Erkennung und Behebung von Gray-Fehlern

Fortschrittliche Multi-AZ-Resilienzmuster

Fortschrittliche Multi-AZ-Resilienzmuster: Erkennung und Behebung von Gray-Fehlern

Table of Contents

Fortgeschrittene Multi-AZ-Resilienzmuster

Datum der Veröffentlichung:11. Juli 2023(Dokumentversionen)

Viele Kunden führen ihre Workloads in hochverfügbaren Konfigurationen mit Multi-Availability Zone (AZ) aus. Diese Architekturen funktionieren gut bei binären Ausfällen, stoßen jedoch häufig auf Probleme mitgrauMisserfolge. Die Symptome dieser Art von Versagen können subtil sein und sich einer schnellen und eindeutigen Erkennung entziehen. Dieses Whitepaper enthält Anleitungen zur Instrumentierung von Workloads, um die Auswirkungen von Grauausfällen, die auf eine einzelne Availability Zone beschränkt sind, zu erkennen und dann Maßnahmen zu ergreifen, um diese Auswirkungen in der Availability Zone zu mindern.

Einführung

Dieses Dokument soll Ihnen helfen, belastbare Multi-AZ-Architekturen effektiver zu implementieren. Eine der besten Praktiken für den Aufbau widerstandsfähiger Systeme inVirtuelle private Cloud von Amazon(VPC-) Netzwerke sind fürjeden Workload auf mehrere Availability Zones verteilen.

EinVerfügbarkeitszoneist ein oder mehrere diskrete Rechenzentren mit redundanter Stromversorgung, Netzwerk und Konnektivität. Durch die Verwendung mehrerer Availability Zones können Sie Workloads betreiben, die höher verfügbar, fehlertoleranter und skalierbarer sind, als dies von einem einzelnen Rechenzentrum aus möglich wäre.

VieleAWSDienstleistungen, wieAutomatische Skalierung von Amazon Elastic Compute Cloud (EC2)oderAmazon Relational Database Service(Amazon RDS), stellen Sie eine Multi-AZ-Konfiguration bereit. Für diese Dienste müssen Sie keine zusätzlichen Observabilitäts- oder Failover-Tools entwickeln. Sie machen Workloads widerstandsfähig gegenüber leicht erkennbaren binären Fehlermodi innerhalb einerAWS-Regiondie sich auf eine einzelne Availability Zone auswirken. Dies kann ein vollständiger physischer Hardwarefehler, ein Stromausfall oder ein latenter Softwarefehler sein, der einen Großteil der Ressourcen betrifft.

Aber es gibt noch eine andere Kategorie von Fehlern, die alsgraue Ausfälle, deren Erscheinungsformen subtil sind und sich einer schnellen und eindeutigen Entdeckung entziehen. Dies wiederum führt zu längeren Zeiten, um die durch den Ausfall verursachten Auswirkungen zu mildern. Dieses Whitepaper konzentriert sich auf die Auswirkungen, die Grauausfälle auf Multi-AZ-Architekturen haben können, wie sie erkannt werden können und wie sie schließlich gemildert werden können.

Die in diesem Whitepaper enthaltenen Leitlinien beziehen sich hauptsächlich auf bestimmte Klassen von Workloads, die:

- Verwenden Sie in erster Linie zonalAWSDienstleistungen
- Die Widerstandsfähigkeit einer einzelnen Region muss verbessert werden
- sind bereit, erhebliche Investitionen zu tätigen, um die erforderlichen Beobachtbarkeits- und Resilienzmuster zu entwickeln

Bei diesen Workloads sind Sie möglicherweise nicht bereit, einige oder alle der in???, oder haben nicht die Möglichkeit, mehrere Regionen zu verwenden. Diese Arten von Workloads stellen wahrscheinlich nur einen kleinen Teil Ihres Gesamtportfolios dar. Daher sollten diese Leitlinien auf Workload-Ebene und nicht auf Plattformebene betrachtet werden.

Graue Fehler

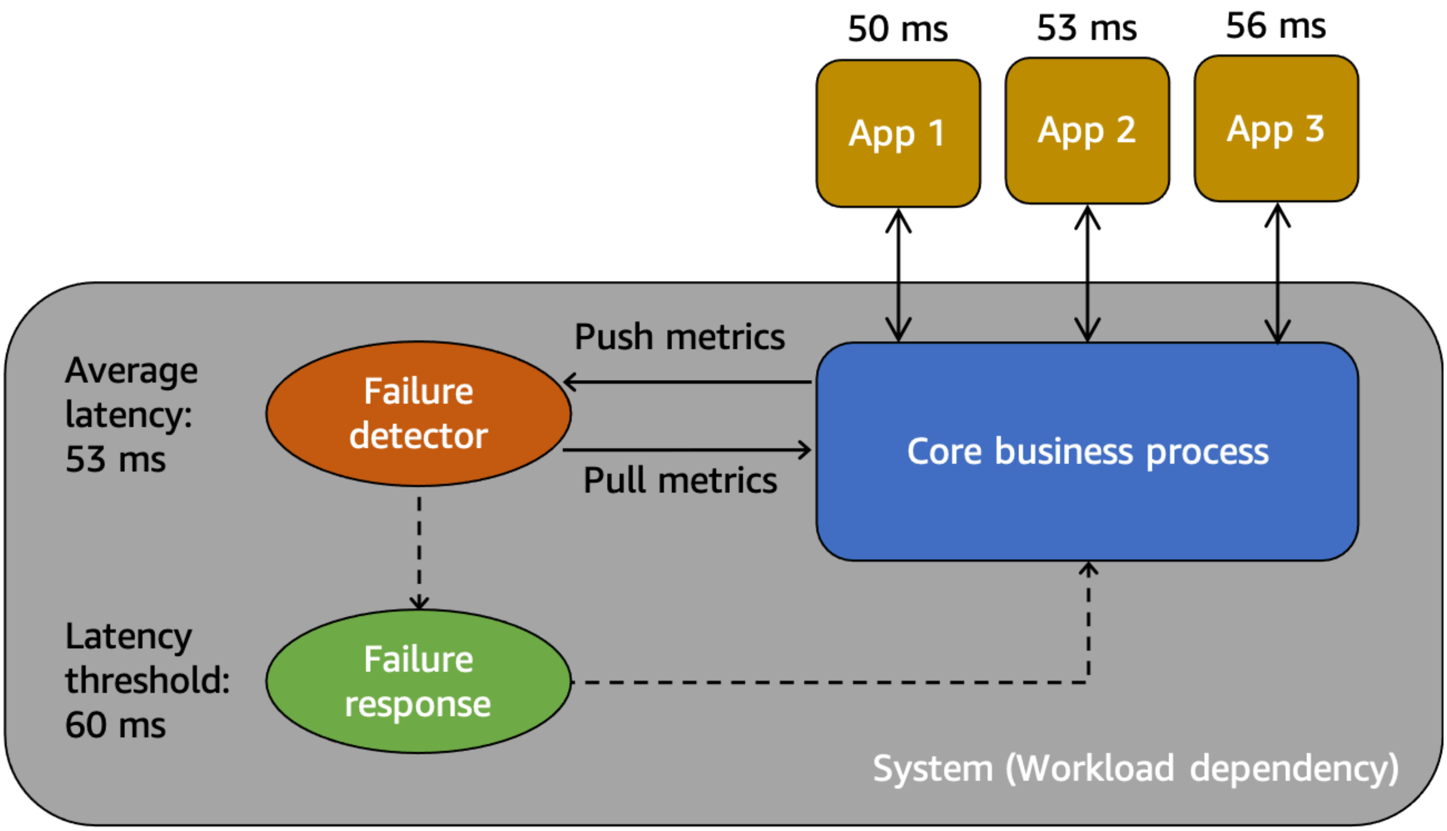
Graue Ausfälle sind definiert durch das Merkmal vondifferentielle Beobachtbarkeit, was bedeutet, dass verschiedene Entitäten den Fehler unterschiedlich beobachten. Lassen Sie uns definieren, was das bedeutet.

Differenzielle Beobachtbarkeit

Die Workloads, die Sie betreiben, haben in der Regel Abhängigkeiten. Dies können zum Beispiel dieAWSCloud-Dienste, die Sie für die Erstellung Ihres Workloads verwenden, oder ein Drittanbieter-Identitätsanbieter (IdP), den Sie für den Verbund verwenden. Diese Abhängigkeiten implementieren fast immer ihre eigene Observability und zeichnen Kennzahlen zu Fehlern, Verfügbarkeit und Latenz auf, die unter anderem durch die Nutzung durch den Kunden generiert werden. Wenn ein Schwellenwert für eine dieser Metriken überschritten wird, ergreift die Abhängigkeit normalerweise Maßnahmen, um ihn zu korrigieren.

Diese Abhängigkeiten haben in der Regel mehrere Nutzer ihrer Dienste. Verbraucher implementieren auch ihre eigene Observability und zeichnen Metriken und Protokolle über ihre Interaktionen mit ihren Abhängigkeiten auf. Sie zeichnen beispielsweise auf, wie viel Latenz bei Festplattenlesevorgängen besteht, wie viele API-Anfragen fehlgeschlagen sind oder wie lange eine Datenbankabfrage gedauert hat.

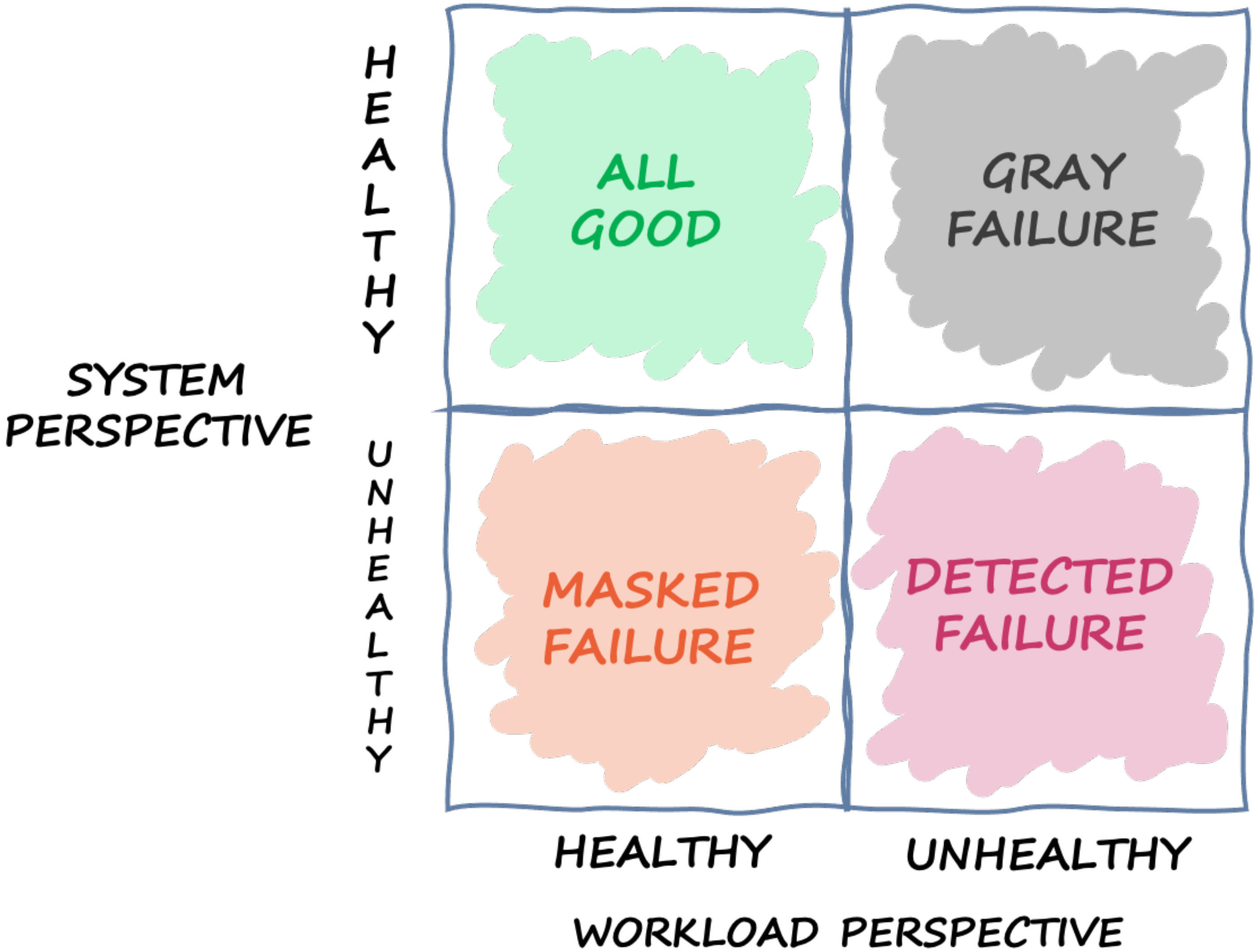
Diese Interaktionen und Messungen sind in der folgenden Abbildung in einem abstrakten Modell dargestellt.



Ein abstraktes Modell zum Verständnis grauer Fehler

Zuerst haben wir dieSystem, was in diesem Szenario eine Abhängigkeit für die Verbraucher App 1, App 2 und App 3 darstellt. Das System verfügt über einen Fehlerdetektor, der Kennzahlen untersucht, die anhand des Kerngeschäftsprozesses erstellt wurden. Es verfügt auch über einen Mechanismus zur Fehlerreaktion, um Probleme zu mildern oder zu korrigieren, die vom Fehlerdetektor beobachtet werden. Das System weist eine durchschnittliche Gesamtlatenz von 53 ms auf und hat einen Schwellenwert festgelegt, um den Fehlerreaktionsmechanismus aufzurufen, wenn die durchschnittliche Latenz 60 ms überschreitet. App 1, App 2 und App 3 machen ebenfalls ihre eigenen Beobachtungen zu ihrer Interaktion mit dem System und zeichnen eine durchschnittliche Latenz von 50 ms, 53 ms bzw. 56 ms auf.

Differenzielle Beobachtbarkeit ist die Situation, in der einer der Systemverbraucher feststellt, dass das System nicht betriebsbereit ist, die systemeigene Überwachung das Problem jedoch nicht erkennt oder die Auswirkungen eine Alarmschwelle nicht überschreiten. Stellen wir uns vor, dass App 1 eine durchschnittliche Latenz von 70 ms statt 50 ms aufweist. App 2 und App 3 stellen keine Veränderung ihrer durchschnittlichen Latenzen fest. Dadurch wird die durchschnittliche Latenz des zugrunde liegenden Systems auf 59,66 ms erhöht, der Latenzschwellenwert zur Aktivierung des Fehlerreaktionsmechanismus wird jedoch nicht überschritten. App 1 verzeichnet jedoch einen Anstieg der Latenz um 40%. Dies könnte sich auf die Verfügbarkeit auswirken, indem das konfigurierte Client-Zeitlimit für App 1 überschritten wird, oder es kann zu kaskadierenden Auswirkungen in einer längeren Kette von Interaktionen kommen. Aus Sicht von App 1 ist das zugrunde liegende System, von dem es abhängt, ungesund, aber aus Sicht des Systems selbst sowie von App 2 und App 3 ist das System gesund. Die folgende Abbildung fasst diese verschiedenen Perspektiven zusammen.



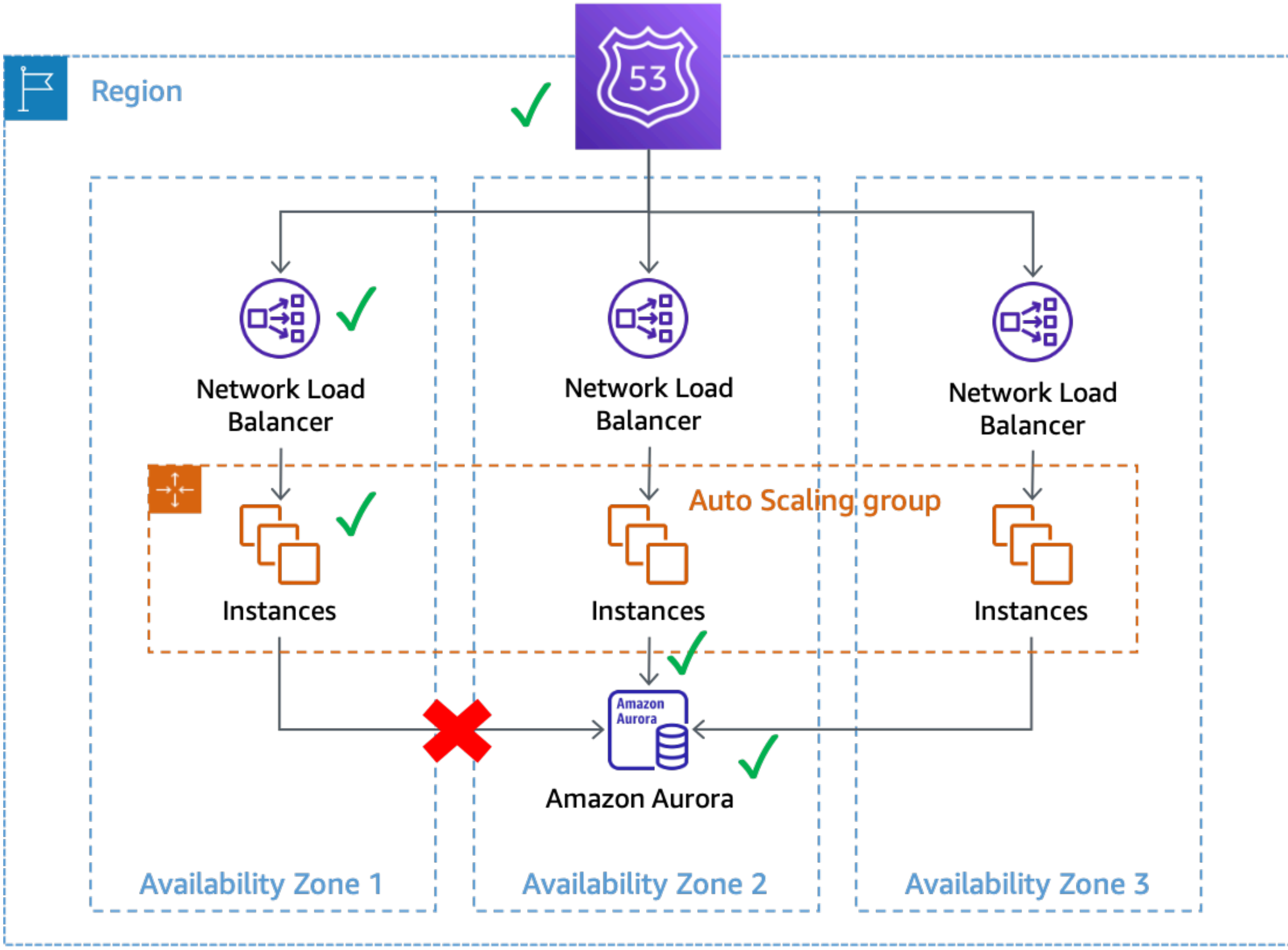
Ein Quadrant, der die verschiedenen Zustände definiert, in denen sich ein System auf der Grundlage verschiedener Perspektiven befinden kann

Der Fehler kann auch diesen Quadranten durchqueren. Ein Ereignis kann als grauer Fehler beginnen, dann zu einem erkannten Fehler werden, dann zu einem maskierten Fehler übergehen und dann möglicherweise wieder zu einem grauen Fehler zurückkehren. Es gibt keinen definierten Zyklus, und es besteht fast immer die Möglichkeit, dass ein Fehler erneut auftritt, bis die Grundursache behoben ist.

Daraus ziehen wir den Schluss, dass sich Workloads nicht immer auf das zugrunde liegende System verlassen können, um den Fehler zu erkennen und zu beheben. Unabhängig davon, wie ausgefeilt und belastbar das zugrundeliegende System ist, besteht immer die Möglichkeit, dass ein Ausfall unentdeckt bleibt oder unter der Reaktionsschwelle bleibt. Die Nutzer dieses Systems, wie App 1, müssen in der Lage sein, die Auswirkungen eines Grauausfalls sowohl schnell zu erkennen als auch zu mindern. Dies erfordert den Aufbau von Beobachtbarkeits- und Wiederherstellungsmechanismen für diese Situationen.

Beispiel für einen Graufehler

Graue Ausfälle können sich auf Multi-AZ-Systeme auswirkenAWS. Nehmen wir zum Beispiel eine Flotte vonAmazon EC2Instanzen in einer Auto Scaling-Gruppe, die in drei Availability Zones bereitgestellt werden. Sie alle stellen eine Verbindung zu einer Amazon Aurora-Datenbank in einer Availability Zone her. Dann tritt ein grauer Fehler auf, der sich auf das Netzwerk zwischen Availability Zone 1 und Availability Zone 2 auswirkt. Die Folge dieser Beeinträchtigung ist, dass ein Prozentsatz der neuen und bestehenden Datenbankverbindungen von Instances in Availability Zone 1 ausfällt. Diese Situation ist in der folgenden Abbildung dargestellt.



Ein grauer Fehler, der sich auf Datenbankverbindungen von Instances in Availability Zone 1 auswirkt

In diesem Beispiel betrachtet Amazon EC2 die Instances in Availability Zone 1 als fehlerfrei, da sie weiterhin bestehenSystem- und Instanzstatuschecks. Amazon EC2 Auto Scaling erkennt auch keine direkten Auswirkungen auf eine Availability Zone und tut dies weiterhinStartkapazität in den konfigurierten Availability Zones. Der Network Load Balancer (NLB) betrachtet die dahinter liegenden Instances ebenfalls als fehlerfrei, ebenso wie die Route 53-Zustandsprüfungen, die am NLB-Endpunkt durchgeführt werden. In ähnlicher Weise betrachtet Amazon Relational Database Service (Amazon RDS) den Datenbank-Cluster als fehlerfrei und tut dies nichteinen automatisierten Failover auslösen. Wir haben viele verschiedene Dienste, deren Service und Ressourcen alle als fehlerfrei einstufen, aber der Workload erkennt einen Fehler, der sich auf die Verfügbarkeit auswirkt. Das ist ein grauer Fehler.

