Detectando e mitigando falhas cinzentas

Padrões de resiliência multi-AZ avançados

Padrões de resiliência multi-AZ avançados: Detectando e mitigando falhas cinzentas

Table of Contents

Resumo e introdução i
Introdução 1
Falhas cinzentas 3
Observabilidade diferencial 3
Exemplo de falha cinzenta 6
Como responder a falhas cinzentas 7
Observabilidade de multi-ZD 10
Detecção de falhas com alarmes compostos do CloudWatch 15
Detectar impacto em uma única Zona de Disponibilidade 15
Confira se o impacto não é regional 17
Garanta que o impacto não seja causado por uma única instância 17
Juntando tudo isso 19
Detecção de falhas usando detecção de discrepâncias 21
Detecção de falhas de recursos zonais de instância única 26
Resumo 29
Padrões de evacuação da zona de disponibilidade 30
Independência da zona de disponibilidade 31
Ambientes de gerenciamento e planos de dados 37
Evacuação controlada por plano de dados 38
Mudança de zona no Application Recovery Controller (ARC) do Route 53 38
Route 53 ARC 40
Uso de um endpoint HTTP autogerenciado 41
Evacuação controlada por ambiente de gerenciamento 49
Resumo 53
Conclusão 54
Apêndice A — Obtendo o ID da zona de disponibilidade 55
Apêndice B – Exemplo de cálculo do qui-quadrado 57
Colaboradores 63
Revisões do documento 64
Avisos 65
AWS Glossário 66
..... lxvii

Padrões de resiliência Multi-AZ avançados

Data de publicação: 11 de julho de 2023 (Revisões do documento)

Muitos clientes executam seus workloads em configurações de zona de multidisponibilidade (AZ) altamente disponíveis. Essas arquiteturas funcionam bem durante eventos de falha binária, mas geralmente encontram problemas com falhas cinzentas. As manifestações desse tipo de falha podem ser sutis e desafiam a detecção rápida e definitiva. Este documento fornece orientação sobre como instrumentar workloads para detectar o impacto de falhas cinzentas isoladas em uma única zona de disponibilidade e, em seguida, tomar medidas para mitigar o impacto na zona de disponibilidade.

Introdução

O objetivo deste documento é ajudá-lo a implementar com mais eficiência arquiteturas Multi-AZ resilientes. Uma das melhores práticas para criar sistemas resilientes nas redes da Amazon Virtual Private Cloud (VPC) é implantar cada workload em várias zonas de disponibilidade.

Uma zona de disponibilidade é um ou mais datacenters discretos com energia, redes e conectividade redundantes. O uso de várias zonas de disponibilidade permite operar workloads com alta disponibilidade, tolerância a falhas e escalabilidade superiores ao que seria possível com um só datacenter.

Muitos serviços de AWS, como o Amazon Elastic Compute Cloud (EC2), o Amazon Elastic Compute Cloud (EC2) Auto Scaling ou o Amazon Relational Database Service (Amazon RDS), fornecem uma configuração multi-AZ. Esses serviços não exigem a criação de nenhuma ferramenta adicional de observabilidade ou failover. Eles tornam os workloads resilientes a modos de falha binária facilmente detectáveis em uma Região da AWS, que afetam uma única zona de disponibilidade. Pode ser uma falha física completa do hardware, perda de energia ou um bug latente de software que afeta a maioria dos recursos.

Mas há outra categoria de falhas denominada falhas cinzentas, cujas manifestações são sutis e desafiam a detecção rápida e definitiva. Isso, por sua vez, resulta em tempos mais longos para mitigar o impacto causado pela falha. Este documento se concentra nos impactos que as falhas cinzentas podem gerar nas arquiteturas multi-AZ, como detectá-las e, por fim, como mitigá-las.

> A orientação fornecida neste whitepaper aplica-se principalmente a classes específicas de cargas de trabalho que:

- Usam principalmente serviços de zona de AWS
- Precisam melhorar a resiliência de uma única região
- Estão dispostas a fazer um investimento significativo para criar os padrões de observabilidade e resiliência necessários

Nesses workloads, talvez você não esteja disposto a fazer algumas ou todas as compensações apresentadas em ???, ou não tenha a opção de usar várias regiões. É provável que esses tipos de workload representem um pequeno subconjunto de seu portfólio geral e, portanto, essa orientação deve ser considerada no nível do workload versus no nível da plataforma.

Falhas cinzentas

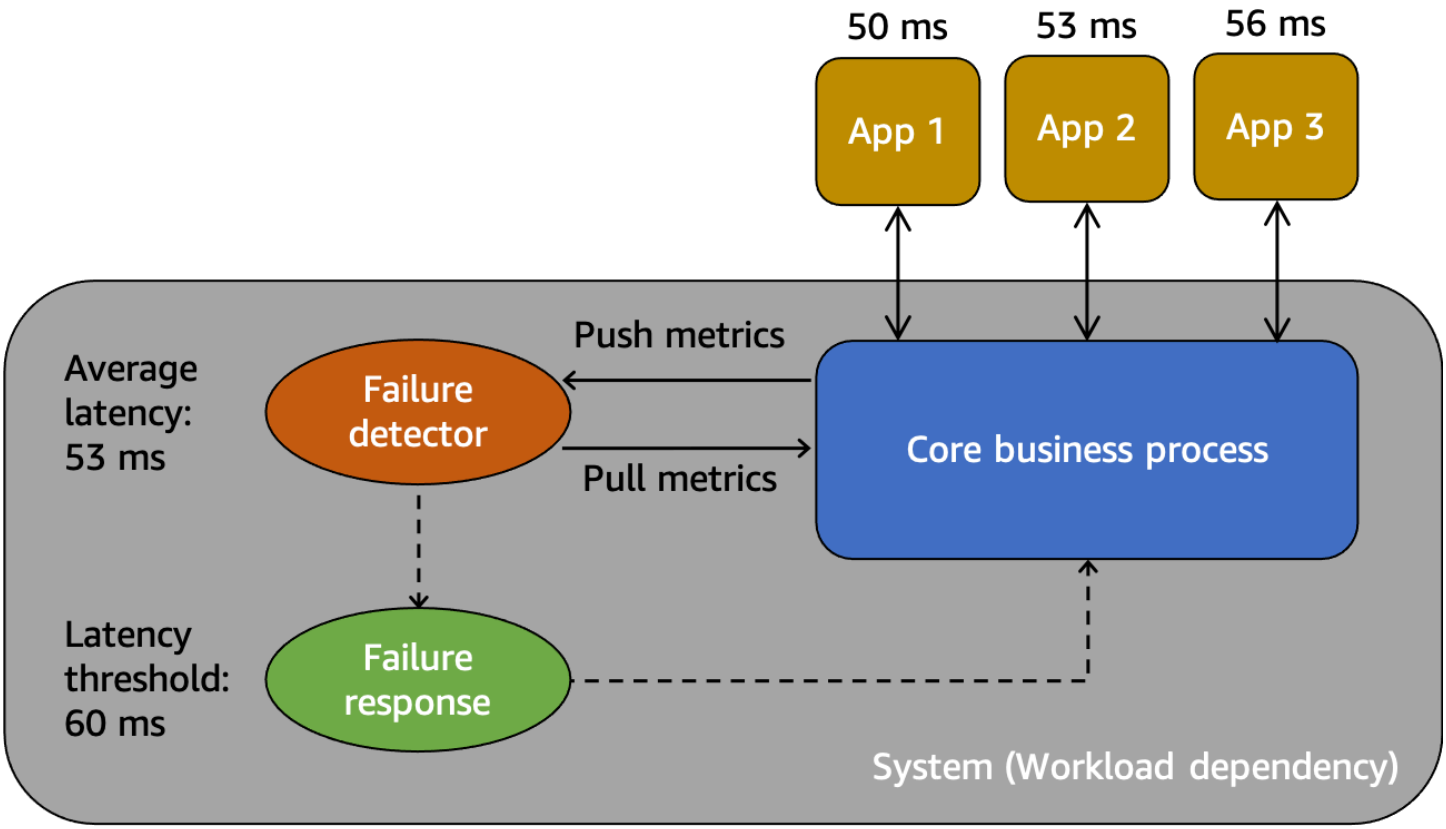
As falhas cinzentas são definidas pela característica da observabilidade diferencial, o que significa que entidades diferentes observam a falha de forma diferente. Vamos definir o que isso significa.

Observabilidade diferencial

As workloads que você opera normalmente têm dependências. Por exemplo, elas podem ser os serviços de nuvem da AWS que você usa para criar sua workload ou um provedor de identidades (IdP) de terceiros que você usa para federação. Essas dependências quase sempre implementam sua própria observabilidade, registrando métricas sobre erros, disponibilidade e latência, entre outras coisas geradas pelo uso do cliente. Quando alguma dessas métricas ultrapassa um limite, a dependência geralmente toma alguma ação para corrigir isso.

Os serviços dessas dependências geralmente têm vários clientes. Os clientes também implementam sua própria observabilidade e criam métricas e logs sobre suas interações com as dependências, registrando coisas como quanta latência existe nas leituras de disco, quantas solicitações de API falharam ou quanto tempo demorou uma consulta ao banco de dados.

Essas interações e medidas são representadas em um modelo abstrato na figura a seguir.

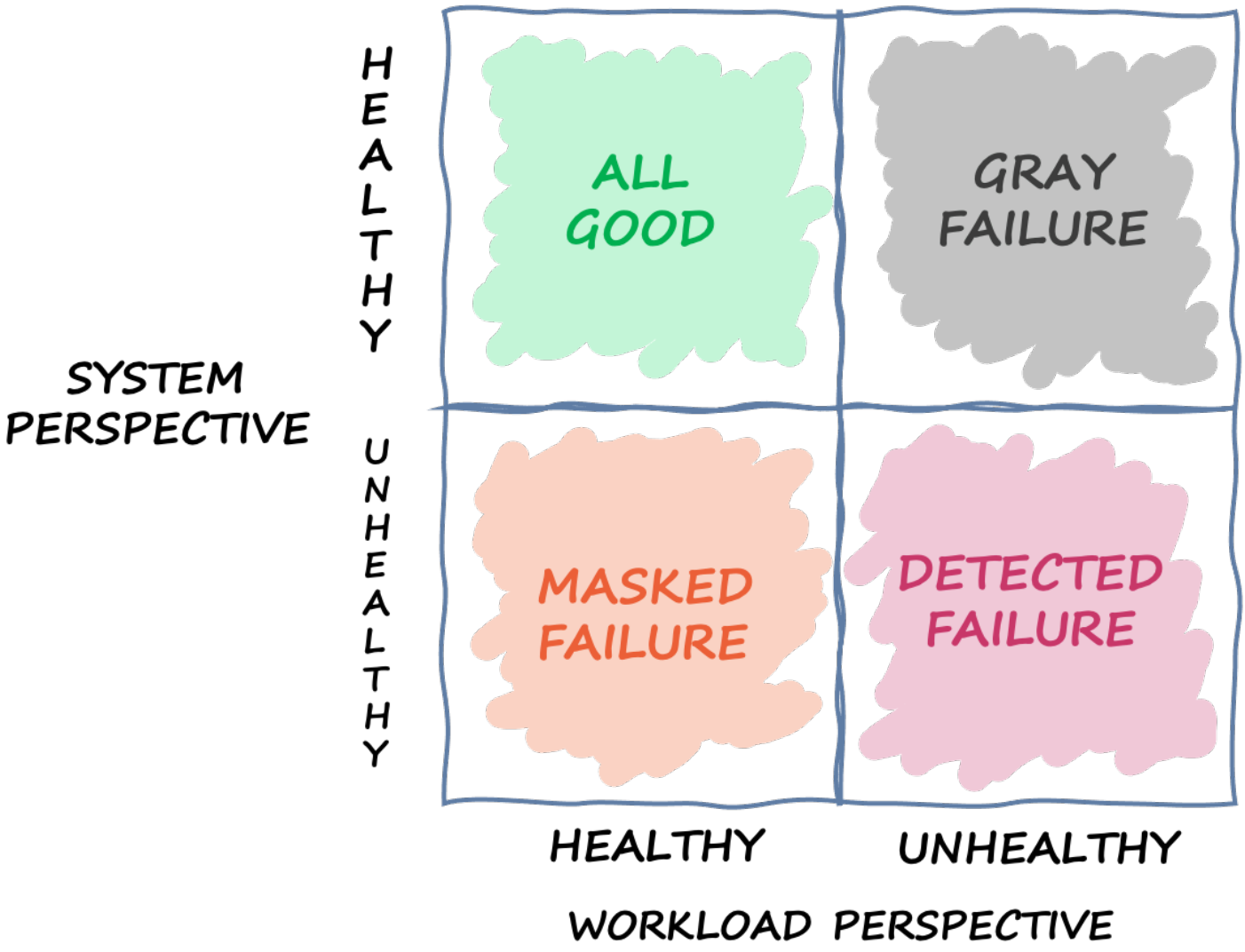


Modelo abstrato para entender as falhas cinzentas

Primeiro, temos o sistema, que é uma dependência dos clientes App 1, App 2 e App 3 neste cenário. O sistema tem um detector de falhas que examina as métricas criadas a partir do processo comercial principal. Ele também tem um mecanismo de resposta a falhas para mitigar ou corrigir problemas observados pelo detector. O sistema tem uma latência média geral de 53 ms e estabeleceu um limite para invocar o mecanismo de resposta a falhas quando a latência média excede 60 ms. App 1, App 2 e App 3 também estão fazendo suas próprias observações sobre suas interações com o sistema, registrando uma latência média de 50 ms, 53 ms e 56 ms, respectivamente.

A observabilidade diferencial acontece quando um dos clientes detecta que o sistema não está íntegro, mas o próprio monitoramento do sistema não detecta o problema ou o impacto não ultrapassa o limite de alarme. Vamos imaginar que o App 1 comece a ter uma latência média de 70 ms, em vez de 50 ms. O App 2 e o App 3 não veem mudança em suas latências médias. Isso aumenta a latência média do sistema subjacente para 59,66 ms, o que não ultrapassa o limite de latência e portanto não ativa o mecanismo de resposta a falhas. No entanto, o App 1 vê um aumento de 40% na latência. Isso pode afetar a disponibilidade ao exceder o tempo limite configurado pelo cliente do App 1, ou pode causar impactos em cascata em uma cadeia mais longa de interações. Do ponto de vista do App 1, o sistema subjacente do qual ele depende não está íntegro, mas do ponto

de vista do próprio sistema, bem como do App 2 e do App 3, o sistema está íntegro. A figura a seguir resume essas diferentes perspectivas.



Quadrante definindo os diferentes estados em que um sistema pode estar com base em perspectivas distintas

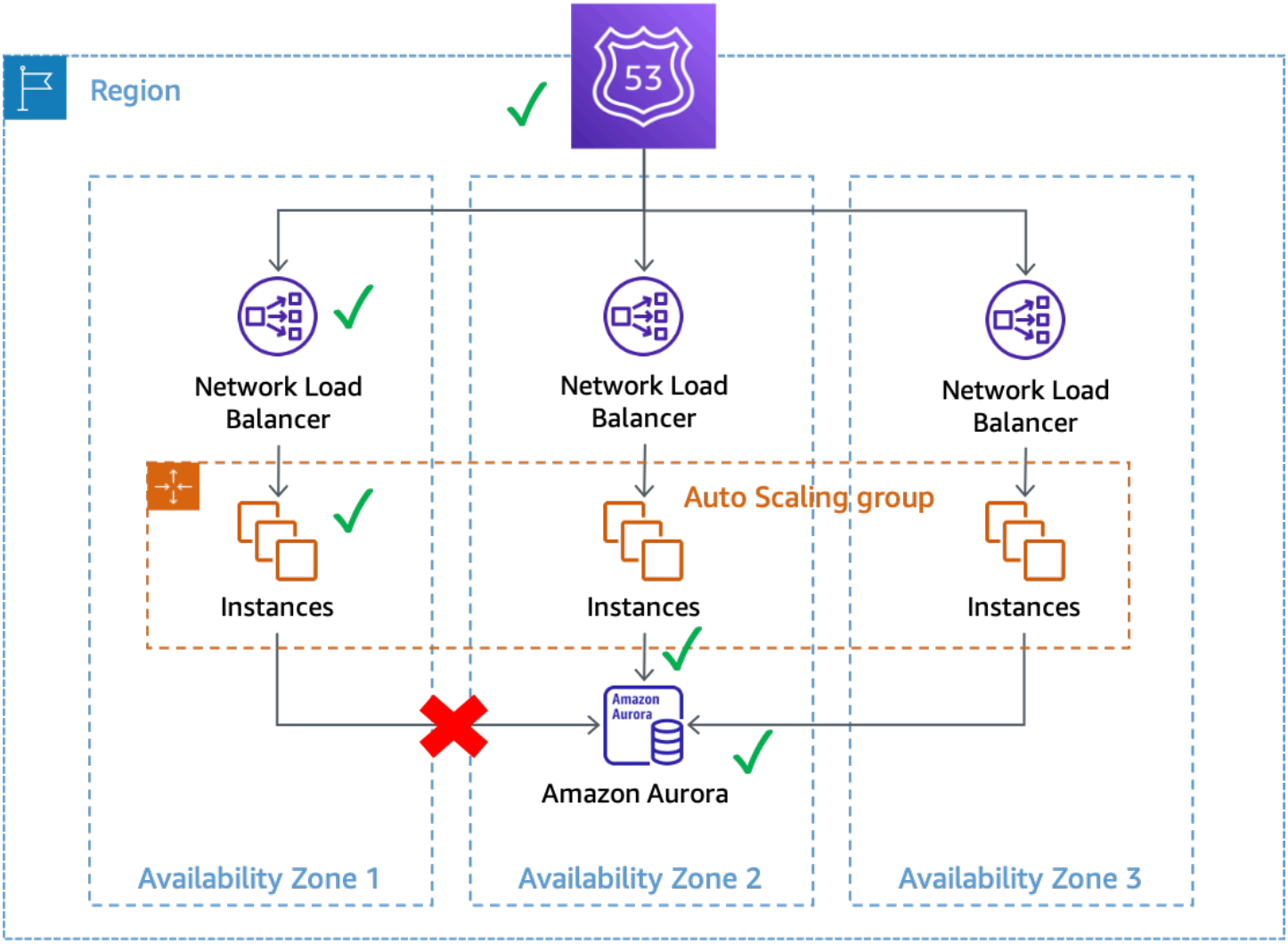
A falha também pode atravessar esse quadrante. Um evento pode começar como uma falha cinzenta, depois se tornar uma falha detectada, depois passar para uma falha mascarada e, talvez, voltar para uma falha cinzenta. Não há um ciclo definido e quase sempre há chances de recorrência da falha até que a causa raiz seja abordada.

A conclusão que tiramos disso é que workloads nem sempre podem contar com o sistema subjacente para detectar e mitigar falhas. Não importa o quão sofisticado e resiliente seja o sistema subjacente, sempre haverá uma chance de uma falha passar despercebida ou permanecer abaixo do limite de reação. Os clientes desse sistema, como o App 1, precisam estar equipados para

detectar e mitigar rapidamente o impacto causado por uma falha cinzenta. Isso requer a construção de mecanismos de observabilidade e recuperação para essas situações.

