AWS Well-Architected Framework

Pilar Confiabilidade

Pilar Confiabilidade: AWS Well-Architected Framework

Table of Contents

Pilar Confiabilidade: AWS Well-Architected Framework

Data de publicação: 27 de junho de 2024 (Revisões do documento)

O foco deste documento é o pilar Confiabilidade do AWS Well-Architected Framework. Ele fornece orientações para ajudar você a aplicar as práticas recomentadas no design, entrega e manutenção dos ambientes da Amazon Web Services (AWS).

Introdução

O AWS Well-Architected Framework ajuda a compreender os prós e os contras das decisões tomadas ao criar workloads na AWS. O uso do Framework ajuda você a aprender as práticas de arquitetura recomendadas para projetar e operar workloads confiáveis, seguras, eficientes, econômicas e sustentáveis na nuvem. Ele fornece uma maneira de avaliar consistentemente suas arquiteturas em relação às práticas recomendadas e identificar áreas de melhoria. Acreditamos que ter workloads bem arquitetadas aumenta muito a probabilidade de sucesso nos negócios.

O AWS Well-Architected Framework é baseado em seis pilares:

- Excelência operacional
- Segurança
- Confiabilidade
- Eficiência de performance
- Otimização de custos
- Sustentabilidade

Este documento enfoca o pilar de confiabilidade e como aplicá-lo às suas soluções. Alcançar essa confiabilidade pode ser desafiador em ambientes on-premises tradicionais devido a pontos únicos de falha, falta de automação e falta de elasticidade. Ao adotar as práticas neste documento, você criará arquiteturas com fortes bases, arquitetura resiliente, gerenciamento de alterações consistente e processos comprovados de recuperação de falhas.

Este documento é destinado a pessoas que ocupam cargos de tecnologia, como diretores de tecnologia (CTOs), arquitetos, desenvolvedores e membros da equipe de operações. Depois de ler este documento, você compreenderá as práticas recomendadas e as estratégias da AWS a serem

usadas ao projetar arquiteturas de nuvem para confiabilidade. Este documento inclui detalhes de implementação de alto nível e padrões de arquitetura, bem como referências a recursos adicionais.

Confiabilidade

O pilar Confiabilidade abrange a capacidade de uma carga de trabalho de executar a função pretendida correta e consistentemente quando esperado. Isso inclui a capacidade de operar e testar a carga de trabalho durante todo o ciclo de vida dela. Este documento fornece orientações detalhadas sobre as práticas recomendadas para a implementação de workloads confiáveis na AWS.

Tópicos

Modelo de responsabilidade compartilhada para resiliência

A resiliência é uma responsabilidade compartilhada entre você e a AWS. É importante que você entenda como a recuperação de desastres (DR) e a disponibilidade, como parte da resiliência, operam nesse modelo compartilhado.

Responsabilidade da AWS: resiliência da nuvem

A AWS é responsável pela resiliência da infraestrutura que executa todos os serviços oferecidos na Nuvem AWS. Essa infraestrutura engloba hardware, software, redes e instalações que executam os serviços da Nuvem AWS. A AWS usa esforços comercialmente razoáveis para tornar esses serviços da Nuvem AWS disponíveis, garantindo que a disponibilidade dos serviços atenda ou exceda os Acordos de Nível de Serviço (SLAs) da AWS.

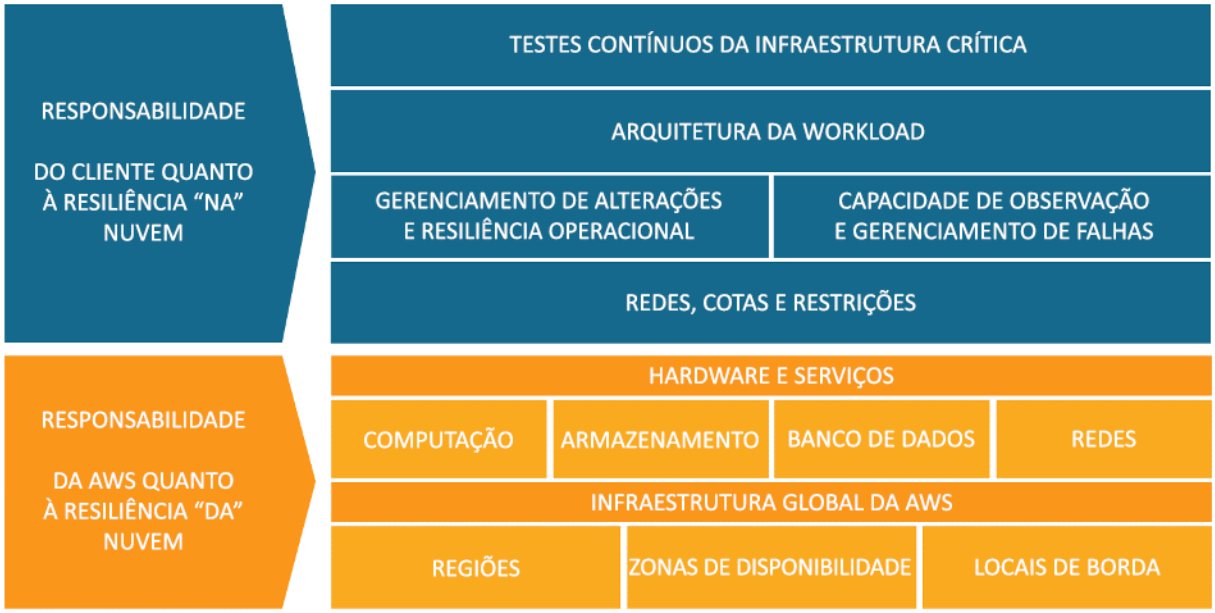
A Infraestrutura de nuvem global da AWS foi projetada para permitir que os clientes criem arquiteturas de workload resilientes. Cada Região da AWS é totalmente isolada e consiste em várias zonas de disponibilidade, que são partições fisicamente isoladas da infraestrutura. As zonas de disponibilidade isolam falhas que poderiam afetar a resiliência da workload, impedindo que elas afetem outras zonas na região. No entanto, ao mesmo tempo, todas as zonas em uma Região da AWS são interconectadas com rede de alta largura de banda e baixa latência, por fibra metropolitana dedicada totalmente redundante, fornecendo rede de alto throughput e baixa latência entre as zonas. Todo o tráfego entre as zonas é criptografado. O desempenho da rede é suficiente para realizar a replicação síncrona entre as zonas. Quando uma aplicação é particionada entre AZs, as empresas

ficam mais bem isoladas e protegidas contra problemas como queda de energia, raios, tornados, furacões, dentre outros.

Responsabilidade do cliente: resiliência na nuvem

Sua responsabilidade é determinada pelos serviços da Nuvem AWS que você escolher. Isso determina o volume do trabalho de configuração que você deve executar como parte de suas responsabilidades de resiliência. Por exemplo, um serviço como o Amazon Elastic Compute Cloud (Amazon EC2) exige que o cliente realize todas as tarefas necessárias de configuração e gerenciamento. Os clientes que implantam instâncias do Amazon EC2 são responsáveis por implantar instâncias do Amazon EC2 em vários locais (como zonas de disponibilidade da AWS), implementar a autorreparação usando serviços como o Auto Scaling, e usando as práticas recomendadas de arquitetura de workload resiliente para aplicações instaladas nas instâncias. Para serviços gerenciados, como o Amazon S3 e o Amazon DynamoDB, a AWS opera a camada de infraestrutura, o sistema operacional e as plataformas, e os clientes acessam os endpoints para armazenar e recuperar dados. Você é responsável por gerenciar a resiliência dos dados incluindo estratégias de backup, versionamento e replicação.

Implantar a workload em várias zonas de disponibilidade em uma Região da AWS faz parte de uma estratégia de disponibilidade desenvolvida para proteger workloads isolando problemas a uma zona de disponibilidade, que usa a redundância de outras zonas de disponibilidade para continuar atendendo às solicitações. Uma arquitetura Multi-AZ também faz parte de uma estratégia de DR desenvolvida para que as workloads sejam mais isoladas e protegidas de problemas como queda de energia, raios, tornados, terremotos, dentre outros. As estratégias de DR também podem usar várias Regiões da AWS. Por exemplo, em uma configuração ativa/passiva, o serviço para a workload faz failover de sua região ativa para sua região de DR caso a região ativa não possa mais atender às solicitações.



Responsabilidade pela resiliência na nuvem e da nuvem para clientes e a AWS.

É possível usar os serviços da AWS para cumprir seus objetivos de resiliência. Como cliente, você é responsável pelo gerenciamento dos seguintes aspectos de seu sistema para obter resiliência na nuvem. Para obter mais detalhes sobre cada serviço específico, consulte a documentação da AWS.

Redes, cotas e restrições

- As práticas recomendadas para essa área do modelo de responsabilidade compartilhada estão descritas em detalhes em Fundamentos.
- Planeje a arquitetura com espaço adequado para escalar e entender as cotas de serviço e as restrições dos serviços incluídos, com base nos aumentos de solicitação de carga esperados quando aplicável.
- Projete a topologia da rede para ser altamente disponível, redundante e escalável.

Gerenciamento de alterações e resiliência operacional

- O gerenciamento de alterações abrange como incluir e gerenciar a alteração em seu ambiente. A implementação da alteração exige criar e manter runbooks atualizados e implementar estratégias para a aplicação e a infraestrutura.

- Uma estratégia resiliente para monitorar os recursos da workload considera todos os componentes, incluindo tanto métricas técnicas como empresariais, notificações, automação e análise.
- As workloads na nuvem devem se adaptar às mudanças na demanda quando ela é reduzida como resultado de condições adversas ou flutuações no uso.

Capacidade de observação e gerenciamento de falhas

- É necessário observar falhas por meio do monitoramento para automatizar a correção, a fim de que as workloads possam resistir às falhas de componentes.
- O gerenciamento de falhas exige backup de dados, aplicação das práticas recomendadas para permitir que a workload resista às falhas de componentes e planejamento da recuperação de desastres.

Arquitetura da carga de trabalho

- A arquitetura da workload inclui como você projeta os serviços em torno dos domínios empresariais, aplica SOA e design de sistema distribuído para evitar falhas e integra recursos como controle de utilização, novas tentativas, gerenciamento de fila, tempo limite e medidas emergenciais.
- Conte com as soluções comprovadas da AWS, a Amazon Builder's Library e os padrões de tecnologia sem servidor para se alinhar às práticas recomendadas e impulsionar as implementações.
- Use a melhoria contínua para decompor o sistema em serviços distribuídos para escalar e inovar mais rapidamente. Use orientações de microsserviços da AWS e opções de serviços gerenciados para simplificar e acelerar sua capacidade de introduzir mudanças e inovar.

Testes contínuos da infraestrutura crítica

- Testar a confiabilidade significa testar nos níveis funcional, de desempenho e caos, além de adotar práticas de análise de incidentes e dia de jogos para desenvolver experiência em resolver problemas que não são bem entendidos.
- Para aplicações híbridas e completas na nuvem, saber como sua aplicação se comporta caso ocorra um problema ou os componentes fiquem inativos permite que você se recupere de interrupções de forma rápida e confiável.

- Crie e documente experimentos repetíveis para entender como seu sistema se comporta quando as coisas não ocorrem como esperado. Esses testes provarão a eficácia de sua resiliência geral e fornecerão um loop de feedback de seus procedimentos operacionais antes de enfrentar cenários de falha reais.