Reaktion auf Grauausfälle

Wenn Sie ein graues Versagen in IhremAWSUmgebung, Sie haben im Allgemeinen drei verfügbare Optionen:

- Tun Sie nichts und warten Sie, bis die Beeinträchtigung beendet ist.
- Wenn die Beeinträchtigung auf eine einzelne Availability Zone beschränkt ist, evakuieren Sie diese Availability Zone.
- Failover zu einem anderenAWS-Regionund nutzen Sie die Vorteile vonAWSRegionale Isolation, um die Auswirkungen zu mildern.

VieleAWSKunden sind mit Option eins für einen Großteil ihrer Workloads einverstanden. Sie akzeptieren eine mögliche VerlängerungZiel für die Wiederherstellungszeit (RTO)mit dem Nachteil, dass sie keine zusätzlichen Observabilitäts- oder Resilienz-Lösungen entwickeln mussten. Andere Kunden entscheiden sich für die Implementierung der dritten Option,Disaster Recovery für mehrere Regionen(DR), als ihr Plan zur Schadensbegrenzung für eine Vielzahl von Ausfallarten. Architekturen mit mehreren Regionen können in diesen Szenarien gut funktionieren. Bei der Verwendung dieses Ansatzes gibt es jedoch einige Kompromisse (sieheAWSGrundlagen mehrerer Regionenfür eine ausführliche Diskussion über Überlegungen zu mehreren Regionen).

Erstens kann der Aufbau und Betrieb einer Architektur mit mehreren Regionen ein herausforderndes, komplexes und potenziell teures Unterfangen sein. Multiregionale Architekturen erfordern eine sorgfältige Abwägung, welcheDR-Strategiedu wählst aus. Es ist möglicherweise steuerlich nicht rentabel, eine aktiv-aktive DR-Lösung für mehrere Regionen zu implementieren, nur um zonale Beeinträchtigungen zu bewältigen, während eine Sicherungs- und Wiederherstellungsstrategie Ihre Anforderungen an die Resilienz möglicherweise nicht erfüllt. Darüber hinaus müssen Failovers für mehrere Regionen in der Produktion kontinuierlich angewendet werden, damit Sie sicher sein können, dass sie bei Bedarf funktionieren. Dies alles erfordert viel Zeit und Ressourcen für die Entwicklung, den Betrieb und das Testen.

Zweitens die Datenreplikation zwischenAWS-Regionenunter Verwendung vonAWSDienste werden heute alle asynchron ausgeführt. Asynchrone Replikation kann zu Datenverlust führen. Dies bedeutet, dass bei einem regionalen Failover die Möglichkeit eines gewissen Datenverlusts und Inkonsistenzen besteht. Ihre Toleranz gegenüber dem Ausmaß des Datenverlusts ist definiert als IhreWiederherstellungspunktziel (RPO). Kunden, für die eine hohe Datenkonsistenz eine Voraussetzung ist, müssen Abgleichssysteme einrichten, um diese Konsistenzprobleme zu beheben, sobald die primäre Region wieder verfügbar ist. Oder sie müssen ihre eigenen synchronen Replikations- oder Dual-Write-Systeme aufbauen, was erhebliche Auswirkungen auf die Reaktionslatenz, die Kosten und die Komplexität haben kann. Sie machen die sekundäre Region auch für jede Transaktion zu einer festen Abhängigkeit, was die Verfügbarkeit des Gesamtsystems potenziell verringern kann.

Schließlich ist für viele Workloads, die einen Active-/Standby-Ansatz verwenden, ein Zeitaufwand ungleich Null erforderlich, um den Failover in eine andere Region durchzuführen. Ihr Workload-Portfolio muss möglicherweise in der Primärregion in einer bestimmten Reihenfolge heruntergefahren werden, Verbindungen müssen abgebaut oder bestimmte Prozesse gestoppt werden. Dann müssen die Dienste möglicherweise in einer bestimmten Reihenfolge wieder verfügbar gemacht werden. Möglicherweise müssen auch neue Ressourcen bereitgestellt werden oder es dauert einige Zeit, bis die erforderlichen Gesundheitschecks bestanden sind, bevor sie in Betrieb genommen werden. Dieser Failover-Prozess kann als eine Phase der vollständigen Nichtverfügbarkeit erlebt werden. Damit befassen sich RTOs.

Innerhalb einer Region vieleAWSDienste bieten eine stark konsistente Datenpersistenz. Amazon RDS Multi-AZ-Bereitstellungen verwendensynchrone Replikation. Amazon Simple Storage Service(Amazon S3) bietetstarkread-after-writeKonsistenz. Amazon Elastic Blockspeicher(Amazon EBS) -Angeboteabsturzkonsistente Snapshots mit mehreren Volumes. Amazon DynamoDBDosesehr konsistente Lesevorgänge durchführen. Diese Funktionen können Ihnen helfen, in einer einzelnen Region ein niedrigeres RPO (in den meisten Fällen ein RPO von Null) zu erreichen als in Architekturen mit mehreren Regionen.

Die Evakuierung einer Availability Zone kann zu einem niedrigeren RTO führen als eine Strategie mit mehreren Regionen, da Ihre Infrastruktur und Ressourcen bereits in allen Availability Zones bereitgestellt sind. Anstatt sorgfältig anzuordnen, dass Dienste heruntergefahren und gesichert werden oder Verbindungen unterbrochen werden, können Multi-AZ-Architekturen auch dann weiter statisch arbeiten, wenn eine Availability Zone beeinträchtigt ist. Anstelle einer Phase vollständiger Nichtverfügbarkeit, wie sie bei einem regionalen Failover auftreten kann, kommt es bei einer Evakuierung der Availability Zone bei vielen Systemen möglicherweise nur zu einer geringfügigen Beeinträchtigung, da die Arbeit auf die verbleibenden Availability Zones verlagert wird. Wenn das System so konzipiert wurdestatisch stabilBei einem Ausfall der Availability Zone (in diesem Fall würde das bedeuten, dass Kapazität in den anderen Availability Zones vorab bereitgestellt wird, um die Last zu absorbieren), sehen Kunden des Workloads möglicherweise überhaupt keine Auswirkungen.

Es ist möglich, dass sich die Beeinträchtigung einer einzelnen Availability Zone auf eine oder mehrere auswirktAWS Regionale Dienstezusätzlich zu Ihrer Arbeitsbelastung.

Wenn Sie regionale Auswirkungen beobachten, sollten Sie das Ereignis als regionale Servicebeeinträchtigung behandeln, obwohl die Ursache dieser Auswirkungen in einer einzelnen Availability Zone liegt. Die Evakuierung einer Availability Zone wird diese Art von Problem nicht lösen. Verwenden Sie Ihre vorhandenen Reaktionspläne, um auf eine Beeinträchtigung des regionalen Dienstes zu reagieren, wenn diese auftritt.

Der Rest dieses Dokuments konzentriert sich auf die zweite Option, die Evakuierung der Availability Zone, um niedrigere RTOs und RPOs bei Single-AZ-Grauausfällen zu erreichen. Diese Muster können dazu beitragen, den Wert und die Effizienz von Multi-AZ-Architekturen zu verbessern, und für die meisten Klassen von Workloads können sie die Notwendigkeit verringern, Architekturen mit mehreren Regionen zu erstellen, um diese Art von Ereignissen zu bewältigen.

Multi-AZ-Beobachtbarkeit

Um eine Availability Zone während eines Ereignisses evakuieren zu können, das auf eine einzelne Availability Zone beschränkt ist, müssen Sie zunächst feststellen können, dass der Fehler tatsächlich auf eine einzelne Availability Zone beschränkt ist. Dies erfordert einen genauen Überblick darüber, wie sich das System in jeder Availability Zone verhält. Viele AWS Dienste bieten out-of-the-box Kennzahlen, die Ihnen betriebliche Einblicke in Ihre Ressourcen geben. Amazon EC2 bietet beispielsweise zahlreiche Metriken wie CPU Auslastung, Lese- und Schreibvorgänge auf der Festplatte sowie ein- und ausgehender Netzwerkverkehr.

Wenn Sie Workloads erstellen, die diese Services nutzen, benötigen Sie jedoch mehr Transparenz als nur diese Standardmetriken. Sie möchten Einblick in das Kundenerlebnis haben, das Ihr Workload bietet. Darüber hinaus müssen Ihre Kennzahlen auf die Availability Zones abgestimmt sein, in denen sie erstellt werden. Dies ist der Einblick, den Sie benötigen, um unterschiedlich beobachtbare Graufehler zu erkennen. Dieses Maß an Transparenz erfordert Instrumentierung.

Die Instrumentierung erfordert das Schreiben von explizitem Code. Dieser Code sollte beispielsweise aufzeichnen, wie lange Aufgaben dauern, zählen, wie viele Elemente erfolgreich waren oder nicht, Metadaten zu den Anfragen sammeln und so weiter. Außerdem müssen Sie im Voraus Schwellenwerte definieren, um zu definieren, was als normal angesehen wird und was nicht. Sie sollten Ziele und verschiedene Schweregrade für Latenz, Verfügbarkeit und Fehleranzahl in Ihrem Workload skizzieren. Der Artikel Instrumenting Distributed Systems for Operational Visibility bietet eine Reihe von bewährten Methoden in der Amazon Builders' Library.

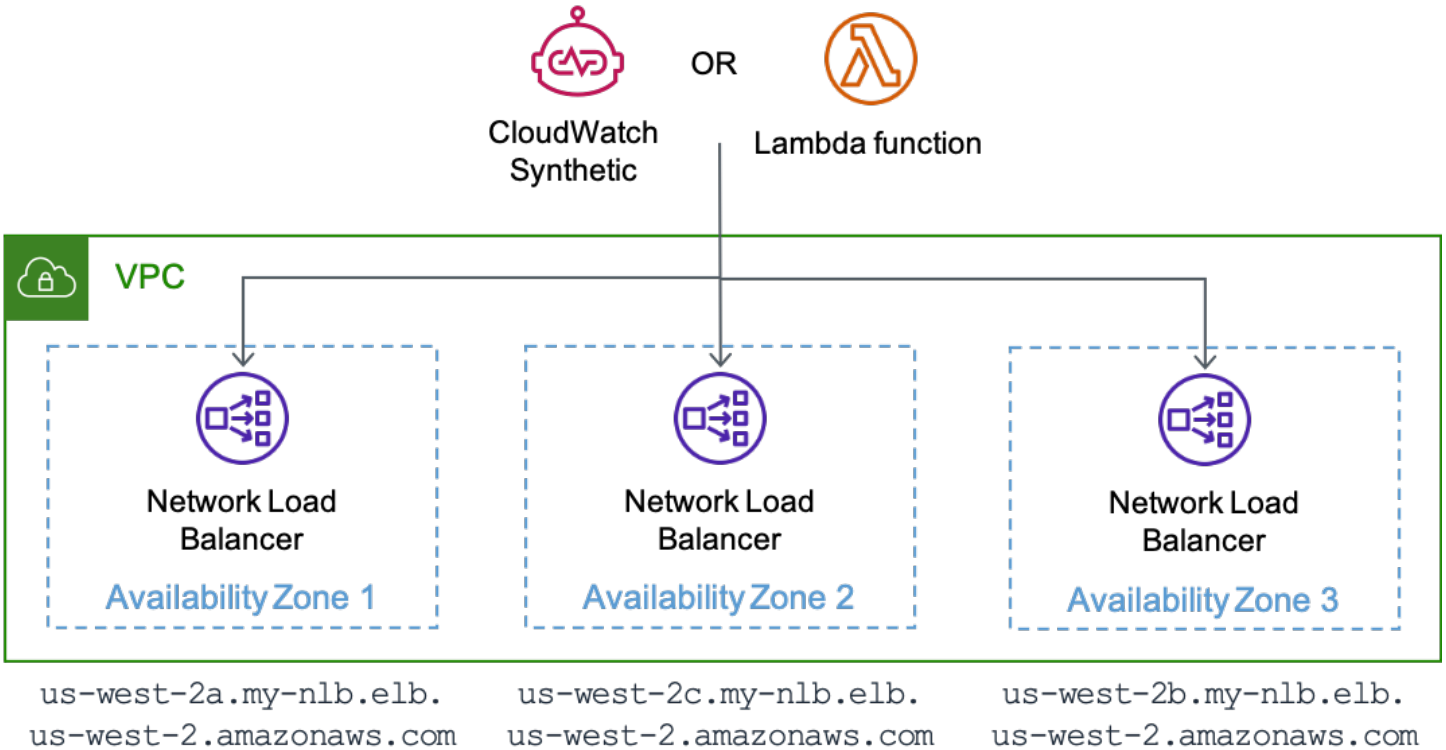
Metriken sollten sowohl serverseitig als auch clientseitig generiert werden. Eine bewährte Methode zur Generierung von kundenseitigen Kennzahlen und zum besseren Verständnis des Kundenerlebnisses ist die Verwendung von Canaries, einer Software, die Ihre Arbeitslast regelmäßig überprüft und Kennzahlen aufzeichnet.

Neben der Erstellung dieser Kennzahlen müssen Sie auch deren Kontext verstehen. Eine Möglichkeit, dies zu tun, ist die Verwendung von Dimensionen. Dimensionen geben einer Metrik eine eindeutige Identität und helfen zu erklären, was Ihnen die Metriken sagen. Für Metriken, die zur Identifizierung von Fehlern in Ihrem Workload verwendet werden (z. B. Latenz, Verfügbarkeit oder Fehleranzahl), müssen Sie Dimensionen verwenden, die Ihren Grenzen zur Fehlerisolierung entsprechen.

Wenn Sie beispielsweise einen Webservice in einer Region über mehrere Availability Zones hinweg mit einem M odel-view-controller (MVC) -Webframework ausführen, sollten Sie`Region`,, `Availability Zone ID``ControllerAction`, und `InstanceId` als Dimensionen für Ihre Dimensionssätze verwenden (wenn Sie Microservices verwenden, könnten Sie den Dienstnamen und die HTTP Methode anstelle der Controller- und Aktionsnamen verwenden). Dies liegt daran, dass Sie davon ausgehen, dass verschiedene Arten von Fehlern durch diese Grenzen isoliert werden. Sie würden nicht erwarten, dass ein Fehler im Code Ihres Webdienstes, der die Möglichkeit beeinträchtigt, Produkte aufzulisten, sich auch auf die Startseite auswirkt. Ebenso würden Sie nicht erwarten, dass ein voller EBS Volume auf einer einzelnen EC2 Instanz andere EC2 Instanzen davon abhält, Ihre Webinhalte bereitzustellen. Mithilfe der Dimension Availability Zone ID können Sie die Auswirkungen auf die Availability Zone überall einheitlich identifizieren. AWS-Konten Sie können die Availability Zone ID in Ihren Workloads auf verschiedene Arten finden. Einige Beispiele finden Sie unter. Anhang A — Abrufen der Availability Zone-ID

Dieses Dokument verwendet in den Beispielen hauptsächlich Amazon EC2 als Rechenressource, `InstanceId` könnte aber durch eine Container-ID für die Rechenressourcen Amazon Elastic Container Service (AmazonECS) und Amazon Elastic Kubernetes Service (AmazonEKS) als Komponenten Ihrer Dimensionen ersetzt werden.

Ihre Kanaren können auch`Controller`,`Action`, und `Region` als Dimensionen in ihren Metriken verwenden`AZ-ID`, wenn Sie zonale Endpunkte für Ihren Workload haben. Richten Sie in diesem Fall Ihre Canaries so aus, dass sie in der Availability Zone laufen, die sie gerade testen. Dadurch wird sichergestellt, dass, wenn sich ein isoliertes Availability Zone-Ereignis auf die Availability Zone auswirkt, in der Ihr Canary läuft, keine Metriken aufgezeichnet werden, die eine andere Availability Zone, die getestet wird, als ungesund erscheinen lassen. Ihr Canary kann beispielsweise jeden zonalen Endpunkt anhand seiner zonalen Namen auf einen Dienst hinter einem Network Load Balancer (NLB) oder Application Load Balancer (ALB) testen. DNS



Ein Canary, der auf CloudWatch Synthetics läuft, oder eine AWS Lambda Funktion, die jeden zonalen Endpunkt eines testet NLB

