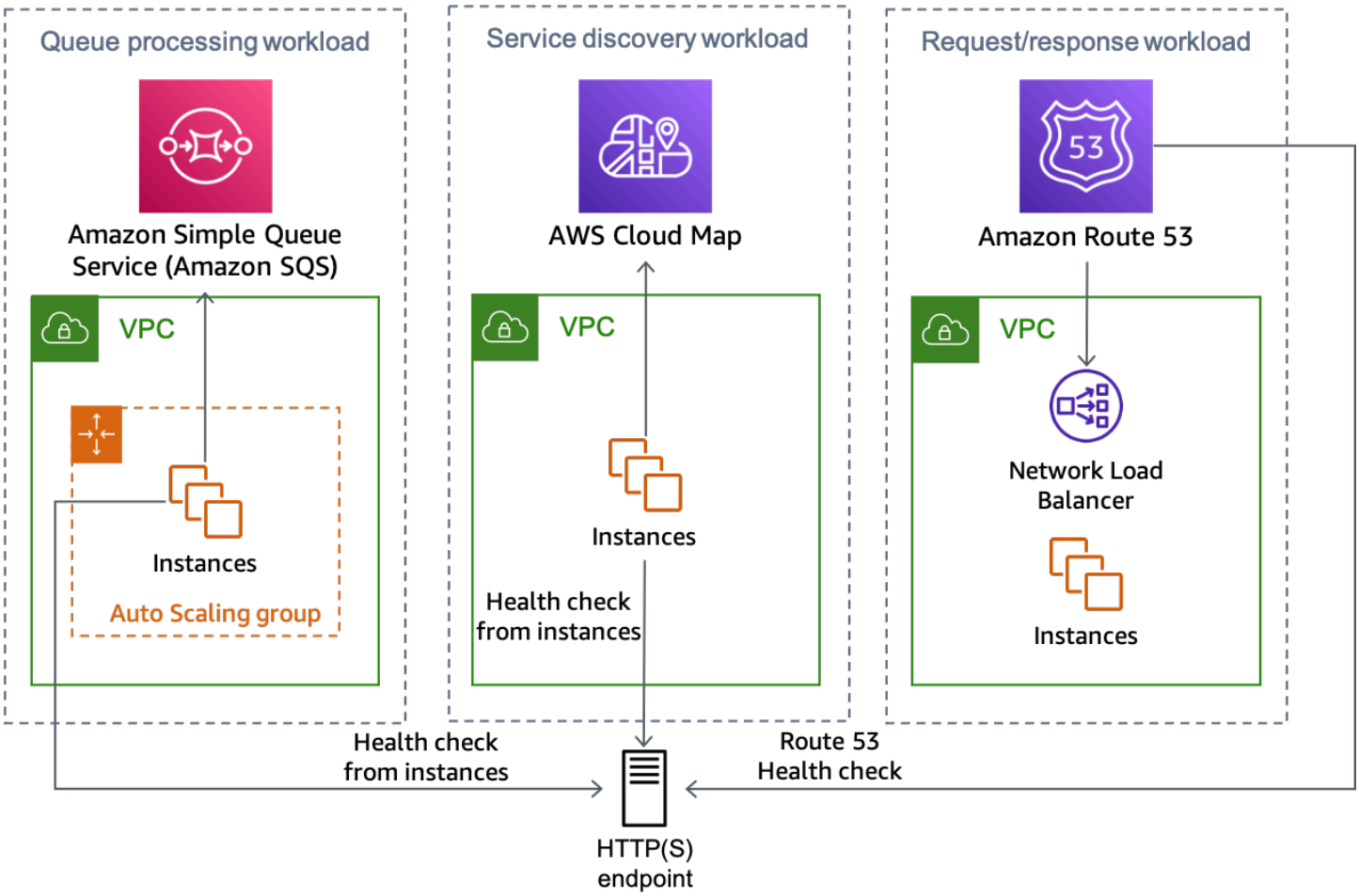
Uso de un punto de conexión HTTP autoadministrado.

También puede implementar esta solución administrando su propio punto de conexión HTTP que indica el estado de una zona de disponibilidad concreta. Esto permite especificar manualmente cuándo una zona de disponibilidad tiene un estado incorrecto en función de la respuesta del punto de conexión HTTP. Esta solución tiene un costo menor que usar Route 53 ARC, pero es más costosa que el cambio de zona y requiere la administración de una infraestructura adicional. Tiene la ventaja de ser mucho más flexible para diferentes escenarios.

El patrón se puede usar con las arquitecturas NLB o ALB y con las comprobaciones de estado de Route 53. También se puede usar en arquitecturas sin equilibrio de carga, como los sistemas de detección de servicios o procesamiento de colas, donde los nodos de trabajo realizan sus propias comprobaciones de estado. En estos casos, los hosts pueden usar un subproceso en segundo plano donde realizan periódicamente una solicitud al punto de conexión HTTP con su AZ-ID (consulte Apéndice A: Obtener el ID de la zona de disponibilidad para obtener más información sobre cómo encontrarlo) y reciben una respuesta sobre el estado de la zona de disponibilidad.

Si se ha declarado que la zona de disponibilidad tiene un estado incorrecto, tienen varias opciones para responder. Pueden optar por no pasar una comprobación de estado externa procedente de fuentes como ELB o Route 53, o comprobaciones de estado personalizadas en las arquitecturas de detección de servicios para que parezcan tener un estado incorrecto para esos servicios. También pueden responder inmediatamente con un error si reciben una solicitud, lo que permite al cliente retroceder y volver a intentarlo. En las arquitecturas basadas en eventos, los nodos pueden dejar de procesar el trabajo de forma intencionada, por ejemplo, devolviendo intencionadamente un mensaje de SQS a la cola. En las arquitecturas de enrutadores de trabajo donde un servicio central programa el trabajo en hosts específicos, también se puede utilizar este patrón. El enrutador puede comprobar el estado de una zona de disponibilidad antes de seleccionar un trabajador, un punto de conexión o una celda. En las arquitecturas de detección de servicios que utilizan AWS Cloud Map, puede detectar los puntos de conexión proporcionando un filtro en la solicitud, por ejemplo, un AZ-ID.

En la siguiente figura, se muestra cómo se puede utilizar este enfoque para varios tipos de cargas de trabajo.