Princípios de design

Na nuvem, há uma série de princípios que podem ajudar a aumentar a confiabilidade. Lembre-se disso ao discutirmos as melhores práticas:

- Recuperar-se automaticamente de falhas: ao monitorar os principais indicadores de desempenho (KPIs) de uma workload, você poderá acionar a automação quando um limite for violado. Esses KPIs devem ser uma medida do valor empresarial, não dos aspectos técnicos da operação do serviço. Isso permite a notificação automática e o rastreamento de falhas, além de processos de recuperação automatizados que solucionam ou reparam a falha. Com uma automação mais sofisticada, é possível antecipar e corrigir falhas antes que elas ocorram.
- Testar os procedimentos de recuperação: em um ambiente on-premises, muitas vezes os testes são realizados para provar que a workload funciona em um cenário específico. Normalmente, o teste não é usado para validar estratégias de recuperação. Na nuvem, você pode testar o comportamento de falha da carga de trabalho e validar os procedimentos de recuperação. É possível usar a automação para simular falhas diferentes ou para recriar cenários que levaram a falhas no passado. Essa abordagem expõe caminhos de falha que você pode testar e corrigir antes que ocorra um cenário de falha real, o que reduz os riscos.
- Escalar horizontalmente para aumentar a disponibilidade agregada da workload: substitua um recurso grande por vários recursos pequenos para reduzir o impacto de uma única falha na workload geral. Distribua as solicitações por vários recursos menores para garantir que elas não compartilhem um ponto de falha comum.
- Parar de adivinhar a capacidade: uma causa comum de falha nas workloads on-premises é a saturação de recursos, quando as demandas impostas a uma workload excedem a capacidade dela (esse geralmente é o objetivo dos ataques de negação de serviço). Na nuvem, você pode monitorar a demanda e a utilização da carga de trabalho e automatizar a adição ou a remoção de recursos para manter o nível ideal e atender à demanda, sem aprovisionamento em excesso ou subprovisionamento. Ainda há limites, mas alguns podem ser controlados e outros podem ser gerenciados (consulte Gerencie cotas e restrições de serviço).

- Gerenciar alterações por meio da automação: as alterações na infraestrutura devem ser feitas usando a automação. Dentre aquelas que precisam ser gerenciadas estão as alterações na automação, que podem ser acompanhadas e analisadas.

Definições

Este whitepaper aborda a confiabilidade na nuvem, descrevendo as melhores práticas para estas quatro áreas:

- Fundamentos
- Arquitetura da carga de trabalho
- Gerenciamento de mudanças
- Gerenciamento de falhas

Para obter confiabilidade, você deve começar com os fundamentos: um ambiente em que cotas de serviço e topologia de rede acomodam a carga de trabalho. A arquitetura de cargas de trabalho do sistema distribuído deve ser projetada para evitar e mitigar falhas. A carga de trabalho deve processar as alterações na demanda ou nos requisitos e ser projetada para detectar falhas e se reparar automaticamente.

Tópicos

Resiliência e os componentes da confiabilidade

A confiabilidade de uma workload na nuvem depende de vários fatores, sendo o principal a resiliência:

- Resiliência é a capacidade de uma workload se recuperar de interrupções de infraestrutura ou serviço, adquirir dinamicamente recursos de computação para atender à demanda e mitigar interrupções, como configurações incorretas ou problemas transitórios de rede.

Os outros fatores que afetam a confiabilidade da carga de trabalho são:

- Excelência operacional, que inclui automação de alterações, uso de playbooks para responder a falhas e Análises da prontidão operacional (ORRs) para confirmar que os aplicativos estão prontos para operações de produção.
- Segurança, que inclui a prevenção de danos a dados ou infraestrutura de agentes mal-intencionados, o que pode afetar a disponibilidade. Por exemplo, criptografe backups para garantir que os dados estejam seguros.
- Eficiência de performance, que inclui o projeto para taxas máximas de solicitação e a minimização de latências para sua carga de trabalho.
- Otimização de custos, que inclui compensações, como se você deseja gastar mais em instâncias do EC2 para alcançar estabilidade estática ou confiar em escalabilidade automática quando mais capacidade for necessária.

A resiliência é o foco principal deste whitepaper.

Os outros quatro aspectos também são importantes e são cobertos por seus respectivos pilares do AWS Well-Architected Framework. Muitas das práticas recomendadas aqui também resolvem esses aspectos da confiabilidade, mas o foco está na resiliência.

Disponibilidade

Disponibilidade (também conhecida como disponibilidade de serviço) é uma métrica que costuma ser usada para medir quantitativamente a resiliência, bem como um objetivo de resiliência de destino.

- Disponibilidade é a porcentagem de tempo em que uma workload está disponível para uso.

Disponível para uso significa que ela executa a função combinada quando necessário.

Esse percentual é calculado em períodos como um mês, um ano ou os últimos três anos. Aplicando a interpretação mais estrita possível, a disponibilidade diminui sempre que o aplicativo não está operando normalmente, incluindo interrupções programadas e não programadas. Definimos a disponibilidade da seguinte maneira:

$$Disponibilidade = \frac{Disponível\ para\ tempo\ de\ uso}{Tempo\ total}$$

- A disponibilidade é um percentual do tempo de atividade (como 99,9%) em um período (geralmente, de um mês ou de um ano).
- Uma forma comum de abreviar essas informações é mencionar apenas o “número de noves”. Por exemplo, “cinco noves” significa 99,999% disponível.
- Alguns clientes preferem excluir o tempo de inatividade do serviço programado (por exemplo, manutenção planejada) do tempo total na fórmula. No entanto, isso não é aconselhado, uma vez que os usuários podem preferir usar o serviço durante essas horas.

Veja a seguir uma tabela das metas de design de disponibilidade de aplicativo comuns e a máxima duração das interrupções que podem ocorrer em um ano sem que o objetivo deixe de ser atingido. A tabela contém exemplos dos tipos de aplicativos comuns em cada nível de disponibilidade. Neste documento, vamos nos referir a esses valores.

Disponibilidade	Indisponibilidade máxima (por ano)	Categorias de aplicativo
99%	3 dias e 15 horas	Tarefas de processamento em lote, extração de dados, transferência e carregamento
99,9%	8 horas e 45 minutos	Ferramentas internas como gerenciamento de conhecimento e acompanhamento de projeto
99,95%	4 horas e 22 minutos	Comércio online, ponto de vendas
99,99%	52 minutos	Entrega de vídeo, workloads de difusão
99,999%	5 minutos	Transações em caixas eletrônicos, workloads de telecomunicações

Medição da disponibilidade com base em solicitações. No seu serviço, pode ser mais fácil contar as solicitações bem-sucedidas e com falha, em vez do “tempo disponível para uso”. Nesse caso, o seguinte cálculo pode ser usado:

$$Disponibilidade = \frac{Respostas\ bem\text{-}sucedidas}{Solicitações\ válidas}$$

Isso é medido com frequência para períodos de um minuto ou de cinco minutos. Então, um percentual de tempo ativo mensal (medição da disponibilidade baseada em tempo) pode ser calculado da média desses períodos. Se nenhuma solicitação for recebida em determinado período, ela será contada como 100% disponível nesse período.

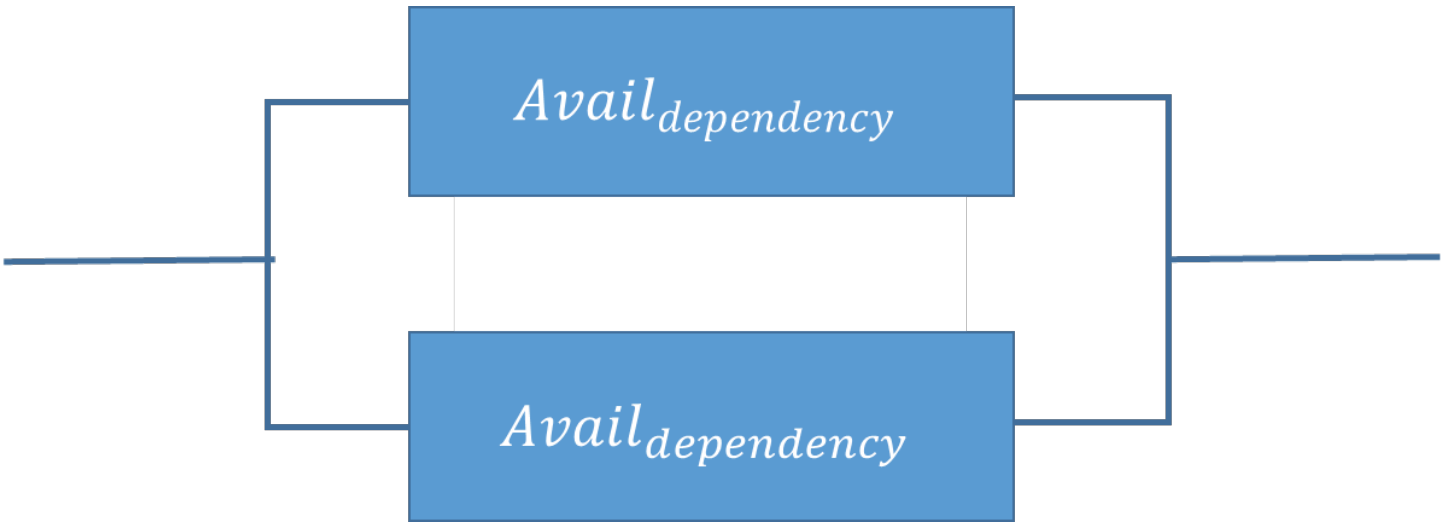
Cálculo de disponibilidade com dependências rígidas. Muitos sistemas têm dependências rígidas de outros sistemas, em que uma interrupção de um sistema dependente leva diretamente a uma interrupção do sistema que o invocou. Isso é o oposto de uma dependência flexível, em que uma falha do sistema dependente é compensada no aplicativo. Quando ocorrem essas dependências rígidas, a disponibilidade do sistema invocador é o produto das disponibilidades dos sistemas dependentes. Por exemplo, se você tiver um sistema projetado para uma disponibilidade de 99,99% que tenha uma dependência rígida de outros dois sistemas independentes, cada um projetado para uma disponibilidade de 99,99%, a carga de trabalho teoricamente poderá atingir uma disponibilidade de 99,97%:

$Disponibilidade_{invok} \times Disponibilidade_{dep1} \times Disponibilidade_{dep2} = Disponibilidade_{carga\ de\ trabalho}$

99,99% × 99,99% × 99,99% = 99,97%

Portanto, é importante entender suas dependências e as respectivas metas de design de disponibilidade ao calcular as suas próprias.

Cálculo de disponibilidade com componentes redundantes. Quando um sistema envolve o uso de componentes independentes e redundantes (por exemplo, recursos redundantes em diferentes zonas de disponibilidade), a disponibilidade teórica é calculada como 100% menos o produto das taxas de falha dos componentes. Por exemplo, se um sistema usar dois componentes independentes, cada um com uma disponibilidade de 99,9%, a disponibilidade efetiva dessa dependência será 99,9999%:



$Disponibilidade_{econômico} = Disponibilidade_{MAX} - ((100\% - Disponibilidade_{dependência}) \times (100\% - Disponibilidade_{dependência}))$

99,9999% = 100% − (0,1%×0,1%)

Cálculo de atalho: se as disponibilidades de todos os componentes em seu cálculo consistirem unicamente em nove dígitos, você poderá somar a contagem do número de nove dígitos para obter a resposta. No exemplo acima, dois componentes independentes e redundantes com disponibilidade de três noves resultam em seis noves.