Durch die Erstellung von Metriken mit diesen Dimensionen können Sie CloudWatch Amazon-Alarme einrichten, die Sie benachrichtigen, wenn innerhalb dieser Grenzen Änderungen der Verfügbarkeit oder Latenz auftreten. Sie können diese Daten auch schnell mithilfe von Dashboards analysieren. Um sowohl Metriken als auch Protokolle effizient zu nutzen, CloudWatch bietet Amazon das eingebettete Metrikformat (EMF), mit dem Sie benutzerdefinierte Metriken in Protokolldaten einbetten können. CloudWatchextrahiert automatisch die benutzerdefinierten Metriken, sodass Sie sie visualisieren und als Alarm auslösen können. AWS bietet mehrere Client-Bibliotheken für verschiedene Programmiersprachen, die den Einstieg erleichternEMF. Sie können mit AmazonEC2, Amazon ECS EKS AWS Lambda, Amazon und lokalen Umgebungen verwendet werden. Da Metriken in Ihre Protokolle eingebettet sind, können Sie Amazon CloudWatch Contributor Insights auch verwenden, um Zeitreihendiagramme zu erstellen, in denen Daten von Mitwirkenden angezeigt werden. In diesem Szenario könnten wir Daten gruppiert nach Dimensionen wie `AZ-IDInstanceId`, oder `Controller` sowie nach jedem anderen Feld im Protokoll wie `SuccessLatency` oder anzeigen. `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
}
```


Dieses Protokoll hat drei Gruppen von Dimensionen. Sie werden in der Reihenfolge ihrer Granularität von Instanz zu Availability Zone zu Region weitergeleitet (`Controller`und `Action` sind in diesem Beispiel immer enthalten). Sie unterstützen die Erstellung von Alarmen für Ihren gesamten Workload, die darauf hinweisen, wenn eine bestimmte Controller-Aktion in einer einzelnen Instanz, in einer einzelnen Availability Zone oder innerhalb eines Ganzen AWS-Region Auswirkungen hat. Diese Dimensionen werden für die Anzahl der HTTP Antwortmetriken 2xx, 3xx, 4xx und 5xx sowie für die Latenz erfolgreicher Anforderungsmetriken verwendet (wenn die Anfrage fehlschlägt, wird auch eine Metrik für die Latenz fehlgeschlagener Anfragen aufgezeichnet). Das Protokoll zeichnet auch andere Informationen auf, z. B. den HTTP Pfad, die Quell-IP des Anforderers und ob für diese Anfrage der lokale Cache aktualisiert werden musste. Diese Datenpunkte können dann verwendet werden, um die Verfügbarkeit und Latenz der einzelnen Datenpunkte zu berechnen, die der Workload API bereitstellt.

> Ein Hinweis zur Verwendung von HTTP Antwortcodes für Verfügbarkeitsmetriken
>
> In der Regel können Sie 2xx- und 3xx-Antworten als erfolgreich und 5xx als Fehlschläge betrachten. 4xx-Antwortcodes liegen irgendwo dazwischen. In der Regel werden sie aufgrund eines Client-Fehlers generiert. Vielleicht liegt ein Parameter außerhalb des zulässigen Bereichs, was zu einer 400-Antwort führt, oder sie fordern etwas an, das nicht existiert, was zu einer 404-Antwort führt. Sie würden diese Antworten nicht mit der Verfügbarkeit Ihres Workloads verrechnen. Dies könnte jedoch auch das Ergebnis eines Fehlers in der Software sein.
>
> Wenn Sie beispielsweise eine strengere Eingabeüberprüfung eingeführt haben, die eine Anfrage ablehnt, die zuvor erfolgreich gewesen wäre, könnte die Antwort von 400 als Rückgang der Verfügbarkeit gewertet werden. Oder vielleicht drosseln Sie den Kunden und geben eine Antwort von 429 zurück. Die Drosselung eines Kunden schützt zwar Ihren Service, um seine Verfügbarkeit aufrechtzuerhalten, aus Sicht des Kunden ist der Service jedoch nicht verfügbar, um seine Anfrage zu bearbeiten. Sie müssen entscheiden, ob 4xx-Antwortcodes Teil Ihrer Verfügbarkeitsberechnung sind oder nicht.

In diesem Abschnitt wurde zwar die Verwendung CloudWatch als Methode zur Erfassung und Analyse von Metriken beschrieben, dies ist jedoch nicht die einzige Lösung, die Sie verwenden können. Sie können sich dafür entscheiden, Metriken auch an Amazon Managed Service for Prometheus und Amazon Managed Grafana, eine Amazon DynamoDB-Tabelle, zu senden oder eine Überwachungslösung eines Drittanbieters zu verwenden. Entscheidend ist, dass die von Ihrem Workload generierten Metriken einen Kontext zu den Grenzen der Fehlerisolierung Ihres Workloads enthalten müssen.

Mit Workloads, die Metriken erzeugen, deren Dimensionen auf die Grenzen der Fehlerisolierung ausgerichtet sind, können Sie eine Beobachtbarkeit einrichten, die isolierte Ausfälle in der Availability Zone erkennt. In den folgenden Abschnitten werden drei sich ergänzende Ansätze zur

Erkennung von Ausfällen beschrieben, die auf die Beeinträchtigung einer einzelnen Availability Zone zurückzuführen sind.

Themen

Fehlererkennung mit CloudWatch kombinierten Alarmen

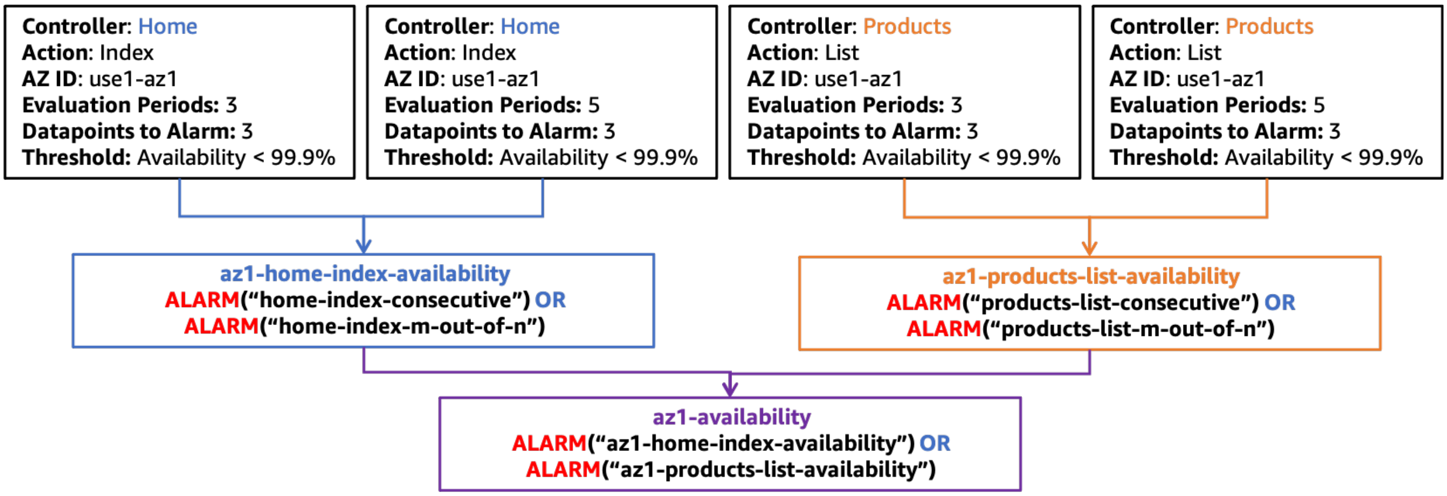
Bei CloudWatch Metriken ist jeder Dimensionssatz eine eindeutige Metrik, und Sie können für jede einzelne einen CloudWatch Alarm erstellen. Sie können dann CloudWatch zusammengesetzte Amazon-Alarme erstellen, um diese Metriken zu aggregieren.

Um Auswirkungen genau zu erkennen, werden in den Beispielen in diesem paper zwei verschiedene CloudWatch Alarmstrukturen für jede Dimension verwendet, auf die sie den Alarm einstellen. Jeder Alarm verwendet einen Zeitraum von einer Minute, was bedeutet, dass die Metrik einmal pro Minute ausgewertet wird. Beim ersten Ansatz werden drei aufeinanderfolgende Datenpunkte für Sicherheitsverletzungen verwendet, indem die Bewertungszeiträume und Datenpunkte auf „Alarm" auf drei gesetzt werden, was einer Gesamtauswirkung von drei Minuten entspricht. Beim zweiten Ansatz wird ein „M aus N" verwendet, wenn 3 beliebige Datenpunkte innerhalb eines Fünf-Minuten-Zeitfensters verletzt werden, indem die Bewertungszeiträume auf fünf und die Datenpunkte auf Alarm auf drei gesetzt werden. Auf diese Weise kann sowohl ein konstantes Signal als auch ein Signal erkannt werden, das über einen kurzen Zeitraum schwankt. Die hier angegebenen Zeitdauern und die Anzahl der Datenpunkte sind nur ein Vorschlag. Verwenden Sie Werte, die für Ihre Workloads sinnvoll sind.

Erkennen Sie Auswirkungen in einer einzelnen Availability Zone

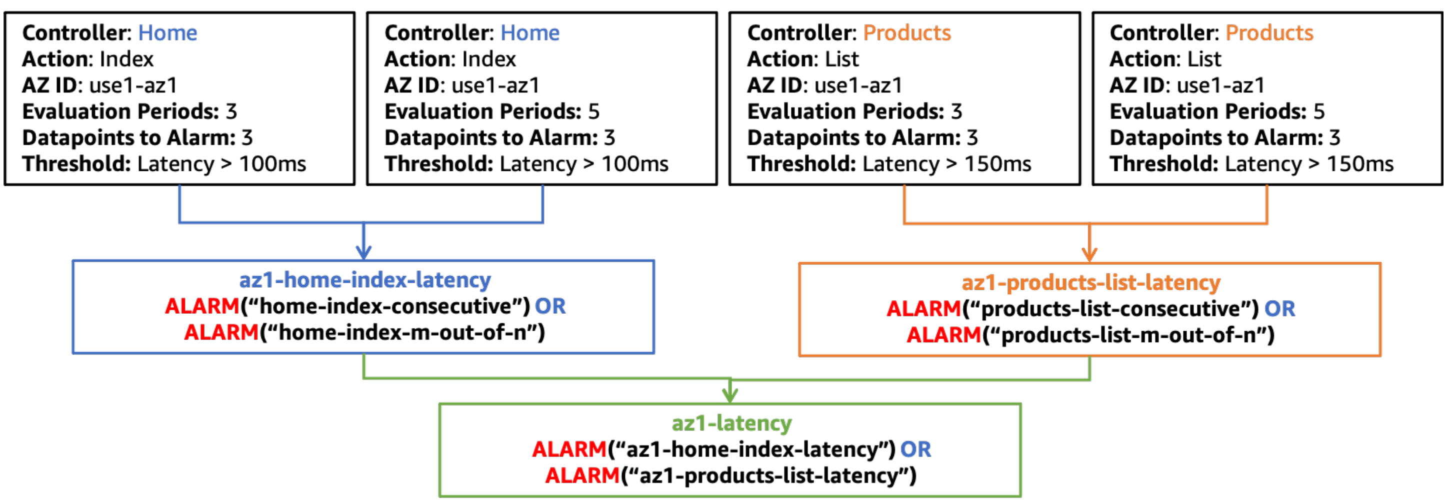
Stellen Sie sich anhand dieses Konstrukts einen Workload vor `ControllerAction`, der`InstanceId`,`AZ-ID`, und `Region` als Dimensionen verwendet. Der Workload besteht aus zwei Controllern, Products und Home, und einer Aktion pro Controller, List und Index. Er ist in drei Availability Zones in der `us-east-1` Region tätig. Sie würden in jeder Availability Zone zwei Alarme für die Verfügbarkeit `Controller` und `Action` eine Kombination davon sowie jeweils zwei Alarme für die Latenz einrichten. Anschließend können Sie optional für jede `Controller` `Action`

Kombination einen zusammengesetzten Verfügbarkeitsalarm erstellen. Schließlich erstellen Sie einen zusammengesetzten Alarm, der alle Verfügbarkeitsalarme für die Availability Zone zusammenfasst. Dies wird in der folgenden Abbildung für eine einzelne Availability Zone dargestellt`use1-az1`, wobei der optionale zusammengesetzte Alarm für jede `Controller Action` Kombination verwendet wird (ähnliche Alarme wären auch für die Availability Zones `use1-az2` und `use1-az3` Availability Zones vorhanden, werden aber der Einfachheit halber nicht dargestellt).



Verbundalarmstruktur für die Verfügbarkeit in *`use1-az1`*

Sie würden auch eine ähnliche Alarmstruktur für die Latenz erstellen, wie in der nächsten Abbildung dargestellt.

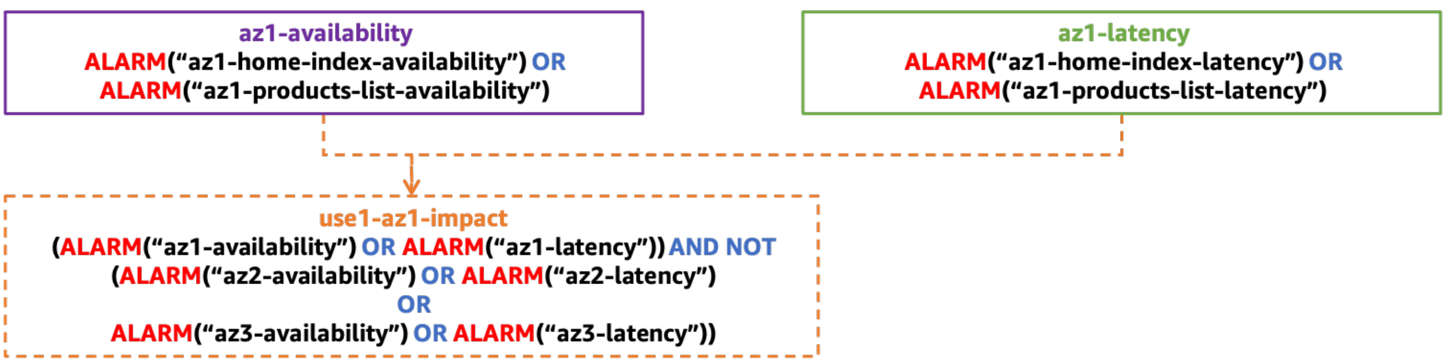


Verbundalarmstruktur für Latenz in *`use1-az1`*

Bei den restlichen Zahlen in diesem Abschnitt werden nur die Alarme `az1-availability` und die `az1-latency` zusammengesetzten Alarme auf der obersten Ebene angezeigt. Diese zusammengesetzten Alarme `az1-availability` und informieren Sie darüber`az1-latency`, ob in einer bestimmten Availability Zone für einen Teil Ihrer Workload entweder die Verfügbarkeit unter oder die Latenz über definierte Schwellenwerte steigt. Möglicherweise sollten Sie auch erwägen, den Durchsatz zu messen, um Auswirkungen zu erkennen, die verhindern, dass Ihre Arbeitslast in einer einzelnen Availability Zone bearbeitet wird. Sie können auch Alarme, die auf den von Ihren Kanaren ausgegebenen Messdaten basieren, in diese zusammengesetzten Alarme integrieren. Auf diese Weise wird durch den Alarm eine Warnung ausgelöst, wenn entweder auf der Server- oder der Clientseite Auswirkungen auf die Verfügbarkeit oder Latenz festgestellt werden.

Stellen Sie sicher, dass die Auswirkungen nicht regional sind

Ein weiterer Satz zusammengesetzter Alarme kann verwendet werden, um sicherzustellen, dass nur ein isoliertes Availability Zone-Ereignis zur Aktivierung des Alarms führt. Dies wird erreicht, indem sichergestellt wird, dass sich ein zusammengesetzter Availability Zone-Alarm im `ALARM` Status befindet, während sich die zusammengesetzten Alarme für die anderen Availability Zones im `OK` Status befinden. Dies führt zu einem zusammengesetzten Alarm pro Availability Zone, die Sie verwenden. Ein Beispiel ist in der folgenden Abbildung dargestellt (denken Sie daran, dass es Alarme für Latenz und Verfügbarkeit in `use1-az2` und`use1-az3`,`az2-latency`,`az2-availability`, und`az3-latency`, gibt`az3-availability`, die der Einfachheit halber nicht abgebildet sind).



Verbundalarmstruktur zur Erkennung von Stößen, isoliert auf eine einzelne AZ

Stellen Sie sicher, dass die Auswirkungen nicht durch eine einzelne Instanz verursacht werden

Eine einzelne Instanz (oder ein kleiner Prozentsatz Ihrer gesamten Flotte) kann unverhältnismäßige Auswirkungen auf die Verfügbarkeits- und Latenzkennzahlen haben, sodass die gesamte Availability Zone als betroffen erscheinen könnte, obwohl dies in Wirklichkeit nicht der Fall ist. Es ist schneller und genauso effektiv, eine einzelne problematische Instanz zu entfernen, als eine Availability Zone zu evakuieren.

Instances und Container werden in der Regel als kurzlebige Ressourcen behandelt und häufig durch Dienste wie ersetzt. AWS Auto Scaling Es ist schwierig, jedes Mal, wenn eine neue Instance erstellt wird, einen neuen CloudWatch Alarm zu erstellen (aber mit den Lifecycle-Hooks von Amazon EventBridge oder Amazon EC2 Auto Scaling ist das sicherlich möglich). Stattdessen können Sie CloudWatch Contributor Insights verwenden, um die Anzahl der Mitwirkenden an Verfügbarkeits- und Latenzmetriken zu ermitteln.

Beispielsweise können Sie für eine HTTP Webanwendung eine Regel erstellen, um die wichtigsten Mitwirkenden für HTTP 5xx-Antworten in jeder Availability Zone zu ermitteln. Dadurch wird ermittelt, welche Instances zu einem Rückgang der Verfügbarkeit beitragen (unsere oben definierte Verfügbarkeitsmetrik basiert auf dem Vorhandensein von 5xx-Fehlern). Erstellen Sie anhand des EMF Protokollbeispiels eine Regel mit dem Schlüssel von`InstanceId`. Filtern Sie dann das Protokoll nach dem `HttpResponseCode` Feld. Dieses Beispiel ist eine Regel für die `use1-az1` Availability Zone.