Varios tipos de carga de trabajo que pueden utilizar la solución de punto de conexión HTTP

Hay varias formas de implementar el enfoque de punto de conexión HTTP, dos de las cuales se describen a continuación.

Uso de Amazon S3

Este patrón se presentó originalmente en esta entrada de blog sobre la recuperación de desastres en varias regiones. Puede usar el mismo patrón para la evacuación de una zona de disponibilidad.

En este escenario, debe crear conjuntos de registros de recursos de Route 53 DNA para cada registro de DNS zonal, como en el escenario de Route 53 ARC anterior, así como las comprobaciones de estado asociadas. Sin embargo, para esta implementación, en lugar de asociar las comprobaciones de estado con los controles de enrutamiento de Route 53 ARC, se configuran para usar un punto de conexión HTTP y se invierten para evitar que un deterioro en Amazon S3 desencadene accidentalmente una evacuación. La comprobación de estado se considera que tiene

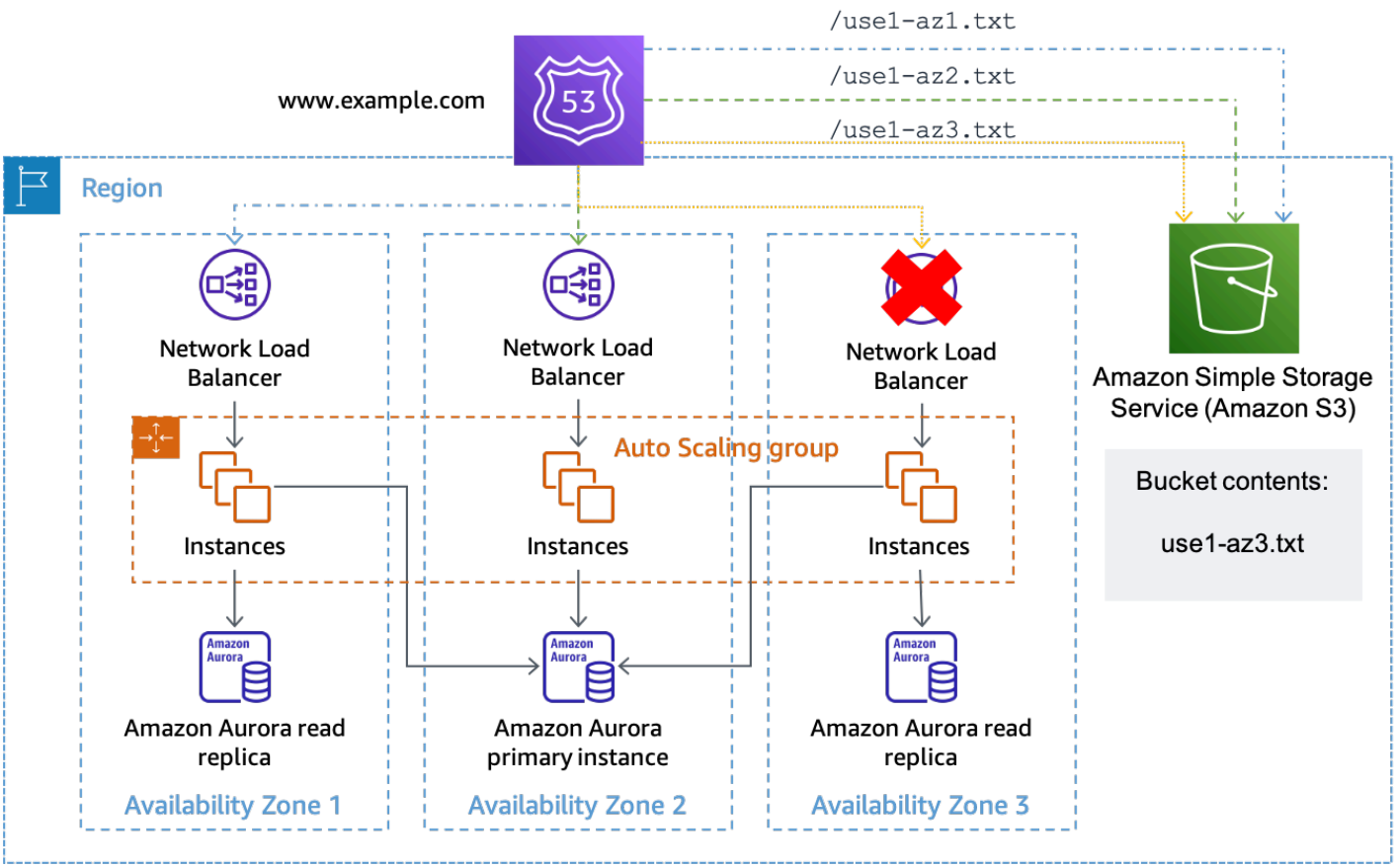
un buen estado cuando el objeto está ausente y un estado incorrecto cuando el objeto está presente. Esta configuración se muestra en la siguiente tabla.

Tabla 4: Configuración de registros de DNS para usar las comprobaciones de estado de Route 53 por zona de disponibilidad

Tipo de comprobación de estado: supervisar un punto de conexión Protocolo: HTTPS ID: `dddd-4444` URL: `https://bucketname.s3.us-east-1.amazonaws.com/use1-az1.txt`	Tipo de comprobación de estado: supervisar un punto de conexión Protocolo: HTTPS ID: `eeee-5555` URL: `https://bucketname.s3.us-east-1.amazonaws.com/use1-az3.txt`	Tipo de comprobación de estado: supervisar un punto de conexión Protocolo: HTTPS ID: `ffff-6666` URL: `https://bucketname.s3.us-east-1.amazonaws.com/use1-az2.txt`	←	Comprobaciones de estado
↑	↑	↑		
Política de enrutamiento: ponderada Nombre: `www.example.com` Tipo: A (alias)	Política de enrutamiento: ponderada Nombre: `www.example.com` Tipo: A (alias)	Política de enrutamiento: ponderada Nombre: `www.example.com` Tipo: A (alias)	←	Los registros A de alias de nivel superior y ponderados uniformemente apuntan a puntos de conexión específicos de NLB AZ