Cálculo da disponibilidade da dependência. Algumas dependências fornecem orientações sobre a disponibilidade delas, incluindo metas de design da disponibilidade para muitos serviços da AWS. Porém, em casos em que isso não está disponível (por exemplo, um componente em que o fabricante não publica informações de disponibilidade), uma maneira de estimar é determinar o Tempo médio entre a falha (MTBF) e o Tempo médio para recuperação (MTTR). É possível estabelecer a estimativa de disponibilidade da seguinte maneira:

$$Disp._{EST.} = \frac{MTBF}{MTBF + MTTR}$$

Por exemplo, se o MTBF for de 150 dias e o MTTR for de 1 hora, a estimativa de disponibilidade será de 99,97%.

Para obter mais detalhes, consulte Availability and Beyond: Understanding and improving the resilience of distributed systems on AWS (Disponibilidade e além: entender e melhorar a resiliência de sistemas distribuídos na AWS), o que pode ajudar a calcular a disponibilidade.

Custos da disponibilidade. Desenvolver aplicativos para níveis mais altos de disponibilidade costuma resulta em mais custos. Assim, é importante identificar as verdadeiras necessidades de disponibilidade antes de iniciar o design do aplicativo. Altos níveis de disponibilidade impõem exigências maiores para teste e validação sob cenários de falha exaustivos. Eles exigem automação para recuperação de todos os tipos de falhas, e exigem que todos os aspectos das operações do sistema sejam criados e testados de modo similar conforme os mesmos padrões. Por exemplo, as ações de adicionar ou remover capacidade, implantar ou reverter software atualizado ou alterações de configuração ou migrar dados do sistema devem ser feitas conforme a meta de disponibilidade desejada. Aumentando os custos de desenvolvimento de software, em níveis muito altos de disponibilidade, a inovação é prejudicada devido à necessidade de avançar mais lentamente na implantação dos sistemas. Assim, a orientação é ter cautela ao aplicar os padrões e considerar o objetivo de disponibilidade adequado para todo o ciclo de vida de operação do sistema.

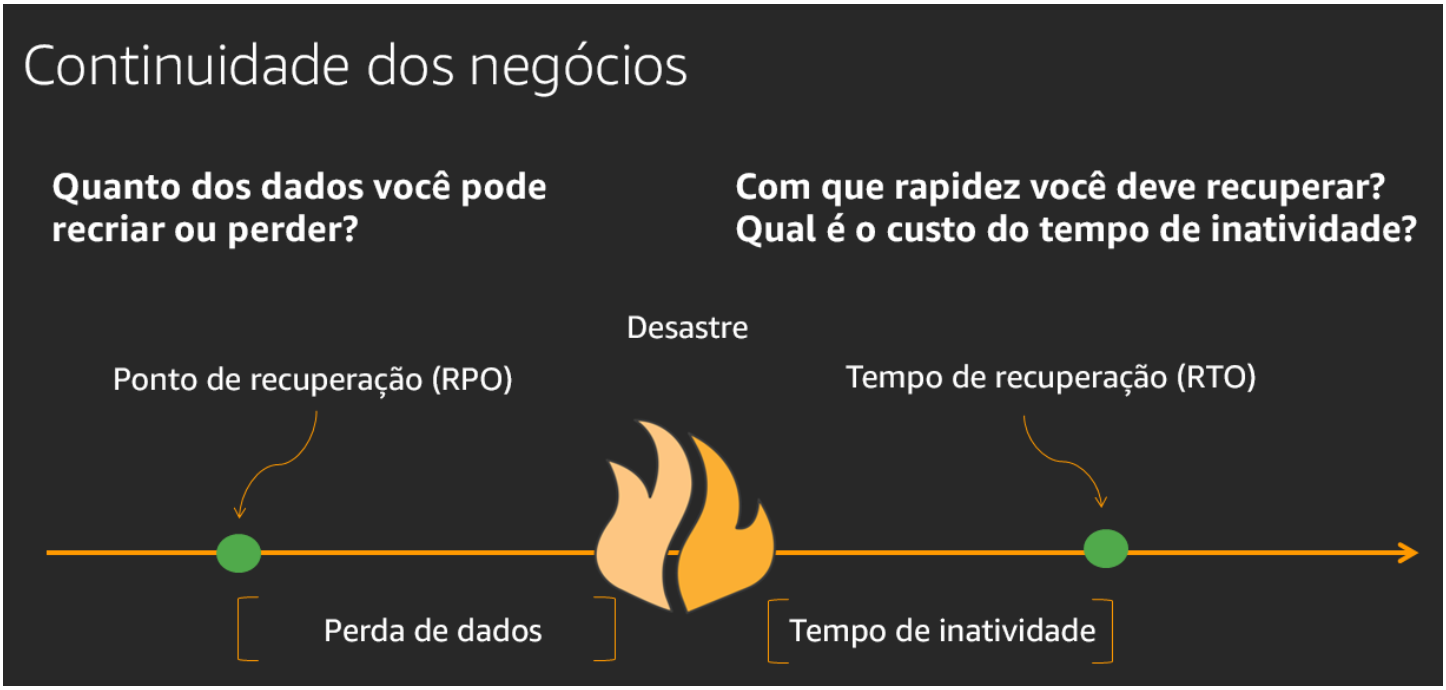
A seleção de dependências também contribui para o aumento dos custos em sistemas que operam com metas de design de disponibilidade maiores. Com esses objetivos maiores, o conjunto de software ou serviços que podem ser escolhidos como dependências diminui com base em quais receberam os investimentos elevados descritos anteriormente. Conforme o objetivo de design de disponibilidade aumenta, é comum encontrar menos serviços multiuso (como um banco de dados relacional) e mais serviços direcionados a uma finalidade específica. Isso acontece porque estes são mais fáceis de avaliar, testar e automatizar e têm menos potencial de interações inesperadas com funcionalidade incluída, mas não utilizada.

Objetivos da recuperação de desastres (DR)

Além dos objetivos de disponibilidade, a estratégia de resiliência também deve incluir os objetivos de recuperação de desastres (DR) para recuperar a workload, em caso de ocorrência de um desastre. A recuperação de desastres se concentra em objetivos de recuperação de uma única vez em resposta a desastres naturais, a falhas técnicas em grande escala ou a ameaças humanas, como ataques ou erros. Isso difere da disponibilidade que mede a resiliência por um período em resposta a falhas de componentes, a picos de carga ou a erros de software.

Objetivo de tempo de recuperação (RTO) Definido pela organização. RTO é o atraso máximo aceitável entre a interrupção do serviço e a restauração do serviço. Isso determina o que é considerado uma janela de tempo aceitável quando o serviço está indisponível.

Objetivo de ponto de recuperação (RPO) Definido pela organização. RPO é o tempo máximo aceitável desde o último ponto de recuperação de dados. Isso determina o que é considerado uma perda aceitável de dados entre o último ponto de recuperação e a interrupção do serviço.



A relação do RPO (Objetivo de ponto de recuperação), RTO (Objetivo do tempo de recuperação) e o evento de desastre.

O RTO é semelhante ao MTTR, pois ambos medem o tempo entre o início de uma interrupção e a recuperação da carga de trabalho. No entanto, MTTR é um valor médio assumido em vários eventos que afetam a disponibilidade ao longo de um período, enquanto RTO é um destino, ou valor máximo permitido, para um único evento que afeta a disponibilidade.

Compreensão das necessidades de disponibilidade

É comum pensar inicialmente na disponibilidade de um aplicativo como um único objetivo para o aplicativo como um todo. Porém, com uma inspeção mais detalhada, é comum descobrirmos que determinados aspectos de um aplicativo ou serviço têm requisitos de disponibilidade diferentes. Por exemplo, alguns sistemas podem priorizar a capacidade de receber e armazenar novos dados antes de recuperar dados existentes. Outros sistemas priorizam operações em tempo real a operações que mudam a configuração ou o ambiente de um sistema. Os serviços podem ter requisitos de disponibilidade muito altos durante determinados horários do dia, mas podem tolerar períodos muito mais longos de interrupção fora desses horários. Estas são algumas das maneiras como você pode dividir um único aplicativo nas partes que o compõem e avaliar os requisitos de disponibilidade de

cada uma. O benefício de fazer isso é concentrar os esforços (e as despesas) de disponibilidade conforme as necessidades específicas, em vez de projetar todo o sistema conforme o requisito mais rígido.

Recomendação

Avalie de modo crítico os aspectos exclusivos de suas aplicações e, quando adequado, diferenci e as metas de design de disponibilidade e de recuperação de desastres para refletirem as necessidades de sua empresa.

Na AWS, normalmente dividimos os serviços em “plano de dados” e “ambiente de gerenciamento”. O plano de dados é responsável por prestar serviço em tempo real, enquanto os planos de controle são usados para configurar o ambiente. Por exemplo, instâncias do Amazon EC2, bancos de dados do Amazon RDS e operações de leitura/gravação de tabelas do Amazon DynamoDB são operações de plano de dados. Em contraste, iniciar novas instâncias do EC2 ou bancos de dados do RDS ou adicionar ou alterar metadados de tabela no DynamoDB são consideradas operações na camada de controle. Embora altos níveis de disponibilidade sejam importantes para todas essas capacidades, os planos de dados costumam ter metas de design de disponibilidade maiores que os planos de controle. Portanto, as workloads com requisitos de alta disponibilidade devem evitar a dependência de tempo de execução nas operações do ambiente de gerenciamento.

Muitos dos clientes da AWS adotam uma abordagem similar para avaliar de modo crítico suas aplicações e identificar subcomponentes com diferentes necessidades de disponibilidade. As metas de design de disponibilidade são personalizadas para os diferentes aspectos, e os esforços de trabalho adequados são executados para projetar o sistema. A AWS possui experiência significativa de aplicações de engenharia com uma série de metas de design de disponibilidade, incluindo serviços com 99,999% ou mais de disponibilidade. Os arquitetos de soluções (SAs) da AWS podem ajudar você a projetar adequadamente conforme suas metas de disponibilidade. Envolver a AWS no início do processo de design melhora nossa capacidade de ajudar você a atingir suas metas de disponibilidade. O planejamento da disponibilidade não é feito apenas antes do início de sua carga de trabalho. Isso também é feito de modo contínuo para refinar seu design conforme você ganha experiência operacional, aprende com eventos do mundo real e tolera falhas de diferentes tipos. Assim você pode aplicar o esforço de trabalho adequado para melhorar sua implementação.

As necessidades de disponibilidade necessárias para uma carga de trabalho devem estar alinhadas à necessidade e à criticidade da empresa. Primeiro, definindo a estrutura de criticidade empresarial com RTO, RPO e disponibilidade definidos, você pode avaliar cada carga de trabalho. Essa

abordagem exige que as pessoas envolvidas na implementação da carga de trabalho tenham conhecimento da estrutura e do impacto que sua carga de trabalho tem sobre as necessidades empresariais.

Fundamentos

Os requisitos fundamentais são aqueles que têm um escopo que vai além de uma única carga de trabalho ou projeto. Antes de criar a arquitetura de um sistema, é necessário instaurar os requisitos fundamentais que influenciam a confiabilidade. Por exemplo, você deve ter largura de banda de rede suficiente no datacenter.

Em um ambiente local, esses requisitos podem causar longos prazos de entrega devido a dependências e, portanto, devem ser incorporados durante o planejamento inicial. No entanto, com a AWS, a maior parte desses requisitos fundamentais já está incorporada ou pode ser resolvida conforme necessário. A nuvem foi projetada para ser praticamente ilimitada, portanto, é responsabilidade da AWS atender ao requisito de capacidade suficiente para as redes e a computação, deixando você livre para alterar o tamanho e as alocações de recursos sob demanda.