```
{
    "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
    }
}
```

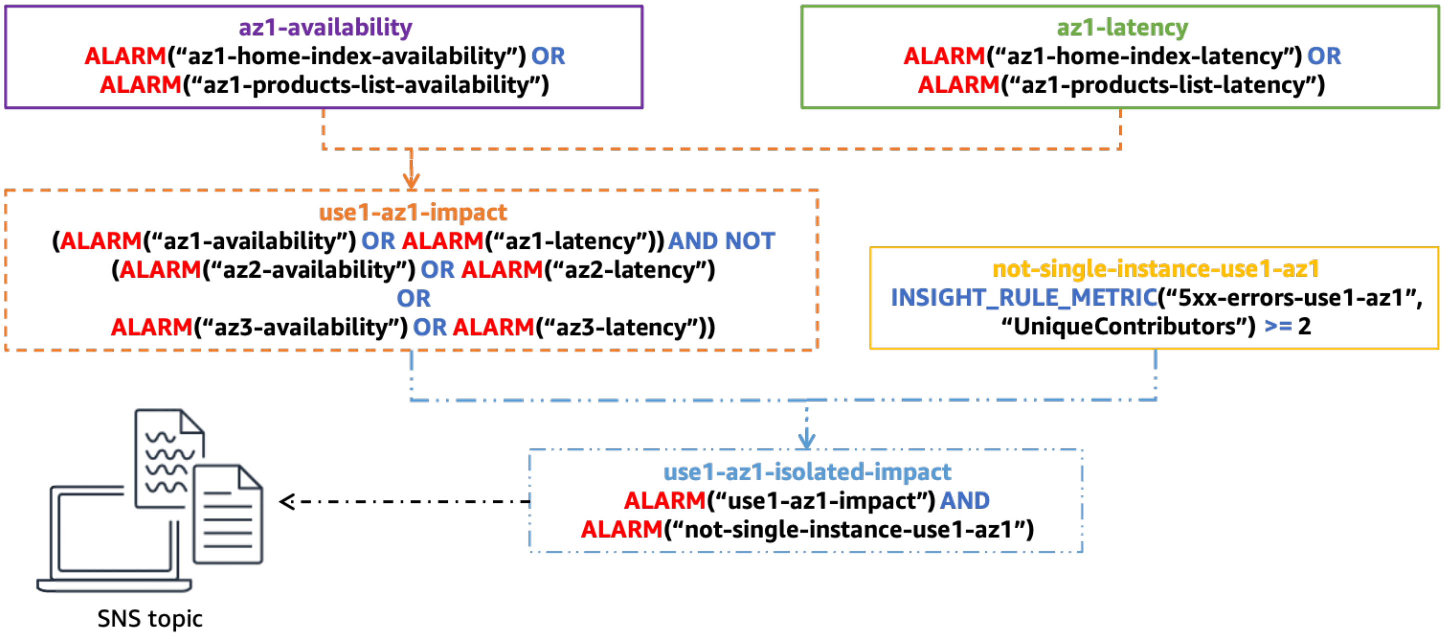
CloudWatch Alarme können ebenfalls auf der Grundlage dieser Regeln erstellt werden. Sie können Alarme auf der Grundlage von Contributor Insights-Regeln mithilfe von metrischer Mathematik und der `INSIGHT_RULE_METRIC` Funktion mit der `UniqueContributors` Metrik erstellen. Sie können auch zusätzliche Contributor Insights-Regeln mit CloudWatch Alarmen für Messwerte wie Latenz oder Fehlerzahlen sowie Alarme für die Verfügbarkeit erstellen. Diese Alarme können zusammen mit den kombinierten Alarmen der isolierten Availability Zone verwendet werden, um sicherzustellen, dass einzelne Instanzen den Alarm nicht auslösen. Die Metrik für die Insights-Regel für `use1-az1` könnte wie folgt aussehen:

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

Sie können einen Alarm definieren, wenn diese Metrik einen Schwellenwert überschreitet, in diesem Beispiel zwei. Er wird aktiviert, wenn der einzelne Mitwirkende an 5xx-Antworten diesen Schwellenwert überschreitet, was darauf hindeutet, dass die Auswirkungen auf mehr als zwei Instanzen zurückzuführen sind. Dieser Alarm verwendet den Vergleich „Größer als“ statt „Weniger als“, um sicherzustellen, dass ein Nullwert für einzelne Mitwirkende den Alarm nicht auslöst. Dies zeigt Ihnen, dass die Auswirkung nicht auf eine einzelne Instanz zurückzuführen ist. Passen Sie diesen Schwellenwert an Ihre individuelle Arbeitslast an. Als allgemeine Richtlinie gilt, dass dieser Wert mindestens 5% der gesamten Ressourcen in der Availability Zone ausmachen sollte. Mehr als 5% Ihrer Ressourcen, die betroffen sind, weisen bei ausreichender Stichprobengröße eine statistische Signifikanz auf.

Zusammenführung

Die folgende Abbildung zeigt die vollständige zusammengesetzte Alarmstruktur für eine einzelne Availability Zone:



Vollständige Verbundalarmstruktur zur Bestimmung der Einzel-AZ-Auswirkung

Der letzte zusammengesetzte Alarm,, wird aktiviert`use1-az1-isolated-impact`, wenn der zusammengesetzte Alarm, der auf die Auswirkungen der isolierten Availability Zone aufgrund von Latenz oder Verfügbarkeit hinweist`use1-az1-aggregate-alarm`, sich im `ALARM` Status befindet und wenn der Alarm, der auf der Contributor Insights-Regel für dieselbe Availability Zone basiert`not-single-instance-use1-az1`, ebenfalls im `ALARM` Status ist (was bedeutet, dass es sich bei der Auswirkung um mehr als eine einzelne Instanz handelt). Sie würden diesen Alarm-Stapel für jede Availability Zone erstellen, die Ihr Workload verwendet.

Sie können diesem letzten Alarm eine [Amazon Simple Notification Service](https://aws.amazon.com/sns/) (AmazonSNS) -Warnung hinzufügen. Alle vorherigen Alarme wurden ohne Aktion konfiguriert. Die Warnung könnte einen Bediener per E-Mail benachrichtigen, um eine manuelle Untersuchung einzuleiten. Es könnte auch eine Automatisierung zur Evakuierung der Availability Zone einleiten. Ein Wort der Vorsicht ist jedoch bei der Gebäudeautomatisierung geboten, um auf diese Warnmeldungen zu reagieren. Nach einer Evakuierung der Availability Zone sollte das Ergebnis sein, dass die erhöhten Fehlerquoten verringert werden und der Alarm wieder in einen Zustand übergeht. `OK` Wenn es in einer anderen Availability Zone zu Auswirkungen kommt, ist es möglich, dass durch die Automatisierung eine zweite oder dritte Availability Zone evakuiert wird, wodurch möglicherweise die gesamte verfügbare

Kapazität des Workloads entfernt wird. Die Automatisierung sollte überprüfen, ob bereits eine Evakuierung durchgeführt wurde, bevor Maßnahmen ergriffen werden. Möglicherweise müssen Sie auch Ressourcen in anderen Availability Zones skalieren, bevor eine Evakuierung erfolgreich ist.

Wenn Sie Ihrer MVC Web-App neue Controller oder Aktionen oder einen neuen Microservice oder generell zusätzliche Funktionen hinzufügen, die Sie separat überwachen möchten, müssen Sie in diesem Setup nur einige Alarme ändern. Sie erstellen neue Verfügbarkeits- und Latenzalarme für diese neue Funktionalität und fügen diese dann den entsprechenden zusammengesetzten Verfügbarkeits- und Latenzalarmen hinzu, `az1-latency` wie `az1-availability` in dem Beispiel, das wir hier verwendet haben. Die übrigen zusammengesetzten Alarme bleiben statisch, nachdem sie konfiguriert wurden. Dies macht die Einführung neuer Funktionen mit diesem Ansatz zu einem einfacheren Prozess.

Fehlererkennung mithilfe der Ausreißererkennung

Eine Lücke zum vorherigen Ansatz könnte entstehen, wenn Sie erhöhte Fehlerraten in mehreren Availability Zones feststellen, die aus einem unkorrelierten Grund auftreten. Stellen Sie sich ein Szenario vor, in dem Sie EC2 Instances in drei Availability Zones bereitgestellt haben und Ihr Schwellenwert für den Verfügbarkeitsalarm bei 99% liegt. Dann kommt es zu einer Beeinträchtigung einer einzelnen Availability Zone, wodurch viele Instanzen isoliert werden und die Verfügbarkeit in dieser Zone auf 55% sinkt. Gleichzeitig, aber in einer anderen Availability Zone, verbraucht eine einzelne EC2 Instanz den gesamten Speicherplatz auf ihrem EBS Volume und kann keine Protokolldateien mehr schreiben. Dadurch gibt sie zwar Fehler zurück, besteht aber trotzdem die Integritätsprüfungen des Load Balancers, da diese das Schreiben einer Protokolldatei nicht auslösen. Dies führt dazu, dass die Verfügbarkeit in dieser Availability Zone auf 98% sinkt. In diesem Fall würde Ihr einzelner Availability Zone-Auswirkungsalarm nicht aktiviert werden, da Sie eine Beeinträchtigung der Verfügbarkeit in mehreren Availability Zones feststellen. Sie könnten jedoch trotzdem fast alle Auswirkungen abmildern, indem Sie die beeinträchtigte Availability Zone evakuieren.

Bei einigen Arten von Workloads können in allen Availability Zones konsistent Fehler auftreten, bei denen die vorherige Verfügbarkeitsmetrik möglicherweise nicht hilfreich ist. Nehmen wir AWS Lambda zum Beispiel. AWS ermöglicht es Kunden, ihren eigenen Code für die Ausführung in der Lambda-Funktion zu erstellen. Um den Dienst nutzen zu können, müssen Sie Ihren Code einschließlich Abhängigkeiten in eine ZIP Datei hochladen und den Einstiegspunkt für die Funktion definieren. Aber manchmal verstehen Kunden diesen Teil falsch, weil sie beispielsweise eine kritische Abhängigkeit in der ZIP Datei vergessen oder den Methodennamen in der Lambda-Funktionsdefinition falsch eingeben. Dies führt dazu, dass die Funktion nicht aufgerufen werden kann, was zu einem Fehler führt. AWS Lambda sieht ständig solche Fehler, aber sie deuten nicht darauf hin, dass etwas unbedingt ungesund ist. Eine Beeinträchtigung der Availability Zone kann jedoch auch dazu führen, dass diese Fehler auftreten.

Um ein Signal in dieser Art von Rauschen zu finden, können Sie mithilfe der Ausreißererkennung feststellen, ob die Anzahl der Fehler zwischen den Availability Zones statistisch signifikant schwankt. Wir sehen zwar Fehler in mehreren Availability Zones, aber wenn es wirklich in einer Availability Zone zu einem Ausfall kommen sollte, würden wir davon ausgehen, dass die Fehlerrate in dieser Availability Zone im Vergleich zu den anderen deutlich höher oder möglicherweise deutlich niedriger ist. Aber um wie viel höher oder niedriger?

Eine Möglichkeit, diese Analyse durchzuführen, besteht darin, einen Chi-Quadrat-Test (x^2) zu verwenden, um statistisch signifikante Unterschiede in den Fehlerraten zwischen Availability Zones zu erkennen (es gibt viele verschiedene Algorithmen für die Erkennung von Ausreißern). Schauen wir uns an, wie der Chi-Quadrat-Test funktioniert.

Ein Chi-Quadrat-Test bewertet die Wahrscheinlichkeit, dass eine Verteilung der Ergebnisse wahrscheinlich ist. In diesem Fall interessiert uns die Verteilung der Fehler auf eine bestimmte Gruppe von. AZs Betrachten Sie für dieses Beispiel vier Availability Zones, um die Mathematik zu vereinfachen.

Stellen Sie zunächst die Nullhypothese auf, die definiert, was Ihrer Meinung nach das Standardergebnis ist. In diesem Test geht die Nullhypothese davon aus, dass Sie erwarten, dass Fehler gleichmäßig auf jede Availability Zone verteilt werden. Generieren Sie dann die alternative Hypothese, dass die Fehler nicht gleichmäßig verteilt sind, was auf eine Beeinträchtigung der Availability Zone hindeutet. Jetzt können Sie diese Hypothesen anhand von Daten aus Ihren Metriken testen. Zu diesem Zweck testen Sie Ihre Metriken in einem Fünf-Minuten-Fenster. Angenommen, Sie erhalten 1000 veröffentlichte Datenpunkte in diesem Fenster, in dem Sie insgesamt 100 Fehler sehen. Sie gehen davon aus, dass bei einer gleichmäßigen Verteilung die Fehler in jeder der vier Availability Zones in 25% der Fälle auftreten würden. Nehmen wir an, die folgende Tabelle zeigt, was Sie erwartet haben, verglichen mit dem, was Sie tatsächlich gesehen haben.

Tabelle 1: Erwartete und tatsächlich festgestellte Fehler im Vergleich

AZ	Expected	Tatsächliche
`use1-az1`	25	20
`use1-az2`	25	20

AZ	Expected	Tatsächliche
`use1-az3`	25	25
`use1-az4`	25	35

Sie sehen also, dass die Verteilung in Wirklichkeit nicht gleichmäßig ist. Sie könnten jedoch glauben, dass dies auf einen gewissen Grad an Zufälligkeit der von Ihnen untersuchten Datenpunkte zurückzuführen ist. Es besteht eine gewisse Wahrscheinlichkeit, dass ein solcher Verteilungstyp im Stichprobensatz vorkommt, und trotzdem wird davon ausgegangen, dass die Nullhypothese wahr ist. Dies führt zu der folgenden Frage: Wie hoch ist die Wahrscheinlichkeit, ein mindestens so extremes Ergebnis zu erzielen? Wenn diese Wahrscheinlichkeit unter einem definierten Schwellenwert liegt, lehnen Sie die Nullhypothese ab. Um statistisch signifikant zu sein, sollte diese Wahrscheinlichkeit 5% oder weniger betragen. [1]

[1] Craparo, Robert M. (2007). „Signifikanzniveau“. In Salkind, Neil J. Enzyklopädie für Messung und Statistik 3. Thousand Oaks, CA: SAGE Veröffentlichungen. S. 889—891. ISBN1-412-91611-9.

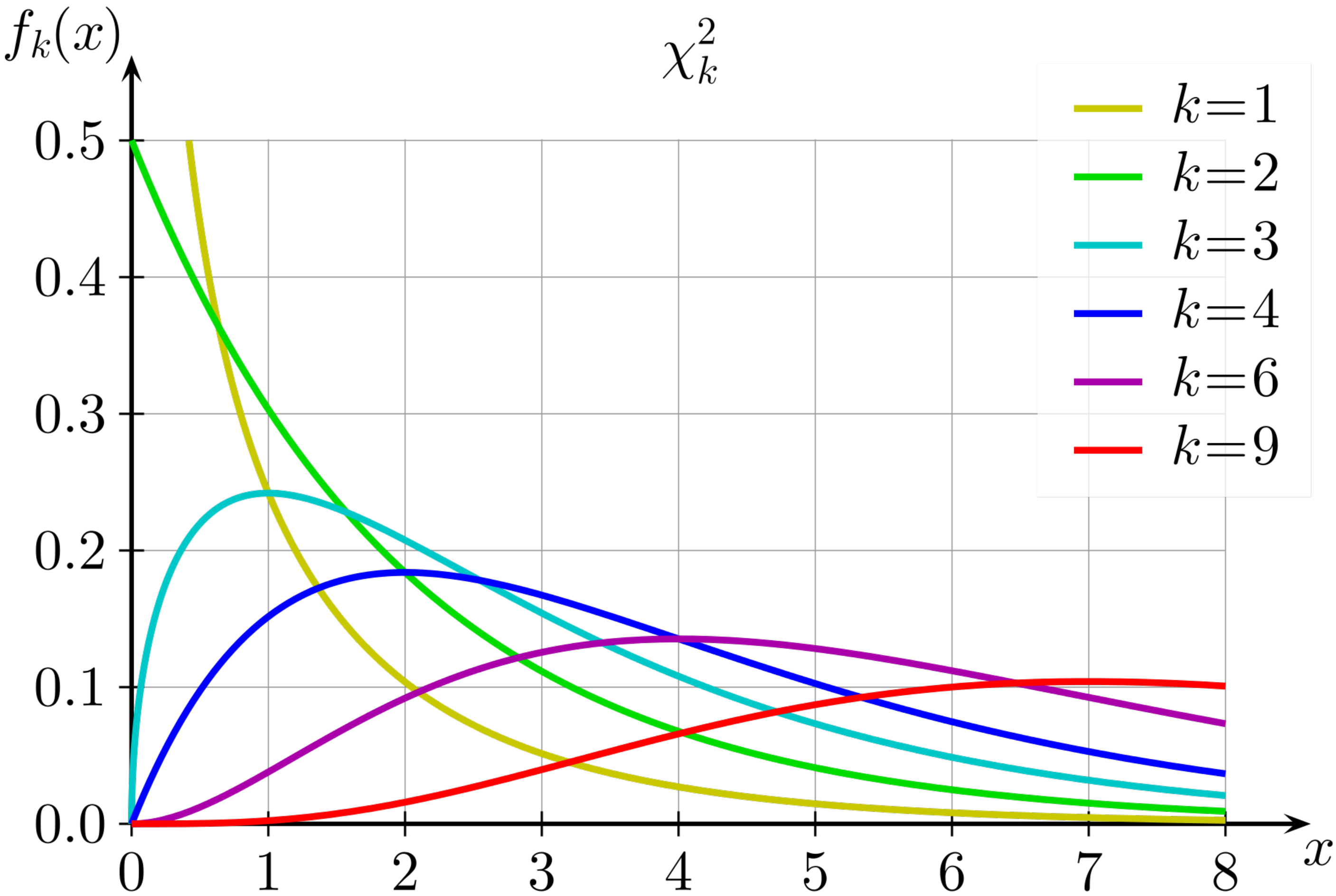
Wie berechnet man die Wahrscheinlichkeit dieses Ergebnisses? Sie verwenden die x/2-Statistik, die sehr gut untersuchte Verteilungen liefert und anhand dieser Formel die Wahrscheinlichkeit bestimmt werden kann, dass ein so extremes oder extremeres Ergebnis erzielt wird.

$$
\begin{aligned}
E_i &= expected\ observations\ of\ type\ i \\
O_i &= actual\ observations\ of\ type\ i \\
\chi^2 &= \sum \frac{(O_i - E_i)^2}{E_i}
\end{aligned}
\tag{1}
$$

In unserem Beispiel ergibt das:

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

Also, was 6 bedeutet das in Bezug auf unsere Wahrscheinlichkeit? Sie müssen sich eine Chi-Quadrat-Verteilung mit dem entsprechenden Freiheitsgrad ansehen. Die folgende Abbildung zeigt mehrere Chi-Quadrat-Verteilungen für unterschiedliche Freiheitsgrade.



Chi-Quadrat-Verteilungen für verschiedene Freiheitsgrade

Der Freiheitsgrad wird berechnet, wenn er um eins kleiner ist als die Anzahl der Auswahlmöglichkeiten im Test. In diesem Fall beträgt der Freiheitsgrad drei, da es vier Availability

Zones gibt. Dann möchten Sie die Fläche unter der Kurve (das Integral) für x ≥ 6 im Diagramm k = 3 ermitteln. Sie können diesen Wert auch anhand einer vorberechneten Tabelle mit häufig verwendeten Werten approximieren.

Tabelle 2: Kritische Werte im Chi-Quadrat

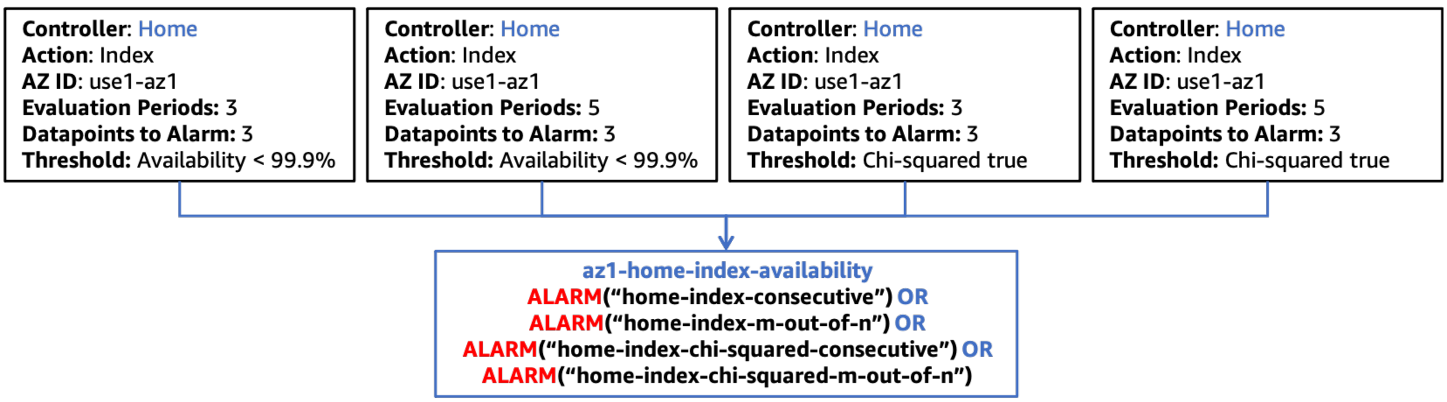
Freiheits grade	Wahrscheinlichkeit geringer als der kritische Wert				
	0,75	0,90	0,95	0,99	0,999
1	1,323	2,706	3,841	6,635	10,828
2	2,773	4,605	5,991	9,210	13,816
3	4,108	6,251	7,815	11,345	16,266
4	5,385	7,779	9,488	13,277	18,467

Bei drei Freiheitsgraden liegt der Chi-Quadrat-Wert von sechs zwischen den Wahrscheinlichkeitsspalten 0,75 und 0,9. Das bedeutet, dass die Wahrscheinlichkeit, dass diese Verteilung eintritt, bei über 10% liegt, was nicht weniger als der Schwellenwert von 5% ist. Daher akzeptieren Sie die Nullhypothese und stellen fest, dass es keinen statistisch signifikanten Unterschied bei den Fehlerraten zwischen den Availability Zones gibt.

Die Durchführung eines Chi-Quadrat-Statistiktests wird in der CloudWatch metrischen Mathematik nicht nativ unterstützt. Daher müssen Sie die entsprechenden Fehlermetriken in einer Rechenumgebung wie Lambda sammeln CloudWatch und den Test in einer Rechenumgebung ausführen. Sie können sich dafür entscheiden, diesen Test auf MVC Controller-/Aktionsebene oder individueller Microservice-Ebene oder auf Availability Zone-Ebene durchzuführen. Sie müssen abwägen, ob sich eine Beeinträchtigung der Availability Zone auf jeden Controller/jede Aktion oder jeden Microservice gleichermaßen auswirken würde, oder ob ein DNS Ausfall Auswirkungen auf einen Dienst mit geringem Durchsatz haben könnte und nicht auf einen Dienst mit hohem Durchsatz, der die Auswirkungen in aggregierter Form maskieren könnte. Wählen Sie in beiden Fällen die entsprechenden Dimensionen aus, um die Abfrage zu erstellen. Der Grad der Granularität wirkt sich auch auf die resultierenden CloudWatch Alarme aus, die Sie erstellen.

Erfassen Sie die Metrik zur Fehleranzahl für jede AZ und jeden Controller/jede Aktion in einem bestimmten Zeitfenster. Berechnen Sie zunächst, ob das Ergebnis des Chi-Quadrat-Tests entweder wahr (es gab eine statistisch signifikante Verzerrung) oder falsch (es gab keinen, das heißt, die Nullhypothese gilt). Wenn das Ergebnis falsch ist, veröffentlichen Sie einen Datenpunkt von 0 in Ihrem Metrik-Stream für Chi-Quadrat-Ergebnisse für jede Availability Zone. Wenn das Ergebnis wahr ist, veröffentlichen Sie einen 1-Datenpunkt für die Availability Zone mit den Fehlern, die am weitesten vom erwarteten Wert entfernt sind, und eine 0 für die anderen (Beispielcode, der in einer Lambda-Funktion verwendet werden kann, finden Sie unter). Anhang B — Beispiel für eine Chi-Quadrat-Berechnung Sie können den gleichen Ansatz wie bei den vorherigen Verfügbarkeitsalarmen verfolgen, indem Sie einen CloudWatch metrischen Alarm 3 in einer Reihe und einen CloudWatch metrischen Alarm 3 von 5 basierend auf den Datenpunkten erstellen, die von der Lambda-Funktion erzeugt werden. Wie in den vorherigen Beispielen kann dieser Ansatz so geändert werden, dass mehr oder weniger Datenpunkte in einem kürzeren oder längeren Fenster verwendet werden.

Fügen Sie diese Alarme dann zu Ihrem bestehenden Availability Zone-Verfügbarkeitsalarm für die Kombination aus Controller und Aktion hinzu, wie in der folgenden Abbildung dargestellt.



Integration des Chi-Quadrat-Statistiktests mit zusammengesetzten Alarmen

Wie bereits erwähnt, müssen Sie, wenn Sie neue Funktionen in Ihren Workload integrieren, nur die entsprechenden CloudWatch metrischen Alarme erstellen, die für diese neue Funktion spezifisch sind, und die nächste Stufe in der zusammengesetzten Alarm-Hierarchie aktualisieren, um diese Alarme einzubeziehen. Der Rest der Alarmstruktur bleibt statisch.

Fehlererkennung bei zonalen Ressourcen einer einzelnen Instanz

In einigen Fällen verfügen Sie möglicherweise über eine einzelne aktive Instanz einer zonalen Ressource, in den meisten Fällen Systeme, die eine Einzelschreiber-Komponente wie eine relationale Datenbank (wie AmazonRDS) oder einen verteilten Cache (wie Amazon ElastiCache (OSSRedis)) benötigen. Wenn sich eine Beeinträchtigung einer einzelnen Availability Zone auf die Availability Zone auswirkt, in der sich die primäre Ressource befindet, kann dies Auswirkungen auf jede Availability Zone haben, die auf die Ressource zugreift. Dies könnte dazu führen, dass Verfügbarkeitsschwellenwerte in jeder Availability Zone überschritten werden, was bedeutet, dass beim ersten Ansatz die einzelne Availability Zone, von der die Auswirkungen ausgehen, nicht korrekt identifiziert werden würde. Darüber hinaus würden Sie wahrscheinlich in jeder Availability Zone ähnliche Fehlerraten feststellen, was bedeutet, dass das Problem auch bei der Ausreißeranalyse nicht erkannt werden würde. Das bedeutet, dass Sie zusätzliche Beobachtbarkeit implementieren müssen, um dieses Szenario gezielt zu erkennen.

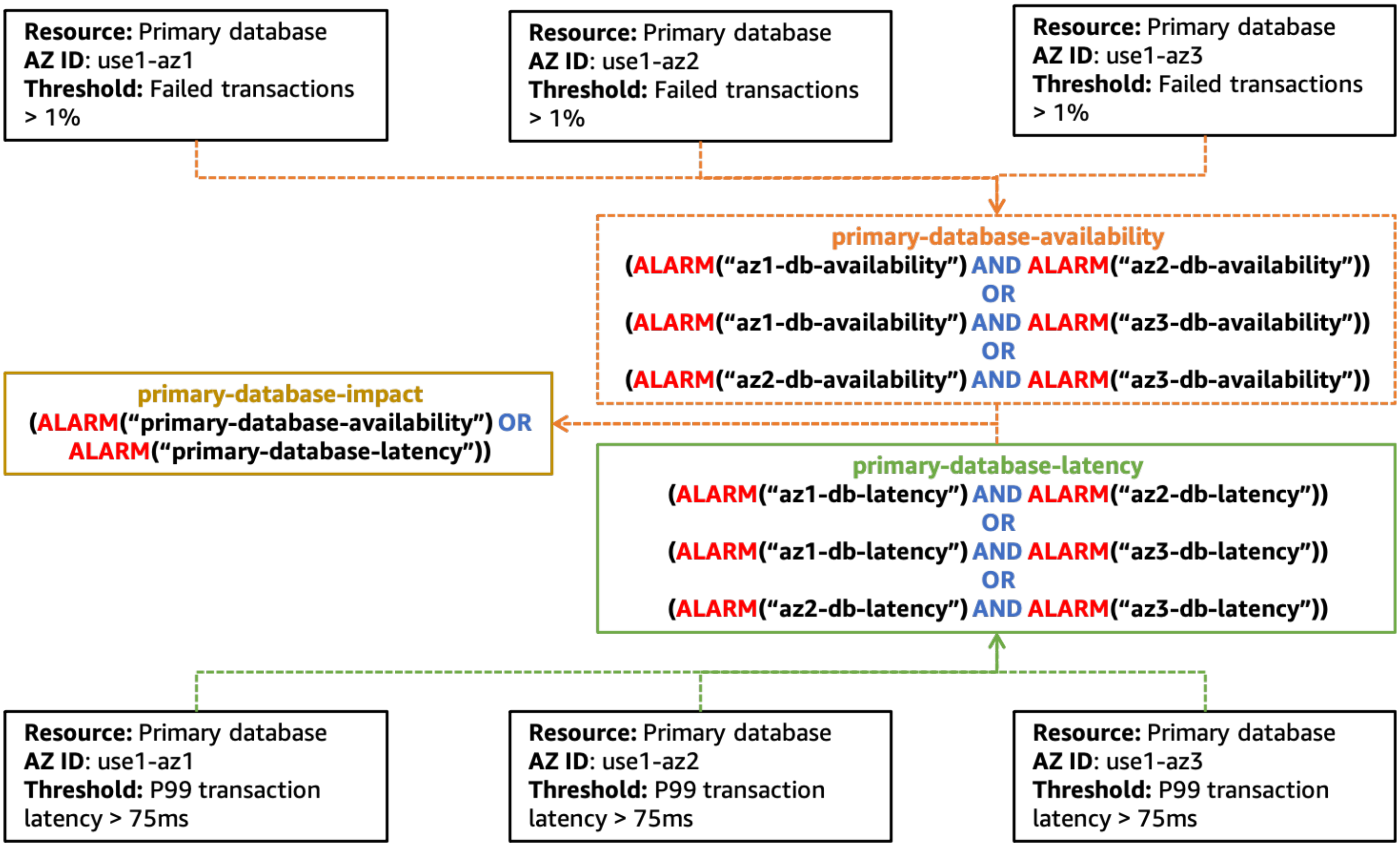
Es ist wahrscheinlich, dass die Ressource, um die Sie sich Sorgen machen, ihre eigenen Messwerte über ihren Zustand erstellt, aber während einer Beeinträchtigung der Availability Zone ist diese Ressource möglicherweise nicht in der Lage, diese Metriken zu liefern. In diesem Szenario sollten Sie Alarme erstellen oder aktualisieren, um zu wissen, wann Sie im Blindflug sind. Wenn es wichtige Messwerte gibt, die Sie bereits überwachen und bei denen Sie Alarme auslösen, können Sie den Alarm so konfigurieren, dass die fehlenden Daten als Sicherheitsverletzung behandelt werden. Auf diese Weise können Sie feststellen, ob die Ressource keine Daten mehr meldet, und Sie können diese Alarme in a row und m von n einschließen, die zuvor verwendet wurden.

Es ist auch möglich, dass in einigen Kennzahlen, die den Zustand der Ressource angeben, ein Datenpunkt mit einem Wert von Null veröffentlicht wird, wenn keine Aktivität stattfindet. Wenn die Beeinträchtigung Interaktionen mit der Ressource verhindert, können Sie den Ansatz fehlender Daten für diese Art von Kennzahlen nicht verwenden. Sie möchten wahrscheinlich auch keinen Alarm auslösen, wenn der Wert Null ist, da es legitime Szenarien geben könnte, in denen dieser Wert innerhalb der normalen Schwellenwerte liegt. Der beste Ansatz zur Erkennung dieser Art von Problemen besteht darin, Metriken von den Ressourcen zu erstellen, die diese Abhängigkeit verwenden. In diesem Fall möchten wir mithilfe zusammengesetzter Alarme Auswirkungen in mehreren Availability Zones erkennen. Diese Alarme sollten eine Handvoll kritischer Metrikkategorien verwenden, die sich auf die Ressource beziehen. Nachfolgend sind einige Beispiele aufgeführt:

- Durchsatz — Die Rate der eingehenden Arbeitseinheiten. Dies können Transaktionen, Lese- und Schreibvorgänge usw. sein.
- Verfügbarkeit — Messen Sie die Anzahl erfolgreicher und fehlgeschlagener Arbeitseinheiten.
- Latenz — Messen Sie mehrere Perzentile der Latenz für erfolgreiche Arbeit in kritischen Vorgängen.

  Auch hier können Sie die Alarme „In einer Reihe“ und „m aus n“ für jede Metrik in jeder Metrikkategorie, die Sie messen möchten, erstellen. Nach wie vor können diese zu einem zusammengesetzten Alarm kombiniert werden, um festzustellen, ob diese gemeinsam genutzte Ressource die Ursache für Auswirkungen in allen Availability Zones ist. Sie möchten in der

Lage sein, mit den kombinierten Alarmen Auswirkungen auf mehr als eine Availability Zone zu identifizieren, aber die Auswirkungen müssen nicht unbedingt alle Availability Zones betreffen. Die allgemeine Verbundalarmstruktur für diese Art von Ansatz ist in der folgenden Abbildung dargestellt.



Ein Beispiel für die Erstellung von Alarmen zur Erkennung von Auswirkungen auf mehrere Availability Zones, die durch eine einzelne Ressource verursacht werden

Sie werden feststellen, dass dieses Diagramm weniger aussagekräftig ist, welche Art von metrischen Alarmen verwendet werden sollte und wie die Hierarchie der zusammengesetzten Alarme aussieht. Das liegt daran, dass es schwierig sein kann, diese Art von Problem zu erkennen, und es ist daher erforderlich, sorgfältig auf die richtigen Signale für die gemeinsam genutzte Ressource zu achten. Diese Signale müssen möglicherweise auch auf spezifische Weise bewertet werden.

Darüber hinaus sollten Sie beachten, dass der `primary-database-impact` Alarm keiner bestimmten Availability Zone zugeordnet ist. Das liegt daran, dass sich die primäre Datenbank-Instance in jeder Availability Zone befinden kann, für deren Verwendung sie konfiguriert ist, und es keine CloudWatch Metrik gibt, die angibt, wo sie sich befindet. Wenn Sie sehen, dass dieser Alarm aktiviert wird, sollten Sie ihn als Signal verwenden, dass möglicherweise ein Problem mit der Ressource vorliegt, und einen Failover zu einer anderen Availability Zone einleiten, falls dieser nicht automatisch durchgeführt wurde. Nachdem Sie die Ressource in eine andere Availability Zone verschoben haben, können Sie abwarten, ob Ihr isolierter Availability Zone-Alarm aktiviert ist, oder Sie können Ihren Availability Zone-Evakuierungsplan präventiv aufrufen.

Übersicht

In diesem Abschnitt wurden drei Ansätze beschrieben, mit deren Hilfe Beeinträchtigungen in einzelnen Availability Zones identifiziert werden können. Jeder Ansatz sollte zusammen verwendet werden, um einen ganzheitlichen Überblick über den Zustand Ihres Workloads zu erhalten.

Der CloudWatch kombinierte Alarmansatz ermöglicht es Ihnen, Probleme zu finden, bei denen der Verfügbarkeitsunterschied statistisch nicht signifikant ist, z. B. Verfügbarkeiten von 98% (die beeinträchtigte Availability Zone), 100% und 99,99%, die nicht auf eine einzelne, gemeinsam genutzte Ressource zurückzuführen sind.

Die Ausreißererkennung hilft dabei, Beeinträchtigungen einzelner Availability Zones zu erkennen, wenn in mehreren Availability Zones unkorrelierte Fehler auftreten, die alle Ihren Alarmschwellenwert überschreiten.

Und schließlich hilft die Identifizierung der Beeinträchtigung der zonalen Ressource einer einzelnen Instanz dabei, herauszufinden, ob sich eine Beeinträchtigung der Availability Zone auf eine Ressource auswirkt, die von mehreren Availability Zones gemeinsam genutzt wird.

Die Alarme, die sich aus jedem dieser Muster ergeben, können zu einer CloudWatch zusammengesetzten Alarmhierarchie zusammengefasst werden, um zu ermitteln, wann Beeinträchtigungen einzelner Availability Zones auftreten und sich auf die Verfügbarkeit oder Latenz Ihrer Arbeitslast auswirken.

Evakuierungsmuster der Availability Zone

Nachdem die Auswirkungen in einer einzelnen Availability Zone festgestellt wurden, besteht der nächste Schritt darin, diese Availability Zone zu evakuieren. Es gibt zwei Ergebnisse, die mit der Evakuierung erreicht werden müssen.