Exemplo de falha cinzenta

Falhas cinzentas podem ter impacto em sistemas Multi-ZD na AWS. Por exemplo, pegue uma frota de instâncias do Amazon EC2 em um grupo do Auto Scaling implantado em três Zonas de Disponibilidade. Todos eles se conectam a um banco de dados Amazon Aurora em uma Zona de Disponibilidade. Em seguida, há uma falha cinzenta que afeta a rede entre as Zonas de Disponibilidade 1 e 2. O resultado desse problema são falhas em uma porcentagem das conexões de banco de dados, novas e já existentes, das instâncias na Zona de Disponibilidade 1. Essa situação é ilustrada na figura a seguir.



Uma falha cinzenta que afeta as conexões do banco de dados de instâncias na Zona de Disponibilidade 1

Neste exemplo, o Amazon EC2 vê as instâncias na Zona de Disponibilidade 1 como íntegras porque elas continuam passando pelas verificações de status do sistema e da instância. O Amazon EC2 Auto Scaling também não detecta impacto direto em nenhuma Zona de Disponibilidade e continua lançando capacidade nas Zonas configuradas. O Network Load Balancer (NLB) também considera as instâncias por trás dele tão íntegras quanto as verificações de integridade do Route 53 que são realizadas no endpoint de NLB. Da mesma forma, o Amazon Relational Database Service (Amazon RDS) vê o cluster de banco de dados como íntegro e não aciona um failover automático. São muitos serviços diferentes que percebem seus serviços e recursos como íntegros, mas a workload detecta uma falha que afeta a disponibilidade. Isso é uma falha cinzenta.

Como responder a falhas cinzentas

Quando uma falha cinzenta ocorre no seu ambiente da AWS, geralmente há três opções disponíveis:

- Não fazer nada e esperar o problema passar.
- Se o problema estiver isolado em uma única Zona de Disponibilidade, evacuar essa Zona.
- Fazer um failover para outra Região da Região da AWS e usar os benefícios do isolamento regional AWS para mitigar o impacto.

Muitos clientes da AWS se contentam com a primeira opção para a maioria de suas workloads. Eles aceitam ter um objetivo de tempo de recuperação (RTO) possivelmente maior desde que não precisem criar soluções adicionais de observabilidade ou resiliência. Outros clientes optam por implementar a terceira opção, a Recuperação de desastres multirregional (DR), como plano de mitigação para vários modos de falha. Arquiteturas multirregionais podem funcionar bem nesses cenários. No entanto, há algumas desvantagens ao usar essa abordagem (consulte Fundamentos Multirregionais da AWS para ver uma discussão completa sobre considerações multirregionais).

Primeiramente, criar e operar uma arquitetura multirregional pode ser uma tarefa desafiadora, complexa e potencialmente cara. As arquiteturas multirregionais exigem uma análise cuidadosa de qual estratégia de DR será selecionada. Talvez não seja fiscalmente viável implementar uma solução de DR ativa e multirregional apenas para lidar com problemas zonais, enquanto uma estratégia de backup e restauração pode não atender aos seus requisitos de resiliência. Além disso, os failovers multirregionais precisam ser praticados continuamente na produção para que funcionem quando necessário. Tudo isso exige muito tempo e recursos de criação, operação e testagem.

Em segundo lugar, a replicação de dados nas Regiões da AWS usando os serviços da AWS é feita de forma assíncrona hoje. Replicação assíncrona pode ocasionar perda de dados. Isso significa que,

durante um failover regional, há uma chance de perda e inconsistência de dados. Sua tolerância à quantidade de perda de dados é definida como seu objetivo de ponto de recuperação (RPO). Os clientes, para quem uma forte consistência de dados é um requisito, precisam criar sistemas de reconciliação para corrigir esses problemas de consistência quando a Região principal estiver disponível novamente. Ou então, precisam criar seus próprios sistemas de replicação síncrona ou de gravação dupla, o que pode ter impactos significativos na latência, no custo e na complexidade da resposta. Eles também tornam a região secundária uma dependência rígida para cada transação, o que pode potencialmente reduzir a disponibilidade do sistema como um todo.

Por fim, para muitas workloads que usam uma abordagem ativa/em espera, é necessário um tempo diferente de zero para realizar o failover em outra Região. Talvez seja necessário reduzir seu portfólio de workloads na Região principal em uma ordem específica, drenar conexões ou interromper processos específicos. Em seguida, talvez seja necessário recuperar os serviços em uma ordem específica. Novos recursos também podem precisar ser provisionados ou exigir tempo para serem aprovados nas verificações de integridade necessárias antes de serem colocados em serviço. Esse processo de failover pode ser um período de total indisponibilidade. É com isso que os RTOs se preocupam.

Dentro de uma Região, muitos serviços da AWS oferecem uma persistência de dados altamente consistente. As implantações Multi-ZD do Amazon RDS usam replicação síncrona. O Amazon Simple Storage Service (Amazon S3) oferece uma forte consistência de read-after-write. O Amazon Elastic Block Storage (Amazon EBS) oferece snapshots de vários volumes consistentes em caso de falha. O Amazon DynamoDB pode realizar leituras altamente consistentes. Esses recursos podem ajudar você a obter um RPO menor (na maioria dos casos, um RPO zero) em uma única Região do que em arquiteturas multirregionais.

A evacuação de uma Zona de Disponibilidade pode ter um RTO menor do que uma estratégia multirregional, porque sua infraestrutura e recursos já estão provisionados nas Zonas de Disponibilidade. Em vez de precisar solicitar cuidadosamente que os serviços sejam desativados e reativados, ou drenar as conexões, as arquiteturas Multi-ZD podem continuar operando de forma estática quando uma Zona de Disponibilidade estiver prejudicada. Em vez de um período de indisponibilidade total que pode ocorrer durante um failover Regional, em uma evacuação da Zona de Disponibilidade, muitos sistemas podem sofrer apenas uma pequena degradação, à medida que o trabalho é transferido para as demais Zonas. Se o sistema tiver sido projetado para ser estaticamente estável a uma falha na Zona de Disponibilidade (nesse caso, isso significaria ter capacidade pré-provisionada em outras Zonas para absorver a carga), os clientes da workload podem não ver nenhum impacto.

É possível que o comprometimento de uma única Zona de Disponibilidade afete um ou mais serviços regionais da AWS, além de sua workload. Se você observar um impacto Regional, trate o evento como um problema no serviço regional, embora a origem desse impacto seja uma única Zona de Disponibilidade. A evacuação de uma Zona de Disponibilidade não mitigará esse tipo de problema. Use os planos de resposta em vigor para responder a um problema no serviço regional quando isso ocorrer.

O restante deste documento se concentra na segunda opção, evacuar a Zona de Disponibilidade, como forma de obter RTOs e RPOs menores para falhas cinzentas em ZD únicas. Esses padrões podem ajudar a obter melhor valor e eficiência das arquiteturas Multi-ZD e, para a maioria das classes de workload, podem reduzir a necessidade de criar arquiteturas multirregionais para lidar com esses tipos de eventos.

Observabilidade de multi-ZD

Para poder evacuar uma Zona de Disponibilidade durante um evento isolado em uma única Zona, primeiro você deve conseguir detectar se a falha está, de fato, isolada. Isso requer uma visibilidade muito precisa do comportamento do sistema em cada Zona de Disponibilidade. Muitos serviços da AWS fornecem métricas prontas para uso que fornecem informações operacionais sobre seus recursos. Por exemplo, o Amazon EC2 fornece várias métricas, como utilização da CPU, leituras e gravações de disco e entrada e saída de tráfego de rede.

No entanto, ao criar workloads que usam esses serviços, você precisa de mais visibilidade do que apenas essas métricas padrão. Você quer visibilidade da experiência do cliente fornecida pela sua workload. Além disso, você precisa que suas métricas estejam alinhadas às Zonas de Disponibilidade onde elas estão sendo produzidas. Esse é o insight de que você precisa para detectar falhas cinzentas observáveis de maneira diferente. Esse nível de visibilidade requer instrumentação.

A instrumentação requer escrever código explícito. Esse código deve fazer coisas como registrar a duração das tarefas, contar quantos itens foram bem-sucedidos ou falharam, coletar metadados sobre as solicitações e assim por diante. Você também precisa definir limites com antecedência para configurar o que é considerado normal e o que não é. Você precisa delinear objetivos e diferentes limites de severidade para latência, disponibilidade e contagens de erros na sua workload. O artigo da Amazon Builders' Library Instrumentando sistemas distribuídos para visibilidade operacional apresenta várias práticas recomendadas.

As métricas devem ser geradas tanto do lado do servidor quanto do lado do cliente. Uma prática recomendada para gerar métricas do lado do cliente e entender a experiência do cliente é usar o Canário, um software que examina regularmente sua workload e registra as métricas.

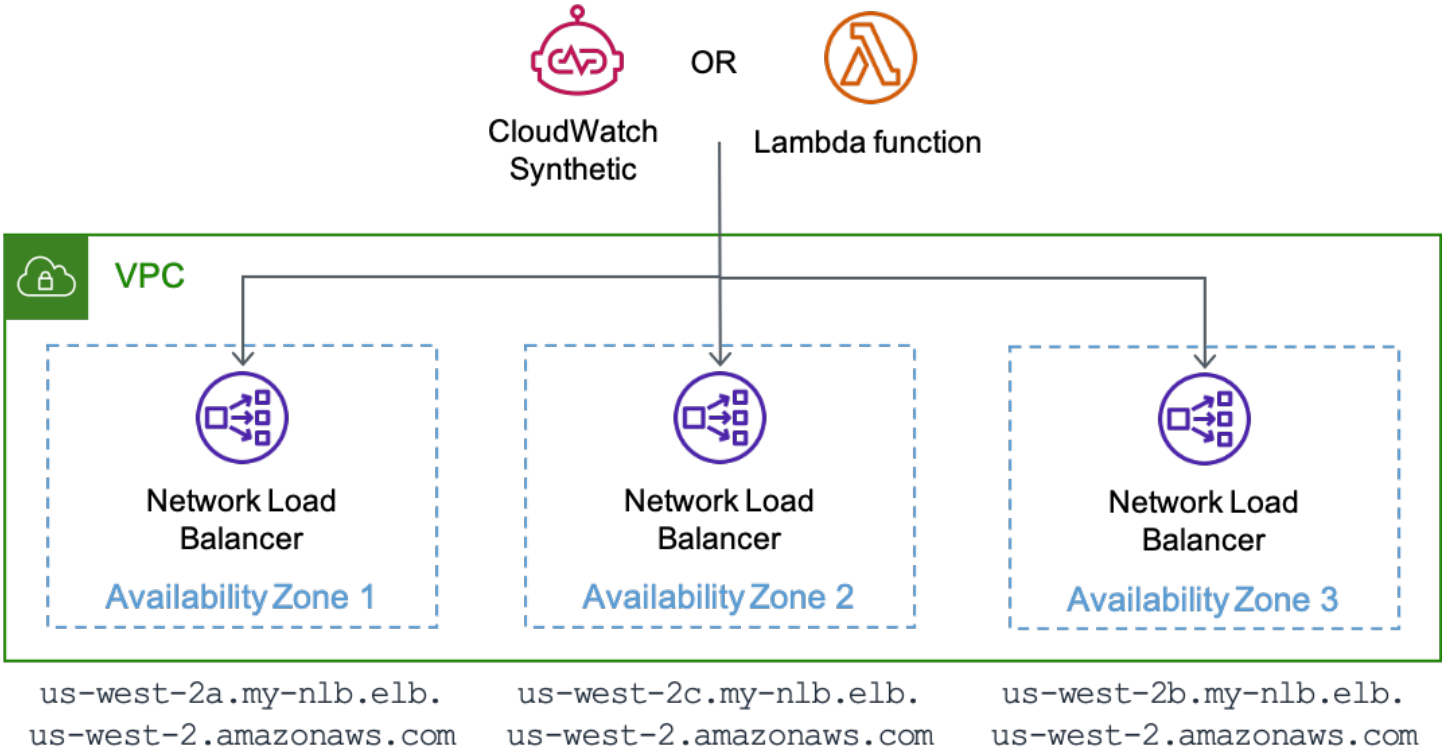
Além de produzir essas métricas, você também precisa entender o contexto delas. Para isso, você pode usar dimensões. As dimensões dão à métrica uma identidade única e ajudam a explicar o que as métricas estão dizendo a você. Para métricas usadas para identificar falhas na sua workload (por exemplo, latência, disponibilidade ou contagem de erros), você precisa usar dimensões que se alinhem aos limites de isolamento de falhas.

Por exemplo, se você estiver executando um serviço web em uma Região, em várias Zonas de Disponibilidade, usando uma estrutura web Model-View-Controller (MVC), você deve usar `Region`, `Availability Zone ID`, `Controller`, `Action` e `InstanceId` como dimensões para seus

conjuntos de dimensões. Se estiver usando microsserviços, você pode usar o nome do serviço e o método HTTP em vez dos nomes do controlador e da ação. Isso ocorre porque você espera que diferentes tipos de falhas sejam isolados por esses limites. Você não esperaria que um bug no código do seu serviço web que afetasse sua capacidade de listar produtos também afetasse a página inicial. Da mesma forma, você não esperaria que um volume completo do EBS em uma única instância do EC2 afetasse outras instâncias do EC2, impedindo a veiculação do seu conteúdo. A dimensão ID da Zona de Disponibilidade é o que permite identificar os impactos relacionados à Zona de Disponibilidade de forma consistente em todas as Contas da AWS. Você pode encontrar o ID da Zona de Disponibilidade nas suas workloads de várias maneiras diferentes. Consulte Apêndice A — Obtendo o ID da zona de disponibilidade para ver alguns exemplos.

Embora este documento use principalmente o Amazon EC2 como recurso computacional nos exemplos, `InstanceId` pode ser substituído por um ID de contêiner para os recursos computacionais do Amazon Elastic Container Service (Amazon ECS) e do Amazon Elastic Kubernetes Service (Amazon EKS) como componentes de suas dimensões.

Seus canários também podem usar `Controller`, `Action`, `AZ-ID` e `Region` como dimensões em suas métricas se você tiver endpoints zonais para sua workload. Nesse caso, alinhe seus canários para que sejam executados na Zona de Disponibilidade que estão testando. Isso garante que, se um evento isolado de Zona de Disponibilidade estiver afetando a Zona na qual seu canário está sendo executado, ele não registre métricas que façam com que uma Zona de Disponibilidade diferente sendo testada pareça não estar íntegra. Por exemplo, seu canário pode testar cada endpoint zonal para um serviço por trás de um Network Load Balancer (NLB) ou de um Application Load Balancer (ALB) usando seus nomes DNS zonais.



Um canário em execução no CloudWatch Synthetics ou uma função da AWS Lambda testando cada endpoint zonal de um NLB