Valor: `us-east-1b.load-balancer-name.elb.us-east-1.amazonaws.com`	Valor: `us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com`	Valor: `us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`
Peso: `100`	Peso: `100`	Peso: `100`
Evaluar el estado del destino: `true`	Evaluar el estado del destino: `true`	Evaluar el estado del destino: `true`

Supongamos que la zona de disponibilidad `us-east-1a` está asignada a `use1-az3` en la cuenta donde tenemos una carga de trabajo y queremos realizar una evacuación de la zona de disponibilidad. Para el conjunto de registros de recursos creado para `us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com`, se asociará una comprobación de estado que compruebe la URL`https://bucket-name.s3.us-east-1.amazonaws.com/use1-az3.txt`. Cuando desee iniciar una evacuación de una zona de disponibilidad para `use1-az3`, cargue un archivo con el nombre `use1-az3.txt` en el bucket mediante la CLI o la API. No es necesario que el archivo contenga ningún contenido, pero sí que debe ser público para que la comprobación de estado de Route 53 pueda acceder a él. En la siguiente figura, se muestra cómo se utiliza esta implementación para evacuar `use1-az3`.



Uso de Amazon S3 como destino para una comprobación de estado de Route 53

Uso de API Gateway y DynamoDB

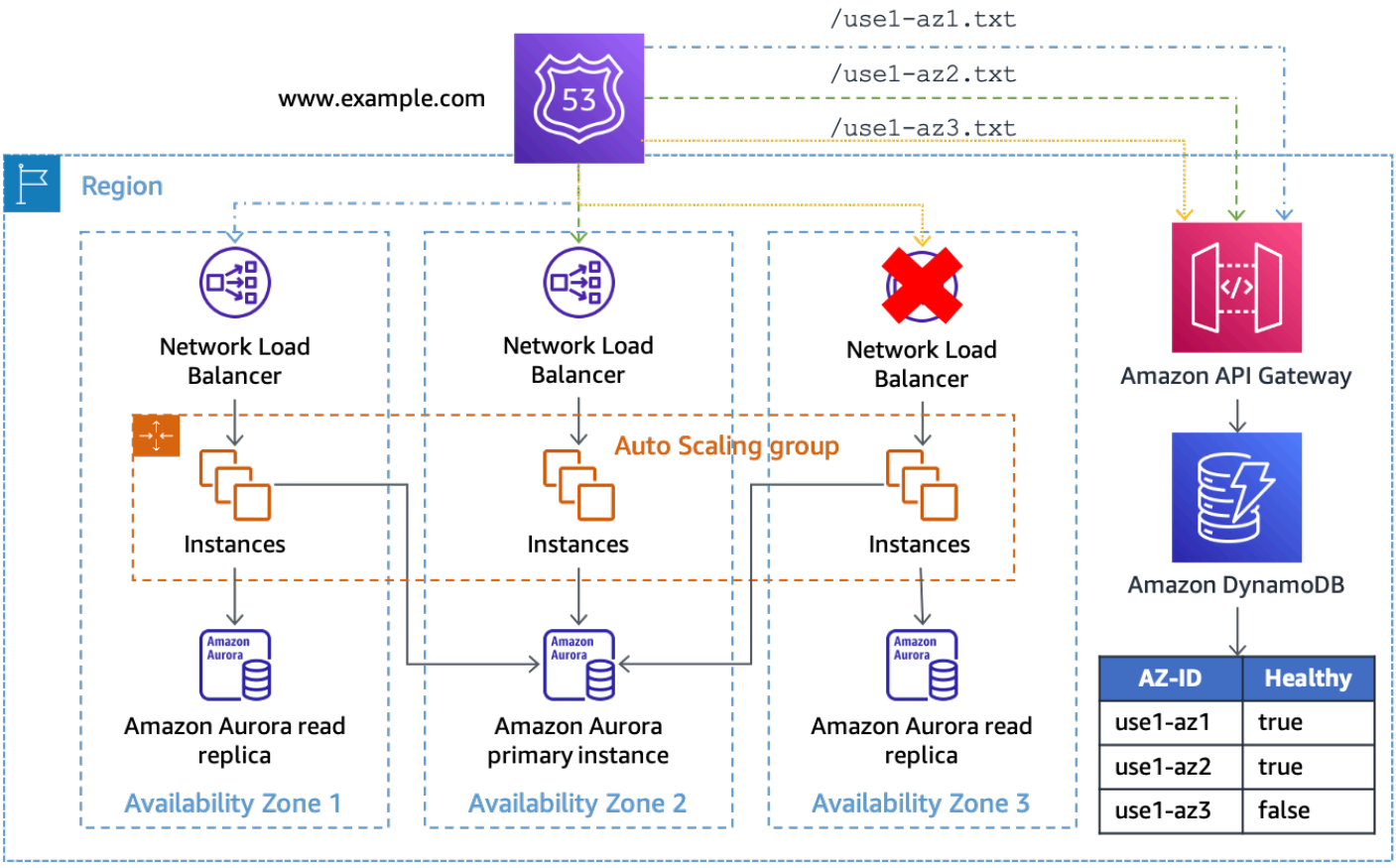
La segunda implementación de este patrón usa una API de REST de Amazon API Gateway. La API se configura con una integración de servicios en Amazon DynamoDB, donde se almacena el estado de cada zona de disponibilidad en uso. Esta implementación es más flexible que el enfoque de Amazon S3, pero requiere crear, utilizar y supervisar más infraestructuras. También se puede usar tanto con las comprobaciones de estado de Route 53 como con las comprobaciones de estado realizadas por hosts individuales.

Si utiliza esta solución con una arquitectura NLB o ALB, configure sus registros de DNS de la misma manera que en el ejemplo anterior de Amazon S3, excepto que debe cambiar la ruta de comprobación de estado para utilizar el punto de conexión de API Gateway y proporcionar el `AZ-ID` en la URL. Por ejemplo, si API Gateway se ha configurado con un dominio personalizado `az-status.example.com`, la solicitud completa de `use1-az1` aparecerá como `https://az-status.example.com/status/use1-az1`. Cuando desee iniciar una evacuación de una zona de disponibilidad, puede crear o actualizar un elemento de DynamoDB mediante la CLI o la API. El

elemento utiliza el `AZ-ID` como clave principal y tiene un atributo booleano denominado `Healthy` que se utiliza para indicar cómo responde API Gateway. A continuación, se muestra un ejemplo de código utilizado en la configuración de API Gateway para realizar esta determinación.

```
#set($inputRoot = $input.path('$'))
#if ($inputRoot.Item.Healthy['BOOL'] == (false))
    #set($context.responseOverride.status = 500)
#end
```

Si el atributo es `true` (o no está presente), API Gateway responde a la comprobación de estado con un HTTP 200; si es false, responde con un HTTP 500. Esta implementación se muestra en la siguiente figura.