Zunächst möchten Sie das Senden von Arbeit an die betroffene Availability Zone beenden. Dies könnte in verschiedenen Architekturen unterschiedliche Dinge bedeuten. Bei einem Request/ Response-Workload würde dies bedeuten, dass Dinge wie HTTP- oder gRPC-Anfragen Ihrer Kunden nicht mehr an den Load Balancer oder andere Ressourcen in der Availability Zone gesendet werden. In einem System zur Stapelverarbeitung oder zur Warteschlangenverarbeitung kann dies bedeuten, dass die Rechenressourcen in der betroffenen Availability Zone daran gehindert werden, Arbeit zu verarbeiten. Sie müssen außerdem verhindern, dass Ressourcen in den nicht betroffenen Availability Zones mit Ressourcen in der betroffenen Availability Zone interagieren, z. B. eine EC2-Instance, die Datenverkehr an eineSchnittstelle VPC-Endpunktin der betroffenen Availability Zone oder beim Herstellen einer Verbindung mit der primären Instanz einer Datenbank.

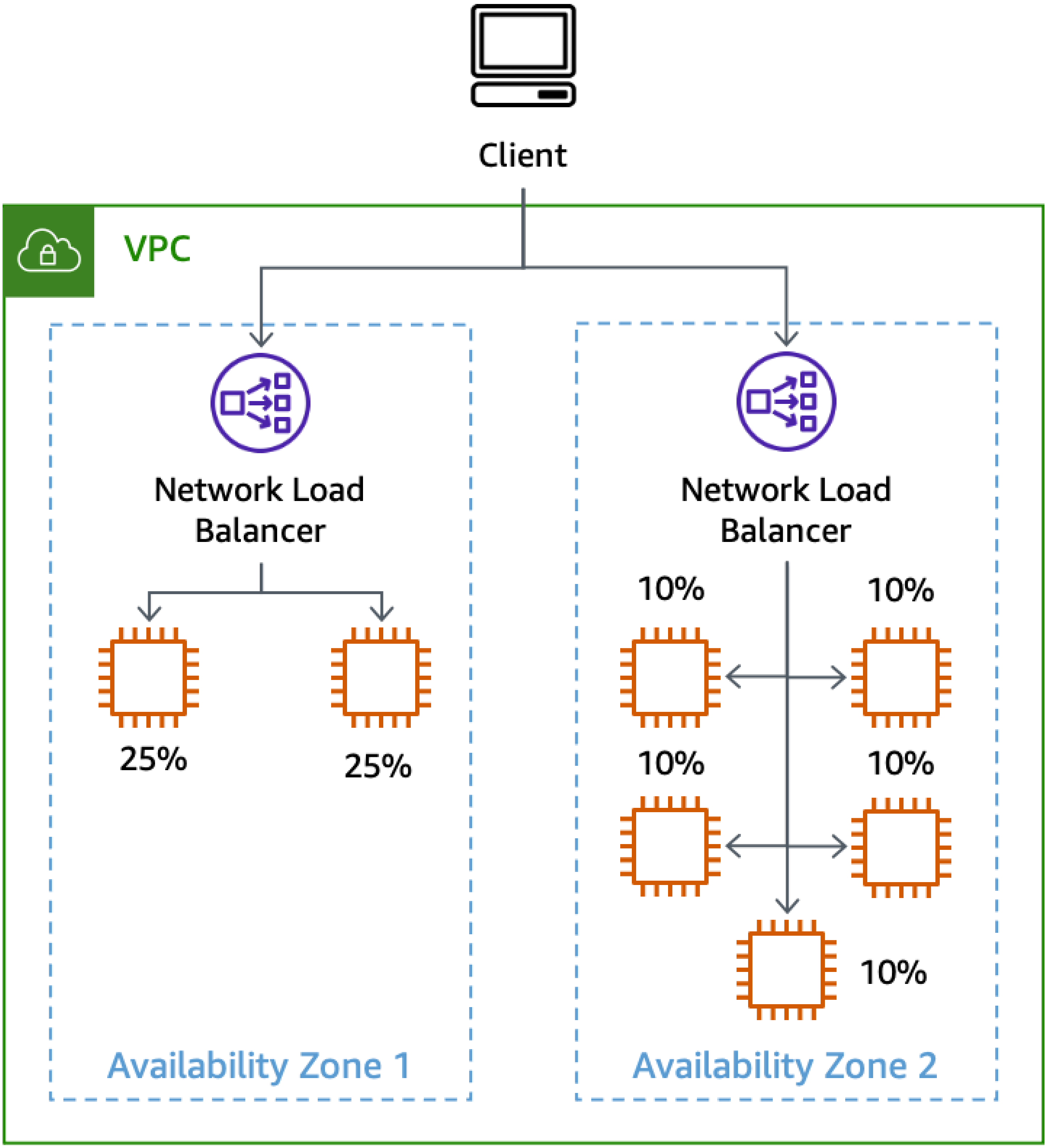
Das zweite Ergebnis besteht darin, die Bereitstellung neuer Kapazitäten in der betroffenen Availability Zone zu verhindern. Dies ist wichtig, da neue Ressourcen wie EC2-Instances oder Container, die in der betroffenen Availability Zone bereitgestellt werden, wahrscheinlich die gleichen Auswirkungen haben wie bestehende Ressourcen. Da das erste Ergebnis verhindert, dass ihnen Arbeit zugestellt wird, können sie außerdem die Last, für die sie bereitgestellt wurden, nicht aufnehmen. Dies führt zu einer erhöhten Belastung der vorhandenen Ressourcen, was letztendlich dazu führen kannbraun werdenoder vollständige Nichtverfügbarkeit des Workloads. Es sind mehrere Auto-Scaling-Dienste verfügbar inAWSwo dies zutrifft:Amazon EC2 Automatische Skalierung,Automatische Skalierung von Anwendungen, undAWS Auto Scaling. Darüber hinaus Dienste wie Amazon ECS, Amazon EKS undAWS Batchkann die Arbeit auf Hosts in allen Availability Zones in einer VPC als Teil ihres normalen Betriebs planen.

Themen

Verfügbarkeit, Zonenunabhängigkeit

Um das erste Ergebnis zu erzielen, d. h. die Übertragung von Arbeit an die betroffene Availability Zone einzustellen, müssen Sie Folgendes implementieren:Verfügbarkeitszone: Independence(AZI), manchmal auch genanntVerfügbarkeitszone Affinity. Dieses Architekturmuster isoliert Ressourcen innerhalb einer Availability Zone und verhindert Interaktionen zwischen Ressourcen in verschiedenen Availability Zones, sofern dies nicht unbedingt erforderlich ist, z. B. die Verbindung zu einer primären Datenbankinstanz in einer anderen Availability Zone.

Bei einem Workload vom Typ Anforderung/Antwort müssen Sie bei der Implementierung von AZIzonenübergreifendes Load Balancing für Application Load Balancers deaktivieren(WEISS),Klassische Load Balancer(CLB) undNetzwerk-Load-Balancer(NLB) (Der zonenübergreifende Lastenausgleich ist für NLBs standardmäßig deaktiviert). Die Deaktivierung des zonenübergreifenden Lastenausgleichs bringt einige Kompromisse mit sich. Wenn Sie den zonenübergreifenden Lastenausgleich deaktivieren,Der Verkehr wird gleichmäßig auf jede Availability Zone aufgeteiltunabhängig davon, wie viele Instanzen es in jedem gibt. Wenn Sie über unausgeglichene Ressourcen oder Auto Scaling-Gruppen verfügen, kann dies die Ressourcen in einer Availability Zone, die über weniger Ressourcen verfügt als andere, zusätzlich belasten. Dies ist in der folgenden Abbildung dargestellt, wo zwei Instances in Availability Zone 1 jeweils 25% der Last und die fünf Instances in Availability Zone 2 jeweils 10% der Last erhalten.



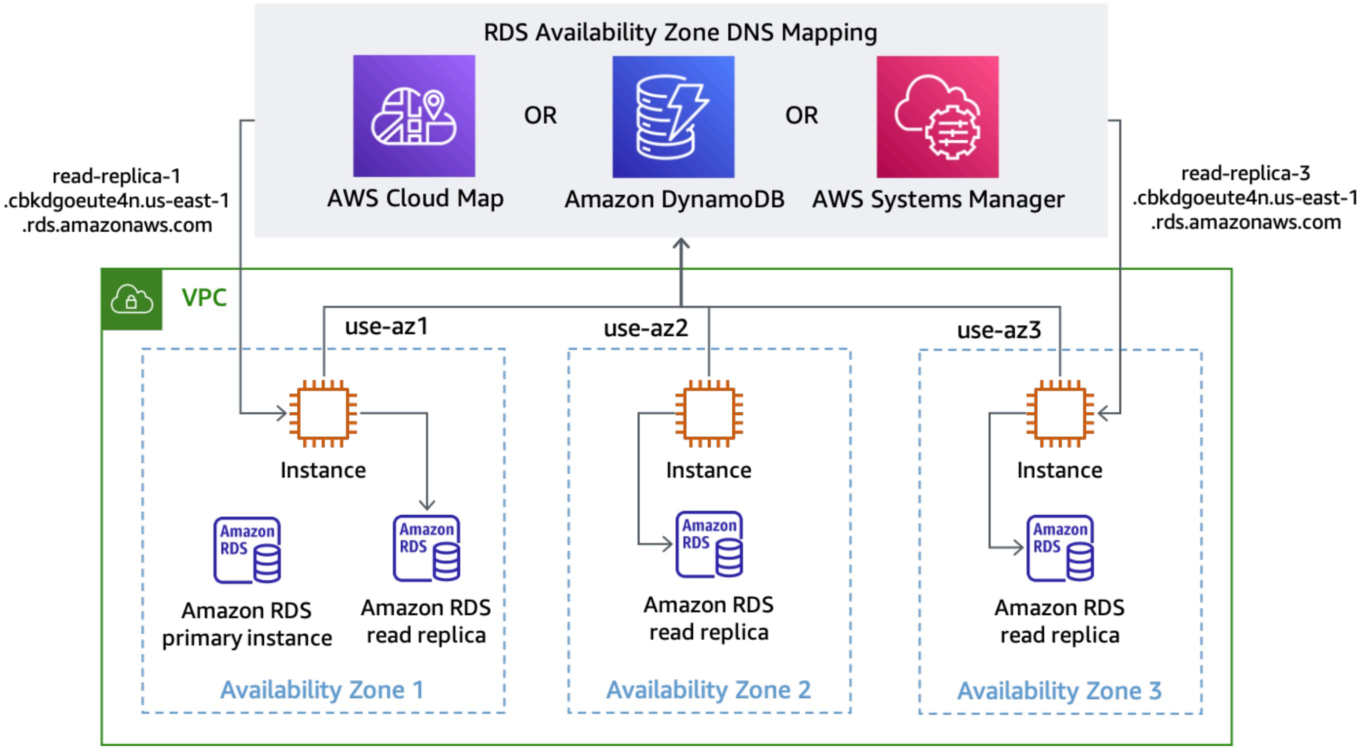
Die Auswirkung der Deaktivierung des zonenübergreifenden Lastenausgleichs bei unausgeglichenen Instances

Andere zonale Dienste, die Sie verwenden, müssen ebenfalls mithilfe von AZI-Mustern implementiert werden, um eine effektive Evakuierung der Availability Zone zu unterstützen. VPC-Endpunkte bieten beispielsweise Schnittstellenspezifische DNS-Namen für jede Availability ZoneDer Schnittstellenendpunkt wird in zur Verfügung gestellt.

Eine Herausforderung bei der Implementierung von AZI sind Datenbanken, insbesondere weil die meisten relationalen Datenbanken jeweils nur einen einzigen primären Writer unterstützen. Bei der Kommunikation mit der primären Instance müssen Sie möglicherweise eine Availability Zone-Grenze überschreiten. VieleAWSDatenbankdienste unterstützen eine benutzerdefinierte Multi-AZ-Konfiguration und verfügen über eine integrierte Multi-AZ-Failover-Funktion, wieAmazon RDSoderAmazonas Aurora. In vielen Ausfallszenarien kann der Service die Auswirkungen erkennen und automatisch ein Failover der Datenbank in eine andere Availability Zone durchführen, wenn ein Problem auftritt. Bei einem grauen Ausfall erkennt der Service jedoch möglicherweise nicht, welche Auswirkungen sich auf Ihren Workload auswirken, oder die Auswirkungen hängen möglicherweise überhaupt nicht mit der Datenbank zusammen. In diesen Fällen können Sie, sobald Sie Auswirkungen in einer Availability Zone feststellen, manuell einen Failover auslösen, um die primäre Datenbank zu verschieben. Auf diese Weise können Sie effektiv auf eine Beeinträchtigung einer einzelnen Availability Zone reagieren.

Wenn Sie Read Replicas mit diesen Datenbanken verwenden, sollten Sie AZI auch für diese Datenbanken implementieren, da Sie ein Failover eines Read Replicas nicht auf eine andere Availability Zone durchführen können, wie dies bei der primären Datenbank der Fall ist. Wenn Sie ein einzelnes Read Replica in Availability Zone 1 haben und Instances in drei Availability Zones so konfiguriert sind, dass sie es verwenden, wirkt sich eine Beeinträchtigung, die sich auf Availability Zone 1 auswirkt, auch auf den Betrieb in den anderen beiden Availability Zones aus. Das sind die Auswirkungen, die Sie verhindern möchten.

Für RDS-Instances erhalten Sie einen DNS-Endpunkt für den Zugriff auf das Replikat in einer bestimmten Availability Zone. Um AZI zu erreichen, benötigen Sie ein Read Replica pro Availability Zone und eine Möglichkeit, damit Ihre Anwendung weiß, welcher Replikat-Endpunkt für die Availability Zone verwendet werden muss, in der sie sich befindet. Ein Ansatz, den Sie wählen können, besteht darin, die Availability Zone-ID als Teil der Datenbank-ID zu verwenden, etwa `wieuse1-az1-read-replica.cbkdgoeute4n.us-east-1.rds.amazonaws.com`. Sie können dies auch mithilfe von Service Discovery tun (z. B. mitAWS Cloud Map) oder nach einer einfachen Karte suchen, die in gespeichert istAWSSystems Manager-Parameterspeicheroder eine DynamoDB-Tabelle. Dieses Konzept ist in der folgenden Abbildung dargestellt.

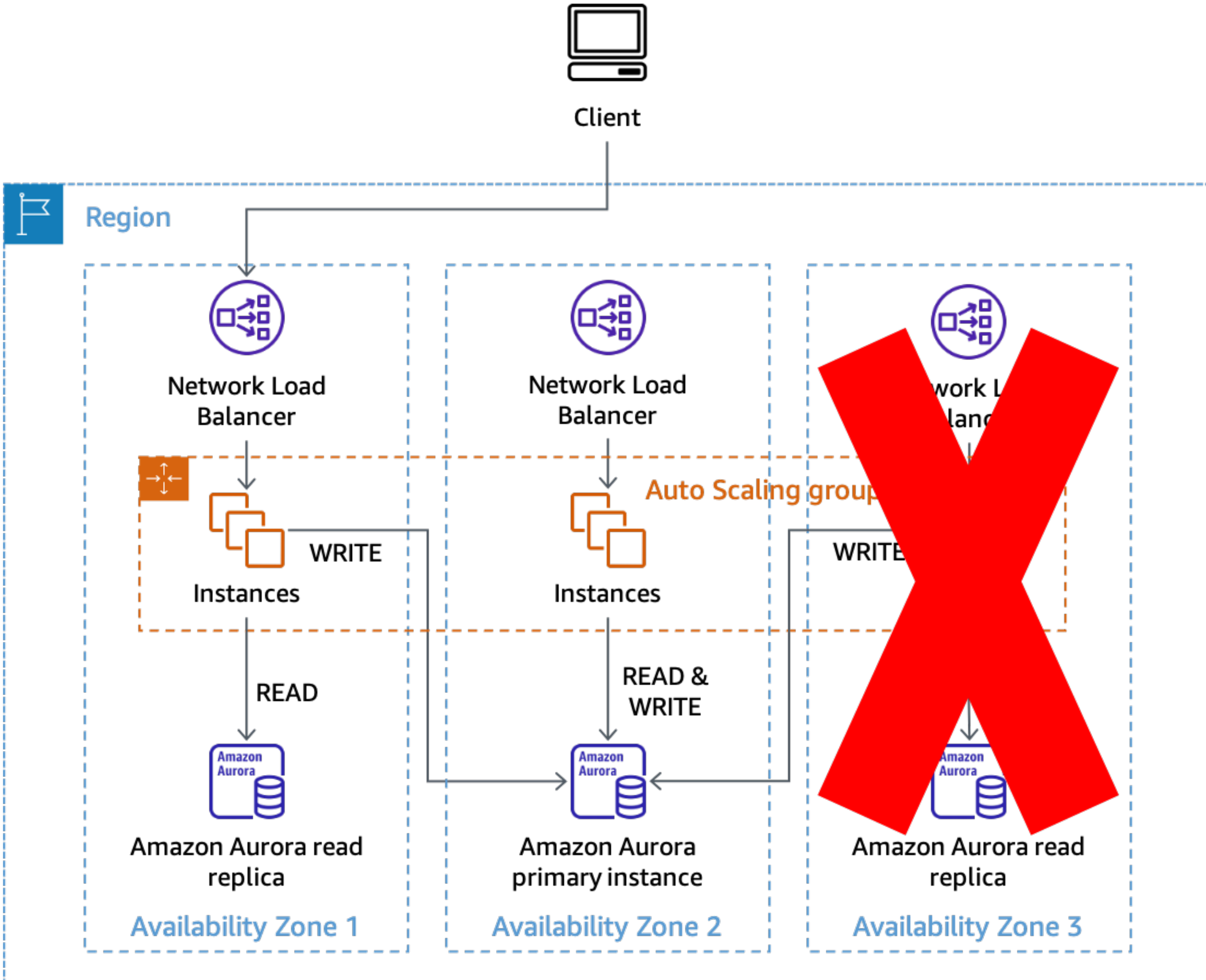


Ermittlung der DNS-Namen von RDS-Endpunkten für jede Availability Zone

Die Standardkonfiguration von Amazon Aurora besteht darin, eine bereitzustelleneinziger Reader-Endpunktdas Lastausgleich von Anfragen auf die verfügbaren Read Replicas durchführt. Um AZI mit Aurora zu implementieren, können Sie einebenutzerdefinierter Endpunktfür jedes Read-Replikat mit demANYgeben Sie ein (damit Sie bei Bedarf ein Read Replica hochstufen können). Benennen Sie den benutzerdefinierten Endpunkt anhand der Availability Zone-ID, in der das Replikat bereitgestellt wird. Anschließend können Sie den vom benutzerdefinierten Endpunkt bereitgestellten DNS-Namen verwenden, um eine Verbindung zu einem bestimmten Read Replica in einer bestimmten Availability Zone herzustellen, wie in der folgenden Abbildung dargestellt.

Einen benutzerdefinierten Endpunkt für ein Aurora-Read Replica verwenden

Wenn Ihr System auf diese Weise aufgebaut ist, wird die Evakuierung der Availability Zone zu einer viel einfacheren Aufgabe. In der folgenden Abbildung sind beispielsweise bei einer Beeinträchtigung der Availability Zone 3 sowohl Lese- als auch Schreibvorgänge in den Availability Zones 1 und 2 nicht betroffen.



Verwendung von AZI zur Vermeidung von Auswirkungen auf Amazon Aurora Read Replicas

Wenn die Availability Zone 2 betroffen wäre, würden die Lesevorgänge in den Availability Zones 1 und 3 alternativ weiterhin erfolgreich sein. Wenn Amazon Aurora dann nicht automatisch ein Failover der primären Datenbank durchgeführt hat, können Sie manuell einen Failover zu einer anderen Availability Zone aufrufen, um die Fähigkeit zur Verarbeitung von Schreibvorgängen wiederherzustellen. Dieser Ansatz verhindert, dass Sie Konfigurationsänderungen an Ihren Datenbankverbindungen vornehmen müssen, wenn Sie eine Availability Zone evakuieren müssen. Wenn Sie die erforderlichen Änderungen minimieren und den Prozess so einfach wie möglich gestalten, wird er zuverlässiger.

Steuerungsebenen und Datenebenen

Bevor wir zu den tatsächlichen Mustern kommen, die Sie für die Evakuierung einer Availability Zone verwenden können, müssen wir die Konzepte von Steuerungsebenen und Datenebenen erörtern.AWSunterscheidet in unseren Diensten zwischen Steuerebenen und Datenebenen. Steuerungsebenen sind die Maschinen, die daran beteiligt sind, Änderungen an einem System vorzunehmen — Ressourcen hinzuzufügen, Ressourcen zu löschen, Ressourcen zu ändern — und diese Änderungen an die Stellen zu übertragen, an denen sie wirksam werden müssen, z. B. beim Aktualisieren einer Netzwerkkonfiguration für ein ALB oder beim Erstellen einesAWS LambdaFunktion.

Datenebenen sind die Hauptfunktion dieser Ressourcen, z. B. die laufende EC2-Instance oder das Abrufen von Elementen aus einer Amazon DynamoDB-Tabelle oder das Einfügen von Elementen in eine Amazon DynamoDB-Tabelle. Eine detailliertere Erläuterung der Steuerungsebenen und Datenebenen finden Sie unterStatische Stabilität mithilfe von Availability ZonesundAWSGrenzen der Fehlerisolierung.

Beachten Sie für die Zwecke dieses Dokuments, dass Steuerungsebenen in der Regel mehr bewegliche Teile und Abhängigkeiten aufweisen als Datenebenen. Dadurch ist die Wahrscheinlichkeit, dass die Steuerebene beeinträchtigt wird, statistisch höher als die Datenebene. Dies ist besonders relevant für Dienste, die AZI bereitstellen, wie Amazon EC2 und EBS, da Teile dieser Dienste über Steuerungsebenen verfügen, die auch zonenunabhängig sind und bei einem Single-AZ-Ereignis beeinträchtigt werden können.

Aktionen auf der Steuerungsebene können zwar verwendet werden, um eine AZ-Evakuierung durchzuführen, auf der Grundlage der vorherigen Informationen haben sie jedoch möglicherweise eine geringere Erfolgswahrscheinlichkeit, insbesondere bei einem Störungsereignis. Um die Wahrscheinlichkeit zu erhöhen, dass die Auswirkungen erfolgreich gemildert werden, können Sie zwei verschiedene Muster verwenden. Das erste Muster stützt sich nur auf Aktionen auf der Datenebene, um zunächst die Auswirkungen zu mildern, indem verhindert wird, dass Arbeit an die betroffene Availability Zone weitergeleitet wird oder dass Arbeit in der betroffenen Availability Zone beendet wird. Anschließend kann mit dem zweiten Muster versucht werden, die Konfiguration der Ressourcen mit Aktionen auf der Steuerungsebene zu aktualisieren, um sowohl zu verhindern, dass Kapazität in der betroffenen Availability Zone bereitgestellt wird, als auch die Kommunikation zwischen den Availability Zones mit dieser Availability Zone zu beenden.

Die in diesem Abschnitt erörterten Wiederherstellungsmuster sindgroße rote Knöpfe. Sie sind die Mechanismen, die Sie verwenden, um schnell groß angelegte Maßnahmen zu ergreifen, ähnlich wie beim Ziehen einesAndonkabel an einer Montagelinie. Sie gehen davon aus, dass die Workloads bereits Strategien ausprobiert haben, wiewiederholen Sie den Vorgang mit exponentiellem Backoff mit Jitterin ihrem Code, um vorübergehende Fehler zu überwinden. Dies bedeutet, dass, wenn isolierte Auswirkungen auf die Availability Zone erkannt werden, deren Auswirkungen auf die Verfügbarkeit oder Latenz so schwerwiegend sind, dass die Availability Zone evakuiert werden muss, um sie wirksam zu mindern.

Datenplangesteuerte Evakuierung

Es gibt mehrere Lösungen, die Sie implementieren können, um eine Availability Zone-Evakuierung mithilfe von Aktionen nur auf der Datenebene durchzuführen. In diesem Abschnitt werden drei davon und die Anwendungsfälle beschrieben, in denen Sie möglicherweise einen dem anderen vorziehen sollten.

Wenn Sie eine dieser Lösungen verwenden, müssen Sie sicherstellen, dass Sie in den verbleibenden Availability Zones über ausreichend Kapazität verfügen, um die Last der Availability Zone zu bewältigen, von der Sie wegwechseln. Der stabilste Weg, dies zu erreichen, besteht darin, die erforderliche Kapazität in jeder Availability Zone vorab bereitzustellen. Wenn Sie drei Availability Zones verwenden, würden Sie in jeder Zone 50% der für die Bewältigung Ihrer Spitzenlast erforderlichen Kapazität bereitstellen, sodass Ihnen beim Verlust einer einzigen Availability Zone immer noch 100% Ihrer benötigten Kapazität zur Verfügung stehen, ohne sich auf eine Steuerungsebene verlassen zu müssen, um mehr bereitzustellen.

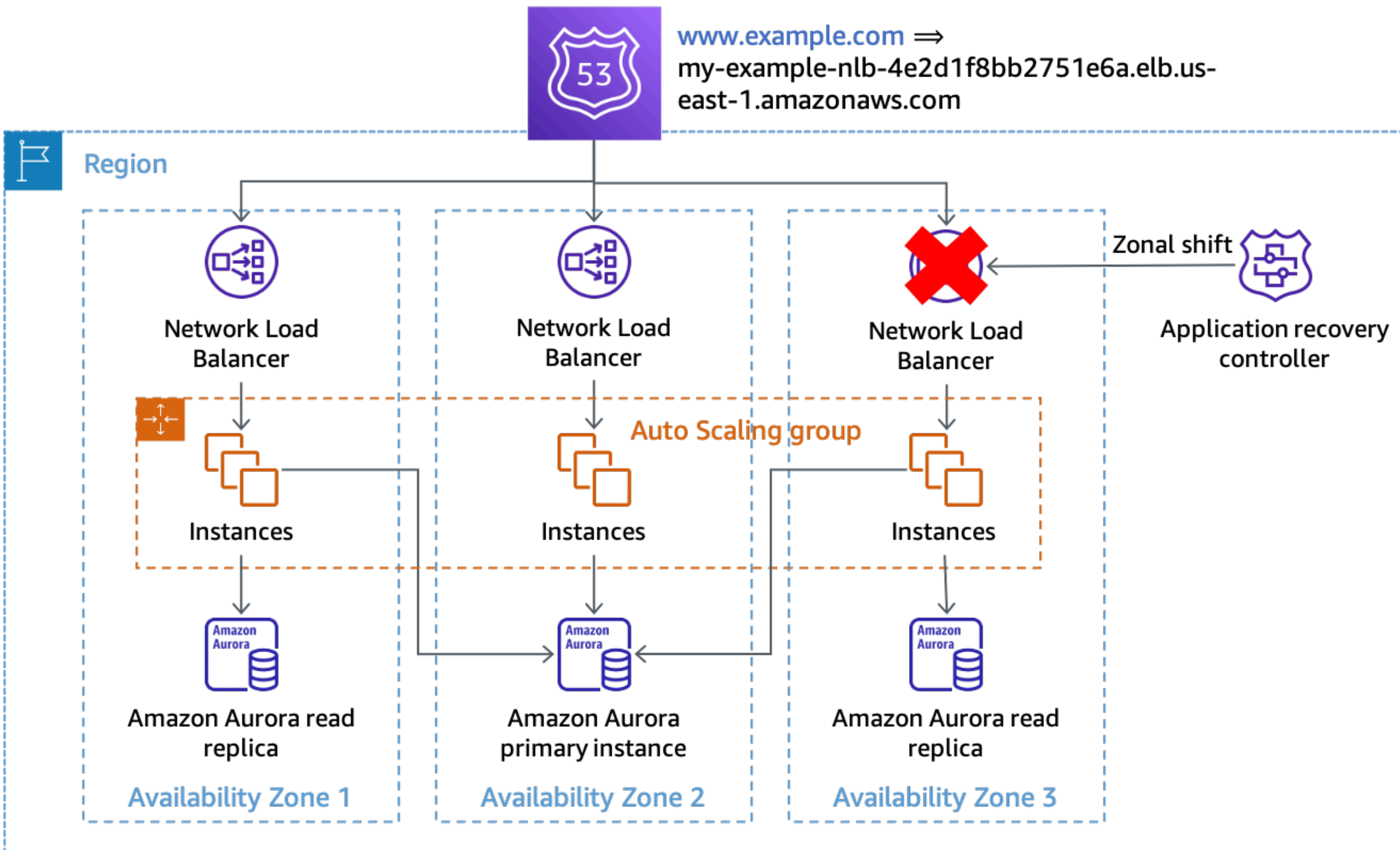
Wenn Sie EC2 Auto Scaling verwenden, stellen Sie außerdem sicher, dass Ihre Auto Scaling-Gruppe (ASG) während der Schicht nicht skaliert, sodass Sie am Ende der Schicht immer noch über genügend Kapazität in der Gruppe verfügen, um Ihren Kundenverkehr abzuwickeln. Sie können dies tun, indem Sie sicherstellen, dass die gewünschte Mindestkapazität Ihrer ASG Ihre aktuelle Kundenlast bewältigen kann. Sie können auch sicherstellen, dass Ihr ASG nicht versehentlich skaliert, indem Sie in Ihren Kennzahlen Durchschnittswerte verwenden und nicht Perzentilmetriken wie P90 oder P99 als Ausreißer verwenden.

Während einer Schicht sollten die Ressourcen, die den Verkehr nicht mehr bedienen, eine sehr geringe Auslastung aufweisen, aber die anderen Ressourcen werden ihre Auslastung mit dem neuen Verkehr erhöhen, sodass der Durchschnitt ziemlich konstant bleibt, wodurch eine Scale-In-Aktion verhindert würde. Schließlich können Sie die Gesundheitseinstellungen für Zielgruppen auch verwenden fürALBundNLBum DNS-Failover entweder mit einem Prozentsatz oder einer Anzahl intakter Hosts anzugeben. Dadurch wird verhindert, dass Datenverkehr an eine Availability Zone weitergeleitet wird, die nicht über genügend gesunde Hosts verfügt.

Zonale Verschiebung im Route 53 Application Recovery Controller (ARC)

Die erste Lösung für die Evakuierung von Availability ZonesZonenverschiebung auf der Route 53 ARC. Diese Lösung kann für Anforderungs-/Antwort-Workloads verwendet werden, die eine NLB oder ALB als Eingangspunkt für den Kundenverkehr verwenden.

Wenn Sie feststellen, dass eine Availability Zone beeinträchtigt ist, können Sie mit Route 53 ARC eine Zonenverschiebung einleiten. Sobald dieser Vorgang abgeschlossen ist und vorhandene zwischengespeicherte DNS-Antworten ablaufen, werden alle neuen Anfragen nur an Ressourcen in den verbleibenden Availability Zones weitergeleitet. Die folgende Abbildung zeigt, wie die zonale Verschiebung funktioniert. In der folgenden Abbildung haben wir einen Route 53-Aliasdatensatz für`www.example.com`das deutet darauf hin`my-example-nlb-4e2d1f8bb2751e6a.elb.us-east-1.amazonaws.com`. Die Zonenverschiebung wird für Availability Zone 3 durchgeführt.



Zonale Verschiebung

Befindet sich im Beispiel die primäre Datenbankinstanz nicht in Availability Zone 3, ist die Durchführung der Zonenverschiebung die einzige Aktion, die erforderlich ist, um das erste Ergebnis der Evakuierung zu erzielen. Dadurch wird verhindert, dass Arbeit in der betroffenen Availability Zone verarbeitet wird. Befindet sich der primäre Knoten in Availability Zone 3, könnten Sie in Abstimmung mit der Zonenverschiebung einen manuell initiierten Failover durchführen (der auf der Amazon RDS-Steuerungsebene basiert), sofern Amazon RDS nicht bereits automatisch ein Failover durchgeführt hat. Dies gilt für alle datenplangesteuerten Lösungen in diesem Abschnitt.

Sie sollten die Zonenverschiebung mithilfe von CLI-Befehlen oder der API einleiten, um die Abhängigkeiten zu minimieren, die für den Start der Evakuierung erforderlich sind. Je einfacher der Evakuierungsprozess ist, desto zuverlässiger wird er sein. Die spezifischen Befehle können in einem lokalen Runbook gespeichert werden, auf das Techniker im Bereitschaftsdienst problemlos zugreifen können. Zonal Shift ist die bevorzugte und einfachste Lösung für die Evakuierung einer Availability Zone.

Route 53 ARC

Die zweite Lösung nutzt die Funktionen von Route 53 ARC, um den Zustand bestimmter DNS-Einträge manuell zu spezifizieren. Diese Lösung hat den Vorteil, dass sie die hochverfügbare Route 53 ARC-Cluster-Datenebene verwendet, wodurch sie widerstandsfähig gegenüber Beeinträchtigungen durch bis zu zwei verschiedeneAWS-Regionen. Es hat den Nachteil zusätzlicher Kosten und erfordert eine zusätzliche Konfiguration von DNS-Einträgen. Um dieses Muster zu implementieren, müssen Sie Aliasdatensätze für das erstellenAvailability Zonenspezifische DNS-Namenbereitgestellt vom Load Balancer (ALB oder NLB). Dies ist in der folgenden Tabelle dargestellt.

Tabelle 3: Route 53-Aliaseinträge, die für die zonalen DNS-Namen des Load Balancers konfiguriert sind

Routing-Richtlinie: gewichtet	Routing-Richtlinie:gewichtet	Routing-Richtlinie:gewichtet
Name (Name: `www.example.com`	Name (Name: `www.example.com`	Name (Name: `www.example.com`
Typ:`A`(Alias)	Typ: `A`(Alias)	Typ: `A`(Alias)
Value (Wert): `us-east-1b.load-balancer-name.elb.us-east-1.amazonaws.com`	Wert: `us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com`	Wert: `us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`
Gewicht:`100`	Gewicht: `100`	Gewicht: `100`

Bewerten Sie die Gesundheit des Ziels: wahr	Bewerten Sie den Zustand des Ziels: `true`	Bewerten Sie den Zustand des Ziels: `true`

Für jeden dieser DNS-Einträge würden Sie eine Route 53-Integritätsprüfung konfigurieren, die mit einem Route 53-ARC verknüpft ist.Routing-Steuerung. Wenn Sie eine Evakuierung der Availability Zone einleiten möchten, setzen Sie den Status der Routing-Kontrolle auf`Off`.AWSempfiehlt, dies mithilfe der CLI oder API zu tun, um die Abhängigkeiten zu minimieren, die erforderlich sind, um die Evakuierung der Availability Zone zu starten. Alsbeste Praxis, sollten Sie eine lokale Kopie der Route 53 ARC-Cluster-Endpunkte aufbewahren, damit Sie diese nicht aus der ARC-Steuerebene abrufen müssen, wenn Sie eine Evakuierung durchführen müssen.

Um die Kosten bei der Verwendung dieses Ansatzes zu minimieren, können Sie einen einzigen Route 53-ARC-Cluster erstellen und die Systemdiagnosen in einem einzigen Schritt durchführen.AWS-Kontoundteilen Sie die Gesundheitschecks mit anderenAWS-Kontenin Ihrer Organisation. Wenn Sie diesen Ansatz wählen, sollten Sie dieID der Verfügbarkeitszone(AZ-ID) (zum Beispiel`use1-az1`) statt des Namens der Availability Zone (zum Beispiel`us-east-1a`) für Ihre Routing-Kontrollen. WeilAWSordnet die physische Availability Zone nach dem Zufallsprinzip den Availability Zone-Namen für jede Zone zuAWS-Konto, die Verwendung der AZ-ID bietet eine konsistente Möglichkeit, auf dieselben physischen Standorte zu verweisen. Wenn Sie eine Evakuierung der Availability Zone einleiten, sagen wir für`use1-az2`, die Route 53-Datensätze in jedemAWS-Kontosollten sicherstellen, dass sie das AZ-ID-Mapping verwenden, um den richtigen Gesundheitscheck für jeden NLB-Datensatz zu konfigurieren.

Nehmen wir zum Beispiel an, wir haben eine Route 53-Zustandsprüfung, die mit einer Route 53 ARC-Routingsteuerung verknüpft ist für`use1-az2`, mit einer ID von`0385ed2d-d65c-4f63-a19b-2412a31ef431`. Wenn in einem anderenAWS-Kontoder diesen Gesundheitscheck nutzen will,`us-east-1c`wurde zugeordnet`use1-az2`, du müsstest den`use1-az2`Gesundheitscheck für das Protokoll`us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`. Sie würden die Gesundheitscheck-ID verwenden`0385ed2d-d65c-4f63-a19b-2412a31ef431`mit diesem Ressourcendatensatz.

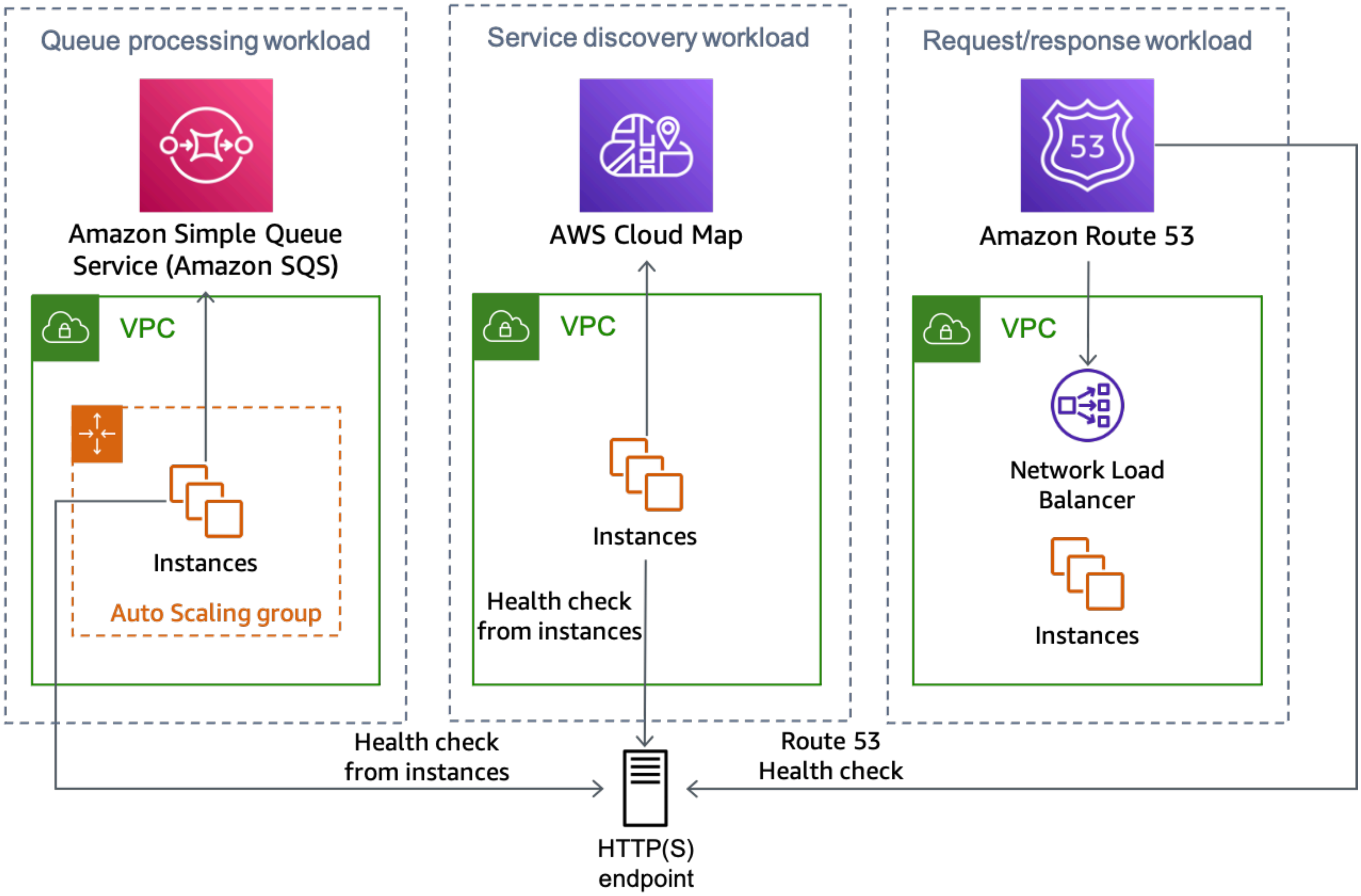
Verwendung eines selbstverwalteten HTTP-Endpunkts

Sie können diese Lösung auch implementieren, indem Sie Ihren eigenen HTTP-Endpunkt verwalten, der den Status einer bestimmten Availability Zone angibt. Damit können Sie anhand der Antwort des HTTP-Endpunkts manuell angeben, wann eine Availability Zone fehlerhaft ist. Diese Lösung kostet weniger als die Verwendung von Route 53 ARC, ist aber teurer als Zonal Shift und erfordert die Verwaltung zusätzlicher Infrastruktur. Es hat den Vorteil, dass es für verschiedene Szenarien viel flexibler ist.

Das Muster kann mit NLB- oder ALB-Architekturen und Route 53-Gesundheitschecks verwendet werden. Es kann auch in Architekturen ohne Lastausgleich eingesetzt werden, wie etwa in Systemen zur Serviceerkennung oder zur Verarbeitung von Warteschlangen, in denen Worker-Knoten ihre eigenen Integritätsprüfungen durchführen. In diesen Szenarien können die Hosts einen Hintergrund-Thread verwenden, in dem sie regelmäßig mit ihrer AZ-ID eine Anfrage an den HTTP-Endpunkt stellen (sieheAnhang A — Abrufen der Availability Zone-ID um zu erfahren, wie Sie das finden) und erhalten Sie eine Antwort über den Zustand der Availability Zone.

Wenn die Availability Zone für fehlerhaft erklärt wurde, haben sie mehrere Optionen, wie sie reagieren können. Sie können sich dafür entscheiden, eine externe Integritätsprüfung aus Quellen wie ELB, Route 53 oder benutzerdefinierte Integritätsprüfungen in Service Discovery-Architekturen nicht zu bestehen, sodass sie für diese Dienste als fehlerhaft erscheinen. Sie können auch sofort mit einem Fehler antworten, falls sie eine Anfrage erhalten, sodass der Client einen Rückzieher machen und es erneut versuchen kann. In ereignisgesteuerten Architekturen können Knoten absichtlich daran scheitern, Arbeit zu verarbeiten, z. B. indem sie absichtlich eine SQS-Nachricht an die Warteschlange zurücksenden. In Router-Architekturen, in denen zentrale Dienstpläne auf bestimmten Hosts funktionieren, können Sie dieses Muster ebenfalls verwenden. Der Router kann den Status einer Availability Zone überprüfen, bevor er einen Worker, einen Endpunkt oder eine Zelle auswählt. In Service Discovery-Architekturen, die Folgendes verwendenAWS Cloud Map, du kannstEntdecken Sie Endpunkte, indem Sie in Ihrer Anfrage einen Filter angeben, wie zum Beispiel eine AZ-ID.