Ao produzir métricas com essas dimensões, você pode estabelecer alarmes do Amazon CloudWatch que notificam você quando mudanças na disponibilidade ou na latência ocorrem dentro desses limites. Você também pode analisar rapidamente esses dados usando painéis. Para usar métricas e logs de forma eficiente, o Amazon CloudWatch oferece o formato de métrica incorporada (EMF) que permite incorporar métricas personalizadas com dados de log. O CloudWatch extrai automaticamente as métricas personalizadas para que você possa visualizá-las e colocar alarmes nelas. A AWS fornece várias bibliotecas de cliente para diferentes linguagens de programação que facilitam o uso inicial do EMF. Elas podem ser usados com Amazon EC2, Amazon ECS, Amazon EKS, AWS Lambda e em ambientes on-premises. Com métricas incorporadas aos seus registros, você também pode usar o Amazon CloudWatch Contributor Insights para criar gráficos de séries temporais que exibem dados dos colaboradores. Nesse cenário, podemos exibir dados agrupados por dimensões como `AZ-ID`, `InstanceId` ou `Controller`, bem como qualquer outro campo no log, como `SuccessLatency` ou `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
}
```

Esse log tem três conjuntos de dimensões. Eles progridem em ordem de granularidade, da instância, à Zona de Disponibilidade e depois à Região (`Controller` e `Action` estão sempre incluídos neste exemplo). Eles oferecem suporte à criação de alarmes em toda a sua workload, que indicam quando há impacto em uma ação específica do controlador em uma única instância, em uma única Zona

de Disponibilidade ou em toda a Região da AWS. Essas dimensões são usadas para a contagem das métricas de resposta HTTP 2xx, 3xx, 4xx e 5xx, bem como para a latência das métricas de solicitação bem-sucedida (se a solicitação falhar, ela também registra uma métrica para a latência da solicitação com falha). O log também grava outras informações, como o caminho HTTP, o IP de origem do solicitante e se essa solicitação exigiu que o cache local fosse atualizado. Esses pontos de dados podem então ser usados para calcular a disponibilidade e a latência de cada API fornecida pela workload.

> Uma observação sobre o uso de códigos de resposta HTTP para métricas de disponibilidade
>
> Normalmente, é possível considerar respostas 2xx e 3xx como bem-sucedidas e 5xx como falhas. Os códigos de resposta 4xx ficam em algum lugar intermediário. Normalmente, elas são produzidas devido a um erro do cliente. Talvez um parâmetro esteja fora do alcance, levando a uma resposta 400, ou talvez o cliente esteja solicitando algo que não existe, resultando em uma resposta 404. Você não contaria essas respostas com base na disponibilidade da sua workload. No entanto, isso também pode ser resultado de um bug no software.
>
> Por exemplo, se você introduziu uma validação de entrada mais rígida, que rejeita uma solicitação que teria sido bem-sucedida antes, a resposta 400 pode contar como uma queda na disponibilidade. Ou talvez você esteja realizando controle de utilização do cliente e retornando uma resposta 429. Embora o controle de utilização proteja seu serviço para manter a disponibilidade, do ponto de vista do cliente, o serviço não está disponível para o processamento da solicitação. Você precisará decidir se os códigos de resposta 4xx fazem parte do cálculo da disponibilidade.

Embora esta seção tenha descrito o uso do CloudWatch como forma de coletar e analisar métricas, ele não é a única solução. Você também pode optar por enviar métricas para o Amazon Managed Service for Prometheus, Amazon Managed Grafana, para uma tabela do Amazon DynamoDB ou usar uma solução de monitoramento de terceiros. O principal é: as métricas que sua workload produz devem conter contexto sobre os limites de isolamento de falhas da workload.

Com workloads que produzem métricas com dimensões alinhadas aos limites de isolamento de falhas, você pode criar uma observabilidade que detecte falhas isoladas de Zona de Disponibilidade. As seções a seguir descrevem três abordagens complementares para detectar falhas decorrentes do comprometimento de uma única Zona de Disponibilidade.

Tópicos

- Detecção de falhas com alarmes compostos do CloudWatch
- Detecção de falhas usando detecção de discrepâncias
- Detecção de falhas de recursos zonais de instância única
- Resumo

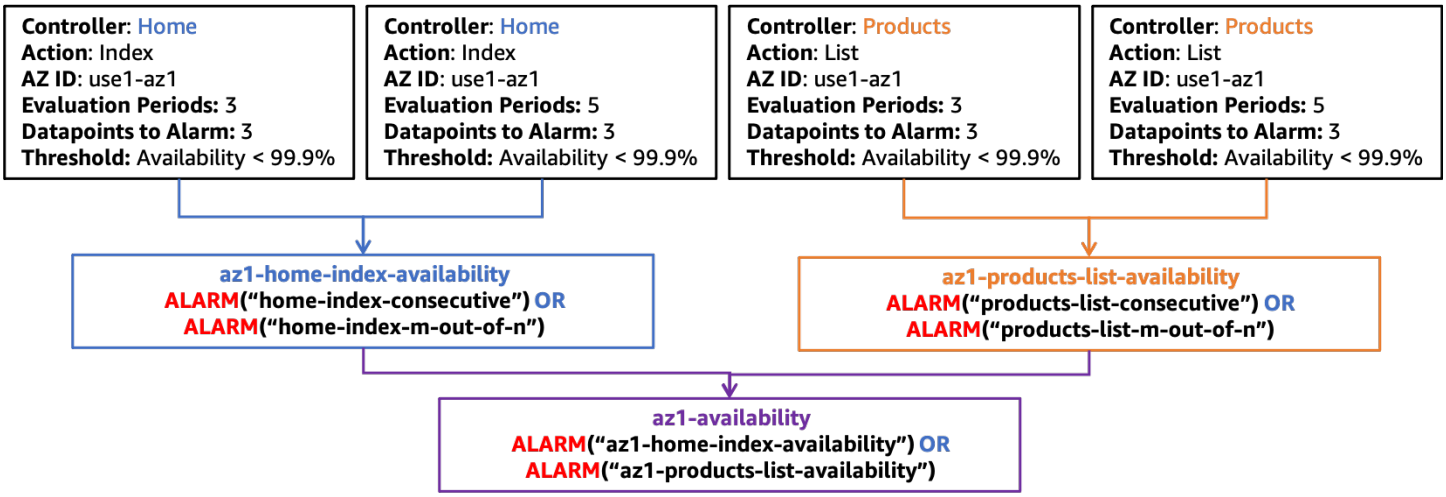
Detecção de falhas com alarmes compostos do CloudWatch

Nas métricas do CloudWatch, cada conjunto de dimensões é uma métrica exclusiva, e você pode criar um alarme do CloudWatch em cada uma. Em seguida, você pode criar alarmes compostos do Amazon CloudWatch para agregar essas métricas.

Para detectar com precisão o impacto, os exemplos neste artigo usarão duas estruturas de alarme diferentes do CloudWatch para cada dimensão definida em que o alarme é ativado. Cada alarme usará um período de um minuto, o que significa que a métrica é avaliada uma vez por minuto. A primeira abordagem usará três pontos de dados de violação consecutivos, definindo os Períodos de Avaliação e os Pontos de Dados para Alarme como três, o que significa impacto por um total de três minutos. A segunda abordagem usará um “M de N” quando quaisquer 3 pontos de dados em uma janela de cinco minutos forem violados, definindo os Períodos de Avaliação como cinco e os Pontos de Dados para Alarme como três. Isso permite a detecção de um sinal constante, bem como um que flutua em um curto espaço de tempo. As durações e o número de pontos de dados contidos aqui são uma sugestão. Use valores que façam sentido para suas workloads.

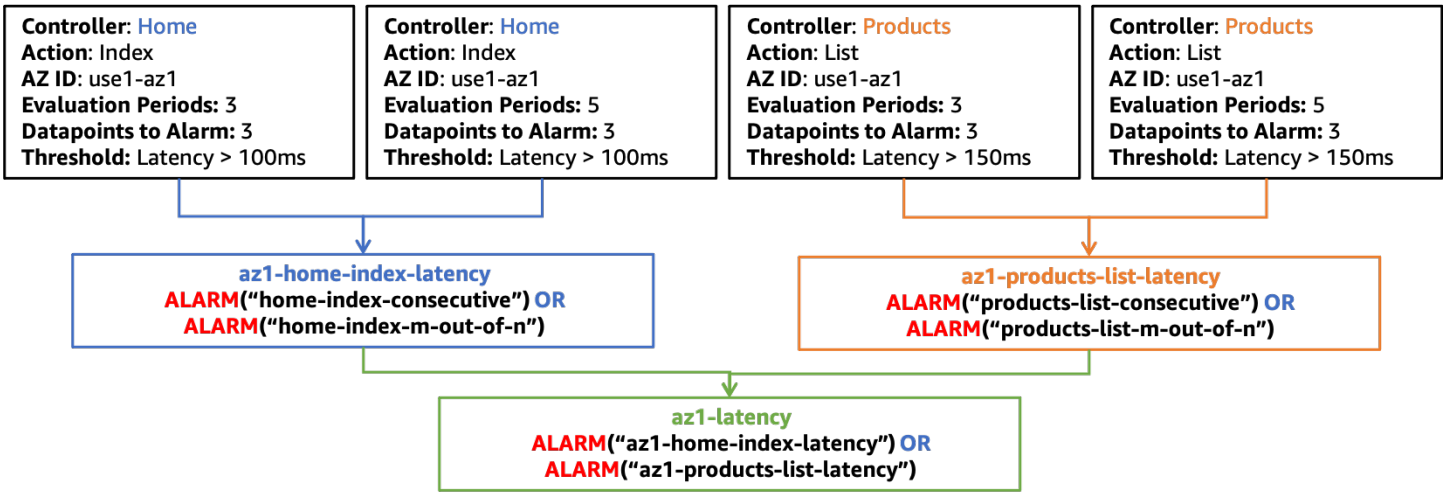
Detectar impacto em uma única Zona de Disponibilidade

Usando essa estrutura, considere uma workload que usa `Controller`, `Action`, `InstanceId`, `AZ-ID` e `Region` como dimensões. A workload tem dois controladores, Products e Home, e uma ação por controlador, List e Index, respectivamente. Ela opera em três Zonas de Disponibilidade na Região `us-east-1`. Você criaria dois alarmes de disponibilidade para cada combinação de `Controller` e `Action` em cada Zona de Disponibilidade, bem como dois alarmes de latência para cada um. Em seguida, você pode optar por criar um alarme composto para verificar a disponibilidade de cada combinação de `Controller` e `Action`. Por fim, você cria um alarme composto que agrega todos os alarmes de disponibilidade da Zona de Disponibilidade. Isso é mostrado na figura a seguir para uma única Zona, `use1-az1`, usando o alarme composto opcional para cada combinação de `Controller` e `Action` (alarmes semelhantes também existiriam para as Zonas `use1-az2` e `use1-az3`, mas não são mostrados para simplificar).



Estrutura de alarme composta para disponibilidade em *use1-az1*

Você também construiria uma estrutura de alarme semelhante para latência, mostrada na figura a seguir.

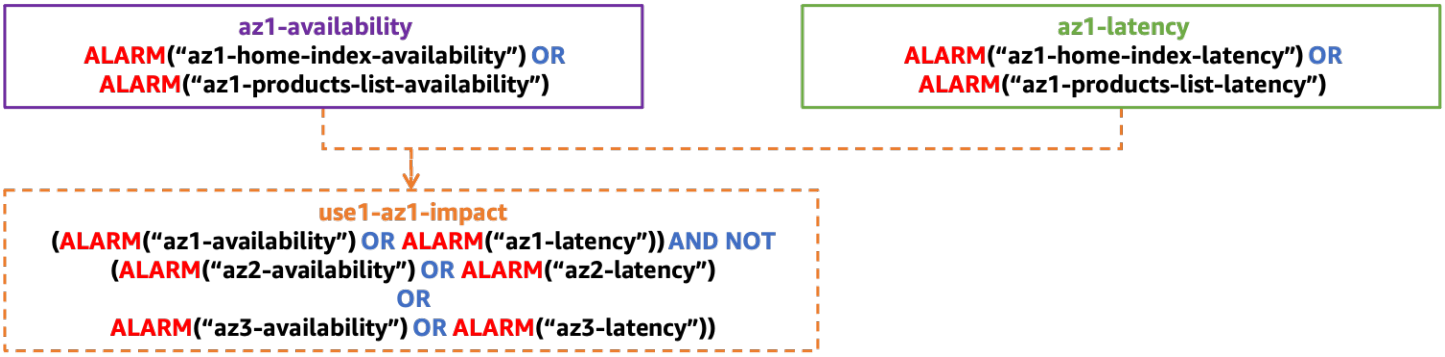


Estrutura de alarme composta para latência em *use1-az1*

Para o restante das figuras desta seção, somente os alarmes compostos `az1-availability` e `az1-latency` serão mostrados no nível superior. Esses alarmes, `az1-availability` e `az1-latency`, dirão se a disponibilidade ficar abaixo ou se a latência ultrapassar os limites definidos em uma Zona de Disponibilidade específica, em qualquer parte da sua workload. Você também pode considerar a medição do throughput para detectar o impacto que impede que sua workload em uma única Zona receba trabalho. Você também pode integrar alarmes produzidos a partir das métricas emitidas por seus canários nesses alarmes compostos. Dessa forma, se o lado do servidor ou o lado do cliente perceberem impactos na disponibilidade ou na latência, o alarme criará um alerta.

Confira se o impacto não é regional

Outro conjunto de alarmes compostos pode ser usado para garantir que somente um evento isolado da Zona de Disponibilidade ative o alarme. Para isso acontecer, um alarme composto da Zona de Disponibilidade é colocado no estado `ALARM` enquanto os alarmes compostos das outras Zonas são colocados no estado `OK`. Isso resultará em um alarme composto por Zona de Disponibilidade usada. Um exemplo é mostrado na figura a seguir (lembre-se de que há alarmes de latência e disponibilidade em `use1-az2` e `use1-az3`, `az2-latency`, `az2-availability`, `az3-latency` e `az3-availability`, que não estão ilustrados para simplificar).



Estrutura de alarme composto para detectar impactos isolados em uma única ZD

Garanta que o impacto não seja causado por uma única instância

Uma única instância (ou uma pequena porcentagem de sua frota geral) pode causar um impacto desproporcional nas métricas de disponibilidade e latência, o que pode fazer com que toda a Zona de Disponibilidade pareça estar afetada, quando na verdade não está. É mais rápido e eficaz remover uma única instância problemática do que evacuar uma Zona.

As instâncias e os contêineres geralmente são tratados como recursos efêmeros, frequentemente substituídos por serviços como AWS Auto Scaling. É difícil criar um novo alarme do CloudWatch toda vez que uma nova instância é criada, mas certamente é possível usando os ganchos de ciclo de vida do Amazon EventBridge ou do Amazon EC2 Auto Scaling. Em vez disso, você pode usar o CloudWatch Contributor Insights para identificar a quantidade de colaboradores nas métricas de disponibilidade e latência.

Como exemplo, para um aplicativo web HTTP, você pode criar uma regra para identificar os principais colaboradores para respostas HTTP 5xx em cada Zona de Disponibilidade. Isso identificará quais instâncias estão contribuindo para uma queda na disponibilidade (nossa métrica de disponibilidade definida acima é determinada pela presença de erros 5xx). Usando o exemplo de log

EMF, crie uma regra usando uma chave de `InstanceId`. Em seguida, filtre o log usando o campo `HttpResponseCode`. Este exemplo é uma regra para a Zona de Disponibilidade `use1-az1`.