Uso de API Gateway y DynamoDB como el destino de las comprobaciones de estado de Route 53

En esta solución, debe usar API Gateway delante de DynamoDB para poder hacer que el punto de conexión sea de acceso público y manipular la URL de la solicitud para convertirla en una solicitud `GetItem` de DynamoDB. La solución también ofrece flexibilidad si desea incluir datos adicionales en la solicitud. Si desea crear estados más detallados, por ejemplo, por aplicación, puede configurar

la URL de comprobación de estado para proporcionar un ID de aplicación en la ruta o cadena de consulta que también coincida con el elemento de DynamoDB.

El punto de conexión de estado de la zona de disponibilidad se puede implementar de forma centralizada para que varios recursos de comprobación de estado en las Cuentas de AWS puedan utilizar la misma vista coherente del estado de la zona de disponibilidad (lo que garantiza que la API de REST de API Gateway y la tabla de DynamoDB estén escaladas para manejar la carga) y eliminar la necesidad de compartir las comprobaciones de estado de Route 53.

La solución también se puede escalar en varias Regiones de AWS mediante una tabla global de Amazon DynamoDB y una copia de la API de REST de API Gateway en cada región. Esto evita que la solución dependa de una sola región y aumenta su disponibilidad. Puede implementar la solución en tres o cinco regiones, y consultar en cada una de ellas el estado de las zonas de disponibilidad, utilizando el resultado de la mayoría de los puntos de conexión para garantizar el quórum. En última instancia, esto permite replicar las actualizaciones de manera coherente en la tabla global, así como mitigar los deterioros que puedan impedir que un punto de conexión responda. Por ejemplo, si utiliza cinco regiones y tres puntos de conexión indican que una zona de disponibilidad tiene un estado incorrecto, un punto de conexión indica que la zona de disponibilidad tiene un estado correcto y un punto de conexión no responde, deberá tratar la zona de disponibilidad como con un estado incorrecto. También puede crear una comprobación de estado calculada por Route 53 utilizando un cálculo de m de n para aplicar esta lógica y determinar el estado de la zona de disponibilidad.

Si está creando una solución para que los hosts individuales la utilicen como un mecanismo para determinar el estado de su zona de disponibilidad, como alternativa, en lugar de proporcionar un mecanismo de extracción para las comprobaciones de estado, puede utilizar las notificaciones push. Una forma de hacerlo es con un tema de SNS al que estén suscritos los consumidores. Cuando desee desencadenar el interruptor de circuito, publique un mensaje en el tema de SNS en el que se indique qué zona de disponibilidad está deteriorada. Este enfoque presenta algunas contrapartidas respecto al anterior. Elimina la necesidad de crear y operar la infraestructura de API Gateway, y ejecutar la administración de la capacidad. También puede proporcionar una convergencia más rápida del estado de la zona de disponibilidad. Sin embargo, elimina la posibilidad de realizar consultas ad hoc y se basa en la Política de reintentos de entrega de SNS para garantizar que cada punto de conexión reciba la notificación. También requiere que cada carga de trabajo o servicio cree una forma de recibir la notificación de SNS y tomar medidas al respecto.

Por ejemplo, cada nueva instancia o contenedor de EC2 que se lance deberá suscribirse al tema con un punto de conexión HTTP durante su arranque. A continuación, cada instancia debe implementar un software que escuche en el punto de conexión donde se entrega la notificación. Asimismo,

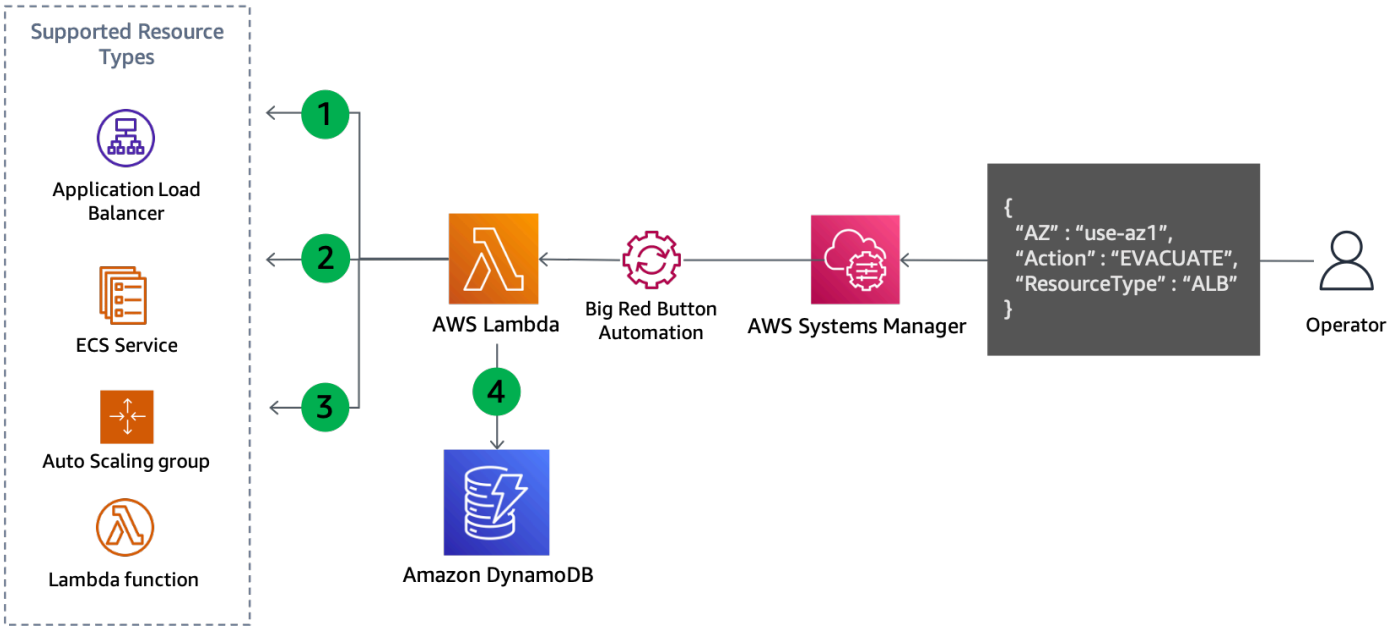
si la instancia se ve afectada por el evento, es posible que no reciba la notificación push y siga funcionando. Por el contrario, con una notificación pull, la instancia sabrá si su solicitud de extracción falla y podrá elegir qué acción realizar como respuesta.