Die folgende Abbildung zeigt, wie dieser Ansatz für mehrere Arten von Workloads verwendet werden kann.



Die HTTP-Endpunktlösung kann von mehreren Workload-Typen verwendet werden

Es gibt mehrere Möglichkeiten, den HTTP-Endpunktansatz zu implementieren. Zwei davon werden im Folgenden beschrieben.

Verwenden von Amazon S3

Dieses Muster wurde ursprünglich in diesem vorgestelltBlogbeitragfür die Notfallwiederherstellung in mehreren Regionen. Sie können dasselbe Muster für die Evakuierung der Availability Zone verwenden.

In diesem Szenario würden Sie Route 53-DNS-Ressourceneintragssätze für jeden zonalen DNS-Eintrag erstellen, genau wie derRoute 53 ARCdas obige Szenario sowie die zugehörigen Gesundheitschecks. Für diese Implementierung werden die Integritätsprüfungen jedoch nicht mit Route 53 ARC-Routing-Steuerelementen verknüpft, sondern so konfiguriert, dass sie eineHTTP-Endpunktund sind invertiert, um zu verhindern, dass eine Beeinträchtigung in Amazon S3 versehentlich eine Evakuierung auslöst. Der Gesundheitscheck wird berücksichtigtgesundwenn das

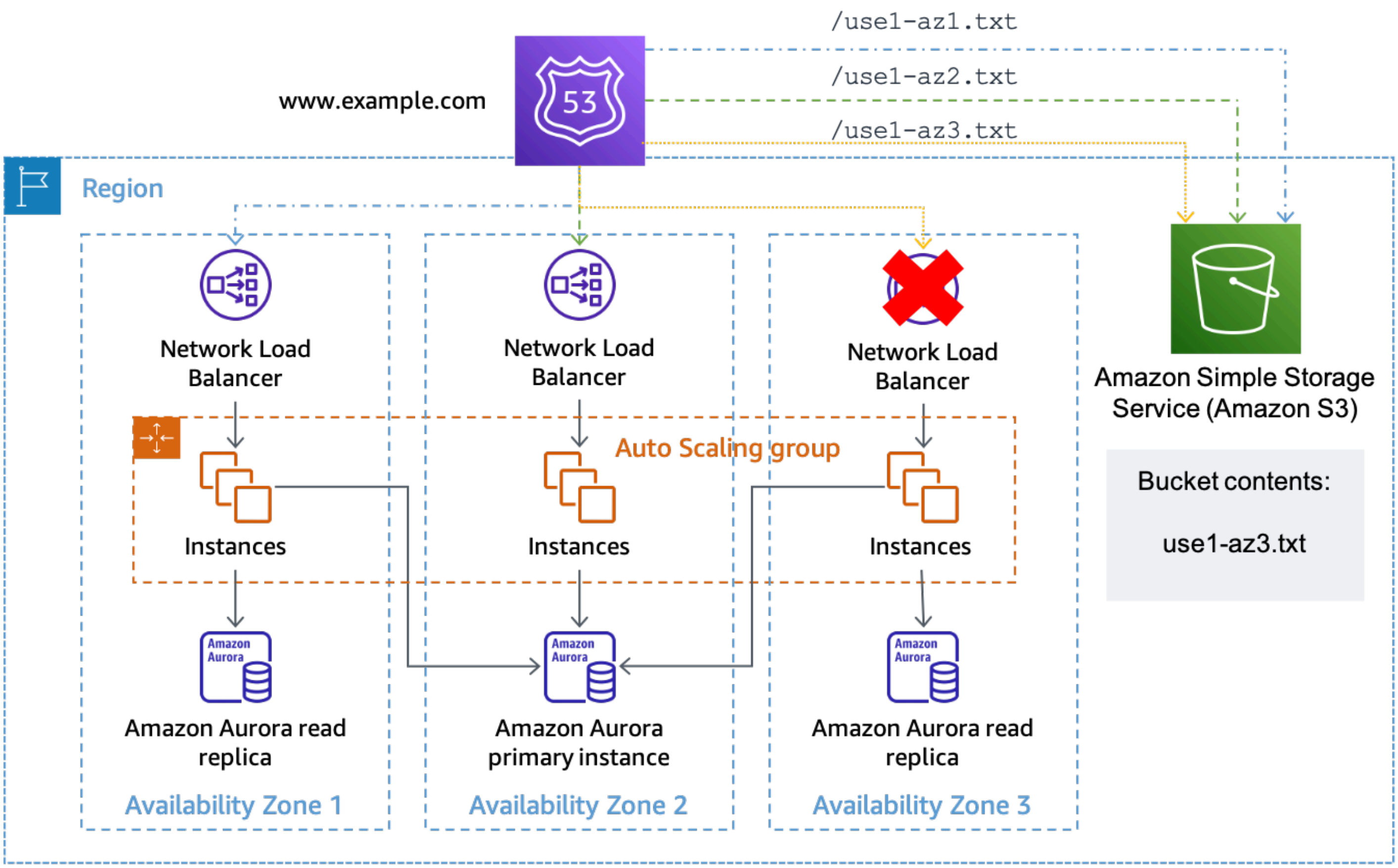
Objekt abwesend ist undungesundwenn das Objekt vorhanden ist. Dieses Setup ist in der folgenden Tabelle dargestellt.

Tabelle 4: Konfiguration des DNS-Eintrags für die Verwendung von Route 53-Zustandsprüfungen pro Availability Zone

Typ des Gesundheitschecks: einen Endpunkt überwachen Protocol (Protokoll): `HTTPS` ID:`dddd-4444` URL:`https://bu` *`name`* `.s3.us-east-1.amazonaws.com/use1-az1.txt`	Typ des Gesundheitschecks: einen Endpunkt überwachen Protocol (Protokoll): `HTTPS` ID:`eeee-5555` URL: `https://bucket` *`name`* `.s3.us-east-1.amazonaws.com/use1-az3.txt`	Typ des Gesundheitschecks: einen Endpunkt überwachen Protocol (Protokoll): `HTTPS` ID:`ffff-6666` URL:`https://bu` *`name`* `.s3.us-east-1.amazonaws.com/use1-az2.txt`	←	Gesundheitschecks
↑	↑	↑		
Routing-Richtlinie: gewichtet Name (Name: `www.example.com` Typ:A(Alias)	Routing-Richtlinie :gewichtet Name (Name: `www.example.com` Typ: A(Alias)	Routing-Richtlinie :gewichtet Name (Name: `www.example.com` Typ: A(Alias)	←	Gleichmäßig gewichtete Alias-A-Datensätze auf oberster Ebene verweisen auf NLB AZ-spezifische Endpunkte

Value (Wert): `us-east-1b.load-balancer-name.elb.us-east-1.amazonaws.com`	Value (Wert): `us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com`	Value (Wert): `us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`		
Gewicht:`100`	Gewicht: `100`	Gewicht: `100`		
Bewerten Sie die Gesundheit des Ziels:`true`	Bewerten Sie den Zustand des Ziels: `true`	Bewerten Sie den Zustand des Ziels: `true`		

Nehmen wir an, dass die Availability Zone`us-east-1a`ist zugeordnet`use1-az3`in dem Konto, in dem wir einen Workload haben, für den wir eine Evakuierung der Availability Zone durchführen möchten. Für den Ressourcendatensatz, der erstellt wurde für`us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com`würde einen Gesundheitscheck verknüpfen, der die URL testet`https://`*`bucket-name`*`.s3.us-east-1.amazonaws.com/use1-az3.txt`. Wenn Sie eine Evakuierung der Availability Zone für einleiten möchten`use1-az3`, lade eine Datei mit dem Namen hoch`use1-az3.txt`mit der CLI oder API zum Bucket. Die Datei muss keinen Inhalt enthalten, aber sie muss öffentlich sein, damit der Route 53-Gesundheitscheck darauf zugreifen kann. Die folgende Abbildung zeigt, dass diese Implementierung zur Evakuierung verwendet wird.`use1-az3`.



Amazon S3 als Ziel für einen Route 53-Gesundheitscheck verwenden

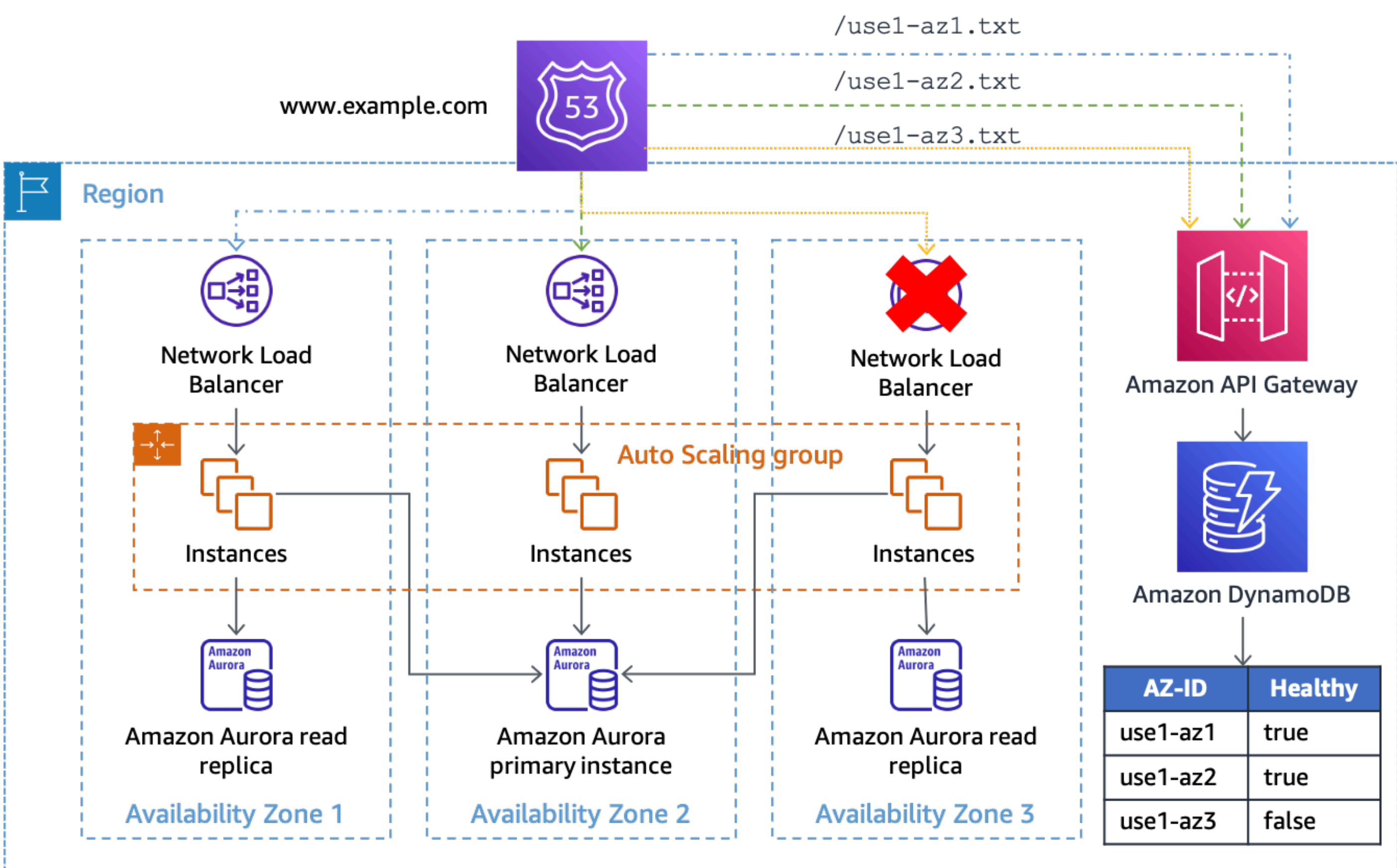
Verwendung von API Gateway und DynamoDB

Die zweite Implementierung dieses Musters verwendet eine[Amazon-API-Gateway REST-API](#). Die API ist konfiguriert mit einem[Serviceintegration](#)zu Amazon DynamoDB, wo der Status für jede verwendete Availability Zone gespeichert wird. Diese Implementierung ist flexibler als der Amazon S3-Ansatz, erfordert jedoch den Aufbau, Betrieb und Überwachung einer größeren Infrastruktur. Es kann auch sowohl für Route 53-Gesundheitschecks als auch für Gesundheitsprüfungen verwendet werden, die von einzelnen Hosts durchgeführt werden.

Wenn Sie diese Lösung mit einer NLB- oder ALB-Architektur verwenden, richten Sie Ihre DNS-Einträge auf die gleiche Weise wie im obigen Amazon S3-Beispiel ein, außer dass Sie den Pfad zur Integritätsprüfung ändern, um den API Gateway-Endpunkt zu verwenden, und geben Sie die`AZ-ID`im URL-Pfad. Wenn das API-Gateway beispielsweise mit einer benutzerdefinierten Domäne von konfiguriert ist`az-status.example.com`, die vollständige Anfrage für`use1-az1`würde aussehen wie`https://az-status.example.com/status/use1-az1`. Wenn Sie eine Evakuierung der Availability Zone einleiten möchten, können Sie ein DynamoDB-Element mithilfe der CLI oder API erstellen oder aktualisieren. Der Artikel verwendet die`AZ-ID`als Primärschlüssel und hat dann ein boolesches Attribut namens`Healthy`welche verwendet werden, geben an, wie API Gateway reagiert. Im Folgenden finden Sie einen Beispielcode, der in der API-Gateway-Konfiguration verwendet wird, um diese Entscheidung zu treffen.

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

Wenn das Attribut ist`true`(oder nicht vorhanden), API Gateway reagiert auf die Integritätsprüfung mit einem HTTP 200. Wenn dieser Wert falsch ist, antwortet es mit HTTP 500. Diese Implementierung ist in der folgenden Abbildung dargestellt.



Verwendung von API Gateway und DynamoDB als Ziel von Route 53-Gesundheitschecks

In dieser Lösung müssen Sie API Gateway vor DynamoDB verwenden, damit Sie den Endpunkt öffentlich zugänglich machen und die Anforderungs-URL in eine`GetItem`Anfrage für DynamoDB. Die Lösung bietet auch Flexibilität, wenn Sie zusätzliche Daten in die Anfrage aufnehmen möchten. Wenn

Sie beispielsweise detailliertere Statusangaben erstellen möchten, z. B. pro Anwendung, können Sie die Integritätsprüfungs-URL so konfigurieren, dass sie eine Anwendungs-ID im Pfad oder in der Abfragezeichenfolge bereitstellt, die auch mit dem DynamoDB-Element abgeglichen wird.

Der Status-Endpunkt der Availability Zone kann zentral bereitgestellt werden, sodass mehrere Ressourcen für die Integritätsprüfung überAWS-Kontenkönnen alle dieselbe konsistente Ansicht des Zustands der Availability Zone verwenden (wobei sichergestellt wird, dass Ihre API-Gateway-REST-API und Ihre DynamoDB-Tabelle so skaliert sind, dass sie die Last bewältigen) und macht die gemeinsame Nutzung von Route 53-Zustandsprüfungen überflüssig.

Die Lösung könnte auch auf mehrere skaliert werdenAWS-Regionenunter Verwendung einesGlobale Amazon DynamoDB-Tabelleund eine Kopie der API Gateway-REST-API in jeder Region. Dadurch wird verhindert, dass diese Lösung von einer einzelnen Region abhängig ist, und ihre Verfügbarkeit wird erhöht. Sie könnten die Lösung in drei oder fünf Regionen bereitstellen und in jeder Region den Zustand der Availability Zone abfragen und dabei die Ergebnisse der meisten Endpunkte verwenden, um das Quorum sicherzustellen. Dies ermöglicht letztendlich eine konsistente Replikation von Updates in der gesamten globalen Tabelle und verringert Beeinträchtigungen, die einen Endpunkt daran hindern könnten, zu reagieren. Wenn Sie beispielsweise fünf Regionen verwenden und drei Endpunkte eine Availability Zone als fehlerhaft melden, ein Endpunkt die Availability Zone als fehlerfrei meldet und ein Endpunkt nicht reagiert, würden Sie sich dafür entscheiden, die Availability Zone als fehlerhaft zu behandeln. Sie könnten auch eine erstellenRoute 53 berechneter Gesundheitscheckunter Verwendung einesm von nBerechnungum diese Logik auszuführen, um den Zustand der Availability Zone zu ermitteln.

Wenn Sie eine Lösung entwickeln würden, die einzelne Hosts als Mechanismus zur Bestimmung des Zustands ihrer AZ verwenden können, können Sie als Alternative Push-Benachrichtigungen verwenden, anstatt einen Pull-Mechanismus für Zustandsprüfungen bereitzustellen. Eine Möglichkeit, dies zu tun, ist ein SNS-Thema, das Ihre Verbraucher abonnieren. Wenn Sie den Schutzschalter auslösen möchten, veröffentlichen Sie im SNS-Thema eine Meldung, in der angegeben wird, welche Availability Zone beeinträchtigt ist. Dieser Ansatz geht Kompromisse mit ersteren ein. Dadurch entfällt die Notwendigkeit, die API-Gateway-Infrastruktur zu erstellen und zu betreiben und ein Kapazitätsmanagement durchzuführen. Es kann möglicherweise auch zu einer schnelleren Konvergenz des Availability Zone-Status führen. Es verhindert jedoch die Möglichkeit, Ad-hoc-Abfragen durchzuführen, und stützt sich auf dieRichtlinie zur Wiederholung der SNS-Zustellungum sicherzustellen, dass jeder Endpunkt die Benachrichtigung erhält. Außerdem muss jeder Workload oder Dienst eine Möglichkeit finden, die SNS-Benachrichtigung zu empfangen und entsprechende Maßnahmen zu ergreifen.

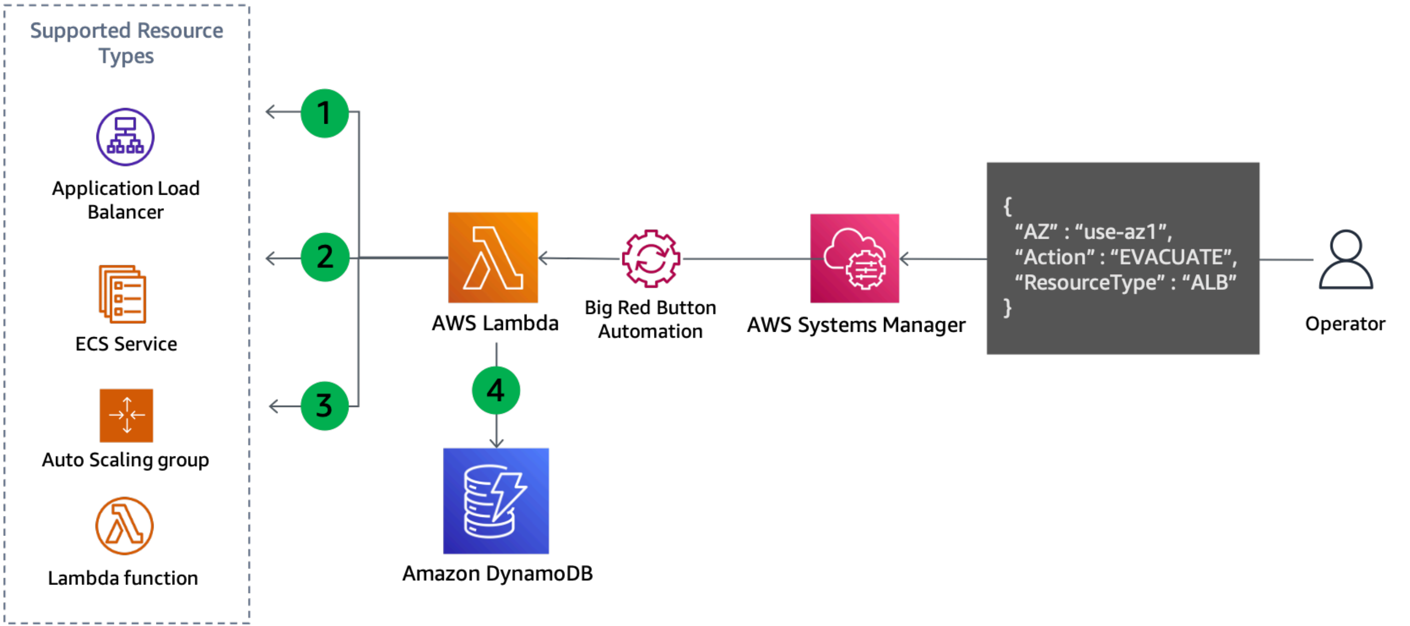
Beispielsweise muss jede neue EC2-Instance oder jeder neue Container, der gestartet wird, das Thema mit einem HTTP-Endpunkt während des Bootstraps abonnieren. Anschließend muss jede Instanz eine Software implementieren, die auf diesem Endpunkt abhört, an dem die Benachrichtigung zugestellt wird. Wenn die Instance von dem Ereignis betroffen ist, erhält sie außerdem möglicherweise keine Push-Benachrichtigung und arbeitet weiter. Bei einer Pull-Benachrichtigung weiß die Instance dagegen, ob ihre Pull-Anfrage fehlschlägt, und kann wählen, welche Maßnahmen als Reaktion ergriffen werden sollen.

Eine zweite Möglichkeit, Push-Benachrichtigungen zu senden, ist LongliveWebSocketVerbindungen. Amazon API Gateway kann verwendet werden, um eine bereitzustellenWebSocketAPIdass Verbraucher eine Verbindung herstellen und eine Nachricht empfangen können, wennvom Backend gesendet. Mit einemWebSocket, können Instances sowohl regelmäßige Pulls durchführen, um sicherzustellen, dass ihre Verbindung einwandfrei ist, als auch Push-Benachrichtigungen mit niedriger Latenz erhalten.

Kontrollflugzeuggesteuerte Evakuierung

Das erste Muster verwendet Operationen auf der Datenebene, um die Ausführung von Arbeiten in einer betroffenen Availability Zone zu verhindern und die Auswirkungen eines Ereignisses zu mildern. Möglicherweise verwenden Sie jedoch eine Architektur, die keine Load Balancer verwendet, oder bei der die Konfiguration eines Zustandschecks pro Host nicht möglich ist. Oder Sie möchten möglicherweise verhindern, dass neue Kapazitäten in der betroffenen Availability Zone bereitgestellt werden, indem Sie Auto Scaling oder die normale Arbeitsplanung verwenden.

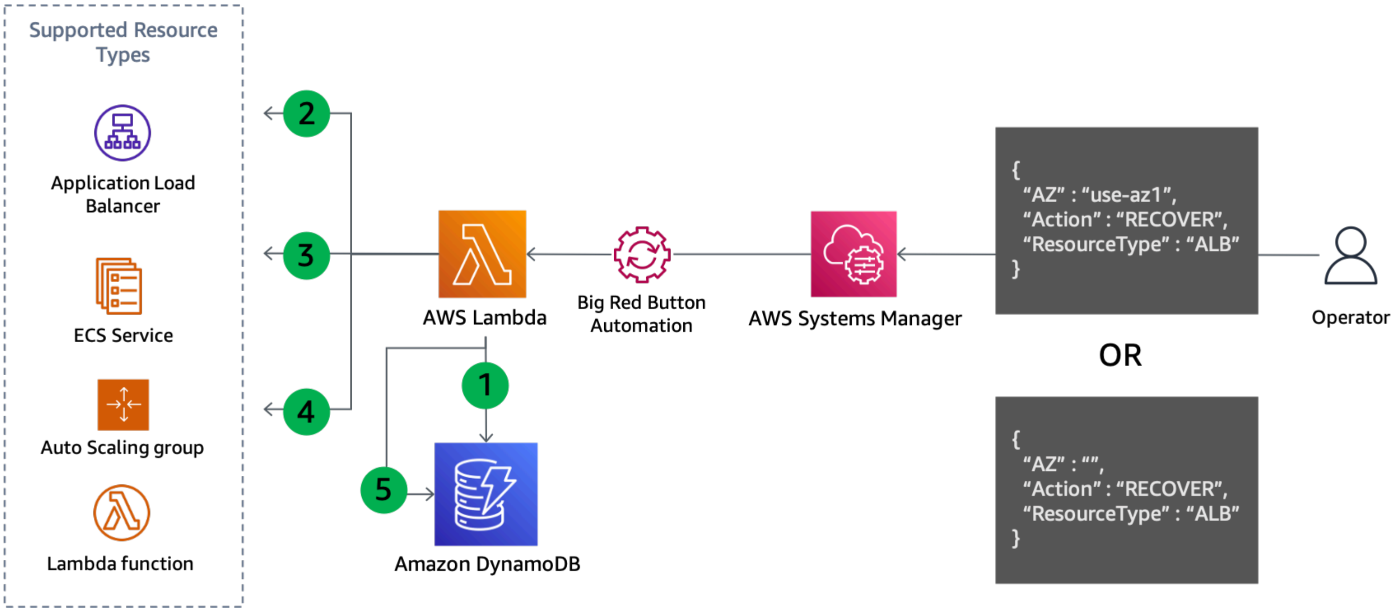
Um beide Situationen zu lösen, sind Aktionen auf der Steuerungsebene erforderlich, um die Konfiguration der Ressource zu aktualisieren. Das Muster funktioniert für jeden Service, dessen Netzwerkkonfiguration aktualisiert werden kann, z. B. EC2 Auto Scaling, Amazon ECS, Lambda und mehr. Es erfordert das Schreiben von Code für jeden Dienst, aber die Geschäftslogik folgt einem Standardmuster. Der Code sollte lokal von einem Operator ausgeführt werden, der auf das Ereignis reagiert, um die erforderlichen Abhängigkeiten zu minimieren. Der grundlegende Ablauf der Skriptlogik ist in der folgenden Abbildung dargestellt.



Aktualisierung der Kontrollebene zur Evakuierung einer Availability Zone

1. Das Skript listet alle Ressourcen des angegebenen Typs auf, z. B. Auto Scaling-Gruppe, ECS-Service oder Lambda-Funktion, und ruft ihre Subnetze aus den Ressourceninformationen ab. Die unterstützten Ressourcen hängen davon ab, wofür das Skript konfiguriert wurde, um es zu unterstützen.
2. Es bestimmt, welche Subnetze entfernt werden sollten, indem der Availability Zone-Name jedes Subnetzes mit der zugeordneten Availability Zone-ID verglichen wird, die als Eingabeparameter angegeben wurde.
3. Die Netzwerkkonfiguration der Ressource wird aktualisiert, um die identifizierten Subnetze zu entfernen.
4. Die Details des Updates werden in einer DynamoDB-Tabelle aufgezeichnet. Die Availability Zone-ID wird gespeichert alsPartitionsschlüsselund die Ressourcen-ARN oder der Name wird gespeichert alsSortierschlüssel. Die entfernten Subnetze werden als String-Array gespeichert. Schließlich wird der Ressourcentyp auch gespeichert und als Hash-Schlüssel für eineGlobaler Sekundärindex(GSI).

Da Schritt vier die vorgenommenen Aktualisierungen aufzeichnet, bietet sich dieser Ansatz auch an, um leicht rückgängig zu machen, wenn Sie zur Wiederherstellung bereit sind, wie in der folgenden Abbildung dargestellt.



Aktualisierung der Steuerungsebene zur Wiederherstellung nach der Evakuierung der Availability Zone

Schritte zur Wiederherstellung:

1. Fragen Sie die GSI ab, um die Subnetze für jede Ressource des angegebenen Typs in der angegebenen Availability Zone (oder alle Availability Zones, falls keine angegeben ist) zu entfernen.
2. Beschreiben Sie jede Ressource, die in der DynamoDB-Abfrage gefunden wurde, um ihre aktuelle Netzwerkkonfiguration abzurufen.
3. Kombinieren Sie die Subnetze aus der aktuellen Netzwerkkonfiguration mit denen, die aus der DynamoDB-Abfrage abgerufen wurden.
4. Aktualisieren Sie die Netzwerkkonfiguration der Ressource mit dem neuen Subnetzsatz.
5. Entfernen Sie den Datensatz aus der DynamoDB-Tabelle, nachdem das Update erfolgreich abgeschlossen wurde.

Dieses generalisierte Muster verhindert sowohl die Weiterleitung von Arbeit an die betroffene Availability Zone als auch die Bereitstellung neuer Kapazitäten dort. Im Folgenden finden Sie Beispiele dafür, wie dies für verschiedene Dienste erreicht wird.

- Lambda— Aktualisiere die FunktionenVPC-Konfigurationum die Subnetze in der angegebenen Availability Zone zu entfernen.

- Auto Scaling-Gruppe—Entfernen Sie die Subnetze aus der ASG-Konfigurationwas diese Kapazität in den verbleibenden Availability Zones ersetzen wird.
- Amazon ECS—Aktualisieren Sie die VPC-Konfiguration des ECS-Serviceum die Subnetze zu entfernen.
- Amazon EKS— BewerbenVerderbungenan Knoten in der betroffenen Availability Zone, um bestehende Pods zu entfernen und zu verhindern, dass dort weitere Pods geplant werden.

Jeder Dienst reagiert unterschiedlich auf das Konfigurationsupdate. Amazon ECS folgt beispielsweise demBereitstellungskonfiguration des Dienstes nach einem Updateund lösen eine fortlaufende Bereitstellung oder eine blaue/grüne Bereitstellung neuer Aufgaben aus.

Diese Updates können die Arbeit für einige Workloads zu schnell in die intakten Availability Zones verlagern. Sie sind zwar so konfiguriert, dass sie bei einem Ausfall statisch stabil sind (in den verbleibenden Availability Zones ist genügend Kapazität vorinstalliert, um die Arbeit der betroffenen Availability Zone zu bewältigen), Sie sollten aber auch die Kapazität der betroffenen Availability Zone schrittweise abbauen.

> Wenn Sie planen, die Netzwerkkonfiguration Ihrer Auto Scaling-Gruppe zu aktualisieren, ist dies eine Zielgruppe für einen Load Balancer mit zonenübergreifendem Load BalancingBehinderte, folgen Sie dieser Anleitung.
>
> Auto Scaling reagiert auf diese Änderung mit seinerLogik zur Neuverteilung der Availability Zone. Es werden Instances in den anderen Availability Zones gestartet, um Ihre gewünschte Kapazität zu erreichen, und Instances in der Availability Zone, die Sie entfernt haben, beendet. Der Load Balancer verteilt den Traffic jedoch weiterhin gleichmäßig auf jede Availability Zone, einschließlich der, die Sie aus der ASG entfernt haben, während die Instances beendet werden. Dies könnte dazu führen, dass die verbleibende Kapazität in dieser Availability Zone ausgeschöpft wird, bis alle Instances dort erfolgreich beendet sind. Dies ist das gleiche Problem, das in beschrieben wirdVerfügbarkeit, Zonenunabhängigkeitbezüglich des Ungleichgewichts in der Availability Zone, wenn der zonenübergreifende Lastenausgleich deaktiviert ist. Um dies zu verhindern, können Sie entweder:
>
> - Führen Sie immer zuerst die Evakuierung Ihrer Availability Zone durch, sodass der Traffic nur auf die verbleibenden Availability Zones aufgeteilt wird.
> - Spezifizieren Sie einminimale Anzahl intakter Ziele mit DNS-Failoverum Ihre erforderliche Mindestanzahl für diese Availability Zone zu erreichen.

Dadurch wird sichergestellt, dass kein Traffic an die Availability Zone gesendet wird, die Sie entfernt haben, nachdem die Instances beendet wurden.

Übersicht

In der folgenden Tabelle werden die Vor- und Nachteile der beschriebenen Evakuierungsmuster zusammengefasst.

Tabelle 5: Vor- und Nachteile des Evakuierungsmusters

Ansatz	Profis	Nachteile
Datenplangesteuerte Evakuierung	Verlässt sich nur auf Aktionen auf der Datenebene Verhindert schnell, dass Arbeit in der betroffenen Availability Zone erledigt wird Flexibler Ansatz für eine zentrale Ansicht des Zustands der Availability Zone	Verhindert nicht, dass Kapazität in einer betroffenen Availability Zone bereitgestellt wird Nicht alle Workload-Typen können diesen Ansatz problemlos verwenden.
Kontrollflugzeuggesteuerte Evakuierung	Verhindert, dass neue Kapazitäten in der betroffenen Availability Zone bereitgestellt werden Entfernt vorhandene Kapazität aus der betroffenen Availability Zone	Verlässt sich auf die Steuerungsebene jedes Dienstes Erfordert, dass Code für jeden Dienst geschrieben wird Muss Service für Service abgeschlossen werden Muss darauf achten, dass die Kapazität während des Updates nicht überlastet wird

Sie werden wahrscheinlich beide Ansätze zusammen als Teil eines Evakuierungsplans für die Availability Zone verwenden. Beginnen Sie mit den von der Datenebene gesteuerten Evakuierungsmaßnahmen, die mit größerer Wahrscheinlichkeit erfolgreich sind, um die Verarbeitung von Arbeiten in der betroffenen Availability Zone schnell zu beenden. Sobald die ersten Auswirkungen abgemildert sind, führen Sie anschließend die von der Kontrollebene kontrollierten Evakuierungsmaßnahmen durch, falls Sie dies für erforderlich halten.

Schlussfolgerung

Dieses Whitepaper gab einen Überblick über Grauausfälle, wie sie sich manifestieren, und erläuterte, warum Sie Beobachtbarkeits- und Evakuierungswerkzeuge entwickeln müssen, um solche Ereignisse zu verhindern, sobald sie auftreten. Im nächsten Abschnitt haben Sie sich mit Multi-AZ-Observability und drei Ansätzen befasst, die Sie implementieren können, um die Auswirkungen einer einzelnen Availability Zone zu erkennen. Im letzten Abschnitt wurden in diesem Dokument zwei allgemeine Ansätze für die Evakuierung der Availability Zone vorgestellt. Der erste Ansatz verwendet Aktionen auf der Datenebene, um zu verhindern, dass Arbeit an die betroffene Availability Zone weitergeleitet wird, während der zweite Ansatz Aktionen auf der Steuerungsebene verwendet, um zu verhindern, dass Kapazität in der betroffenen Availability Zone bereitgestellt wird. Zusammen erreichen diese beiden Ansätze die beiden Ergebnisse, die mit der Evakuierung der Availability Zone angestrebt werden.

Die in diesem Dokument beschriebenen Wiederherstellungsmuster werden wahrscheinlich Teil einer umfassenderen Überwachungs- und Fehlerwiederherstellungslösung sein. Dieser Ansatz zur Behandlung von Grauausfällen in einer einzigen Verfügbarkeitszone erfordert technische Arbeit, um die Instrumente zu ihrer Erkennung sowie die Werkzeuge zu ihrer Erkennung zu entwickeln. Für viele Workloads kann dieser Ansatz jedoch eine einfachere und kostengünstigere Alternative zum Aufbau von Architekturen mit mehreren Regionen sein. Darüber hinaus kann es im Vergleich zu DR für mehrere Regionen dazu beitragen, kleinere RPOs und RTOs zu erreichen (was die Verfügbarkeit des Workloads erhöht).

Anhang A — Abrufen der Availability Zone-ID

Wenn Sie das verwendenAWS.NET SDK (sowie einige andere wieJavaScript) oder wenn Sie Ihr System auf einer EC2-Instance (einschließlich Amazon ECS und Amazon EKS) ausführen, können Sie die Availability Zone-ID direkt abrufen.

- AWS.NET SDK

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

- EC2-Instance-Metadatendienst

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

Auf anderen Plattformen wie Lambda und Fargate müssen Sie den Availability Zone-Namen abrufen und dann die Zuordnung zur Availability Zone-ID finden. Mit dem Availability Zone-Namen finden Sie die Availability Zone-ID wie folgt:

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

Die folgenden Beispiele, um den Availability Zone-Namen zu finden, der im obigen Beispiel verwendet werden soll, wurden in Bash geschrieben, wobei derAWS CLIund das Paketjq. Sie müssen in die Programmiersprache konvertiert werden, die für Ihren Workload verwendet wird.

- Amazon ECS— Wenn der Instance Metadata Service (IMDS) vom Host blockiert wird, können Sie stattdessen die Container-Metadatendatei verwenden.

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

- Fargate(Plattformversion 1.4 oder höher)

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

- Lambda— Die Availability Zone ist der Funktion nicht direkt ausgesetzt. Um es zu finden, müssen Sie mehrere Schritte ausführen. Dazu müssen Sie einen privaten API-Gateway-REST-Endpunkt erstellen, der die IP-Adresse des Anforderers zurückgibt. Dadurch wird die private IP identifiziert, die der Elastic Network-Schnittstelle zugewiesen ist, die von der Funktion verwendet wird.
  - Ruf das Lambda an`GetFunction`API, um die VPC-ID der Funktion zu finden.
  - Rufen Sie den API-Gateway-Dienst auf, um die IP der Funktion abzurufen.
  - Suchen Sie anhand der IP und der VPC-ID die zugehörige Netzwerkschnittstelle und extrahieren Sie die Availability Zone.