```
{
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.InstanceId",
                "IsPresent": true
            },
            {
                "Match": "$.HttpStatusCode",
                "IsPresent": true
            },
            {
                "Match": "$.HttpStatusCode",
                "GreaterThan": 499
            },
            {
                "Match": "$.HttpStatusCode",
                "LessThan": 600
            },
            {
                "Match": "$.AZ-ID",
                "In": ["use1-az1"]
            },
        ],
        "Keys": [
            "$.InstanceId"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/loggroupname"
    ],
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    }
}
```

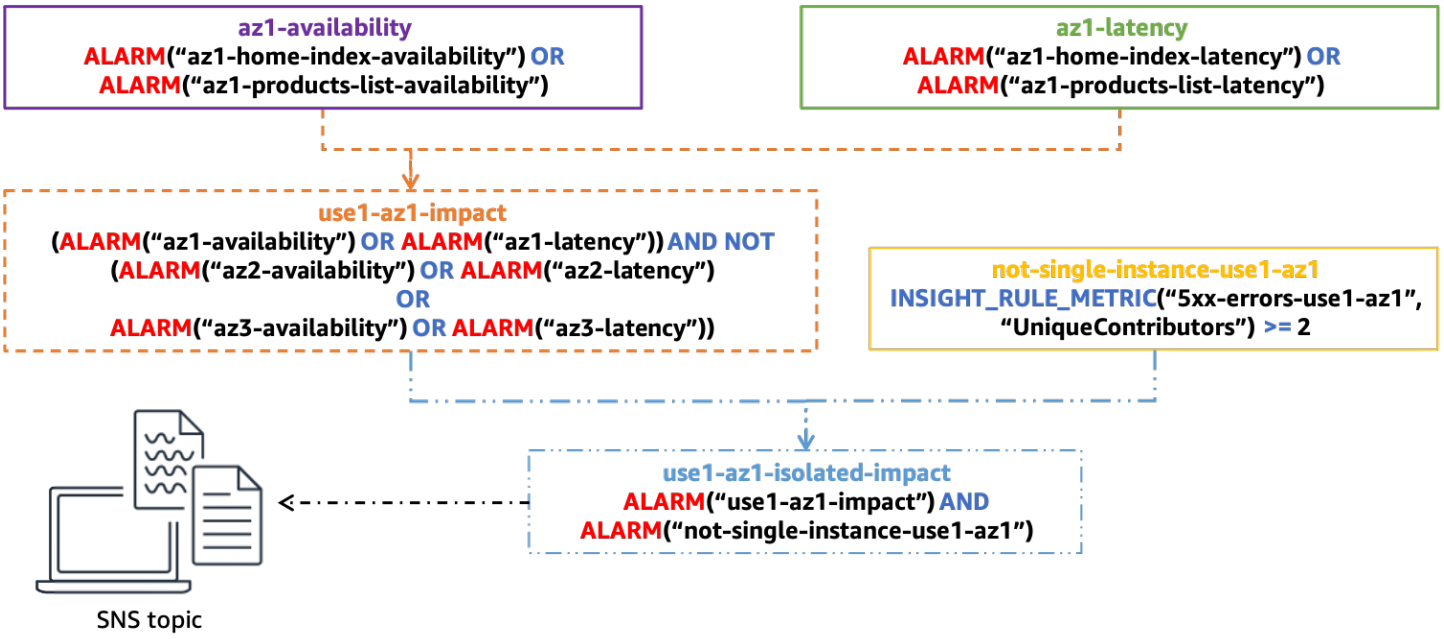
Os alarmes do CloudWatch também podem ser criados com base nessas regras. Você pode criar alarmes com base nas regras do Contributor Insights usando matemática de métrica e a função `INSIGHT_RULE_METRIC` com a métrica `UniqueContributors`. Você também pode criar regras adicionais do Contributor Insights com alarmes do CloudWatch para métricas como latência ou contagens de erros, além daquelas de disponibilidade. Esses alarmes podem ser usados com os alarmes compostos de para impacto em uma Zona de Disponibilidade isolada, de forma a garantir que instâncias únicas não ativem o alarme. A métrica da regra de insights para `use1-az1` pode ser assim:

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

Você pode definir um alarme quando essa métrica ultrapassar um limite; neste exemplo, dois. Ele é ativado quando os colaboradores exclusivos das respostas 5xx ultrapassam esse limite, indicando que o impacto está vindo de mais de duas instâncias. O motivo para esse alarme usar uma comparação “maior que” em vez de “menor que” é para garantir que um valor zero para colaboradores exclusivos não acione o alarme. Isso indica que o impacto não vem de uma única instância. Ajuste esse limite para sua workload individual. Indicamos que esse número seja 5% ou mais do total de recursos na Zona de Disponibilidade. Mais de 5% dos seus recursos sendo afetados mostra significância estatística, considerando um tamanho amostral suficiente.

Juntando tudo isso

A figura a seguir mostra a estrutura de alarme composto completa para uma única Zona de Disponibilidade:



Estrutura completa de alarme composto para determinar o impacto em uma única ZD

O alarme composto final, `use1-az1-isolated-impact`, é ativado quando o alarme composto que indica o impacto isolado na Zona de Disponibilidade devido à latência ou disponibilidade, `use1-az1-aggregate-alarm`, estiver no estado `ALARM` e quando o alarme baseado na regra do Contributor Insights para a mesma ZD, `not-single-instance-use1-az1`, também estiver em estado `ALARM` (o que significa que o impacto é mais do que uma única instância). Você criaria essa pilha de alarmes para cada Zona de Disponibilidade usada pela workload.

Você pode anexar um alerta do Amazon Simple Notification Service (Amazon SNS) a esse alarme final. Todos os alarmes anteriores são configurados sem uma ação. O alerta pode notificar um operador por e-mail para que ele de início a uma investigação manual. Ele também pode iniciar a automação para evacuar a Zona de Disponibilidade. No entanto, cuidado ao criar automações para responder a esses alertas. Após uma evacuação de Zona de Disponibilidade, o resultado deve ser a mitigação das taxas de erro crescentes e a volta do alarme a um estado 0K. Se o impacto ocorrer em outra ZD, é possível que a automação evacue uma segunda ou terceira ZD, potencialmente removendo toda a capacidade disponível da workload. A automação deve verificar se uma evacuação já foi realizada antes de realizar qualquer ação. Você também pode precisar escalar recursos em outras Zonas de Disponibilidade antes que a evacuação seja bem-sucedida.

Ao adicionar novos controladores, ações, um novo microsserviço ou qualquer funcionalidade adicional que você queira monitorar separadamente ao seu aplicativo web MVC, você só precisa modificar alguns alarme nessa configuração. Você criará novos alarmes de disponibilidade e

latência para essa nova funcionalidade e, em seguida, os adicionará aos alarmes compostos de disponibilidade e latência adequados da Zona de Disponibilidade, `az1-latency` e `az1-availability` no exemplo que estamos usando aqui. Os alarmes compostos restantes permanecem estáticos após serem configurados. Isso torna a integração de novas funcionalidades um processo mais simples.

Detecção de falhas usando detecção de discrepâncias

Uma lacuna na abordagem anterior pode surgir quando você vê taxas de erro elevadas em várias Zonas de Disponibilidade que estão ocorrendo por um motivo não correlacionado. Imagine um cenário em que você tenha instâncias do EC2 implantadas em três Zonas de Disponibilidade e em que o limite do seu alarme de disponibilidade seja de 99%. Em seguida, um problema ocorre em uma única ZD, isolando muitas instâncias e fazendo com que a disponibilidade nessa zona caia para 55%. Ao mesmo tempo, mas em uma ZD diferente, uma única instância do EC2 esgota todo o armazenamento em seu volume do EBS e não pode mais gravar arquivos de registros. Ela comece a retornar erros, mas ainda passa pelas verificações de integridade do balanceador de carga, pois elas não acionam a gravação de um arquivo de log. Isso faz com que a disponibilidade caia para 98% nessa ZD. Nesse caso, seu alarme de impacto em uma única Zona de Disponibilidade não seria ativado porque você está vendo impactos em várias ZD. No entanto, você ainda pode mitigar quase todo o impacto evacuando a Zona de Disponibilidade prejudicada.

Em alguns tipos de workload, você pode enfrentar erros de forma consistente em todas as Zonas de Disponibilidade nas quais a métrica de disponibilidade anterior pode não ser útil. Veja AWS Lambda, por exemplo. A AWS permite que os clientes criem seu próprio código para execução na função do Lambda. Para usar o serviço, você precisa carregar seu código em um arquivo ZIP, incluindo dependências, e definir o ponto de entrada para a função. Mas, às vezes, os clientes erram nessa parte. Por exemplo, eles podem esquecer uma dependência crítica no arquivo ZIP ou digitar incorretamente o nome do método na definição da função do Lambda. Isso faz com que a função não seja invocada e resulta em um erro. A AWS Lambda vê esses tipos de erros o tempo todo, mas eles não são indicativos de que algo esteja necessariamente errado. No entanto, coisas como problemas na Zona de Disponibilidade também podem fazer com que esses erros apareçam.

Para encontrar sinal no meio desses ruídos, você pode usar a detecção de discrepâncias para determinar se há uma distorção estatisticamente significativa no número de erros entre as Zonas de Disponibilidade. Embora vejamos erros em várias Zonas de Disponibilidade, se realmente houvesse uma falha em uma delas, veríamos uma taxa de erro muito maior nessa ZD em comparação com as outras, ou potencialmente muito menor. Mas quanto maior ou menor?

Uma forma de fazer essa análise é usar um teste qui-quadrado (X^2) para detectar diferenças estatisticamente significativas nas taxas de erro entre Zonas (há muitos algoritmos diferentes para realizar a detecção de discrepâncias). Vamos ver como funciona o teste qui-quadrado.

Um teste qui-quadrado avalia a probabilidade de alguma distribuição de resultados ocorrer. Nesse caso, estamos interessados na distribuição de erros em algum conjunto definido de ZDs. Neste exemplo, para facilitar a matemática, considere quatro Zonas de Disponibilidade.

Primeiro, estabeleça a hipótese nula, que define o que você acredita ser o resultado padrão. Nesse teste, você espera que os erros sejam distribuídos uniformemente em cada ZD: essa é a hipótese nula. Em seguida, gere a hipótese alternativa: os erros não estão distribuídos uniformemente, indicando um problema em uma Zona de Disponibilidade. Agora você pode testar essas hipóteses usando dados de suas métricas. Para isso, você fará uma amostra de suas métricas em um período de cinco minutos. Suponha que você obtenha 1000 pontos de dados publicados nesse período, no qual você vê 100 erros no total. Você espera que, com uma distribuição uniforme, os erros ocorram 25% das vezes em cada uma das quatro ZDs. Suponha que a tabela a seguir mostre o que você esperava em comparação com o que você realmente viu.

Tabela 1: erros esperados versus erros reais observados

ZD	Esperados	Reais
`use1-az1`	25	20
`use1-az2`	25	20
`use1-az3`	25	25
`use1-az4`	25	35

Você vê que, na realidade, a distribuição não é uniforme. No entanto, você pode acreditar que isso ocorreu devido a algum nível de aleatoriedade nos pontos de dados coletados. Há probabilidade de que esse tipo de distribuição possa ocorrer no conjunto amostral e ainda supor que a hipótese nula seja verdadeira. Isso leva à seguinte pergunta: qual é a probabilidade de obter um resultado pelo menos nesse extremo? Se essa probabilidade estiver abaixo de um limite definido, você rejeita a hipótese nula. Para ser estatisticamente significativa, essa probabilidade deve ser de 5% ou menos[1]

[1] Craparo, Robert M. (2007). “Nível de significância”. Em Salkind, Neil J. Encyclopedia of Measurement and Statistics 3. Thousand Oaks, CA: SAGE Publications. pp. 889—891. ISBN 1-412-91611-9.

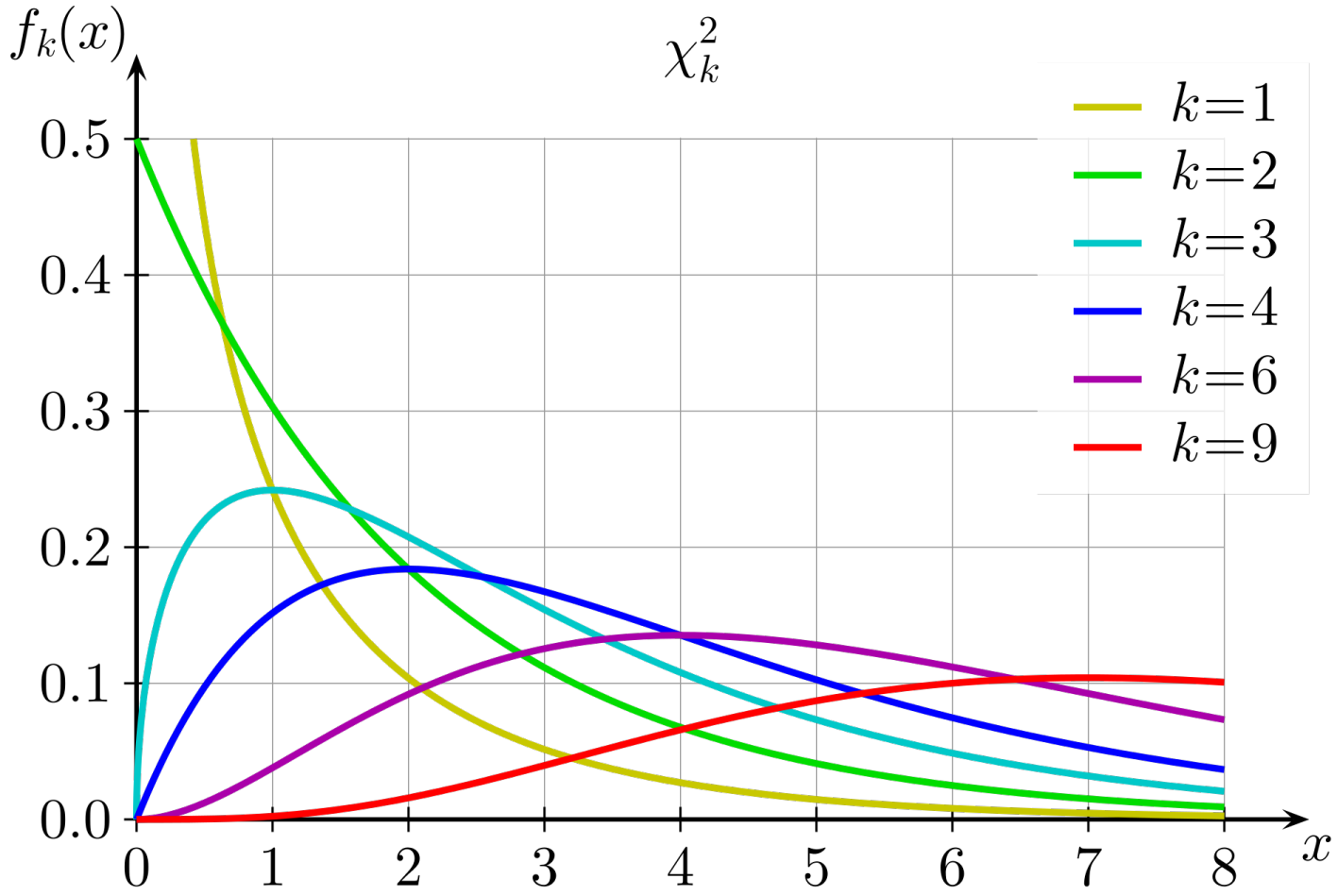
Como você calcula a probabilidade desse resultado? Você usa a estatística X^2, que fornece distribuições muito bem estudadas e pode ser usada para determinar a probabilidade de obter um resultado tão extremo ou mais extremo usando essa fórmula.

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

Para nosso exemplo, isso resulta em:

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

Então, o que 6 significa em termos de probabilidade? Você precisa observar uma distribuição qui-quadrada com o grau apropriado de liberdade. A figura a seguir mostra várias distribuições de qui-quadrado para diferentes graus de liberdade.



Distribuições de qui-quadrado para diferentes graus de liberdade

O grau de liberdade é calculado como um a menos que o número de opções no teste. Nesse caso, como há quatro Zonas de Disponibilidade, o grau de liberdade é três. Então, você quer saber a área sob a curva (a integral) para x ≥ 6 no gráfico k = 3. Você também pode usar uma tabela pré-calculada com valores comumente usados para aproximar esse valor.

Tabela 2: valores críticos do qui-quadrado

Graus de liberdade	Probabilidade menor que o valor crítico				
	0,75	0,90	0,95	0,99	0,999
1	1,323	2,706	3,841	6,635	10,828
2	2,773	4,605	5,991	9,210	13,816

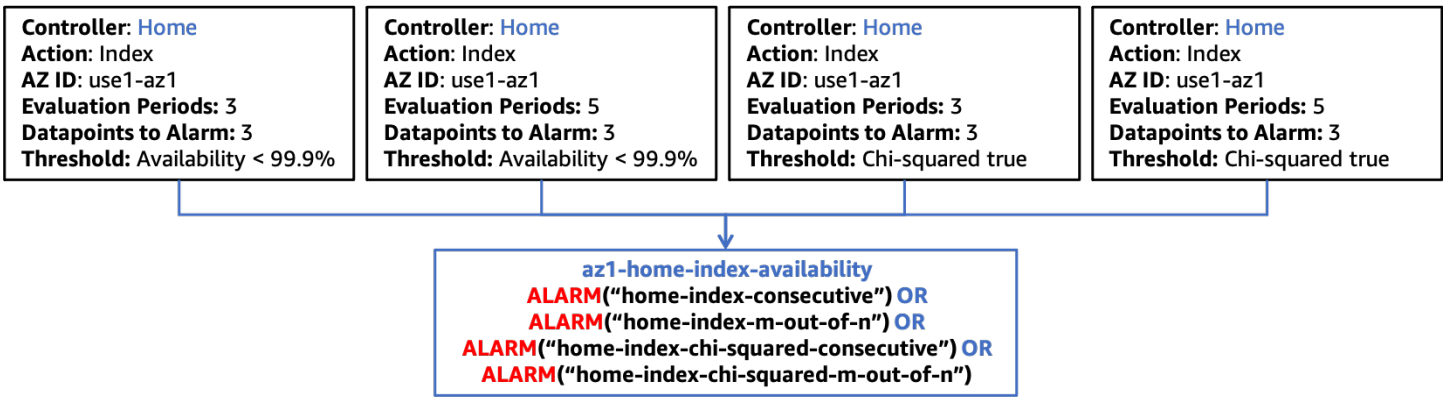
Graus de liberdade	Probabilidade menor que o valor crítico				
	0,75	0,90	0,95	0,99	0,999
3	4,108	6,251	7,815	11,345	16,266
4	5,385	7,779	9,488	13,277	18,467

Para três graus de liberdade, o valor qui-quadrado de seis fica entre as colunas de probabilidade de 0,75 e 0,9. Ou seja, há mais de 10% de chance de que essa distribuição ocorra, o que não é inferior ao limite de 5%. Portanto, você aceita a hipótese nula e determina que não há uma diferença estatisticamente significativa nas taxas de erro entre as Zonas de Disponibilidade.

A realização de um teste estatístico qui-quadrado não tem suporte nativo na matemática de métricas do CloudWatch, então você precisará coletar as métricas de erro aplicáveis e executar o teste em um ambiente computacional, como o Lambda. Você pode realizar esse teste em algo como um controlador/ação MVC ou no nível de microsserviço individual, ou no nível da Zona de Disponibilidade. Você precisará considerar se um problema na Zona de Disponibilidade afetaria igualmente cada Controlador/Ação ou microsserviço, ou se algo como uma falha de DNS poderia causar impacto em um serviço de baixo throughput e não em um serviço de alto throughput, o que poderia mascarar o impacto quando agregado. Em ambos os casos, selecione as dimensões apropriadas para criar a consulta. O nível de granularidade também afetará os alarmes resultantes do CloudWatch que você criar.

Colete a métrica de contagem de erros para cada ZD e para cada Controlador/Ação em um período de tempo especificado. Primeiro, calcule o resultado do teste qui-quadrado como verdadeiro (houve uma distorção estatisticamente significativa) ou falso (não houve, ou seja, a hipótese nula é válida). Se o resultado for falso, publique um ponto de dados 0 em seu fluxo métrico de forma a obter resultados qui-quadrados para cada Zona de Disponibilidade. Se o resultado for verdadeiro, publique um ponto de dados 1 para a Zona de Disponibilidade com os erros mais distantes do valor esperado, e um 0 para os outros (consulte o Apêndice B – Exemplo de cálculo do qui-quadrado para ver um exemplo de código que pode ser usado em uma função do Lambda). Você pode seguir a mesma abordagem dos alarmes de disponibilidade anteriores, usando a criação de um alarme métrico de 3 em uma linha do CloudWatch e um alarme métrico de 3 em 5 do CloudWatch com base nos pontos de dados produzidos pela função do Lambda. Como nos exemplos anteriores, essa abordagem pode ser modificada para usar mais ou menos pontos de dados em um período menor ou mais longo.

Em seguida, adicione esses alarmes ao alarme de disponibilidade existente da ZD na combinação de Controller e Action, mostrada na figura a seguir.



Integração do teste estatístico qui-quadrado com alarmes compostos

Conforme mencionado anteriormente, ao integrar uma nova funcionalidade na sua workload, você só precisa criar os alarmes métricos apropriados do CloudWatch que sejam específicos a essa nova funcionalidade e atualizar o próximo nível na hierarquia de alarmes compostos para inclui-los. O resto da estrutura de alarmes permanece estática.

Detecção de falhas de recursos zonais de instância única

Em alguns casos, você pode ter uma única instância ativa de um recurso zonal, geralmente sistemas que exigem um componente de gravação único, como um banco de dados relacional (por exemplo, o Amazon RDS) ou um cache distribuído (por exemplo, o Amazon ElastiCache for Redis). Se um problema em uma única Zona de Disponibilidade afetar a Zona onde o recurso principal está, isso pode impactar todas as Zonas que acessam o recurso. Isso poderia fazer com que limites de disponibilidade fossem ultrapassados em todas as ZDs, ou seja, a primeira abordagem não identificaria corretamente a única Zona sendo impactada. Além disso, você provavelmente veria taxas de erro semelhantes em cada ZD, o que significa que a análise de discrepância também não detectaria o problema. Isso significa que você precisa implementar observabilidade adicional para detectar esse cenário.

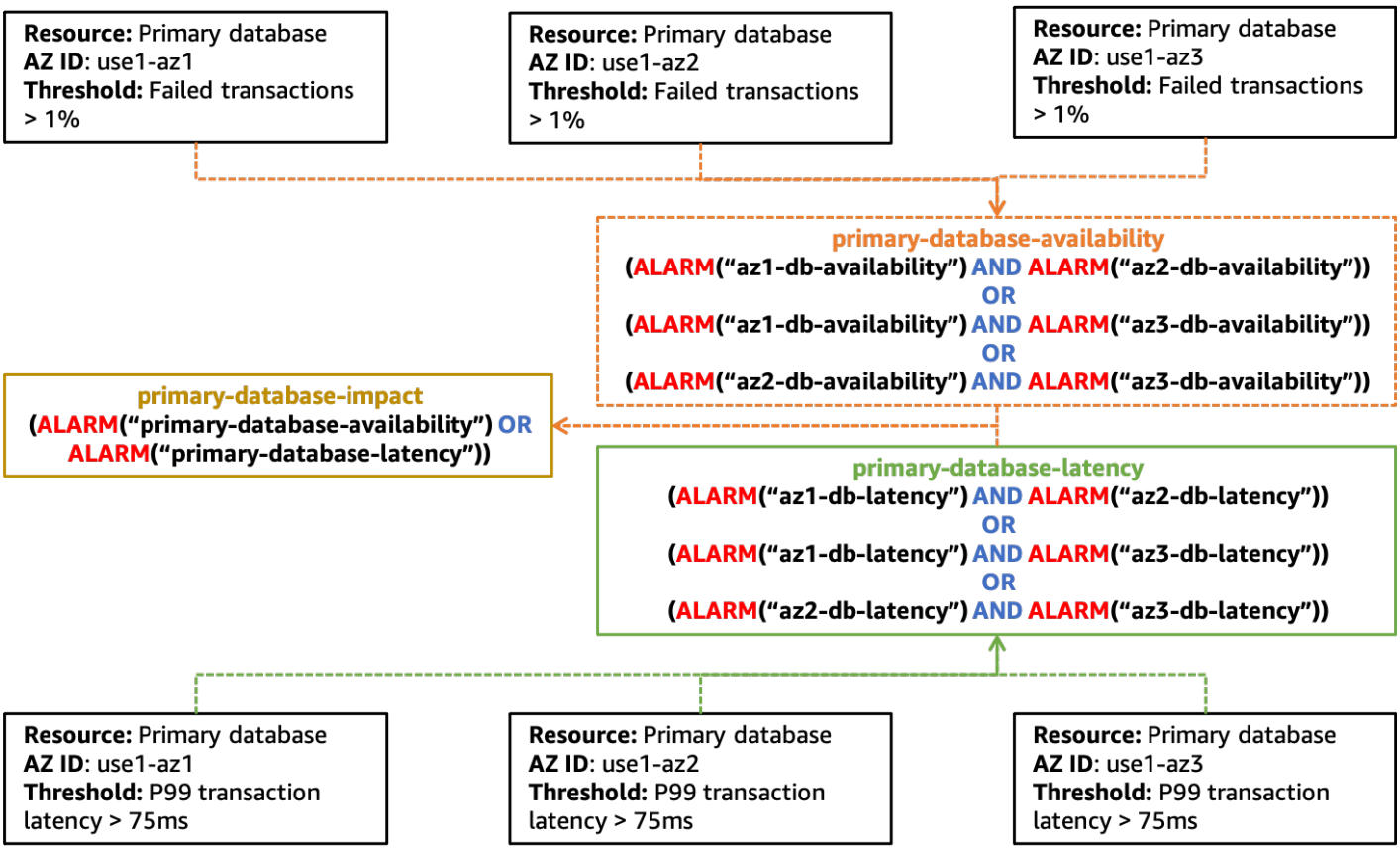
É provável que o recurso que está gerando preocupação produza as próprias métricas sobre sua integridade. Porém, durante um problema de ZD, esse recurso pode não ser capaz de fornecer essas métricas. Nesse cenário, você deve criar ou atualizar alarmes para saber quando está voando às cegas. Se houver métricas importantes que você já monitora e que possuem alarmes, você pode configurar o alarme para tratar os dados ausentes como violação. Isso ajudará você a saber se o

recurso para de reportar dados. Ele pode ser incluído no mesmos alarmes em sequência e m de n usados anteriormente.

Também é possível que, em algumas das métricas que indicam a integridade do recurso, ele publique um ponto de dados de valor zero quando não há atividade. Se o problema estiver impedindo interações com o recurso, não será possível usar a abordagem de dados ausentes para esses tipos de métricas. Você provavelmente também não quer colocar um alarme sobre o valor zero, pois pode haver cenários em que isso esteja dentro dos limites normais. A melhor abordagem para detectar esse tipo de problema é com métricas produzidas pelos recursos que usam essa dependência. Nesse caso, queremos detectar impacto em várias Zonas de Disponibilidade usando alarmes compostos. Esses alarmes precisam usar algumas categorias de métricas críticas relacionadas ao recurso. Alguns exemplos estão listados abaixo:

- Throughput — A taxa de entrada de unidades de trabalho. Isso pode incluir transações, leituras, gravações e assim por diante.
- Disponibilidade — Mede o número de unidades de trabalho bem-sucedidas versus as unidades com falha.
- Latência — Mede vários percentis de latência para um trabalho bem-sucedido realizado em várias operações críticas.

  Mais uma vez, você pode criar os alarmes métricos em sequência e m de n para cada métrica, em cada categoria a ser mensurada. Como antes, eles podem ser combinados em um alarme composto para determinar se esse recurso compartilhado é a fonte de impacto nas ZDs. Você quer poder identificar impacto em mais de uma ZD com os alarmes compostos, mas o impacto não precisa necessariamente ser em todas as ZDs. A estrutura de alarme composto de alto nível para esse tipo de abordagem é mostrada na figura a seguir.



Um exemplo de criação de alarmes para detectar impacto causado por um único recurso em várias Zonas de Disponibilidade

Você notará que esse diagrama é menos prescritivo em relação a quais tipos de alarmes métricos devem ser usados e à hierarquia dos alarmes compostos. Isso ocorre porque descobrir esse tipo de problema pode ser difícil e exigirá atenção cuidadosa aos sinais certos para o recurso compartilhado. Esses sinais também podem precisar ser avaliados de maneiras específicas.

Além disso, observe que o alarme `primary-database-impact` não está associado a uma Zona de Disponibilidade específica. Isso ocorre porque a instância primária do banco de dados pode estar localizada em qualquer ZD configurada para uso. Não há uma métrica do CloudWatch que especifique onde ela está. Quando esse alarme for ativado, veja isso como um sinal de que pode haver um problema com o recurso e inicie um failover para outra Zona de Disponibilidade, caso isso não tenha sido feito automaticamente. Depois de mover o recurso para outra Zona de Disponibilidade, você pode esperar e ver se o alarme para Zona de Disponibilidade isolada está ativado, ou pode optar por invocar preventivamente seu plano de evacuação da Zona.

Resumo

Esta seção descreveu três abordagens para ajudar a identificar problemas em uma única Zona de Disponibilidade. Use as abordagens em conjunto para ter uma visão holística da integridade da sua workload.

A abordagem de alarme composto do CloudWatch permite que você encontre problemas em que a distorção na disponibilidade não é estatisticamente significativa, digamos, disponibilidades de 98% (a Zona de disponibilidade Prejudicada), 100% e 99,99%, que não sejam causados por um único recurso compartilhado.

A detecção de discrepâncias ajudará a detectar problemas em uma única Zona de Disponibilidade onde haja erros não correlacionados em várias ZDs, todos ultrapassando o limite de alarme.

Por fim, identificar degradação de um recurso zonal de única instância ajuda a descobrir quando um problema de Zona de Disponibilidade afeta um recurso compartilhado entre ZDs.

Os alarmes resultantes de cada um desses padrões podem ser combinados em uma hierarquia de alarmes compostos do CloudWatch para descobrir quando ocorrem problemas em uma única Zona de Disponibilidade que impactam a disponibilidade ou a latência da sua workload.

Padrões de evacuação da zona de disponibilidade

Depois de detectar o impacto em uma única zona de disponibilidade, a próxima etapa é evacuar essa zona de disponibilidade. A evacuação precisa alcançar dois resultados.

Primeiro, você deve parar de enviar trabalhos para a zona de disponibilidade afetada. Isso pode ter significados diferentes em arquiteturas diferentes. Em um workload de solicitação/resposta, isso significaria impedir que solicitações HTTP ou gRPC provenientes de seus clientes fossem enviadas para o balanceador de carga ou outros recursos na zona de disponibilidade. Em um sistema de processamento em lote ou em filas, isso pode significar impedir que os recursos computacionais processem o trabalho na zona de disponibilidade afetada. Você também precisará evitar que recursos nas zonas de disponibilidade não afetadas interajam com recursos na zona de disponibilidade afetada, por exemplo, uma instância do EC2 enviando tráfego para um endpoint da VPC de interface na zona de disponibilidade afetada ou se conectando à instância primária de um banco de dados.