Una segunda forma de enviar notificaciones push es con conexiones de WebSocket de larga duración. Amazon API Gateway se puede utilizar para proporcionar una API de WebSocket a la que los consumidores puedan conectarse y recibir un mensaje cuando lo envíe el backend. Con un WebSocket, las instancias pueden realizar extracciones periódicas para garantizar que su conexión esté en buen estado y también recibir notificaciones push de baja latencia.

Evacuación controlada por plano de control

El primer patrón utiliza las operaciones del plano de datos para evitar realizar trabajos en una zona de disponibilidad afectada y mitigar el impacto de un evento. Sin embargo, es posible que esté utilizando una arquitectura que no utilice equilibradores de carga o donde no sea posible configurar una comprobación de estado por cada host. O bien, puede que desee evitar que se despliegue nueva capacidad en la zona de disponibilidad afectada mediante el escalado automático o con una programación de trabajo normal.

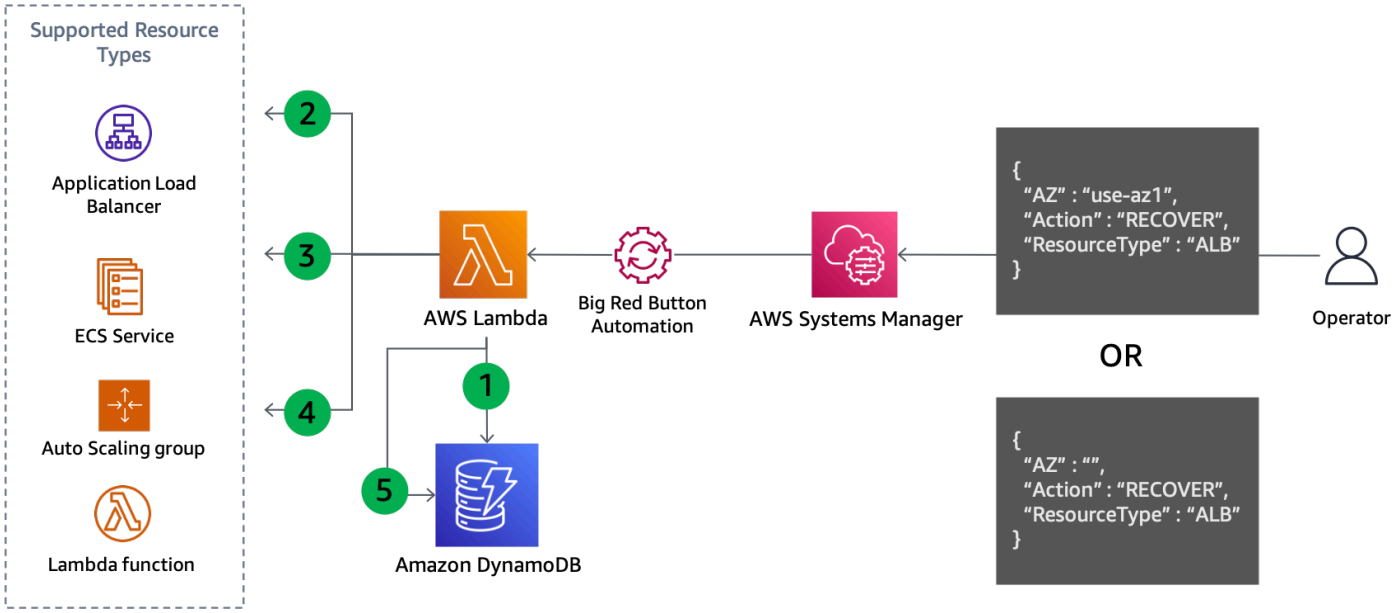
Para abordar ambas situaciones, es necesario realizar acciones en el plano de control para actualizar la configuración del recurso. El patrón funcionará para cualquier servicio cuya configuración de red se pueda actualizar, por ejemplo, EC2 Auto Scaling, Amazon ECS, Lambda, etc. Requiere escribir código para cada servicio, pero la lógica empresarial sigue un patrón estándar. El código debe estar ejecutado localmente por un operador que responda al evento para minimizar las dependencias requeridas. El flujo básico de la lógica del script se muestra en la figura siguiente.



Actualización del plano de control para evacuar una zona de disponibilidad

1. El script muestra todos los recursos del tipo especificado, por ejemplo, el grupo de escalado automático, el servicio de ECS o la función de Lambda, y recupera sus subredes a partir de la información del recurso. Los recursos compatibles dependen de la compatibilidad que haya configurado el script.
2. Determina qué subredes deben eliminarse comparando el nombre de la zona de disponibilidad de cada subred con su ID de zona de disponibilidad asignado que se ha proporcionado como parámetro de entrada.
3. La configuración de red del recurso se actualiza para eliminar las subredes identificadas.
4. Los detalles de la actualización se registran en una tabla de DynamoDB. El ID de la zona de disponibilidad se almacena como la clave de partición y el ARN o el nombre del recurso se almacena como la clave de clasificación. Las subredes que se eliminan se almacenan como una matriz de cadenas. Por último, el tipo de recurso también se almacena y se usa como una clave hash para un índice secundario global (GSI).

Como en el paso cuatro se registran las actualizaciones realizadas, este enfoque también puede invertirse fácilmente cuando esté listo para la recuperación, como se muestra en la siguiente figura.