```
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’)
```

Anhang B — Beispiel für eine Chi-Quadrat-Berechnung

Im Folgenden finden Sie ein Beispiel für die Erfassung von Fehlermetriken und die Durchführung eines Chi-Quadrat-Tests für die Daten. Der Code ist noch nicht produktionsbereit und führt keine notwendige Fehlerbehandlung durch, bietet jedoch einen Machbarkeitsnachweis für die Funktionsweise der Logik. Sie sollten dieses Beispiel an Ihre Bedürfnisse anpassen.

Zunächst wird jede Minute eine Lambda-Funktion von einem Amazon aufgerufenEventBridgegeplante Veranstaltung. Der Inhalt der Veranstaltung wird mit den folgenden Daten konfiguriert:

```
{
  "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" ]
}
```

Die Daten werden verwendet, um die gemeinsamen Daten zu spezifizieren, die zum Abrufen der entsprechenden Daten erforderlich sindCloudWatchMetriken (wie Namespace, Metrikname und Dimensionen) und veröffentlichen Sie dann die Chi-Quadrat-Ergebnisse für jede Availability Zone. Der Code in der Lambda-Funktion sieht mit Python 3.9 wie folgt aus. Auf hoher Ebene sammelt es die angegebenenCloudWatchMetriken für die vorherige Minute, führt den Chi-Quadrat-Test mit diesen Daten durch und veröffentlicht dannCloudWatchMetriken zum Testergebnis für jede angegebene Availability Zone.