O segundo resultado é impedir que novas capacidades sejam provisionadas na zona de disponibilidade afetada. Isso é importante porque provavelmente os novos recursos, como instâncias do EC2 ou contêineres, que foram provisionados na zona de disponibilidade afetada, terão o mesmo impacto que os recursos existentes. Além disso, como o primeiro resultado impede que o trabalho seja enviado para esses novos recursos, eles não conseguem absorver a carga para a qual foram provisionados. Isso leva ao aumento da carga sobre os recursos existentes, o que pode, em última instância, levar ao apagão ou à indisponibilidade total do workload. Há vários serviços de ajuste de escala automático disponíveis em AWS onde isso se aplica: Amazon EC2 Auto Scaling, Application Auto Scaling e AWS Auto Scaling. Além disso, serviços como Amazon ECS, Amazon EKS e AWS Batch podem programar trabalhos em hosts em várias zonas de disponibilidade em uma VPC como parte de sua operação normal.

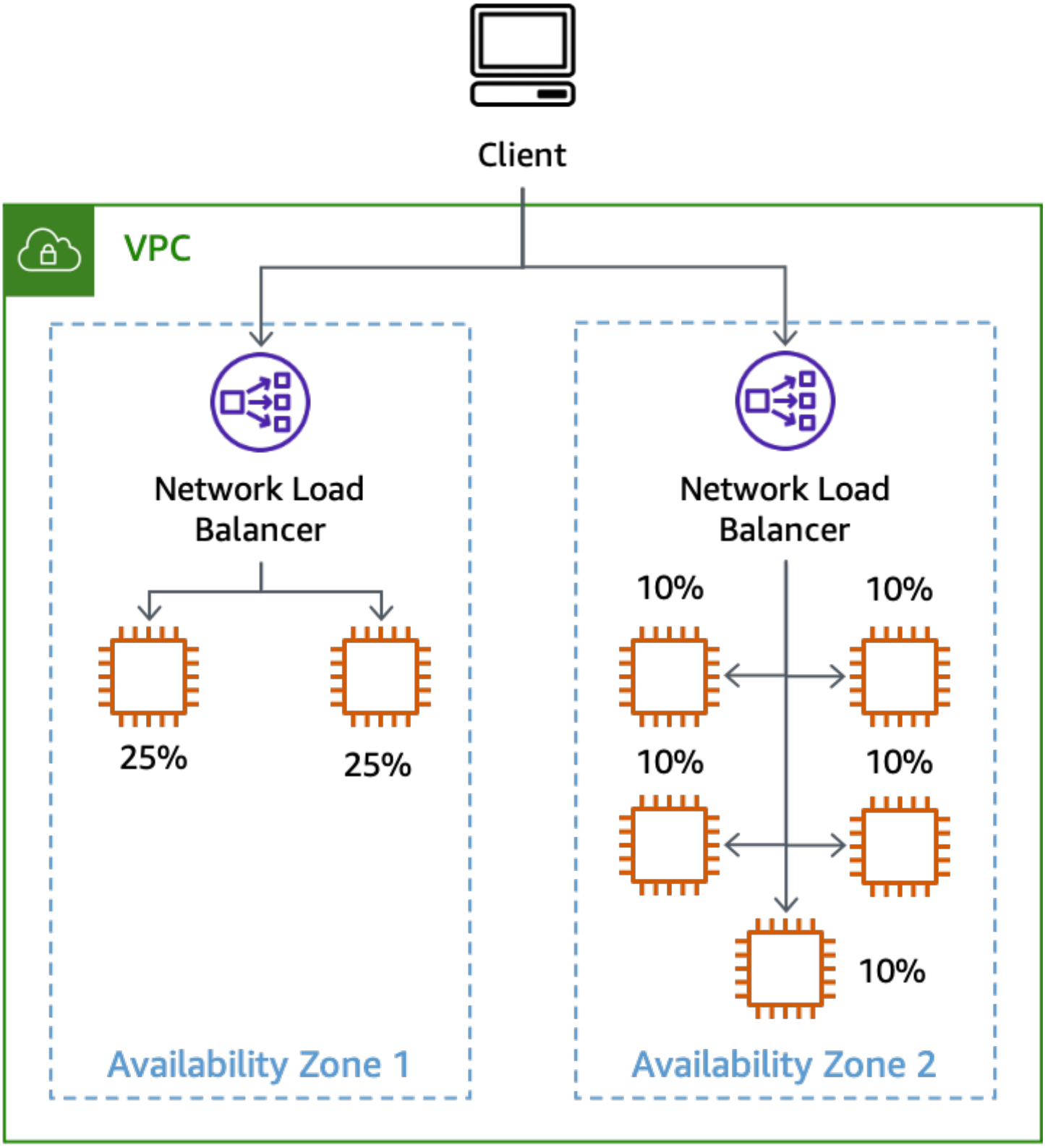
Tópicos

- Independência da zona de disponibilidade
- Ambientes de gerenciamento e planos de dados
- Evacuação controlada por plano de dados
- Evacuação controlada por ambiente de gerenciamento
- Resumo

Independência da zona de disponibilidade

Para alcançar o primeiro resultado, ou seja, parar de enviar trabalho para a zona de disponibilidade afetada, a evacuação exige que você implemente a Independência da zona de disponibilidade (AZI), também chamada de Afinidade da zona de disponibilidade. Esse padrão de arquitetura isola recursos dentro de uma zona de disponibilidade e impede a interação entre recursos em diferentes zonas de disponibilidade, exceto quando for absolutamente necessário, como conectar-se a uma instância de banco de dados primária em uma zona de disponibilidade diferente.

Em um workload do tipo solicitação/resposta, a implementação da AZI exige que você desative o balanceamento de carga entre zonas dos Application Load Balancers (ALB), Classic Load Balancers (CLB) e Network Load Balancers (NLB) (o balanceamento de carga entre zonas fica desativado por padrão para NLBs). Desabilitar balanceamento de carga entre zonas tem algumas desvantagens. Quando você desativa o balanceamento de carga entre zonas, o tráfego é dividido uniformemente entre cada zona de disponibilidade, independentemente de quantas instâncias estejam em cada zona. Se você tiver recursos desbalanceados ou grupos do Auto Scaling, isso pode sobrecarregar os recursos em uma zona de disponibilidade com menos recursos que outras. Isso é mostrado na figura a seguir, onde duas instâncias na zona de disponibilidade 1 estão recebendo 25% da carga e as cinco instâncias na zona de disponibilidade 2 estão recebendo 10% da carga cada uma.



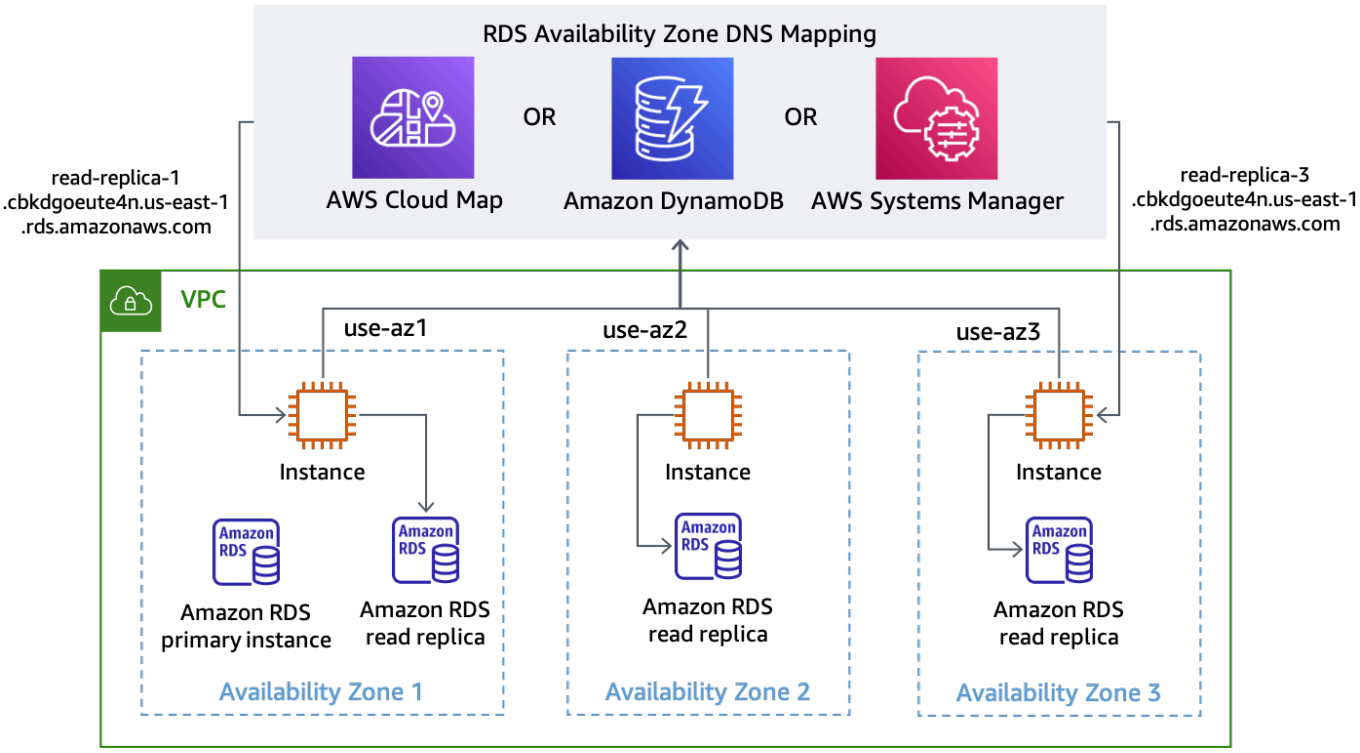
O efeito da desativação do balanceamento de carga entre zonas com instâncias desbalanceadas

Outros serviços zonais que você usa também precisarão ser implementados usando padrões AZI para apoiar a evacuação efetiva da zona de disponibilidade. Por exemplo, os endpoints da VPC de interface fornecem nomes DNS específicos para cada zona de disponibilidade na qual o endpoint da interface é disponibilizado.

Um desafio com a implementação da AZI é com bancos de dados, especialmente porque a maioria dos bancos de dados relacionais suporta apenas um único gravador primário a qualquer momento. Ao se comunicar com a instância primária, talvez seja necessário cruzar o limite da zona de disponibilidade. Muitos serviços de banco de dados AWS oferecem suporte a uma configuração multi-AZ definida pelo usuário e têm um atributo de failover multi-AZ integrado, como Amazon RDS ou Amazon Aurora. Em muitos cenários de falha, o serviço pode detectar o impacto e fazer o failover automático do banco de dados para uma zona de disponibilidade diferente quando ocorre um problema. No entanto, durante uma falha cinzenta, o serviço pode não detectar o impacto que está afetando o workload ou o impacto pode não estar relacionado ao banco de dados. Nesses casos, depois de detectar o impacto em uma zona de disponibilidade, você pode invocar manualmente um failover para mover o banco de dados principal. Isso permite que você reaja de forma eficaz a um simples comprometimento da zona de disponibilidade.

Se você estiver usando réplicas de leitura com esses bancos de dados, talvez também queira implementar a AZI para eles, pois não é possível fazer o failover de uma réplica de leitura para uma zona de disponibilidade diferente, como acontece com o banco de dados primário. Se você tiver uma única réplica de leitura na zona de disponibilidade 1 e as instâncias de três zonas de disponibilidade estiverem configuradas para usá-la, um comprometimento que esteja afetando a zona de disponibilidade 1 também afetará as operações nas outras duas zonas de disponibilidade. Esse é o impacto que você quer evitar.

Para instâncias do RDS, você recebe um endpoint DNS para acessar a réplica em uma zona de disponibilidade específica. Para obter a AZI, você precisa de uma réplica de leitura por zona de disponibilidade e de seu aplicativo saber, de alguma maneira, qual endpoint de réplica usar para a zona de disponibilidade em que está. Uma abordagem que você pode adotar é usar o ID da zona de disponibilidade como parte do identificador do banco de dados, algo como `use1-az1-read-replica.cbkdgoeute4n.us-east-1.rds.amazonaws.com`. Você também pode fazer isso usando a descoberta de serviços (como com AWS Cloud Map) ou pesquisando um mapa simples armazenado no AWS Systems Manager Parameter Store ou em uma tabela do DynamoDB. Esse conceito é mostrado na figura a seguir.

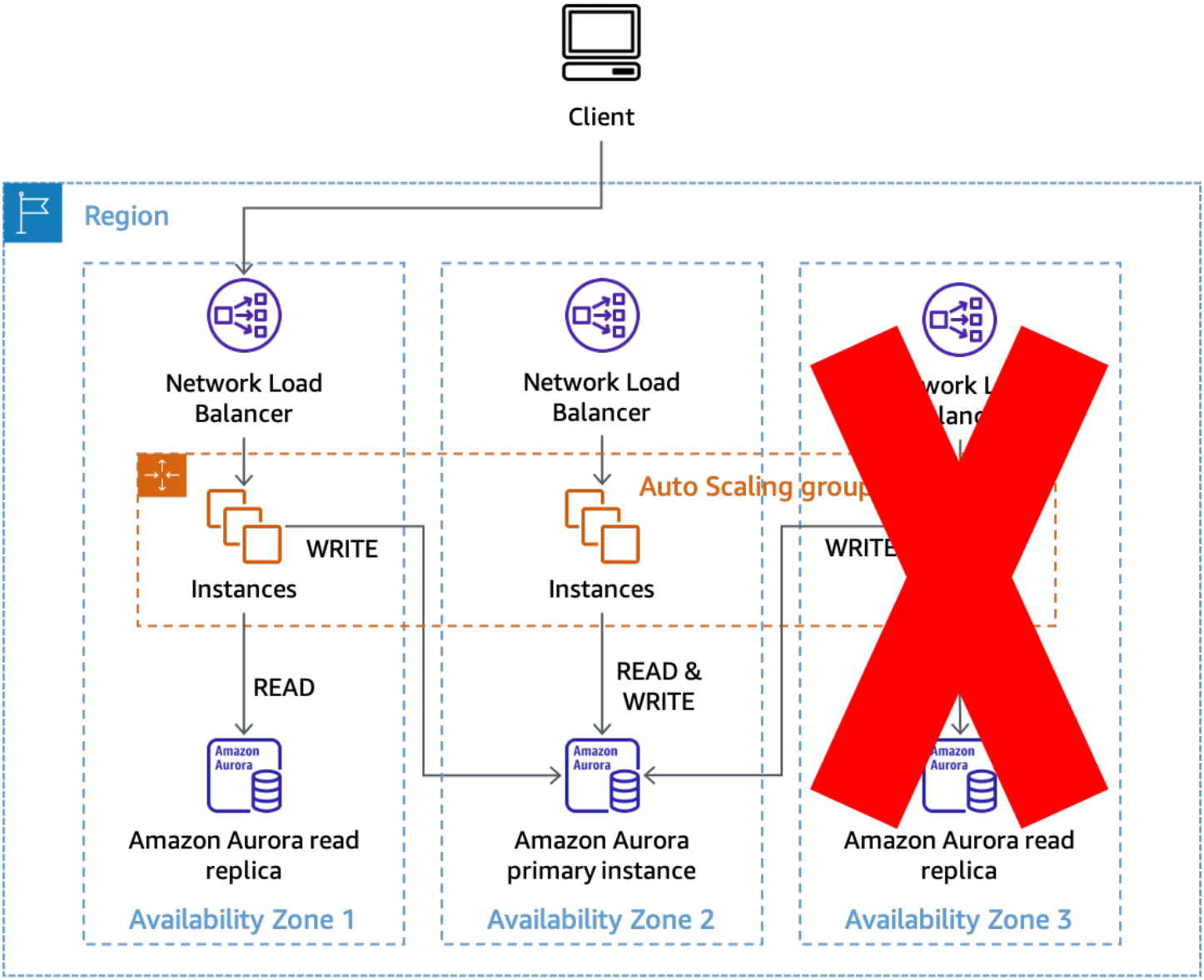


Descoberta de nomes DNS de endpoints do RDS para cada zona de disponibilidade

A configuração padrão do Amazon Aurora é fornecer um único endpoint de leitura que equilibra a carga das solicitações entre as réplicas de leitura disponíveis. Para implementar a AZI usando o Aurora, você pode usar um endpoint personalizado para cada réplica de leitura usando o tipo ANY (para viabilizar a promoção de uma réplica de leitura, se necessário). Atribua um nome ao endpoint personalizado com base na ID da zona de disponibilidade em que a réplica será implantada. Em seguida, você pode usar o nome DNS fornecido pelo endpoint personalizado para se conectar a uma réplica de leitura específica em uma zona de disponibilidade específica, como mostra a figura a seguir.

Uso de um endpoint personalizado para uma réplica de leitura do Aurora

Quando seu sistema é arquitetado dessa forma, a evacuação da zona de disponibilidade torna-se uma tarefa muito mais simples. Por exemplo, na figura a seguir, quando há um comprometimento afetando a zona de disponibilidade 3, as operações de leitura e gravação nas zonas de disponibilidade 1 e 2 não são afetadas.



Usando a AZI para evitar impactos com réplicas de leitura do Amazon Aurora

Como alternativa, se a zona de disponibilidade 2 fosse afetada, as operações de leitura ainda seriam bem-sucedidas nas zonas de disponibilidade 1 e 3. Então, se o Amazon Aurora não fizer o failover automático do banco de dados principal, você poderá invocar manualmente um failover para uma zona de disponibilidade diferente para restaurar a capacidade de processamento de gravações. Essa abordagem evita a necessidade de fazer alterações na configuração das conexões do banco de dados quando for necessário evacuar uma zona de disponibilidade. Ao minimizar as mudanças necessárias e manter o processo o mais simples possível o tornará mais confiável.

Ambientes de gerenciamento e planos de dados

Antes de chegarmos aos padrões reais que você pode usar para realizar a evacuação de uma zona de disponibilidade, precisamos discutir os conceitos de ambientes de gerenciamento e planos de dados. AWS faz uma distinção entre ambientes de gerenciamento e planos de dados em nossos serviços. Os ambientes de gerenciamento são o mecanismo envolvido em fazer alterações em um sistema — adicionar recursos, excluir recursos, modificar recursos — e fazer com que essas alterações sejam propagadas para onde devam entrar em vigor, como atualizar uma configuração de rede para um ALB ou criar uma função de AWS Lambda.

Os planos de dados são a função principal desses recursos, como por exemplo, executar uma instância do EC2 ou obter itens de uma tabela do Amazon DynamoDB ou colocar itens na tabela. Para uma discussão mais detalhada sobre ambientes de gerenciamento e planos de dados, consulte Estabilidade estática usando zonas de disponibilidade e Limites de isolamento de falhas AWS.

Para os fins deste documento, considere que os ambientes de gerenciamento tendem a ter mais partes móveis e dependências que os planos de dados. Isso torna estatisticamente mais provável que o ambiente de gerenciamento fique prejudicado em comparação com o plano de dados. Isso é especialmente relevante para serviços que fornecem AZI, como Amazon EC2 e EBS, porque partes desses serviços têm ambientes de gerenciamento que também são independentes de zona e podem ser afetados durante um evento em uma única ZD.

Embora as ações do ambiente de gerenciamento possam ser usadas para realizar a evacuação da ZD, com base nas informações anteriores, elas podem ter uma probabilidade menor de sucesso, especialmente durante um evento de falha. Para aumentar a probabilidade de mitigar o impacto com sucesso, você pode usar dois padrões diferentes. O primeiro padrão depende apenas das ações do plano de dados para mitigar inicialmente o impacto, impedindo que o trabalho seja encaminhado ou impedindo que o trabalho seja feito na zona de disponibilidade afetada. Em seguida, pode-se tentar o segundo padrão para atualizar a configuração dos recursos com ações do ambiente de gerenciamento para impedir que a capacidade seja provisionada na zona de disponibilidade afetada, bem como interromper a comunicação entre zonas de disponibilidade com a zona de disponibilidade afetada.

Os padrões de recuperação discutidos nesta seção são grandes alertas vermelhos. Eles são os mecanismos que você usará para realizar ações em grande escala, rapidamente, semelhantes a puxar um cabo Andon em uma linha de montagem. Tais mecanismos assumem que os workloads já tentaram estratégias como tentar novamente com recuo exponencial com jitter no código para superar erros transitórios. Isso significa que, quando um impacto isolado na zona de disponibilidade

é detectado, seus efeitos na disponibilidade ou na latência são graves o suficiente para exigir a evacuação da zona de disponibilidade para uma mitigação eficaz.

Evacuação controlada por plano de dados

Há várias soluções que você pode implementar para realizar a evacuação de uma zona de disponibilidade usando ações somente do plano de dados. Esta seção descreverá três delas e os casos de uso em que talvez você opte por escolher um em vez dos outros.

Ao usar qualquer uma dessas soluções, você precisará garantir que tenha capacidade suficiente nas zonas de disponibilidade restantes para lidar com a carga da zona de disponibilidade da qual você está saindo. A maneira mais resiliente de fazer isso é ter a capacidade necessária pré-provisionada em cada zona de disponibilidade. Se estiver usando três zonas de disponibilidade, você teria 50% da capacidade necessária para lidar com o pico de carga implantada em cada uma, de modo que a perda de uma única zona de disponibilidade ainda continuaria com 100% da capacidade necessária sem precisar depender de um ambiente de gerenciamento para provisionar mais.

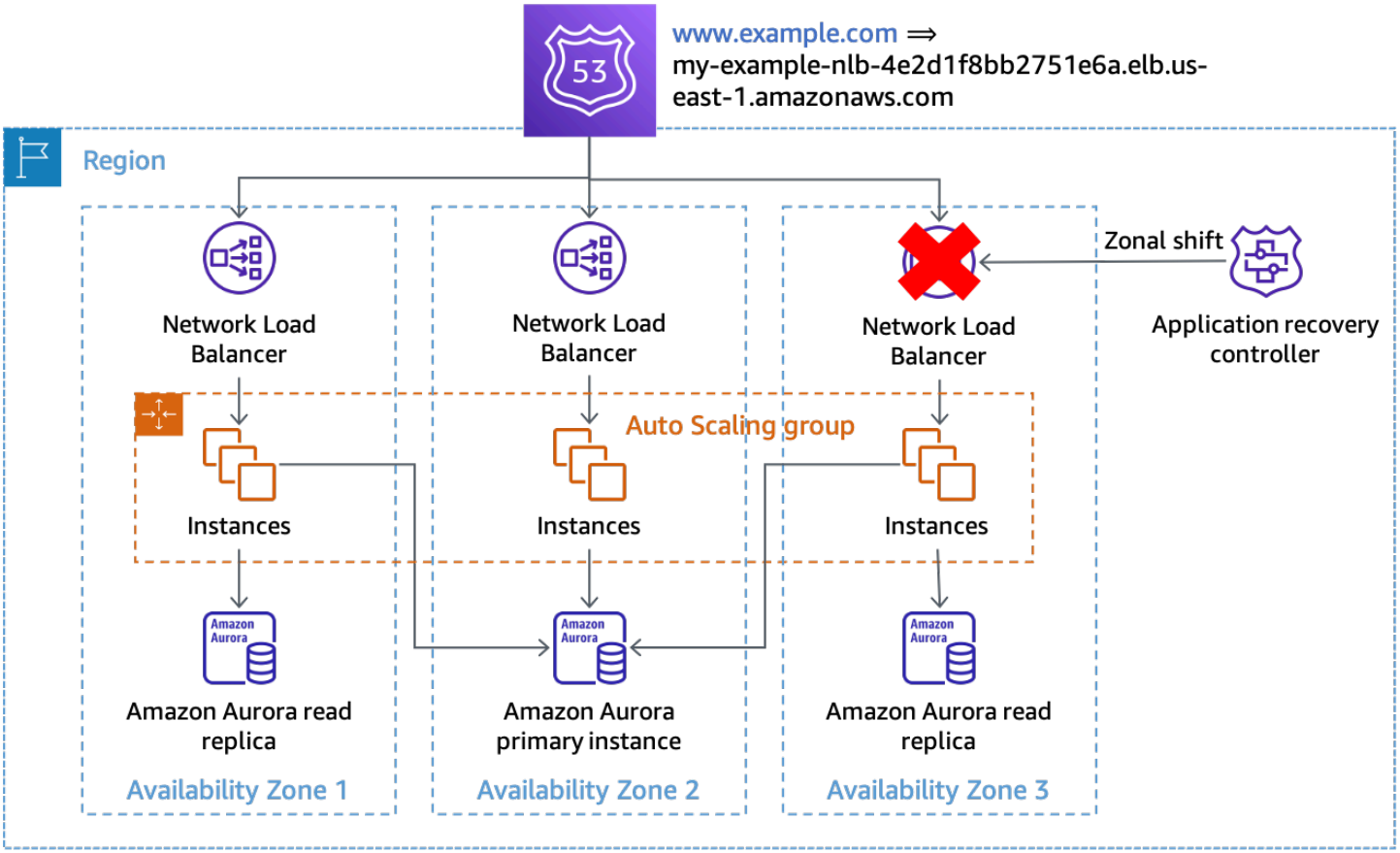
Além disso, se você estiver usando o EC2 Auto Scaling, certifique-se de que seu grupo do Auto Scaling (ASG) não reduza a escala horizontalmente durante a mudança, de forma que, quando a mudança terminar, você ainda tenha capacidade suficiente no grupo para lidar com o tráfego de clientes. Você pode fazer isso garantindo que a capacidade mínima desejada do seu ASG possa lidar com a carga atual de clientes. Você também pode ajudar a garantir que seu ASG não reduza a escala horizontalmente de maneira inadvertida usando médias nas métricas, em vez de métricas percentuais atípicas, como P90 ou P99.

Durante uma mudança, os recursos que não atendem mais ao tráfego devem ter uma utilização muito baixa, mas os outros recursos aumentarão a utilização com o novo tráfego, mantendo a média bastante consistente, o que impediria uma ação para reduzir a escala horizontalmente. Por fim, você também pode usar as configurações de integridade do grupo-alvo para ALB e NLB para especificar o failover de DNS com uma porcentagem ou uma contagem de hosts íntegros. Isso evita que o tráfego seja roteado para uma zona de disponibilidade que não tenha hosts íntegros suficientes.

Mudança de zona no Application Recovery Controller (ARC) do Route 53

A primeira solução para evacuação da zona de disponibilidade usa mudança de zona no ARC do Route 53. Essa solução pode ser usada para workloads de solicitação/resposta que usam um NLB ou ALB como ponto de entrada para o tráfego de clientes.

Ao detectar que uma zona de disponibilidade está comprometida, você pode iniciar uma mudança de zona com o ARC do Route 53. Depois que essa operação for concluída e as respostas de DNS em cache existentes expirarem, todas as novas solicitações serão encaminhadas somente para recursos nas zonas de disponibilidade restantes. A figura a seguir mostra como funciona a mudança de zona. Na figura a seguir, temos um registro de alias `www.example.com` do Route 53 que aponta para `my-example-nlb-4e2d1f8bb2751e6a.elb.us-east-1.amazonaws.com`. A mudança de zona é executada para a zona de disponibilidade 3.



Mudança de zona

No exemplo, se a instância primária do banco de dados não estiver na zona de disponibilidade 3, realizar a mudança de zona é a única ação necessária para alcançar o primeiro resultado de evacuação, impedindo que o trabalho seja processado na zona de disponibilidade afetada. Se o nó primário estivesse na zona de disponibilidade 3, você poderia realizar um failover iniciado manualmente (que depende do ambiente de gerenciamento do Amazon RDS) em coordenação com a mudança de zona, caso o Amazon RDS ainda não tenha feito o failover automático. Isso valerá para todas as soluções controladas pelo plano de dados nesta seção.

Você deve iniciar a mudança de zona usando os comandos da CLI ou a API para minimizar as dependências necessárias para iniciar a evacuação. Quanto mais simples for o processo de evacuação, mais confiável ele será. Os comandos específicos podem ser armazenados em um runbook local que os engenheiros de plantão podem acessar facilmente. A mudança de zona é a solução preferida e mais simples para evacuar uma zona de disponibilidade.

Route 53 ARC

A segunda solução usa os recursos do ARC do Route 53 para especificar manualmente a integridade de registros DNS específicos. Essa solução tem a vantagem de usar o plano de dados de cluster ARC altamente disponível do Route 53, tornando-o resiliente à deficiência de até dois Regiões da AWS diferentes. Ele tem a desvantagem de um custo adicional e requer alguma configuração adicional de registros DNS. Para implementar esse padrão, você precisa criar registros de alias para os nomes DNS específicos da zona de disponibilidade fornecidos pelo balanceador de carga (ALB ou NLB). Isto é mostrado na tabela a seguir.

Tabela 3: Registros de alias do Route 53 configurados para os nomes DNS de zona do balanceador de carga

Política de roteamento: ponderada	Política de roteamento: ponderada	Política de roteamento: ponderada
Nome: `www.example.com`	Nome: `www.example.com`	Nome: `www.example.com`
Tipo: `A` (alias)	Tipo: `A` (alias)	Tipo: `A` (alias)
Value (Valor): `us-east-1b.load-balancer-name.elb.us-east-1.amazonaws.com`	Valor: `us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com`	Valor: `us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`
Peso: `100`	Peso: `100`	Peso: `100`
Avaliar status do destino: true	Avaliar status do destino: `true`	Avaliar status do destino: `true`

Para cada um dos registros DNS, configure uma verificação de integridade do Route 53 associada a um controle de roteamento do ARC do Route 53. Quando quiser iniciar uma evacuação da

zona de disponibilidade, defina o estado do controle de roteamento como `Off`. AWS recomenda fazer isso usando a CLI ou a API para minimizar as dependências necessárias para iniciar a evacuação da zona de disponibilidade. Como prática recomendada, você deve manter uma cópia local dos endpoints do cluster ARC do Route 53 para não precisar recuperá-los do ambiente de gerenciamento do ARC quando precisar realizar uma evacuação.

Para minimizar o custo ao usar essa abordagem, você pode criar um único cluster ARC do Route 53 e verificações de integridade em um único Conta da AWS e compartilhar as verificações de integridade com outras Contas da AWS pessoas na organização. Ao adotar essa abordagem, você deve usar o ID da zona de disponibilidade (AZ-ID) (por exemplo, `use1-az1`) em vez do nome da zona de disponibilidade (por exemplo, `us-east-1a`) para os controles de roteamento. Como AWS mapeia a zona de disponibilidade física aleatoriamente com os nomes das zonas de disponibilidade de cada Conta da AWS, o uso do AZ-ID fornece uma maneira consistente de fazer referência aos mesmos locais físicos. Quando você inicia uma evacuação da zona de disponibilidade, digamos, por `use1-az2`, os conjuntos de registros do Route 53 em cada Conta da AWS devem garantir que usem o mapeamento AZ-ID para configurar a verificação de integridade correta para cada registro de NLB.

Por exemplo, digamos que temos uma verificação de integridade do Route 53 associada a um controle de roteamento ARC do Route 53 para `use1-az2`, com um ID de `0385ed2d-d65c-4f63-a19b-2412a31ef431`. Se uma Conta da AWS diferente que quisesse usar isso para fazer a verificação de integridade, `us-east-1c` estaria mapeado para `use1-az2`, você precisaria usar a verificação de integridade `use1-az2` para o registro `us-east-1c.load-balancer-name.elb.us-east-1.amazonaws.com`. Você usaria o ID da verificação de integridade `0385ed2d-d65c-4f63-a19b-2412a31ef431` com esse conjunto de registros de recursos.

Uso de um endpoint HTTP autogerenciado

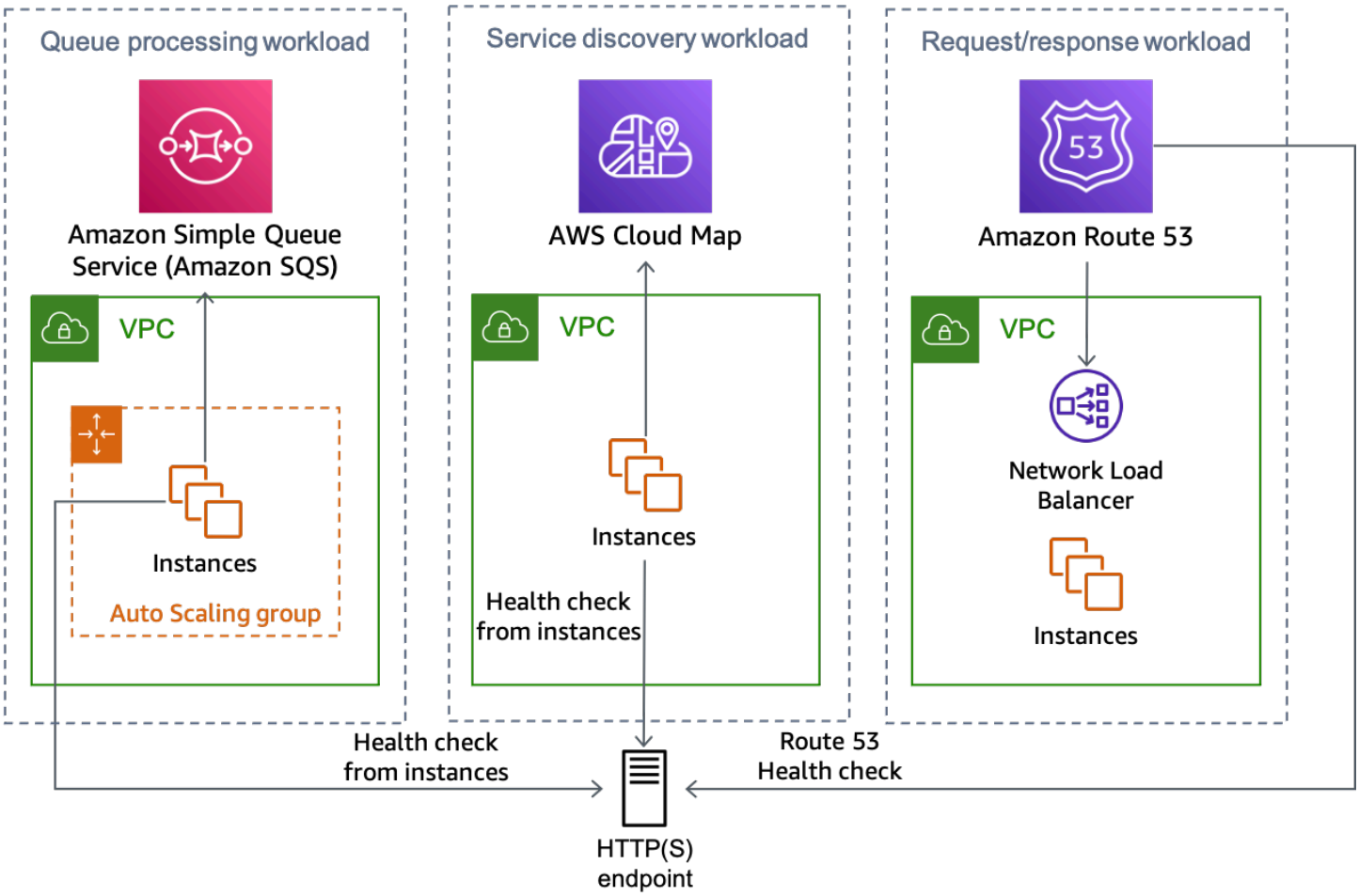
Você também pode implementar essa solução gerenciando seu próprio endpoint HTTP que indica o status de uma zona de disponibilidade específica. Isso permite que você especifique manualmente quando uma zona de disponibilidade não está íntegra com base na resposta do endpoint HTTP. Essa solução custa menos do que usar o ARC do Route 53, mas é mais cara do que a mudança de zona e requer o gerenciamento de infraestrutura adicional. Ela tem a vantagem de ser muito mais flexível para diferentes cenários.

O padrão pode ser usado com arquiteturas NLB ou ALB e com verificações de integridade do Route 53. Também pode ser usada em arquiteturas sem balanceamento de carga, como sistemas de descoberta de serviços ou processamento de filas, nos quais os nós de trabalho realizam suas próprias verificações de integridade. Nesses cenários, os hosts podem usar um thread em segundo

plano em que periodicamente fazem uma solicitação ao endpoint HTTP com seu AZ-ID (consulte Apêndice A — Obtendo o ID da zona de disponibilidade para obter detalhes sobre como encontrar isso) e receber de volta uma resposta sobre a integridade da zona de disponibilidade.

Se a zona de disponibilidade tiver sido declarada não íntegra, eles terão várias opções sobre como responder. Os hosts podem optar por falhar em uma verificação de integridade externa de fontes como ELB, Route 53 ou em verificações de integridade personalizadas em arquiteturas de descoberta de serviços de forma que pareçam não íntegros para os serviços. Também podem responder imediatamente com um erro caso recebam uma solicitação, permitindo que o cliente recue e tente novamente. Em arquiteturas orientadas por eventos, os nós podem falhar intencionalmente no processamento do trabalho, como retornar intencionalmente uma mensagem SQS para a fila. Em arquiteturas de roteadores de trabalho em que um cronograma de serviço central funciona em hosts específicos, você também pode usar esse padrão. O roteador pode verificar o status de uma zona de disponibilidade antes de selecionar um operador, endpoint ou célula. Nas arquiteturas de descoberta de serviços que usam AWS Cloud Map, você pode descobrir endpoints fornecendo um filtro na solicitação, como um AZ-ID.