As seções a seguir explicam melhores práticas que se concentram nessas considerações para fins de confiabilidade.

Tópicos

Gerenciar cotas de serviço e restrições de serviços

Para arquiteturas de carga de trabalho baseadas na nuvem, há cotas de serviço, que também são conhecidas como limites de serviço. Essas cotas existem para evitar o aprovisionamento acidental de mais recursos do que o necessário e para limitar as taxas de solicitação nas operações de API para proteger os serviços contra abuso. Há também restrições de recursos, por exemplo, a taxa de envio de bits por um cabo de fibra óptica ou a quantidade de armazenamento em um disco físico.

Se estiver usando aplicações do AWS Marketplace, será necessário compreender essas limitações. Se você estiver usando software como serviço ou serviços web de terceiros, também precisará estar ciente desses limites.

Práticas recomendadas

- REL01-BP04 Monitorar e gerenciar cotas
- REL01-BP05 Automatizar o gerenciamento de cotas
- REL01-BP06 Garantir que existe uma lacuna suficiente entre as cotas atuais e o uso máximo para acomodar o failover

REL01-BP01 Conhecimento das cotas e restrições de serviço

Esteja ciente das suas cotas padrão e das solicitações de aumento de cota referentes à sua arquitetura de workload. Saiba quais restrições de recursos, como disco ou rede, podem gerar impactos.

Resultado desejado: os clientes conseguem evitar a degradação ou a interrupção do serviço nas Contas da AWS implementando diretrizes adequadas para monitorar as principais métricas, análises da infraestrutura e etapas de remediação da automação, a fim de confirmar que as cotas e as restrições do serviço não foram atingidas, o que poderia causar degradação ou interrupção do serviço.

Antipadrões comuns:

- Implantar uma workload sem compreender as cotas flexíveis ou fixas e seus limites para os serviços utilizados.
- Implantar uma workload de substituição sem analisar e reconfigurar as cotas necessárias ou entrar em contato com o suporte com antecedência.
- Pressupor que os serviços em nuvem não têm limites e os serviços podem ser usados sem considerar taxas, limites, contagens e quantidades.
- Pressupor que as cotas aumentarão automaticamente.
- Não saber o processo e a linha de tempo das solicitações de cota.
- Pressupor que a cota de serviço em nuvem padrão é idêntica para todos os serviços em comparação entre as regiões.
- Pressupor que as restrições do serviço podem ser violadas e os sistemas vão ser escalados automaticamente ou aumentar o limite além das restrições do recurso
- Não testar a aplicação em tráfego de pico a fim de aplicar tensão na utilização de seus recursos.
- Provisionar o recurso sem analisar o tamanho necessário dele.
- Superprovisionar capacidade selecionando tipos de recurso que superam em muito a necessidade real ou os picos esperados.

- Não avaliar os requisitos de capacidade para novos níveis de tráfego antes de um novo evento de cliente ou implantação de uma nova tecnologia.

Benefícios do estabelecimento desta prática recomendada: o monitoramento e o gerenciamento automatizado de cotas de serviço e restrições de recursos podem reduzir as falhas de forma proativa. As alterações nos padrões de tráfego do serviço de um cliente poderão causar interrupção ou degradação se as práticas recomendadas não forem seguidas. Ao monitorar e gerenciar esses valores em todas as regiões e contas, as aplicações podem ter uma resiliência aprimorada em eventos adversos ou não planejados.

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto

Orientações para a implementação

O Service Quotas é um serviço da AWS que ajuda você a gerenciar as cotas de mais de 250 serviços da AWS em um único local. Além de pesquisar os valores de cotas, você também pode solicitar e acompanhar aumentos de cota no console do Service Quotas ou por meio do AWS SDK. O AWS Trusted Advisor oferece uma verificação de cotas de serviço que exibe o uso e as cotas para certos aspectos de alguns serviços. As cotas de serviço padrão por serviço também estão na documentação da AWS por respectivo serviço, por exemplo, consulte Cotas da Amazon VPC.

Alguns limites de serviço, como os limites de taxa para APIs limitadas, são definidos no próprio Amazon API Gateway por meio da configuração de um plano de uso. Alguns limites definidos como configuração em seus respectivos serviços incluem IOPS provisionadas, armazenamento do Amazon RDS alocado e alocações de volume do Amazon EBS. O Amazon Elastic Compute Cloud tem seu próprio painel de limites de serviço, que pode ajudar você a gerenciar sua instância, o Amazon Elastic Block Store e os limites de endereços IP elásticos. Se você tiver um caso de uso em que as cotas de serviço afetam a performance de sua aplicação e elas não forem ajustadas às suas necessidades, entre em contato com o AWS Support para ver se há mitigações.

As cotas de serviço podem ser específicas da região e também pode ser globais por natureza. O uso de um serviço da AWS com a cota atingida fará com que o comportamento dele não seja o esperado e poderá causar interrupção ou degradação do serviço. Por exemplo, a cota de um serviço limita o número de DL Amazon EC2 que pode ser usado em uma região e esse limite poderá ser atingido durante um evento de escalabilidade de tráfego usando grupos do Auto Scaling (ASG).

As cotas de serviço de cada conta devem ser avaliadas regularmente quanto ao uso a fim de determinar quais são os limites de serviço apropriados para a conta em questão. Essas cotas de serviço existem como barreiras de proteção operacionais, a fim de impedir o provisionamento

acidental de recursos além do necessário. Elas também servem para limitar as taxas de solicitação em operações de API para proteger os serviços contra abuso.

Restrições de serviço são diferentes de cotas de serviço. As restrições de serviço representam os limites de um recurso específico conforme definido pelo tipo de recurso em questão. Podem ser a capacidade de armazenamento (por exemplo, o gp2 tem um limite de tamanho de 1 GB a 16 TB) ou o throughput de disco (10 mil iops). É essencial que a restrição de um tipo de recurso seja projetada e avaliada constantemente quanto ao uso que pode atingir o limite. Se uma restrição for atingida de modo inesperado, as aplicações da conta ou os serviços poderão sofrer degradação ou interrupção.

Se houver um caso de uso em que as cotas de serviço afetem a performance de uma aplicação e elas não puderem ser ajustadas às necessidades, entre em contato com o AWS Support para ver se há mitigações. Para obter mais detalhes sobre o ajuste de cotas fixas, consulte REL01-BP03 Acomodar as restrições e as cotas fixas de serviço por meio da arquitetura.

Há uma série de serviços e ferramentas da AWS para ajudar a monitorar e gerenciar o Service Quotas. O serviço e as ferramentas devem ser utilizadas para oferecer verificações automatizadas ou manuais dos níveis de cota.

- O AWS Trusted Advisor oferece uma verificação de cotas de serviço que exibe o uso e cotas para alguns aspectos de alguns serviços. Ele pode ajudar na identificação de serviços que estão próximos da cota.
- O AWS Management Console oferece métodos para exibir valores de cota de serviço, gerenciar, solicitar novas cotas, monitorar o status das solicitações de cota e exibir o histórico de cotas.
- A AWS CLI e os CDKs oferecem métodos programáticos para gerenciar e monitorar automaticamente os níveis e o uso de cotas de serviço.

Etapas da implementação

Para Service Quotas:

- Analisar o AWS Service Quotas.
- Para saber suas cotas de serviço existentes, determine os serviços (como o IAM Access Analyzer) utilizados. Há cerca de 250 serviços da AWS controlados por cotas de serviço. Depois, determine o nome específico da cota de serviço que pode estar sendo usada em cada conta e região. Há cerca de 3 mil nomes de cota de serviço por região.
- Incremente essa análise de cota com o AWS Config para encontrar todos os recursos da AWS utilizados em suas Contas da AWS.

- Use dados do AWS CloudFormation para determinar seus recursos da AWS utilizados. Examine os recursos que foram criados no AWS Management Console ou com o comando `list-stack-resources` da AWS CLI. Você também pode ver no próprio modelo os recursos configurados para implantação.
- Examine o código da implantação para determinar todos os serviços necessários à sua workload.
- Determine as cotas de serviço aplicáveis. Use as informações acessíveis programaticamente por meio do Trusted Advisor e do Service Quotas.
- Estabeleça um método de monitoramento automatizado (consulte REL01-BP02 Gerenciar cotas de serviço de várias contas e regiões e REL01-BP04 Monitorar e gerenciar cotas) para alertar e informar se as cotas de serviço estiverem perto do limite ou o atingirem.
- Estabeleça um método automatizado e programático para conferir se uma cota de serviço foi alterada em uma região, mas não em outras na mesma conta (consulte REL01-BP02 Gerenciar cotas de serviço de várias contas e regiões e REL01-BP04 Monitorar e gerenciar cotas).
- Automatize as verificações de logs e métricas de aplicações para determinar se há erros de restrição de serviço ou cota. Se houver esses erros, envie alertas ao sistema de monitoramento.
- Estabeleça os procedimentos de engenharia para calcular a alteração necessária na cota (consulte REL01-BP05 Automatizar o gerenciamento de cotas) depois de identificar que são necessárias cotas maiores para serviços específicos.
- Crie um fluxo de trabalho de provisionamento e aprovação para solicitar alterações na cota de serviço. Isso deve incluir um fluxo de trabalho de exceção em caso de negação de solicitação ou aprovação parcial.
- Crie um método de engenharia para analisar cotas de serviço antes de provisionar e usar novos serviços da AWS antes de distribuir na produção ou carregar ambientes (por exemplo, conta de teste de carga).

Para restrições de serviço:

- Estabeleça métodos de monitoramento e métricas para alertar sobre recursos que estejam próximos de suas restrições de recurso. Utilize o CloudWatch conforme apropriado para métricas ou monitoramento de logs.
- Estabeleça limites de alerta para cada recurso que tenha uma restrição significativa para a aplicação ou o sistema.
- Crie um fluxo de trabalho e procedimentos de gerenciamento de infraestrutura para alterar o tipo de recurso se a restrição estiver próxima da utilização. Esse fluxo de trabalho deve incluir testes de

carga como prática recomendada para verificar se o novo tipo de recurso é o correto com as novas restrições.

- Migre o recurso identificado para o novo tipo de recurso usando os procedimentos e os processos existentes.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP02 Gerenciar cotas de serviço de várias contas e regiões
- REL01-BP03 Acomodar as restrições e as cotas fixas de serviço por meio da arquitetura
- REL01-BP04 Monitorar e gerenciar cotas
- REL01-BP05 Automatizar o gerenciamento de cotas
- REL01-BP06 Garantir que existe uma lacuna suficiente entre as cotas atuais e o uso máximo para acomodar o failover
- REL03-BP01 Escolher como segmentar a workload
- REL10-BP01 Implantar a workload em vários locais
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP03 Automatizar a reparação em todas as camadas
- REL12-BP05 Testar a resiliência por meio da engenharia do caos

Documentos relacionados:

- AWS Pilar Confiabilidade da Well-Architected Framework: Disponibilidade
- AWS Service Quotas (anteriormente chamado de limites de serviço)
- AWS Trusted Advisor Best Practice Checks (Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Service Limits (Limites de serviço))
- AWS limit monitor on AWS answers (Monitor de limites da AWS em respostas da AWS)
- Amazon EC2 Service Limits (Limites de serviço do Amazon EC2)
- What is Service Quotas? (O que é o Service Quotas?)
- How to Request Quota Increase (Como solicitar aumento de cota)
- Service endpoints and quotas (Endpoints e cotas de serviço)
- Guia do usuário do Service Quotas

- Quota Monitor for AWS (Monitor de cotas da AWS)
- AWS Fault Isolation Boundaries (Limites de isolamento de falhas da AWS)
- Availability with redundancy (Disponibilidade com redundância)
- AWS para dados
- O que significa integração contínua?
- O que significa distribuição contínua?
- Parceiro do APN: parceiros que podem ajudar no gerenciamento de configuração
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Gerenciar o ciclo devida da conta em ambientes de SaaS de conta por locatário na AWS)
- Managing and monitoring API throttling in your workloads (Gerenciar e monitorar o controle de utilização de API em workloads)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Exibir recomendações do AWS Trusted Advisor em grande escala com AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatizar aumentos de limite de serviço e suporte empresarial com AWS Control Tower)

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Exibir e gerenciar cotas para serviços da AWS usando o Service Quotas)
- AWS IAM Quotas Demo (Demonstração de cotas do AWS IAM)

Ferramentas relacionadas:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor

- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP02 Gerenciar cotas de serviço de várias contas e regiões

Se você estiver usando várias contas ou regiões, solicite as cotas adequadas em todos os ambientes nos quais suas workloads de produção são executadas.