Actualización del plano de control para recuperarse de la evacuación de una zona de disponibilidad

Pasos de recuperación:

1. Consulte el GSI para eliminar las subredes de cada recurso del tipo especificado en la zona de disponibilidad especificada (o todas las zonas de disponibilidad, si no se especifica ninguna).
2. Describa cada recurso encontrado en la consulta de DynamoDB para obtener su configuración de red actual.
3. Combine las subredes de la configuración de red actual con las recuperadas de la consulta de DynamoDB.
4. Actualice la configuración de red del recurso con el nuevo conjunto de subredes.
5. Elimine el registro de la tabla de DynamoDB una vez que la actualización se haya completado correctamente.

Este patrón generalizado impide enrutar el trabajo a la zona de disponibilidad afectada e impide que se implemente ninguna nueva capacidad en ella. Los siguientes son ejemplos de cómo se logra esto para diferentes servicios.

- Lambda: actualice la configuración de VPC de la función para eliminar las subredes de la zona de disponibilidad especificada.
- Grupo de escalado automático: elimine las subredes de la configuración de ASG que sustituirán esa capacidad en las zonas de disponibilidad restantes.

- Amazon ECS: actualice la configuración de la VPC del servicio de ECS para eliminar las subredes.
- Amazon EKS: aplique taints a los nodos de la zona de disponibilidad afectada para expulsar los pods existentes y evitar que se programen nuevos pods en ella.

Cada servicio reaccionará de forma diferente a la actualización de la configuración. Por ejemplo, Amazon ECS seguirá la configuración de implementación del servicio después de una actualización y desencadenará una implementación continua o una implementación azul/verde de nuevas tareas.

Estas actualizaciones pueden trasladar el trabajo a las zonas de disponibilidad en buen estado con demasiada rapidez para algunas cargas de trabajo. Aunque está configurado para permanecer estable de forma estática ante el error (con suficiente capacidad aprovisionada previamente en las zonas de disponibilidad restantes para manejar el trabajo de la zona de disponibilidad afectada), es posible que también desee eliminar gradualmente capacidad de la zona de disponibilidad afectada.

Si tiene previsto actualizar la configuración de red de su grupo de escalado automático que es el grupo objetivo de un equilibrador de carga con el equilibro de cargas entre zonas deshabilitado, siga estas instrucciones.

El escalado automático reacciona a este cambio utilizando su lógica de reequilibrio de zonas de disponibilidad. Lanzará instancias en las demás zonas de disponibilidad para cumplir con la capacidad deseada y terminará las instancias en la zona de disponibilidad que haya eliminado. Sin embargo, el equilibrador de carga seguirá dividiendo el tráfico de manera uniforme en cada zona de disponibilidad, incluida la que ha eliminado del ASG, mientras se cancelen las instancias. Esto podría provocar una disminución de la capacidad restante en esa zona de disponibilidad hasta que todas las instancias se terminen correctamente en esa zona. Este es el mismo problema que se describe en Independencia de la zona de disponibilidad en relación con el desequilibrio de las zonas de disponibilidad cuando el equilibrio de carga entre zonas está deshabilitado. Para evitar que esto ocurra, puede hacer lo siguiente:

- Ejecute siempre primero la evacuación de la zona de disponibilidad para que el tráfico solo se divida entre las zonas de disponibilidad restantes
- Especifique un número mínimo de destinos con estado correcto con una conmutación por error de DNS que coincida con el número mínimo de destinos requerido para esa zona de disponibilidad.

Esto ayudará a garantizar que el tráfico no se envíe a la zona de disponibilidad que ha eliminado una vez que las instancias comiencen a terminarse.

Resumen

En la tabla siguiente, se resumen las ventajas y desventajas de los patrones de evacuación descritos.

Tabla 5: Ventajas y desventajas del patrón de evacuación

Enfoque	Ventajas	Desventajas
Evacuación controlada por un plano de datos	Se basa únicamente en las acciones del plano de datos Evita rápidamente que el trabajo se realice en la zona de disponibilidad afectada Un enfoque flexible para una visión centralizada del estado de la zona de disponibilidad	No impide que la capacidad se implemente en una zona de disponibilidad afectada No todos los tipos de carga de trabajo pueden utilizar este enfoque fácilmente
Evacuación controlada por plano de control	Impide que se implemente una nueva capacidad en la zona de disponibilidad afectada Elimina la capacidad existente de la zona de disponibilidad afectada	Se basa en el plano de control de cada servicio Requiere que se escriba un código para cada servicio Debe completarse servicio por servicio Debe tener cuidado de no sobrecargar la capacidad durante la actualización

Es probable que combine ambos enfoques como parte de un plan de evacuación de la zona de disponibilidad. Comience por las acciones de evacuación controladas por un plano de datos que tengan más probabilidades de éxito para detener rápidamente el trabajo de procesamiento en la zona de disponibilidad afectada. A continuación, una vez que se mitigue el impacto inicial, continúe con las acciones de evacuación controladas por el plano de control, si lo considera necesario.

Conclusión

En este documento, se proporciona una visión general de los errores grises, cómo se manifiestan y por qué es necesario crear herramientas de observabilidad y evacuación para mitigar ese tipo de eventos cuando ocurren. En la siguiente sección, se analiza la observabilidad Multi-AZ y los tres enfoques que puede implementar para detectar el impacto en una sola zona de disponibilidad. En la última sección, este documento incluye dos enfoques generales para realizar la evacuación de una zona de disponibilidad. El primer enfoque utiliza acciones del plano de datos para evitar que el trabajo se dirija a la zona de disponibilidad afectada, mientras que el segundo enfoque utiliza acciones del plano de control para evitar que se aprovisione capacidad en la zona de disponibilidad afectada. En conjunto, estos dos enfoques logran los dos resultados que busca la evacuación de la zona de disponibilidad.

Es probable que los patrones de recuperación descritos en este documento formen parte de una solución más amplia de supervisión y recuperación de errores. Este enfoque para hacer frente a los errores grises en una zona de disponibilidad única requiere trabajos de ingeniería para crear la instrumentación necesaria para detectarlos, así como herramientas para responder a ellos. Sin embargo, para muchas cargas de trabajo, este enfoque puede ser una alternativa más sencilla y menos costosa a la creación de arquitecturas multirregionales. Además, puede ayudar a lograr RPO y RTO más pequeños (lo que aumenta la disponibilidad de la carga de trabajo) en comparación con la DR multirregional.