A figura a seguir mostra como essa abordagem pode ser usada para vários tipos de workload.



Vários tipos de workload, em que todos podem usar a solução de endpoint HTTP

Há várias maneiras de implementar a abordagem de endpoint HTTP, duas delas serão descritas a seguir.

Uso do Amazon S3

Esse padrão foi apresentado originalmente nesta postagem do blog sobre recuperação de desastres em várias regiões. Você pode usar o mesmo padrão para a evacuação da zona de disponibilidade.

Nesse cenário, você criaria conjuntos de registros de recursos DNS do Route 53 para cada registro DNS de zona, assim como o cenário ARC do Route 53 acima, bem como as verificações de integridade associadas. No entanto, para essa implementação, em vez de associar as verificações de integridade aos controles de roteamento ARC do Route 53, elas são configuradas para usar um endpoint HTTP e são invertidas para se protegerem contra um comprometimento no Amazon S3 que acidentalmente acione uma evacuação. A verificação de integridade é considerada íntegra quando o

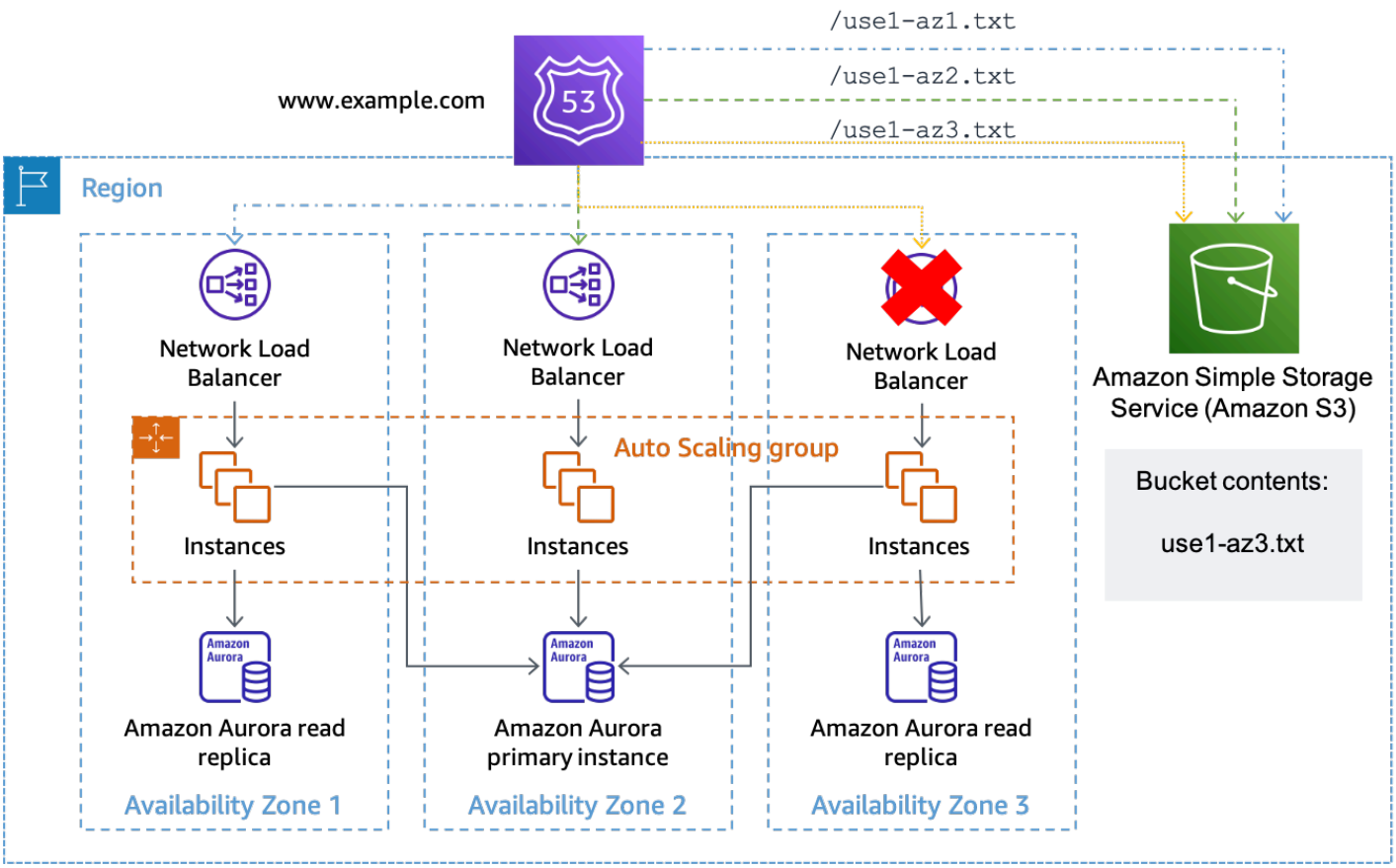
objeto está ausente e não íntegra quando o objeto está presente. Essa configuração é mostrada na tabela a seguir.

Tabela 4: Configuração do registro DNS para usar as verificações de integridade do Route 53 por zona de disponibilidade

Tipo de verificação de integridade: monitorar um endpoint Protocolo: HTTPS ID: dddd-4444 URL: https://*bucketname*.s3.us-east-1.amazonaws.com/use1-az1.txt	Tipo de verificação de integridade: monitorar um endpoint Protocolo: HTTPS ID: eeee-5555 URL: https://*bucketname*.s3.us-east-1.amazonaws.com/use1-az3.txt	Tipo de verificação de integridade: monitorar um endpoint Protocolo: HTTPS ID: ffff-6666 URL: https://*bucketname*.s3.us-east-1.amazonaws.com/use1-az2.txt	←	Verificações de integridade
↑	↑	↑		
Política de roteamento: ponderada Nome: www.example.com Tipo: A (alias) Value (Valor): us-east-1	Política de roteamento: ponderada Nome: www.example.com Tipo: A (alias) Value (Valor): us-east-1	Política de roteamento: ponderada Nome: www.example.com Tipo: A (alias) Value (Valor): us-east-1	←	Os registros de alias A de nível superior e com ponderação uniforme apontam para endpoints específicos da ZD do NLB

`b.load-balancer-name.elb.us-east-1.amazonaws.com`	`a.load-balancer-name.elb.us-east-1.amazonaws.com`	`c.load-balancer-name.elb.us-east-1.amazonaws.com`
Peso: `100`	Peso: `100`	Peso: `100`
Avaliar status do destino: `true`	Avaliar status do destino: `true`	Avaliar status do destino: `true`

Vamos supor que a zona de disponibilidade `us-east-1a` esteja mapeada na conta `use1-az3` em que temos um workload que queremos realizar uma evacuação da zona de disponibilidade. Pois o conjunto de registros de recursos criado para `us-east-1a.load-balancer-name.elb.us-east-1.amazonaws.com` associaria uma verificação de integridade que testa o URL `https://`*`bucket-name`*`.s3.us-east-1.amazonaws.com/use1-az3.txt`. Quando quiser iniciar uma evacuação da zona de disponibilidade para `use1-az3`, faça upload de um arquivo com nome de `use1-az3.txt` para o bucket usando a CLI ou a API. O arquivo não precisa conter nenhum conteúdo, mas precisa ser público para que a verificação de integridade do Route 53 possa acessá-lo. A figura a seguir demonstra que essa implementação está sendo usada para evacuar `use1-az3`.



Uso do Amazon S3 como destino para uma verificação de integridade do Route 53

Uso da API Gateway e o DynamoDB

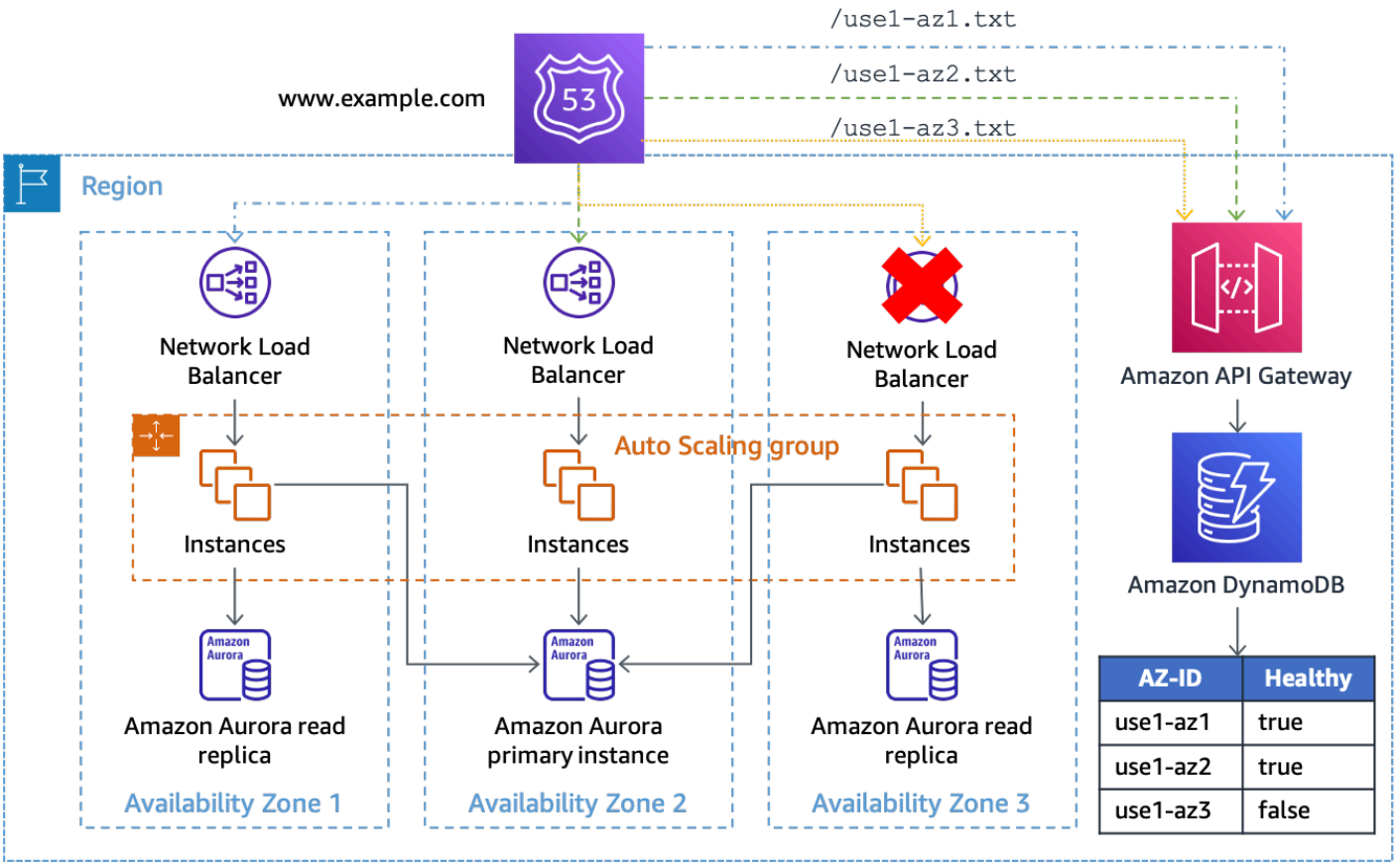
A segunda implementação desse padrão usa uma API REST do Amazon API Gateway. A API é configurada com uma integração de serviço com o Amazon DynamoDB, onde o status de cada zona de disponibilidade em uso fica armazenado. Essa implementação é mais flexível do que a abordagem do Amazon S3, mas exige a criação, a operação e o monitoramento de mais infraestrutura. Também pode ser usada com as verificações de integridade do Route 53 e com as verificações de integridade realizadas por hosts individuais.

Se você estiver usando essa solução com uma arquitetura NLB ou ALB, configure seus registros DNS da mesma forma que o exemplo do Amazon S3 acima, exceto altere o caminho da verificação de integridade para usar o endpoint da API Gateway e forneça o `AZ-ID` no caminho do URL. Por exemplo, se a API Gateway estiver configurada com um domínio personalizado de `az-status.example.com`, a solicitação completa de `use1-az1` seria semelhante a `https://az-status.example.com/status/use1-az1`. Quando quiser iniciar uma evacuação da zona de disponibilidade, você pode criar ou atualizar um item do DynamoDB usando a CLI ou a API. O item

usa o `AZ-ID` como chave primária e, em seguida, tem um atributo booleano chamado `Healthy`, que é usado para indicar como a API Gateway responde. Veja a seguir um exemplo de código usado na configuração da API Gateway para fazer essa determinação.

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

Se o atributo for `true` (ou não presente), a API Gateway responderá à verificação de integridade com um HTTP 200; se for falso, ela responderá com um HTTP 500. Essa implementação é mostrada na figura a seguir.



Uso da API Gateway e o DynamoDB como destino das verificações de integridade do Route 53

Nessa solução, você precisa usar a API Gateway na frente do DynamoDB para tornar o endpoint acessível ao público, bem como manipular a URL da solicitação em uma solicitação de `GetItem` para o DynamoDB. A solução também oferece flexibilidade se você quiser incluir dados adicionais na solicitação. Por exemplo, se quiser criar status mais granulares, como por aplicativo, poderá

configurar a URL da verificação de integridade para fornecer um ID do aplicativo no caminho ou na string de consulta que também corresponda ao item do DynamoDB.

O endpoint de status da zona de disponibilidade pode ser implantado de maneira centralizada para que vários recursos de verificação de integridade em Contas da AWS que podem usar a mesma visão consistente da integridade da zona de disponibilidade (garantindo que a API REST da API Gateway e a tabela do DynamoDB sejam dimensionadas para lidar com a carga) e elimine a necessidade de compartilhar as verificações de integridade do Route 53.

A solução também pode ser escalada em várias Regiões da AWS usando uma tabela global do Amazon DynamoDB e uma cópia da API REST da API Gateway em cada região. Isso evita que a solução dependa de uma única região e aumenta sua disponibilidade. Você pode implantar a solução em três ou cinco regiões e consultar cada uma sobre a integridade da zona de disponibilidade, usando o resultado da maioria dos endpoints para garantir o quórum. Isso permite uma replicação final consistente de atualizações em toda a tabela global, além de mitigar os comprometimentos que podem impedir um endpoint de responder. Por exemplo, se você estiver usando cinco regiões e três endpoints relatarem uma zona de disponibilidade como não íntegra, um endpoint relatar a zona de disponibilidade como íntegra e um endpoint não responder, você deve escolher tratar a zona de disponibilidade como não íntegra. Você também pode criar uma verificação de integridade calculada do Route 53 usando um cálculo m de n para executar a lógica e determinar a integridade da zona de disponibilidade.

Se você estava criando uma solução para hosts individuais usarem como um mecanismo para determinar a integridade de sua ZD, como alternativa, em vez de fornecer um mecanismo de extração para verificações de integridade, você pode usar notificações push. Uma maneira de fazer isso é com um tópico do SNS que seus consumidores assinam. Quando quiser acionar o disjuntor, publique uma mensagem no tópico do SNS indicando qual zona de disponibilidade está comprometida. Essa abordagem tem desvantagens comparada com a anterior. Ela elimina a necessidade de criar e operar a infraestrutura da API Gateway e realizar o gerenciamento de capacidade. Também pode potencialmente fornecer uma convergência mais rápida do estado da zona de disponibilidade. No entanto, ela remove a capacidade de realizar consultas ad hoc e depende da política de novas tentativas de entrega do SNS para garantir que cada endpoint receba a notificação. Também exige que cada workload ou serviço crie uma forma de receber a notificação do SNS e realizar ações sobre ela.

Por exemplo, cada lançamento de uma nova instância ou contêiner do EC2 deverá ser inscrito no tópico com um endpoint HTTP durante a inicialização. Em seguida, cada instância precisa implementar um software que escute esse endpoint em que a notificação é entregue. Além disso,

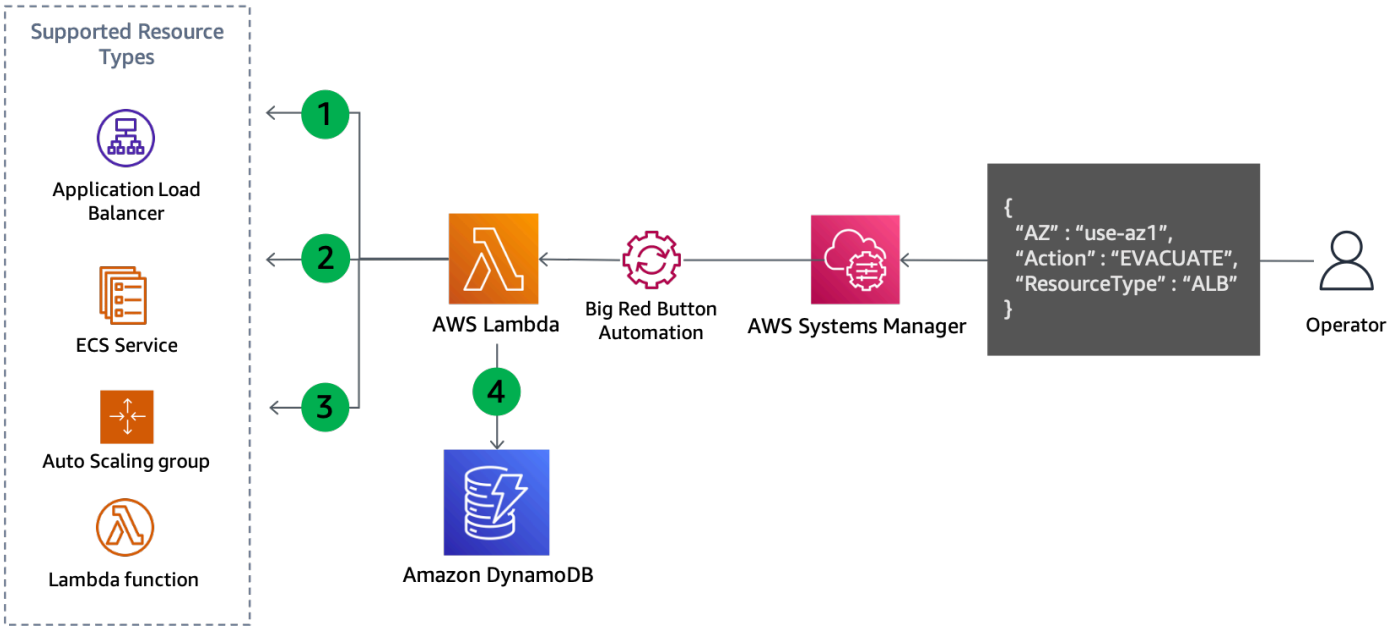
se a instância for afetada pelo evento, ela poderá não receber a notificação push e continuar trabalhando. Por outro lado, com uma notificação pull, a instância saberá se sua solicitação de pull falhou e poderá escolher qual ação tomar em resposta.

Uma segunda forma de enviar notificações push é com conexões WebSocket de longa duração. O Amazon API Gateway pode ser usado para fornecer uma API de WebSocket à qual os consumidores podem se conectar e receber uma mensagem quando enviada pelo back-end. Com um WebSocket, as instâncias podem fazer buscas periódicas para garantir que a conexão esteja íntegra e também receber notificações push de baixa latência.

Evacuação controlada por ambiente de gerenciamento

O primeiro padrão usa operações de plano de dados para evitar a execução de trabalhos em uma zona de disponibilidade afetada para mitigar o impacto de um evento. No entanto, você pode ter uma arquitetura que não usa balanceadores de carga ou em que a configuração de uma verificação de integridade por host não seja viável. Ou, talvez você queira evitar que uma nova capacidade seja implantada na zona de disponibilidade afetada por meio do Auto Scaling ou do agendamento normal de trabalho.

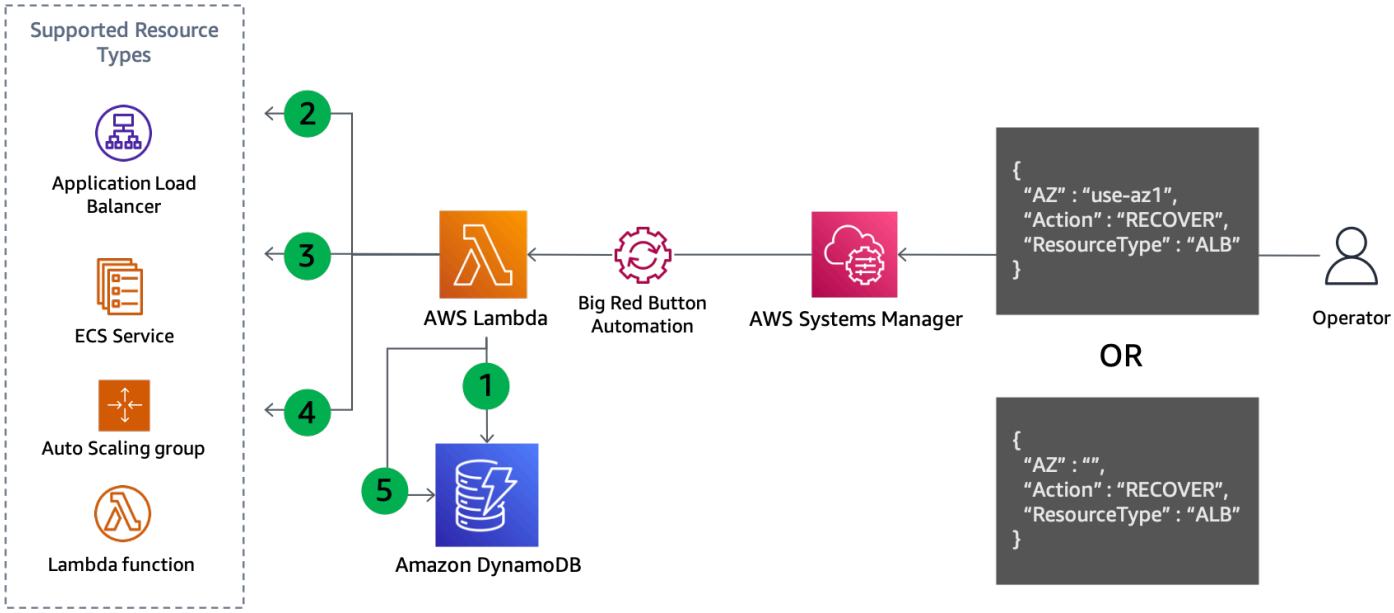
Para lidar com as duas situações, são necessárias ações do ambiente de gerenciamento para atualizar a configuração do recurso. O padrão funcionará para qualquer serviço cuja configuração de rede possa ser atualizada, por exemplo, EC2 Auto Scaling, Amazon ECS, Lambda e muito mais. É necessário escrever código para cada serviço, mas a lógica de negócios segue um padrão básico. O código deve ser executado localmente por um operador que responde ao evento para minimizar as dependências necessárias. O fluxo básico da lógica do script é mostrado na figura a seguir.



Atualização do ambiente de gerenciamento para evacuar uma zona de disponibilidade

1. O script lista todos os recursos do tipo especificado, como grupo Auto Scaling, serviço ECS ou função do Lambda e recupera as sub-redes a partir das informações do recurso. Os recursos suportados dependem do que o script foi configurado para dar suporte.
2. Ele determina quais sub-redes devem ser removidas comparando o nome da zona de disponibilidade de cada sub-rede com o ID da zona de disponibilidade mapeada, fornecida como parâmetro de entrada.
3. A configuração de rede do recurso é atualizada para remover as sub-redes identificadas.
4. Os detalhes da atualização são registrados em uma tabela do DynamoDB. O ID da zona de disponibilidade é armazenado como chave de partição e o ARN ou nome do recurso é armazenado como chave de classificação. As sub-redes que foram removidas são armazenadas como uma matriz de strings. Finalmente, o tipo de recurso também é armazenado e usado como uma chave de hash para um Índice Secundário Global (GSI).

Como a etapa quatro registra as atualizações feitas, essa abordagem também pode ser facilmente reversível quando você estiver pronto para a recuperação, conforme mostrado na figura a seguir.



Atualização do ambiente de gerenciamento para se recuperar da evacuação da zona de disponibilidade

Etapas de recuperação:

1. Consulte o GSI para remover as sub-redes de cada recurso do tipo especificado na zona de disponibilidade especificada (ou todas as zonas de disponibilidade, se não nenhuma for especificada).
2. Descreva cada recurso encontrado na consulta do DynamoDB para obter sua configuração de rede atual.
3. Combine as sub-redes da configuração de rede atual com aquelas recuperadas da consulta do DynamoDB.
4. Atualize a configuração de rede do recurso com o novo conjunto de sub-redes.
5. Remova o registro da tabela do DynamoDB após a atualização ser concluída com sucesso.

Esse padrão generalizado impede o trabalho de roteamento para a zona de disponibilidade afetada e impede que uma nova capacidade seja implantada lá. A seguir estão exemplos de como isso é feito para diferentes serviços.

- Lambda — atualize a configuração de VPC da função para remover as sub-redes na zona de disponibilidade especificada.
- Grupo de Auto Scaling — remova as sub-redes da configuração do ASG que substituirão essa capacidade nas demais zonas de disponibilidade.

- Amazon ECS — atualize a configuração VPC do serviço ECS para remover as sub-redes.
- Amazon EKS — aplique taints nos nós na zona de disponibilidade afetada para remover pods existentes e evitar que outros pods sejam agendados lá.

Cada serviço reagirá de forma diferente à atualização da configuração. Por exemplo, o Amazon ECS seguirá a configuração de implantação do serviço após uma atualização e acionará uma implantação contínua ou uma implantação azul/verde de novas tarefas.

Essas atualizações podem transferir o trabalho para as zonas de disponibilidade íntegras muito rapidamente para alguns workloads. Embora esteja configurado para ser estaticamente estável a falhas (com capacidade pré-provisionada suficiente nas zonas de disponibilidade restantes para lidar com o trabalho da zona de disponibilidade afetada), você também pode querer eliminar gradualmente a capacidade da zona de disponibilidade afetada.

Se você planeja atualizar a configuração de rede do grupo de Auto Scaling que é um grupo-alvo para um balanceador de carga com balanceamento de carga entre zonas desativado, siga esta orientação.

O Auto Scaling reage a essa mudança usando sua lógica de rebalanceamento da zona de disponibilidade. Ele iniciará instâncias nas outras zonas de disponibilidade para atender à capacidade desejada e encerrará instâncias na zona de disponibilidade que você removeu. No entanto, o balanceador de carga continuará dividindo o tráfego uniformemente em cada zona de disponibilidade, incluindo a que você removeu do ASG, enquanto as instâncias estão sendo encerradas. Isso pode reduzir a capacidade restante nessa zona de disponibilidade até que todas as instâncias sejam encerradas com sucesso. Esse é o mesmo problema descrito em Independência da zona de disponibilidade em relação ao desequilíbrio da zona de disponibilidade quando o balanceamento de carga entre zonas está desativado. Para evitar que isso ocorra, você pode:

- Sempre realizar primeiro a evacuação da zona de disponibilidade para que o tráfego seja dividido apenas entre as demais zonas de disponibilidade
- Especificar uma contagem mínima de alvos íntegros com failover de DNS para corresponder à contagem mínima necessária para essa zona de disponibilidade.

Isso ajudará a garantir que o tráfego não seja enviado para a zona de disponibilidade que você removeu depois que as instâncias começarem a ser encerradas.

Resumo

A tabela a seguir resume os prós e os contras dos padrões de evacuação descritos.

Tabela 5: Prós e contras do padrão de evacuação

Abordagem	Prós	Contras
Evacuação controlada por plano de dados	Depende apenas das ações do plano de dados Impede rapidamente que o trabalho seja feito na zona de disponibilidade afetada Abordagem flexível para uma visão centralizada da integrida de da zona de disponibilidade	Não impede que capacidad es sejam implantadas em uma zona de disponibilidade afetada Nem todos os tipos de workload podem usar essa abordagem com facilidade
Evacuação controlada por ambiente de gerenciamento	Impede que novas capacidad es sejam implantadas na zona de disponibilidade afetada Remove a capacidade existente da zona de disponibi lidade afetada	Depende do ambiente de gerenciamento de cada serviço Requer que o código seja escrito para cada serviço Tem que ser executado serviço por serviço É preciso ter cuidado para não sobrecarregar a capacidade durante a atualização

Você provavelmente usará as duas abordagens em conjunto como parte de um plano de evacuação da zona de disponibilidade. Comece com as ações de evacuação controladas pelo plano de dados que têm maior probabilidade de sucesso para interromper rapidamente o trabalho de processamento na zona de disponibilidade afetada. A seguir, assim que o impacto inicial seja mitigado, siga com as ações de evacuação controladas pelo ambiente de gerenciamento, caso julgue necessário.

Conclusão

Este artigo forneceu uma visão geral das falhas cinzentas, como elas se manifestam e descreveu por que você precisa criar ferramentas de observabilidade e evacuação para mitigar esses tipos de eventos quando eles ocorrem. Na seção seguinte, você analisou a observabilidade de multi-ZD e três abordagens que podem ser implementadas para detectar o impacto em uma única Zona de Disponibilidade. Na última seção, este artigo apresentou duas abordagens gerais para realizar a evacuação da Zona de Disponibilidade. A primeira abordagem usa ações do plano de dados para evitar que o trabalho seja roteado para a Zona de Disponibilidade afetada, enquanto a segunda abordagem usa ações do ambiente de gerenciamento para evitar que a capacidade seja provisionada na Zona de Disponibilidade afetada. Juntas, essas abordagens alcançam os dois resultados pretendidos pela evacuação da Zona de Disponibilidade.

Os padrões de recuperação descritos neste artigo provavelmente farão parte de uma solução maior de monitoramento e recuperação de falhas. Essa abordagem para lidar com falhas cinzentas em uma única Zona de Disponibilidade requer trabalho de engenharia para construir a instrumentação necessária para detectá-las, bem como as ferramentas para responder a elas. No entanto, para muitas cargas de trabalho, essa abordagem pode ser uma alternativa mais simples e econômica à criação de arquiteturas multirregionais. Além disso, ela pode ajudar a obter RPOs e RTOs menores (o que aumenta a disponibilidade da carga de trabalho) em comparação com a DR multirregional.

Apêndice A — Obtendo o ID da zona de disponibilidade

Se estiver usando o AWS SDK do .NET (assim como outros, como JavaScript) ou executando o sistema em uma instância do EC2 (incluindo o Amazon ECS e Amazon EKS), você pode obter o ID da zona de disponibilidade diretamente.

- AWS SDK do .NET

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

- Serviço de metadados da instância EC2

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

Em outras plataformas, como Lambda e Fargate, você precisará recuperar o nome da zona de disponibilidade e, em seguida, encontrar o mapeamento para o ID da zona de disponibilidade. Com o nome da zona de disponibilidade, você pode encontrar o ID da zona de disponibilidade da seguinte forma:

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

Os exemplos a seguir para encontrar o nome da zona de disponibilidade a serem usados no exemplo acima foram criados em bash usando o AWS CLI e o pacote jq. Eles deverão ser convertidos para a linguagem de programação usada para seu workload.

- Amazon ECS – Se o serviço de metadados de instância (IMDS) estiver bloqueado pelo host, você poderá usar o arquivo de metadados do contêiner.

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

- Fargate (versão da plataforma 1.4 ou posterior)

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

- Lambda – A zona de disponibilidade não é exposta diretamente à função. Para encontrá-la, você precisa executar várias etapas. Para fazer isso, crie um endpoint REST privado do API Gateway que retorne o endereço IP do solicitante. Isso identificará o IP privado atribuído à interface de rede elástica que está sendo usada pela função.
  - Chame a API `GetFunction` do Lambda para encontrar o ID da VPC da função.
  - Chame o serviço API Gateway para obter o IP da função.
  - Usando o IP e o ID da VPC, encontre a interface de rede associada e extraia a zona de disponibilidade.