```
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)
```

Sie können dann einen Alarm pro AZ erstellen. Das folgende Beispiel ist für`use1-az2`und Alarme für drei einminütige Datenpunkte hintereinander, die einen Maximalwert von 1 haben (1 ist die Metrik, die veröffentlicht wird, wenn der Chi-Quadrat-Test eine statistisch signifikante Abweichung der Fehlerrate feststellt).

```
{
    "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"
    }
}
```

Sie können auch eine erstellenm-of-nalarmieren und kombinieren Sie diese beiden Alarme mit einem Verbundalarm. Sie müssten außerdem dieselben Alarme für jede Controller-/Action-Kombination oder jeden Microservice erstellen, den Sie in jeder Availability Zone haben. Schließlich können Sie den Chi-Quadrat-Kompositalarm dem für die Availability Zone-spezifischen Alarm für jede Kombination aus Controller und Aktion hinzufügen, wie in gezeigtFehlererkennung mithilfe der Ausreißererkennung.

Beitragende Faktoren

Zu den Mitwirkenden an diesem Dokument gehören:

- Michael Haken, Principal Solutions Architect, Amazon Web Services

Dokumentversionen

Abonnieren Sie den RSS-Feed, um über Aktualisierungen dieses Whitepapers informiert zu werden.

Änderung	Beschreibung	Datum
Whitepaper aktualisiert	Aktualisiert mit zusätzlichen Hinweisen zur Beobachtb arkeit und zur Verwendung der neuen Funktion zur zonalen Verschiebung.	11. Juli 2023
Erstveröffentlichung	Das Whitepaper wurde zuerst veröffentlicht.	2. März 2022

> **Note**
>
> Um RSS-Updates zu abonnieren, müssen Sie ein RSS-Plug-In für den von Ihnen verwendeten Browser aktiviert haben.

Hinweise

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.