Resultado desejado: os serviços e as aplicações não devem ser afetados pelo esgotamento da cota de serviço para configurações que abrangem contas ou regiões ou que têm designs de resiliência que usam failover de conta, zona ou região.

Antipadrões comuns:

- Permitir que a utilização de recursos em uma região de isolamento aumente sem nenhum mecanismo para manter a capacidade das demais.
- Configurar manualmente todas as cotas nas regiões de isolamento de forma independente.
- Não considerar o efeito das arquiteturas de resiliência (como ativa ou passiva) em necessidades futuras de cota durante a degradação na região que não é a principal.
- Não avaliar as cotas regularmente e fazer alterações necessárias em cada região e conta nas quais a workload é executada.
- Não utilizar modelos de solicitação de cota para solicitar aumentos em várias regiões e contas.
- Não atualizar as cotas de serviço por imaginar incorretamente que aumentar as cotas tem implicações de custo, como solicitações de reserva computacional.

Benefícios do estabelecimento desta prática recomendada: confirmar que você pode lidar com sua carga atual em contas ou regiões secundárias se os serviços regionais ficarem indisponíveis. Isso pode ajudar a reduzir o número de erros ou níveis de degradações que ocorrem durante a perda da região.

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto

Orientações para a implementação

Cotas de serviço são rastreadas por conta. A menos que especificado de outra forma, cada cota é específica da Região da AWS. Além dos ambientes de produção, gerencie também as cotas em

todos os ambientes aplicáveis que não são de produção, para que os testes e o desenvolvimento não sejam dificultados. Manter um alto grau de resiliência exige que as cotas de serviço sejam avaliadas de forma contínua (sejam elas automatizadas ou manuais).

Com mais workloads abrangendo regiões devido à implementação de designs usando as abordagens Ativo/Ativo, Ativo/Passivo: Quente, Ativo/Passivo: Frio e Ativo/Passivo: Luz piloto, é essencial entender todos os níveis de cota de contas e regiões. Padrões de tráfego passados nem sempre são um bom indicador de que a cota de serviço está definida corretamente.

Igualmente importante, o limite do nome da cota de serviço nem sempre é o mesmo para cada região. Em uma região, o valor pode ser cinco e em outra região pode ser dez. O gerenciamento dessas cotas deve abranger todos os mesmos serviços, contas e regiões para fornecer resiliência consistente sob carga.

Reconcilie todas as diferenças de cota de serviço em todas as diferentes regiões (Região ativa ou Região passiva) e crie processos para reconciliar de forma contínua essas diferenças. Os planos de teste de failovers de região passiva raramente são escalados para a capacidade ativa de pico, o que significa que os exercícios de simulações teóricas e dias de teste podem não encontrar diferenças em cotas de serviço entre regiões e também depois manter os limites corretos.

É muito importante rastrear e avaliar o desvio de cotas de serviço, a condição em que os limites de uma cota de serviço específica são alterados em uma região e não em todas. É necessário pensar em alterar a cota em regiões com tráfego ou que possam ter tráfego.

- Selecione as contas e as regiões relevantes conforme seus requisitos de serviço, de latência, regulatórios e de recuperação de desastres.
- Identifique as cotas de serviço de todas as contas, regiões e zonas de disponibilidade relevantes. O escopo dos limites é definido para conta e região. Esses valores devem ser comparados em relação a diferenças.

Etapas da implementação

- Analise os valores do Service Quotas que possam ter ultrapassado um nível de risco de uso. O AWS Trusted Advisor oferece alertas para violações de limite de 80% e 90%.
- Analise os valores de cotas de serviço em todas as regiões passivas (em um design ativo/passivo). Verifique se a carga será executada com êxito em regiões secundárias em caso de falha na região principal.

- Automatize a avaliação se ocorreu algum desvio de cota de serviço entre as regiões na mesma conta e aja adequadamente para alterar os limites.
- Se as unidades organizações (UO) do cliente estiverem estruturadas da forma compatível, os modelos de cota de serviço deverão ser atualizados para refletir alterações em todas as cotas que devem ser aplicadas a várias regiões e contas.
  - Crie um modelo e associe regiões à alteração de cota.
  - Analise todos os modelos de cota de serviço existentes para todas as alterações necessárias (região, limites e contas).

Recursos

Práticas recomendadas relacionadas:

- REL01-BP01 Conhecimento das cotas e restrições de serviço
- REL01-BP03 Acomodar as restrições e as cotas fixas de serviço por meio da arquitetura
- REL01-BP04 Monitorar e gerenciar cotas
- REL01-BP05 Automatizar o gerenciamento de cotas
- REL01-BP06 Garantir que existe uma lacuna suficiente entre as cotas atuais e o uso máximo para acomodar o failover
- REL03-BP01 Escolher como segmentar a workload
- REL10-BP01 Implantar a workload em vários locais
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP03 Automatizar a reparação em todas as camadas
- REL12-BP05 Testar a resiliência por meio da engenharia do caos

Documentos relacionados:

- AWS Pilar Confiabilidade da Well-Architected Framework: Disponibilidade
- AWS Service Quotas (anteriormente chamado de limites de serviço)
- AWS Trusted Advisor Best Practice Checks (Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Service Limits (Limites de serviço))
- AWS limit monitor on AWS answers (Monitor de limites da AWS em respostas da AWS)
- Amazon EC2 Service Limits (Limites de serviço do Amazon EC2)

- What is Service Quotas? (O que é o Service Quotas?)
- How to Request Quota Increase (Como solicitar aumento de cota)
- Service endpoints and quotas (Endpoints e cotas de serviço)
- Guia do usuário do Service Quotas
- Quota Monitor for AWS (Monitor de cotas da AWS)
- AWS Fault Isolation Boundaries (Limites de isolamento de falhas da AWS)
- Availability with redundancy (Disponibilidade com redundância)
- AWS para dados
- O que significa integração contínua?
- O que significa distribuição contínua?
- Parceiro do APN: parceiros que podem ajudar no gerenciamento de configuração
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Gerenciar o ciclo devida da conta em ambientes de SaaS de conta por locatário na AWS)
- Managing and monitoring API throttling in your workloads (Gerenciar e monitorar o controle de utilização de API em workloads)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Exibir recomendações do AWS Trusted Advisor em grande escala com AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatizar aumentos de limite de serviço e suporte empresarial com AWS Control Tower)

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Exibir e gerenciar cotas para serviços da AWS usando o Service Quotas)
- AWS IAM Quotas Demo (Demonstração de cotas do AWS IAM)

Serviços relacionados:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch

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

REL01-BP03 Acomodar as restrições e as cotas fixas de serviço por meio da arquitetura

Esteja ciente das cotas de serviço, das restrições do serviço e dos limites de recursos físicos que não podem ser alterados. Projete arquiteturas para aplicações e serviços visando evitar que esses limites afetem a confiabilidade.

Os exemplos incluem largura de banda da rede, tamanho da carga útil da invocação da função sem servidor, taxa de intermitência de aceleração para um gateway da API e conexões simultâneas de usuários com um banco de dados.

Resultado desejado: a aplicação ou o serviço tem o desempenho esperado em condições de tráfego normal e alto. Elas foram projetadas para funcionar com as limitações referentes às restrições ficas ou cotas de serviço do recurso.

Antipadrões comuns:

- Escolher um design que usa um recurso de um serviço sem saber que há restrições de design que causarão falha à medida que você escala.
- Usar parâmetros de comparação irrealistas e que atingirão as cotas fixas do serviço durante os testes. Por exemplo, executar testes em um limite de intermitência mas por um período estendido.
- Escolher um design que não possa ser escalado nem modificado caso seja necessário ultrapassar as cotas fixas do serviço. Por exemplo, um tamanho de carga útil do SQS de 256 KB.
- A capacidade de observação não foi projetada nem implementada para monitorar e alertar sobre os limites das cotas de serviço que podem estar em risco durante eventos com tráfego alto.

Benefícios do estabelecimento dessa prática recomendada: verificar se a aplicação será executada em todos os níveis de carga de serviços projetados sem interrupção ou dano.

Nível de risco exposto se esta prática recomendada não é estabelecida: médio

Orientações para a implementação

Ao contrário das cotas de serviço flexíveis ou de recursos que são substituídos com unidades de capacidade mais altas, as cotas fixas dos serviços da AWS não podem ser alteradas. Isso significa que todos esses tipos de serviços da AWS devem ser avaliados com relação a possíveis limites de capacidade rígidos quando usados em um design da aplicação.

Os limites rígidos são mostrados no console do Service Quotas. Se a coluna mostrar `ADJUSTABLE = No`, o serviço tem um limite rígido. Os limites rígidos também são mostrados em algumas páginas de configuração de recursos. Por exemplo, o Lambda tem limites rígidos específicos que não podem ser ajustados.

Como exemplo, ao projetar uma aplicação Python para ser executada em uma função do Lambda, a aplicação deve ser avaliada para determinar se há alguma chance de o Lambda ser executado por mais de 15 minutos. Se código puder ser executado mais do que esse limite de cota de serviço, tecnologias ou designs alterativos devem ser considerados. Se esse limite for atingido depois da implantação na produção, a aplicação sofrerá uma degradação e interrupção até que isso possa ser corrigido. Ao contrário das cotas flexíveis, não há um método para alterar esses limites mesmo sob eventos de emergência de gravidade 1.

Depois que a aplicação for implantada em um ambiente de teste, devem ser usadas estratégias para descobrir se algum limite rígido pode ser atingido. Testes de estresse, testes de carga e testes de caos devem fazer parte do plano de teste de introdução.

Etapas da implementação

- Revise a lista completa de serviços da AWS que poderiam ser usados na fase de design da aplicação.
- Revise os limites da cota flexível e os da cota rígida para todos esses serviços. Nem todos os limites são mostrados no console do Service Quotas. Alguns serviços descrevem esses limites em locais alternativos.
- À medida que você planeja a aplicação, revise os fatores que impulsionam a tecnologia e os negócios da workload, como resultados empresariais, casos de uso, sistemas dependentes, destinos de disponibilidade e objetos de recuperação de desastres. Permita que os fatores que impulsionam a tecnologia e os negócios orientem o processo para identificar o sistema distribuído certo para sua workload.