```
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’)
```

Apêndice B – Exemplo de cálculo do qui-quadrado

Veja a seguir um exemplo de coleta de métricas de erro e realização de um teste qui-quadrado nos dados. O código não está pronto para produção e não executa o tratamento de erros necessário, mas fornece uma prova de conceito sobre como a lógica funciona. Você deve atualizar esse exemplo para atender às suas necessidades.

Primeiro, uma função do Lambda é do invocada a cada minuto por um evento programado do Amazon EventBridge. O conteúdo do evento é configurado com os seguintes dados:

```
{
  "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" ]
}
```

Os dados são usados para especificar os dados comuns necessários para recuperar as métricas apropriadas do CloudWatch (como namespace, nome da métrica e dimensões) e, em seguida, publicar os resultados do qui-quadrado para cada zona de disponibilidade. O código na função do Lambda é semelhante ao código a seguir, usando o Python 3.9. Em um alto nível, ele coleta as métricas especificadas do CloudWatch para o minuto anterior, executa o teste qui-quadrado nesses dados e, em seguida, publica as métricas do CloudWatch sobre o resultado do teste para cada zona de disponibilidade especificada.

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

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

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

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

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

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

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

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

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

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

    for az in detail["azs"]:

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

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

        metric_query["MetricDataQueries"].append(query)

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

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

        data = cw_client.get_metric_data(**metric_query)

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

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

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

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

        if next_token is None:
            break

    return results

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

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

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

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

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

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

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

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

    cw_client.put_metric_data(**metric_query)

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

Em seguida, você já pode criar um alarme por AZ. O exemplo a seguir é para `use1-az2` e gera o alarme para três pontos de dados de um minuto em uma linha que tem um valor máximo igual a 1 (1 é a métrica publicada quando o teste qui-quadrado determina uma distorção estatisticamente significativa na taxa de erro).