Apéndice A: Obtener el ID de la zona de disponibilidad

Si utiliza el SDK de AWS .NET (además de otros, como JavaScript) o ejecuta el sistema en una instancia EC2 (por ejemplo, Amazon ECS y Amazon EKS), puede obtener el ID de zona de disponibilidad directamente.

- SDK de AWS .NET

```
Amazon.Util.EC2InstanceMetadata.GetData(“/placement/availability-zone-id”)
```

- Servicio de metadatos de instancias EC2

```
curl http://169.254.169.254/latest/meta-data/placement/availability-zone-id
```

En otras plataformas, como Lambda y Fargate, necesitará recuperar el nombre de la zona de disponibilidad y, a continuación, buscar la asignación al ID de zona de disponibilidad. Con el nombre de la zona de disponibilidad, puede encontrar el ID de la zona de disponibilidad de la siguiente manera:

```
aws ec2 describe-availability-zones --zone-names $AZ --output json
    --query ‘AvailabilityZones[0].ZoneId’
```

Los siguientes ejemplos para encontrar el nombre de la zona de disponibilidad que se va a utilizar en el ejemplo anterior están escritos en bash utilizando la AWS CLI y el paquete jq. Deberán convertirse al lenguaje de programación utilizado para su carga de trabajo.

- Amazon ECS: si el host bloquea el servicio de metadatos de instancias (IMDS), puede utilizar en su lugar el archivo de metadatos del contenedor.

```
AZ=$(cat $ECS_CONTAINER_METADATA_FILE | jq --raw-output
    .AvailabilityZone)
```

- Fargate (versión 1.4 de la plataforma o posterior)

```
AZ=$(curl $ECS_CONTAINER_METADATA_URI_V4/task | jq --raw-output
    .AvailabilityZone)
```

- Lambda: la zona de disponibilidad no está expuesta directamente a la función. Para encontrarla, debe realizar varios pasos. Para ello, necesitará crear un punto de conexión REST de API Gateway privado que devuelva la dirección IP del solicitante. Esto identificará la IP privada asignada a la interfaz de red elástica que utiliza la función.
  - Llame a la API `GetFunction` de Lambda para buscar el ID de VPC de la función.
  - Llame al servicio de API Gateway para obtener la IP de la función.
  - Con la IP y el ID de VPC, busque la interfaz de red asociada y extraiga la zona de disponibilidad.

```
VPC_ID=$(aws lambda get-function --function-name $ AWS_LAMBDA_FUNCTION_NAME --
region $AWS_REGION --output json --query ‘Configuration.VpcConfig.VpcId’)

MY_IP=$(curl http://whats-my-private-ip.internal)

AZ=$(aws ec2 describe-network-interfaces --filters Name=private-ip-address,Values=
$MY_IP Name=vpc-id,Values=$VPC_ID --region $AWS_REGION --output json –query
 ‘NetworkInterfaces[0].AvailabilityZone’)
```

Apéndice B: Ejemplo de cálculo con chi cuadrado

El siguiente es un ejemplo de recopilación de métricas de error y realización de una prueba de chi cuadrado en los datos. El código no está listo para la producción y no realiza el manejo de errores necesario, pero proporciona una prueba de concepto sobre el funcionamiento de la lógica. Debe actualizar este ejemplo para adaptarlo a sus necesidades.

En primer lugar, un evento programado de Amazon EventBridge invoca una función de Lambda cada minuto. El contenido del evento se configura con los siguientes datos:

```
{
  "timestamp": "2023-03-15T15:26:37.527Z",
  "namespace": "multi-az/frontend",
  "metricName": "5xx",
  "dimensions": [
    { "Name": "Region", "Value": "us-east-1" },
    { "Name": "Controller", "Value": "Home" },
    { "Name": "Action", "Value": "Index" }
  ],
  "period": 60,
  "stat": "Sum",
  "unit": "Count",
  "chiSquareMetricName": "multi-az/chi-squared",
  "azs": [ "use1-az2", "use1-az4", "use1-az6" ]
}
```