- Analise a carga do serviço nas regiões e contas. Muitos limites rígidos são regionais para os serviços. No entanto, alguns limites são por conta.
- Analise arquiteturas de resiliência quanto ao uso de recursos durante uma falha de zona e de região. Na progressão de designs de várias regiões usando as abordagens ativo/ativo, ativo/passivo – quente, ativo/passivo – frio e ativo/passivo – luz-piloto, esses casos de falha resultarão em maior uso. Isso cria um possível caso de uso para atingir limites rígidos.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP01 Conhecimento das cotas e restrições de serviço
- REL01-BP02 Gerenciar cotas de serviço de várias contas e regiões
- REL01-BP04 Monitorar e gerenciar cotas
- REL01-BP05 Automatizar o gerenciamento de cotas
- REL01-BP06 Garantir que existe uma lacuna suficiente entre as cotas atuais e o uso máximo para acomodar o failover
- REL03-BP01 Escolher como segmentar a workload
- REL10-BP01 Implantar a workload em vários locais
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP03 Automatizar a reparação em todas as camadas
- REL12-BP05 Testar a resiliência por meio da engenharia do caos

Documentos relacionados:

- AWS Pilar Confiabilidade da Well-Architected Framework: Disponibilidade
- AWS Service Quotas (anteriormente chamado de limites de serviço)
- AWS Trusted Advisor Best Practice Checks (Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Service Limits (Limites de serviço))
- AWS limit monitor on AWS answers (Monitor de limites da AWS em respostas da AWS)
- Amazon EC2 Service Limits (Limites de serviço do Amazon EC2)
- What is Service Quotas? (O que é o Service Quotas?)
- How to Request Quota Increase (Como solicitar aumento de cota)

- Service endpoints and quotas (Endpoints e cotas de serviço)
- Guia do usuário do Service Quotas
- Quota Monitor for AWS (Monitor de cotas da AWS)
- AWS Fault Isolation Boundaries (Limites de isolamento de falhas da AWS)
- Availability with redundancy (Disponibilidade com redundância)
- AWS para dados
- O que significa integração contínua?
- O que significa distribuição contínua?
- Parceiro do APN: parceiros que podem ajudar no gerenciamento de configuração
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Gerenciar o ciclo devida da conta em ambientes de SaaS de conta por locatário na AWS)
- Managing and monitoring API throttling in your workloads (Gerenciar e monitorar o controle de utilização de API em workloads)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Exibir recomendações do AWS Trusted Advisor em grande escala com AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatizar aumentos de limite de serviço e suporte empresarial com AWS Control Tower)
- Ações, recursos e chaves de condição do Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Exibir e gerenciar cotas para serviços da AWS usando o Service Quotas)
- AWS IAM Quotas Demo (Demonstração de cotas do AWS IAM)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: fechar ciclos e abrir mentes: como controlar sistemas, sejam grandes ou pequenos)

Ferramentas relacionadas:

- AWS CodeDeploy
- AWS CloudTrail

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

REL01-BP04 Monitorar e gerenciar cotas

Avalie seu uso potencial e aumente suas cotas adequadamente, permitindo o crescimento planejado do uso.

Resultado desejado: sistemas ativos e automáticos que gerenciam e monitoram foram implantados. Essas soluções garantem que os limites de uso da cota sejam quase atingidos. Isso seria corrigido proativamente por mudanças na cota solicitada.

Antipadrões comuns:

- Não configurar o monitoramento para verificar os limites da cota de serviço.
- Não configurar o monitoramento de limites rígidos, embora esses valores não possam ser alterados.
- Presumir que o tempo necessário para solicitar e proteger uma mudança de cota flexível seja imediato ou período curto.
- Configurar alarmes para quando as cotas de serviço estiverem sendo atingidas, mas não ter um processo de resposta a um alerta.
- Configurar alarmes apenas para serviços compatíveis com o AWS Service Quotas e não monitorar outros serviços da AWS.
- Não considerar o gerenciamento da cota para designs com resiliência de várias regiões, como as abordagens ativo/ativo, ativo/passivo – quente, ativo/passivo – frio e ativo/passivo – luz-piloto.
- Não avaliar as diferenças de cota entre regiões.
- Não avaliar as necessidades de cada região com relação a uma solicitação de aumento de cota específica.

- Não utilizar modelos para o gerenciamento de cota de várias regiões.

Benefícios do estabelecimento dessa prática recomendada: o rastreamento automático do AWS Service Quotas e o monitoramento do uso em relação a essas cotas permitirão que você veja quando estiver perto de atingir um limite de cota. Também é possível usar esse monitoramento de dados para ajudar a limitar qualquer dano devido à exaustão da cota.

Nível de risco exposto se esta prática recomendada não é estabelecida: médio

Orientações para a implementação

Para dispositivos compatíveis, é possível monitorar as cotas configurando vários serviços diferentes que podem avaliar e, então, enviar alertas ou alarmes. Isso pode auxiliar o monitoramento do uso e alertar quando você estiver se aproximando das cotas. Esses alarmes podem ser acionados pelo AWS Config, por funções do Lambda, pelo Amazon CloudWatch ou pelo AWS Trusted Advisor. Você também pode usar filtros de métrica no CloudWatch Logs para pesquisar e extrair padrões nos logs a fim de determinar se o uso está se aproximando dos limites de cota.

Etapas da implementação

Para monitoramento:

- Capture o consumo atual de recursos (por exemplo, buckets ou instâncias). Use operações de API de serviço, como a API do Amazon EC2 `DescribeInstances` para coletar o consumo atual de recursos.
- Capture as cotas atuais que são essenciais e aplicáveis aos serviços usando:
  - AWS Service Quotas
  - AWS Trusted Advisor
  - Documentação da AWS
  - Páginas específicas de serviços da AWS
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Use o AWS Service Quotas, um serviço da AWS que ajuda você a gerenciar as cotas de mais de 250 serviços da AWS em um único local.
- Use os limites de serviço do Trusted Advisor para monitorar os limites de serviço atuais em vários limites.
- Use o histórico da cota de serviço (console ou AWS CLI) para verificar os aumentos regionais.

- Compare as alterações na cota de serviço em cada região e cada conta para criar equivalência, se necessário.

Para gerenciamento:

- Automático: configure uma regra personalizada do AWS Config para verificar as cotas de serviço nas regiões e comparar as diferenças.
- Automático: configure uma função programada do Lambda para verificar as cotas de serviço nas regiões e comparar as diferenças.
- Manual: verifique as cotas de serviço por meio da AWS CLI, da API ou do Console da AWS para conferir as cotas de serviço nas regiões e comparar as diferenças. Relate as diferenças.
- Se forem identificadas diferenças nas cotas entre as regiões, solicite uma mudança na cota, se necessário.
- Avalie o resultado de todas as solicitações.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP01 Conhecimento das cotas e restrições de serviço
- REL01-BP02 Gerenciar cotas de serviço de várias contas e regiões
- REL01-BP03 Acomodar as restrições e as cotas fixas de serviço por meio da arquitetura
- REL01-BP05 Automatizar o gerenciamento de cotas
- REL01-BP06 Garantir que existe uma lacuna suficiente entre as cotas atuais e o uso máximo para acomodar o failover
- REL03-BP01 Escolher como segmentar a workload
- REL10-BP01 Implantar a workload em vários locais
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP03 Automatizar a reparação em todas as camadas
- REL12-BP05 Testar a resiliência por meio da engenharia do caos

Documentos relacionados:

- AWS Pilar Confiabilidade da Well-Architected Framework: Disponibilidade

- AWS Service Quotas (anteriormente chamado de limites de serviço)
- AWS Trusted Advisor Best Practice Checks (Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Service Limits (Limites de serviço))
- AWS limit monitor on AWS answers (Monitor de limites da AWS em respostas da AWS)
- Amazon EC2 Service Limits (Limites de serviço do Amazon EC2)
- What is Service Quotas? (O que é o Service Quotas?)
- How to Request Quota Increase (Como solicitar aumento de cota)
- Service endpoints and quotas (Endpoints e cotas de serviço)
- Guia do usuário do Service Quotas
- Quota Monitor for AWS (Monitor de cotas da AWS)
- AWS Fault Isolation Boundaries (Limites de isolamento de falhas da AWS)
- Availability with redundancy (Disponibilidade com redundância)
- AWS para dados
- O que significa integração contínua?
- O que significa distribuição contínua?
- Parceiro do APN: parceiros que podem ajudar no gerenciamento de configuração
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Gerenciar o ciclo devida da conta em ambientes de SaaS de conta por locatário na AWS)
- Managing and monitoring API throttling in your workloads (Gerenciar e monitorar o controle de utilização de API em workloads)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Exibir recomendações do AWS Trusted Advisor em grande escala com AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatizar aumentos de limite de serviço e suporte empresarial com AWS Control Tower)
- Ações, recursos e chaves de condição do Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Exibir e gerenciar cotas para serviços da AWS usando o Service Quotas)
- AWS IAM Quotas Demo (Demonstração de cotas do AWS IAM)

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: fechar ciclos e abrir mentes: como controlar sistemas, sejam grandes ou pequenos)

Ferramentas relacionadas:

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

REL01-BP05 Automatizar o gerenciamento de cotas

Implemente ferramentas para alertar você quando os limites estiverem perto de serem atingidos. Ao usar as APIs do AWS Service Quotas, você pode automatizar as solicitações de aumento de cota.

Se você integrar o Configuration Management Database (CMDB) ou sistema de emissão de tíquetes com o Service Quotas, poderá automatizar o acompanhamento de solicitações de aumento de cota e as cotas atuais. Além do AWS SDK, o Service Quotas oferece automação usando o AWS Command Line Interface (AWS CLI).

Antipadrões comuns:

- Acompanhar as cotas e o uso em planilhas.
- Executar relatórios sobre o uso diário, semanal ou mensal e comparar o uso com as cotas.

Benefícios do estabelecimento dessa prática recomendada: O acompanhamento automatizado das cotas de serviço da AWS e o monitoramento do seu uso em relação a essa cota permitem que você veja quando está perto de atingir um limite. Você pode configurar a automação para ajudá-lo a

solicitar um aumento de cota quando necessário. Você pode considerar a redução de algumas cotas quando seu uso estiver na direção oposta para aproveitar os benefícios do risco reduzido (no caso de credenciais comprometidas) e da economia de custos.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- Configure o monitoramento automatizado. Implemente ferramentas usando SDKs para alertar você quando os limites estiverem perto de serem atingidos.
  - Use o Service Quotas e aumente o serviço com uma solução automatizada de monitoramento de cotas, como o AWS Limit Monitor ou uma oferta do AWS Marketplace.
    - O que é o Service Quotas?
    - Monitoramento de cotas na AWS: solução da AWS
  - Configure respostas acionadas com base nos limites de cota por meio do Amazon SNS e das APIs do AWS Service Quotas.
  - Teste a automação.
    - Configure os limites.
    - Integre-se a eventos de alteração do AWS Config, de pipelines de implantação, do Amazon EventBridge ou de terceiros.
    - Defina limites baixos fictícios de cota para testar as respostas.
    - Configure gatilhos para executar a ação adequada mediante notificações e entre em contato com o AWS Support quando necessário.
    - Acione manualmente os eventos de alteração.
    - Execute um dia de jogo para testar o processo de alteração de aumento de cota.

Recursos

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar no gerenciamento de configuração
- AWS Marketplace: produtos CMDB que ajudam a acompanhar os limites
- AWS Service Quotas (anteriormente chamado de limites de serviço)
- Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Limites de serviço)

- Monitoramento de cotas na AWS: solução da AWS
- Amazon EC2 Service Limits
- O que é o Service Quotas?

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP06 Garantir que existe uma lacuna suficiente entre as cotas atuais e o uso máximo para acomodar o failover

Quando um recurso falha ou fica inacessível, ele ainda pode ser contabilizado nas cotas até ser encerrado com êxito. Verifique se as cotas abrangem a sobreposição de recursos inacessíveis ou com falha e suas substituições. Você deve considerar casos de uso como falha de rede, falha na zona de disponibilidade ou falhas regionais ao calcular essa lacuna.