```
{
    "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"
    }
}
```

Você também pode criar um alarme m-of-n e combinar esses dois alarmes em um alarme composto. Você também precisa criar os mesmos alarmes para cada combinação de controlador/ação ou microsserviço que tem em cada zona de disponibilidade. Finalmente, você pode adicionar o alarme composto de qui-quadrado ao alarme específico da zona de disponibilidade para cada combinação de controlador/ação, conforme mostrado em Detecção de falhas usando detecção de discrepâncias.

Colaboradores

Os colaboradores deste documento incluem:

- Michael Haken, Principal Solutions Architect na Amazon Web Services

Revisões do documento

Para ser notificado sobre atualizações nesse whitepaper, inscreva-se no feed RSS.

Alteração	Descrição	Data
Whitepaper atualizado	Atualizado com orientação adicional de observabilidade e para usar o novo atributo de mudança de zona.	11 de julho de 2023
Publicação inicial	Whitepaper publicado pela primeira vez.	2 de março de 2022

> **Note**
>
> Para se inscrever nas atualizações de RSS, você precisa ter um plug-in de RSS habilitado no navegador.

Avisos

Os clientes são responsáveis por fazer a própria avaliação independente das informações contidas neste documento. Este documento: (a) é apenas para fins informativos, (b) representa as ofertas de produto e práticas da AWS, que estão sujeitas a alterações sem aviso prévio e (c) não cria nenhum compromisso ou garantia da AWS e de suas afiliadas, fornecedores ou licenciadores. Os produtos ou serviços da AWS são fornecidos “no estado em que se encontram”, sem garantias, representações ou condições de qualquer tipo, expressas ou implícitas. As responsabilidades e as obrigações da AWS com os seus clientes são controladas por contratos da AWS, e este documento não é parte de, nem modifica, qualquer contrato entre a AWS e seus clientes.

AWS Glossário

Para obter a terminologia mais recente da AWS, consulte o glossário da AWS na Referência do Glossário da AWS.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.