Los datos se utilizan para especificar los datos comunes necesarios para recuperar las métricas de CloudWatch adecuadas (como el espacio de nombres, el nombre de la métrica y las dimensiones) y, a continuación, publicar los resultados de chi cuadrado de cada zona de disponibilidad. El código de la función de Lambda tiene el siguiente aspecto con Python 3.9. En un nivel superior, recopila las métricas de CloudWatch especificadas del minuto anterior, ejecuta la prueba de chi cuadrado con esos datos y, a continuación, publica las métricas de CloudWatch sobre el resultado de la prueba para cada zona de disponibilidad especificada.

```
import os
import boto3
import datetime
import copy
import json
```

```
from datetime import timedelta
from scipy.stats import chisquare
from aws_embedded_metrics import metric_scope

cw_client = boto3.client("cloudwatch", os.environ.get("AWS_REGION", "us-east-1"))

@metric_scope
def handler(event, context, metrics):
    metrics.set_property("Event", json.loads(json.dumps(event, default = str)))
    time = datetime.datetime.strptime(event["timestamp"], "%Y-%m-%dT%H:%M:%S.%fZ")

    # Round down to the previous minute
    end: datetime = roundTime(time)

    # Subtract a minute for the start
    start: datetime = end - timedelta(minutes = 1)

    # Get all the metrics that match the query
    results = get_all_metrics(event, start, end, metrics)
    metrics.set_property("MetricCounts", results)

    # Calculate the chi squared result
    chi_sq_result = chisquare(list(results.values()))
    expected = sum(list(results.values())) / len(results.values())
    metrics.set_property("ChiSquaredResult", chi_sq_result)

    # Put the chi square metrics into CloudWatch
    put_all_metrics(event, results, chi_sq_result[1], expected, start, metrics)

def get_all_metrics(detail: dict, start: datetime, end: datetime, metrics):
    """
    Gets all of the error metrics for each AZ specified
    """
    metric_query = {
        "MetricDataQueries": [
        ],
        "StartTime": start,
        "EndTime": end
    }

    for az in detail["azs"]:

        dim = copy.deepcopy(detail["dimensions"])
        dim.append({"Name": "AZ-ID", "Value": az})
```

```
        query = {
            "Id": az.replace("-", "_"),
            "MetricStat": {
                "Metric": {
                    "Namespace": detail["namespace"],
                    "MetricName": detail["metricName"],
                    "Dimensions": dim
                },
                "Period": int(detail["period"]),
                "Stat": detail["stat"],
                "Unit": detail["unit"]
            },
            "Label": az,
            "ReturnData": True
        }

        metric_query["MetricDataQueries"].append(query)

    metrics.set_property("GetMetricRequest", json.loads(json.dumps(metric_query,
default=str)))
    next_token: str = None
    results = {}

    while True:
        if next_token is not None:
            metric_query["NextToken"] = next_token

        data = cw_client.get_metric_data(**metric_query)

        if next_token is not None:
            metrics.set_property("GetMetricResult::" + next_token,
json.loads(json.dumps(data, default = str)))
        else:
            metrics.set_property("GetMetricResult", json.loads(json.dumps(data, default
= str)))

        for item in data["MetricDataResults"]:
            key = item["Id"].replace("_", "-")
            if key not in results:
              results[key] = 0

            results[key] += sum(item["Values"])
```

```
        if "NextToken" in data:
            next_token = data["NextToken"]

        if next_token is None:
            break

    return results

def put_all_metrics(detail: dict, results: dict, chi_sq_value: float, expected: float,
 timestamp: datetime, metrics):
    """
    Adds the chi squared metric for all AZs to CloudWatch
    """
    farthest_from_expected = None
    if len(results) > 0:
        keys = list(results.keys())
        farthest_from_expected = keys[0]

        for key in keys:
            if abs(results[key] - expected) > abs(results[farthest_from_expected] -
 expected):
                farthest_from_expected = key

    metric_query = {
        "Namespace": detail["namespace"],
        "MetricData": []
    }

    for az in detail["azs"]:
        dim = copy.deepcopy(detail["dimensions"])
        dim.append({"Name": "AZ-ID", "Value": az})

        query = {
            "MetricName": detail["chiSquareMetricName"],
            "Dimensions": dim,
            "Timestamp": timestamp,
        }

        if chi_sq_value <= 0.05 and az == farthest_from_expected:
            query["Value"] = 1
        else:
            query["Value"] = 0

        metric_query["MetricData"].append(query)
```

```
    metrics.set_property("PutMetricRequest", json.loads(json.dumps(metric_query,
 default = str)))

    cw_client.put_metric_data(**metric_query)

def roundTime(dt=None, roundTo=60):
   """Round a datetime object to any time lapse in seconds
   dt : datetime.datetime object, default now.
   roundTo : Closest number of seconds to round to, default 1 minute.
   """
   if dt == None : dt = datetime.datetime.now()
   seconds = (dt.replace(tzinfo=None) - dt.min).seconds
   rounding = (seconds+roundTo/2) // roundTo * roundTo
   return dt + datetime.timedelta(0,rounding-seconds,-dt.microsecond)
```

A continuación, puede crear una alarma por zona de disponibilidad. El siguiente ejemplo corresponde a `use1-az2` y emite una alarma para tres puntos de datos seguidos de un minuto con un valor máximo igual a 1 (1 es la métrica que se publica cuando la prueba de chi cuadrado determina un sesgo estadísticamente significativo en la tasa de error).

```
{
    "Type": "AWS::CloudWatch::Alarm",
    "Properties": {
        "AlarmName": "use1-az2-chi-squared",
        "ActionsEnabled": true,
        "OKActions": [],
        "AlarmActions": [],
        "InsufficientDataActions": [],
        "MetricName": "multi-az/chi-squared",
        "Namespace": "multi-az/frontend",
        "Statistic": "Maximum",
        "Dimensions": [
            {
                "Name": "AZ-ID",
                "Value": "use1-az2"
            },
            {
                "Name": "Action",
                "Value": "Index"
            },
```

```
            {
                "Name": "Region",
                "Value": "us-east-1"
            },
            {
                "Name": "Controller",
                "Value": "Home"
            }
        ],
        "Period": 60,
        "EvaluationPeriods": 3,
        "DatapointsToAlarm": 3,
        "Threshold": 1,
        "ComparisonOperator": "GreaterThanOrEqualToThreshold",
        "TreatMissingData": "missing"
    }
}
```

También puede crear una alarma tipo “m de n” y combinar estas dos alarmas en una alarma compuesta. También deberá crear las mismas alarmas para cada microservicio o combinación de controlador/acción que tenga en cada zona de disponibilidad. Por último, puede añadir la alarma compuesta de chi cuadrado a la alarma específica de la zona de disponibilidad para cada combinación de controlador/acción, como se muestra en Detección de errores utilizando la detección de valores atípicos.

Colaboradores

Los colaboradores de este documento son:

- Michael Haken, arquitecto principal de soluciones, Amazon Web Services

Revisiones del documento

Para recibir notificaciones sobre las actualizaciones de este documento técnico, suscríbase a la fuente RSS.

Cambio	Descripción	Fecha
Documento técnico actualizado	Se ha actualizado con una guía de observabilidad adicional y para utilizar la nueva característica de cambio de zona.	11 de julio de 2023
Publicación inicial	Documento técnico publicado por primera vez.	2 de marzo de 2022

> **Note**
>
> Para suscribirse a las actualizaciones de RSS, debe tener un complemento de RSS habilitado para el navegador que esté utilizando.

Avisos

Es responsabilidad de los clientes realizar su propia evaluación independiente de la información que contiene este documento. El presente documento: (a) tiene solo fines informativos, (b) representa las ofertas y prácticas actuales de los productos de AWS, que están sujetas a cambios sin previo aviso, y (c) no supone ningún compromiso ni garantía por parte de AWS ni sus filiales, proveedores o licenciantes. Los productos o servicios de AWS se proporcionan “tal cual”, sin garantías, declaraciones ni condiciones de ningún tipo, ya sean expresas o implícitas. Las responsabilidades y obligaciones de AWS con respecto a sus clientes se controlan mediante los acuerdos de AWS y este documento no forma parte ni modifica ningún acuerdo entre AWS y sus clientes.

Glosario de AWS

Para ver la terminología más reciente de AWS, consulte el Glosario de AWS en la Referencia de Glosario de AWS.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.