Resultado desejado: falhas pequenas ou grandes em recursos ou na acessibilidade de recursos podem ser cobertas nos limites atuais do serviço. As falhas de zona, falhas de rede ou até mesmo falhas regionais têm sido consideradas no planejamento de recursos.

Antipadrões comuns:

- Configurar cotas de serviço com base nas necessidades atuais sem considerar os cenários de failover.
- Não considerar as entidades principais de estabilidade estática ao calcular a cota de pico de um serviço.
- Não considerar o potencial de recursos inacessíveis no cálculo da cota total necessária para cada região.
- Não considerar os limites de isolamento de falhas de serviço da AWS para alguns serviços e seus padrões de uso possivelmente anormais.

Benefícios do estabelecimento dessa prática recomendada: quando um evento de interrupção do serviço afeta a disponibilidade da aplicação, a nuvem permite implementar estratégias para mitigar ou se recuperar desses eventos. Essas estratégias geralmente incluem a criação de recursos adicionais para substituir os que falharam ou estão inacessíveis. Sua estratégia de cota acomodaria essas condições de failover e não incluiria danos adicionais devido à exaustão do limite de serviço.

Nível de risco exposto se esta prática recomendada não é estabelecida: médio

Orientações para a implementação

Ao avaliar os limites de cota, considere casos de failover que podem ocorrer devido a algum dano. Os seguintes tipos de casos de failover devem ser considerados:

- Uma VPC interrompida ou inacessível.
- Uma sub-rede inacessível.
- Uma zona de disponibilidade foi danificada o suficiente para afetar a acessibilidade de muitos recursos.
- Várias rotas de rede ou pontos de ingresso e egresso são bloqueados ou alterados.
- Uma região foi danificada o suficiente para afetar a acessibilidade de muitos recursos.
- Há vários recursos, mas nem todos são afetados por uma falha em uma região ou zona de disponibilidade.

Falhas como as da lista acima poderiam ser o gatilho para iniciar um evento de failover. A decisão de fazer failover é única para cada situação e cliente, já que o impacto na empresa pode variar drasticamente. No entanto, ao decidir operacionalmente realizar failover de aplicações ou serviços, o planejamento da capacidade de recursos no local de failover e as cotas relacionadas devem ser solucionados antes do evento.

Revise as cotas de cada serviço considerando os picos mais altos do que o normal que podem ocorrer. Esses picos podem estar relacionados aos recursos que podem ser acessados devido às redes ou permissões, mas ainda estão ativos. Os recursos ativos não encerrados ainda serão contabilizados no limite de cota do serviço.

Etapas da implementação

- Verifique se há uma lacuna suficiente entre a cota de serviço e o uso máximo para acomodar um failover ou uma perda de acessibilidade.
- Determine suas cotas de serviço, considerando os padrões de implantação, os requisitos de disponibilidade e o aumento do consumo.
- Solicite aumentos de cota, se necessário. Planeje o tempo necessário para o atendimento das solicitações de aumento de cota.
- Determine os requisitos de confiabilidade (também conhecidos como “número de noves”).

- Estabeleça seus cenários de falha (por exemplo, perda de um componente, uma zona de disponibilidade ou uma região).
- Estabeleça a metodologia de implantação (por exemplo, canário, azul/verde, vermelho/preto ou gradual).
- Inclua uma reserva adequada (por exemplo, 15%) do limite atual.
- Inclua cálculos para estabilidade estática (por zona e região), quando apropriado.
- Planeje o aumento do consumo (por exemplo, monitore suas tendências de consumo).
- Considere o impacto da estabilidade estática das suas workloads mais críticas. Avalie os recursos em conformidade com um sistema estaticamente estável em todas as regiões e zonas de disponibilidade.
- Considere o uso de reservas de capacidade sob demanda para programas a capacidade antecipadamente de qualquer failover. Isso pode ser uma estratégia útil durante as programações empresariais mais críticas para reduzir possíveis riscos de obter a quantidade o tipo certo de recursos durante o failover.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP01 Conhecimento das cotas e restrições de serviço
- REL01-BP02 Gerenciar cotas de serviço de várias contas e regiões
- REL01-BP03 Acomodar as restrições e as cotas fixas de serviço por meio da arquitetura
- REL01-BP04 Monitorar e gerenciar cotas
- REL01-BP05 Automatizar o gerenciamento de cotas
- REL03-BP01 Escolher como segmentar a workload
- REL10-BP01 Implantar a workload em vários locais
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP03 Automatizar a reparação em todas as camadas
- REL12-BP05 Testar a resiliência por meio da engenharia do caos

Documentos relacionados:

- AWS Pilar Confiabilidade da Well-Architected Framework: Disponibilidade

- AWS Service Quotas (anteriormente chamado de limites de serviço)
- AWS Trusted Advisor Best Practice Checks (Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Service Limits (Limites de serviço))
- AWS limit monitor on AWS answers (Monitor de limites da AWS em respostas da AWS)
- Amazon EC2 Service Limits (Limites de serviço do Amazon EC2)
- What is Service Quotas? (O que é o Service Quotas?)
- How to Request Quota Increase (Como solicitar aumento de cota)
- Service endpoints and quotas (Endpoints e cotas de serviço)
- Guia do usuário do Service Quotas
- Quota Monitor for AWS (Monitor de cotas da AWS)
- AWS Fault Isolation Boundaries (Limites de isolamento de falhas da AWS)
- Availability with redundancy (Disponibilidade com redundância)
- AWS para dados
- O que significa integração contínua?
- O que significa distribuição contínua?
- Parceiro do APN: parceiros que podem ajudar no gerenciamento de configuração
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS (Gerenciar o ciclo devida da conta em ambientes de SaaS de conta por locatário na AWS)
- Managing and monitoring API throttling in your workloads (Gerenciar e monitorar o controle de utilização de API em workloads)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations (Exibir recomendações do AWS Trusted Advisor em grande escala com AWS Organizations)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower (Automatizar aumentos de limite de serviço e suporte empresarial com AWS Control Tower)
- Ações, recursos e chaves de condição do Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Exibir e gerenciar cotas para serviços da AWS usando o Service Quotas)

- AWS IAM Quotas Demo (Demonstração de cotas do AWS IAM)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: fechar ciclos e abrir mentes: como controlar sistemas, sejam grandes ou pequenos)

Ferramentas relacionadas:

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

Planejar a topologia da rede

Muitas vezes, as cargas de trabalho estão presentes em vários ambientes. Dentre eles estão vários ambientes de nuvem (acessíveis publicamente e privados) e possivelmente sua infraestrutura de datacenter existente. Os planos devem incluir considerações de rede, como conectividade dentro dos sistemas e entre eles, gerenciamento de endereços IP públicos e privados e resolução de nomes de domínio.

Ao arquitetar sistemas usando redes baseadas em endereço IP, planeje a topologia e o endereçamento de rede, antecipando possíveis falhas e acomodando o crescimento futuro e a integração com outros sistemas e as respectivas redes.

A Amazon Virtual Private Cloud (Amazon VPC) permite provisionar uma seção isolada e privada da Nuvem AWS, onde é possível executar recursos da AWS em uma rede virtual.

Práticas recomendadas

- REL02-BP01 Usar conectividade de rede altamente disponível nos endpoints públicos de workload
- REL02-BP02 Provisionar conectividade redundante entre as redes privadas na nuvem e nos ambientes on-premises
- REL02-BP03 Garantir contas de alocação de sub-rede IP para expansão e disponibilidade
- REL02-BP04 Preferir topologias hub-and-spoke em vez da malha muitos para muitos
- REL02-BP05 Aplicar intervalos de endereços IP privados não sobrepostos a todos os espaços de endereços privados onde estão conectados

REL02-BP01 Usar conectividade de rede altamente disponível nos endpoints públicos de workload

Criar uma conectividade de rede altamente disponível nos endpoints públicos das workloads pode ajudar a reduzir o tempo de inatividade devido à perda de conectividade e melhorar a disponibilidade e o SLA da workload. Para que isso seja possível, use DNS altamente disponível, redes de entrega de conteúdo (CDNs), gateways de API, balanceamento de carga ou proxies reversos.

Resultado desejado: é fundamental planejar, criar e operacionalizar a conectividade de rede altamente disponível para endpoints públicos. Se a workload ficar inacessível devido a uma perda de conectividade, mesmo se ela estiver em execução e indisponível, os clientes verão o sistema como inativo. Ao combinar a conectividade de rede altamente disponível e resiliente para os endpoints públicos da workload, junto com uma arquitetura resiliente para a própria workload, é possível fornecer o melhor nível possível de serviço e disponibilidade possível aos clientes.

O AWS Global Accelerator, o Amazon CloudFront, o Amazon API Gateway, os URLs de função do AWS Lambda, as APIs do AWS AppSync e o Elastic Load Balancing (ELB) fornecem endpoints públicos altamente disponíveis. O Amazon Route 53 fornece um serviço de DNS altamente disponível para a resolução do nome de domínio a fim de verificar se os endereços do endpoint público podem ser resolvidos.

Também é possível avaliar os dispositivos de software do AWS Marketplace com relação ao proxy e ao balanceamento de carga.

Antipadrões comuns:

- Projetar uma workload altamente disponível sem planejar a alta disponibilidade do DNS e da conectividade de rede.

- Usar endereços de internet públicos em instâncias ou contêineres individuais e gerenciar a conectividade com eles por meio de DNS.
- Usar endereços IP em vez de nomes de domínio para localizar serviços.
- Não testar cenários em que a conectividade com os endpoints públicos é perdida.
- Não analisar as necessidades de throughput de rede e os padrões de distribuição.
- Não testar nem se planejar para cenários em que a conectividade de rede da internet com os endpoints públicos da workload possam ser interrompidos.
- Fornecer conteúdo (como páginas da web, ativos estáticos ou arquivos de mídia) para uma grande área geográfica e não usar uma rede de entrega de conteúdo.
- Não se planejar para ataques de negação distribuída de serviços (DDoS). Ataques DDoS representam um risco de obstruir o tráfego legítimo e reduzir a disponibilidade para os usuários.

Benefícios do estabelecimento dessa prática recomendada: projetar-se para uma conectividade de rede resiliente e altamente disponível garante que a workload esteja acessível e disponível para os usuários.

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto

Orientações para a implementação

No centro da criação de uma conectividade de rede altamente disponível com os endpoints públicos está o roteamento do tráfego. Para verificar se o tráfego consegue acessar os endpoints, o DNS deve poder resolver os nomes de domínio para os endereços IP correspondentes. Use um Sistema de Nomes de Domínio (DNS) escalável e altamente disponível, como o Amazon Route 53, para gerenciar os registros de DNS do domínio. Também é possível usar verificações de integridade fornecidas pelo Amazon Route 53. As verificações de integridade conferem se a aplicação está acessível, disponível e funcional, e podem ser configuradas de uma maneira que imitem o comportamento do usuário, como solicitar uma página da web ou um URL específico. Em caso de falha, o Amazon Route 53 responde às solicitações de resolução de DNS e direciona o tráfego somente aos endpoints íntegros. Também é possível considerar o uso dos recursos DNS GEO e Roteamento baseado em latência oferecidos pelo Amazon Route 53.

Para verificar se a própria workload é altamente disponível, use o Elastic Load Balancing (ELB). O Amazon Route 53 pode ser usado para direcionar o tráfego para o ELB, o que distribui o tráfego às instâncias de computação de destino. Também é possível usar o Amazon API Gateway com o AWS Lambda para obter uma solução sem servidor. Os clientes também podem executar workloads em várias Regiões da AWS. Com o padrão multissite ativo/ativo, a workload pode fornecer o tráfego

de várias regiões. Com um padrão multissite ativo/passivo, a workload fornece tráfego da região ativa enquanto os dados são replicados para a região secundária e se torna ativa caso ocorra uma falha na região primaria. As verificações de integridade do Route 53 podem então ser usadas para controlar o failover de DNS de qualquer endpoint em uma região primária para um endpoint em uma região secundária, verificando se a workload está acessível e disponível para os usuários.

O Amazon CloudFront fornece uma API simples para distribuir o conteúdo com baixa latência e altas taxas de transferência de dados atendendo a solicitações usando uma rede de locais de borda ao redor do mundo. As redes de entrega de conteúdo (CDNs) atendem os clientes fornecendo conteúdo localizado ou armazenado em cache em um local próximo ao usuário. Isso também melhora a disponibilidade da aplicação, já que a carga do conteúdo é migrada dos servidores para os locais da borda do CloudFront. Os locais da borda e os caches de borda regionais armazenam cópias em cache do conteúdo próximo aos visualizadores, resultando em recuperação rápida e aumentando a acessibilidade e a disponibilidade da workload.

Para workloads com usuários distribuídos geograficamente, o AWS Global Accelerator ajuda a melhorar a disponibilidade e o desempenho das aplicações. O AWS Global Accelerator fornece endereços IP estáticos anycast que servem como um ponto de entrada fixo para a aplicação hospedada em uma ou mais Regiões da AWS. Isso permite que o tráfego entre na rede global da AWS o mais próximo possível dos usuários, melhorando a acessibilidade e a disponibilidade da workload. O AWS Global Accelerator também monitora a integridade dos endpoints da aplicação usando as verificações de integridade de TCP, HTTP e HTTPS. Qualquer mudança na integridade ou na configuração dos endpoints aciona o redirecionamento do tráfego de usuários para endpoints íntegros que oferecem o melhor desempenho e disponibilidade aos usuários. Além disso, o AWS Global Accelerator tem um design de isolamento de falhas que usa dois endereços IPv4 estáticos que são fornecidos por zonas de rede independentes, aumentando a disponibilidade das aplicações.

Para ajudar a proteger os clientes de ataques DDoS, a AWS oferece o AWS Shield Standard. O Shield Standard vem automaticamente habilitado e protege de ataques de infraestrutura comum (camadas três e quatro) como inundações de SYN/UDP e ataques de reflexão para comportar a alta disponibilidade das aplicações na AWS. Para obter mais proteções contra ataques maiores e mais sofisticados (como inundações de UDP), ataques de exaustão de estado (como inundações de TCP SYN) e para ajudar a proteger as aplicações executadas nos serviços Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator e Route 53, considere o uso do AWS Shield Advanced. Para proteção contra ataques de camada da aplicação como inundações de HTTP POST e GET, use o AWS WAF. O AWS WAF pode usar condições de scripts entre sites, endereços IP, cabeçalhos HTTP, corpo HTTP, strings de URI e injeção de SQL para determinar se uma solicitação deve ser bloqueada ou permitida.

Etapas da implementação

1. Configure DNS de alta disponibilidade: o Amazon Route 53 é um serviço da web de Sistema de Nomes de Domínio (DNS) escalável e altamente disponível. O Route 53 conecta as solicitações dos usuários às aplicações da internet executadas na AWS ou on-premises. Para obter mais informações, consulte Configurar o Amazon Route 53 como serviço DNS.
2. Configure verificações de integridade: ao usar o Route 53, verifique se somente os destinos íntegros podem ser resolvidos. Comece criando verificações de integridade do Route 53 e configurando o failover de DNS. Os aspectos a seguir são importantes a considerar ao configurar verificações de integridade:
   a. How Amazon Route 53 determines whether a health check is healthy (Como o Amazon Route 53 determina se uma verificação de integridade é íntegra)
   b. Criar, atualizar e excluir verificações de integridade
   c. Monitorar o status da verificação de integridade e receber notificações
   d. Práticas recomendadas do Amazon Route 53 DNS
3. Conectar o serviço de DNS aos endpoints.
   a. Ao usar o Elastic Load Balancing como destino do tráfego, crie um registro de alias usando o Amazon Route 53 que aponte para o endpoint regional do balanceador de carga. Durante a criação do registro de alias, defina a opção “Evaluate target health” (Avaliar integridade do destino) como “Yes” (Sim).
   b. Para workloads sem servidor ou APIs privadas quando o API Gateway é usado, use o Route 53 para redirecionar o tráfego para o API Gateway.
4. Decida sobre uma rede de entrega de conteúdo.
   a. Para entregar conteúdo usando locais da borda mais próximos ao usuário, comece entendendo como o CloudFront entrega conteúdo.
   b. Comece com uma distribuição simples do CloudFront. O CloudFront então sabe de onde você quer que o conteúdo seja entregue e os detalhes sobre como rastrear e gerenciar a entrega de conteúdo. É importante entender e considerar os aspectos a seguir ao configurar uma distribuição do CloudFront:
      i. Como funciona o armazenamento em cache com os pontos de presença do CloudFront
      ii. Aumentar a taxa de solicitações fornecidas diretamente de caches do CloudFront (taxa de acertos do cache)
      iii. Usar o Amazon CloudFront Origin Shield
      iv. Otimizar a alta disponibilidade com o failover de origem do CloudFront

5. Configure a proteção da camada da aplicação: o AWS WAF ajuda você a se proteger contra explorações e bots comuns da web que podem afetar a disponibilidade, comprometer a segurança ou consumir recursos em excesso. Para obter uma compreensão mais profunda, veja como o AWS WAF funciona e, quando estiver pronto para implementar proteções contra inundações HTTP POST E GET da camada de aplicações, consulte Getting started with AWS WAF (Conceitos básicos do AWS WAF). Também é possível usar o AWS WAF com o CloudFront. Consulte a documentação sobre como o AWS WAF funciona com os recursos do Amazon CloudFront.
6. Configure proteção adicional contra DDoS: por padrão, todos os clientes da AWS recebem proteção contra ataques de DDoS da camada de transporte e rede comuns e que ocorrem com mais frequência que visam seu site ou sua aplicação com o AWS Shield Standard sem custo adicional. Para proteção adicional de aplicações voltadas para a internet executadas no Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator e Amazon Route 53, considere o AWS Shield Advanced e consulte exemplos de arquiteturas resilientes a DDoS. Para proteger sua workload e seus endpoints públicos de ataques de DDoS, consulte Getting started with AWS Shield Advanced (Conceitos básicos do AWS Shield Advanced).

Recursos

Práticas recomendadas relacionadas:

- REL10-BP01 Implantar a workload em vários locais
- REL10-BP02 Escolher os locais apropriados para sua implantação de vários locais
- REL11-BP04 Confiar no plano de dados e não no ambiente de gerenciamento durante a recuperação
- REL11-BP06 Enviar notificações quando os eventos afetarem a disponibilidade

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede
- O que é o Reachability Analyzer?
- O que é o Reachability Analyzer?
- O que é o Amazon Route 53?
- O que é o Reachability Analyzer?

- Network Connectivity capability - Establishing Your Cloud Foundations (Recurso de conectividade de rede: como estabelecer as bases da nuvem)
- O que é o Amazon API Gateway?
- What are AWS WAF, AWS Shield, and AWS Firewall Manager? (O que são o AWS WAF, o AWS Shield e o AWS Firewall Manager?)
- O que é o Amazon Route 53 Application Recovery Controller?
- Configurar verificações de integridade personalizadas para failover de DNS

Vídeos relacionados:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator(AWS re:Invent 2022: melhore o desempenho e a disponibilidade com o AWS Global Accelerator)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53 (AWS re:Invent 2020: gerenciamento de tráfego global com o Amazon Route 53)
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications (AWS re:Invent 2022: operar aplicações Multi-AZ altamente disponíveis)
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure (AWS re:Invent 2022: aprofundamento na infraestrutura de rede da AWS)
- AWS re:Invent 2022 - Building resilient networks (AWS re:Invent 2022: criar redes resilientes)

Exemplos relacionados:

- Disaster Recovery with Amazon Route 53 Application Recovery Controller (ARC) (Recuperação de desastres com o Amazon Route 53 Application Recovery Controller (ARC))
- Reliability Workshops (Workshops sobre confiabilidade)
- Workshop sobre o AWS Global Accelerator

REL02-BP02 Provisionar conectividade redundante entre as redes privadas na nuvem e nos ambientes on-premises

Implemente redundância nas conexões entre redes privadas na nuvem e ambientes on-premises a fim de obter resiliência de conectividade. Isso pode ser feito por meio da implantação de dois ou mais links e caminhos de tráfego, preservando a conectividade em caso de falhas na rede.

Antipadrões comuns:

- Você depende de apenas uma conexão de rede, o que cria um ponto único de falha.
- Você usa somente um túnel VPN ou vários túneis que terminam na mesma zona de disponibilidade.
- Você depende de um ISP para conectividade VPN, o que pode levar a falhas completas durante interrupções do ISP.
- Não implementar protocolos de roteamento dinâmico, como o BGP, que são cruciais para redirecionar o tráfego durante interrupções na rede.
- Você ignora as limitações de largura de banda dos túneis VPN e superestima as respectivas capacidades de backup.

Benefícios de estabelecer esta prática recomendada: ao implementar conectividade redundante entre seu ambiente de nuvem e o ambiente corporativo ou on-premises, os serviços dependentes entre os dois ambientes podem se comunicar de forma confiável.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: alto

Orientações para a implementação

Ao usar o AWS Direct Connect para conectar sua rede on-premises à AWS, você pode alcançar a máxima resiliência da rede (SLA de 99,99%) utilizando conexões separadas que terminam em dispositivos distintos em mais de um ambiente on-premises e em mais de um local do AWS Direct Connect. Essa topologia oferece resiliência contra falhas de dispositivos, problemas de conectividade e interrupções de locais inteiros. Como alternativa, você pode obter alta resiliência (SLA de 99,9%) usando duas conexões individuais com vários locais (cada local on-premises conectado a um único local do Direct Connect). Essa abordagem oferece proteção contra interrupções de conectividade causadas por cortes na fibra ou falhas de dispositivos, além de ajudar a mitigar falhas de locais inteiros. O kit de ferramentas de resiliência do AWS Direct Connect pode ajudar a projetar sua topologia do AWS Direct Connect.

Você também pode considerar a terminação do AWS Site-to-Site VPN em um AWS Transit Gateway como um backup econômico para sua conexão primária do AWS Direct Connect. Essa configuração possibilita o roteamento multicaminho de custo igual (ECMP) em vários túneis VPN, permitindo um throughput de até 50 Gbps, mesmo que cada túnel VPN tenha um limite de 1,25 Gbps. No entanto, é importante observar que o AWS Direct Connect ainda é a opção mais eficaz para minimizar as interrupções na rede e oferecer conectividade estável.

Ao usar VPNs pela internet para conectar o ambiente de nuvem ao data center on-premises, configure dois túneis VPN como parte de uma única conexão do Site-to-Site VPN. Cada túnel deve terminar em uma zona de disponibilidade diferente para proporcionar alta disponibilidade, usar hardware redundante e evitar falhas no dispositivo on-premises. Além disso, considere várias conexões à internet de diversos provedores de serviços de Internet (ISPs) em seu local on-premises a fim de evitar a interrupção completa da conectividade VPN devido à interrupção de um único ISP. A seleção de ISPs com roteamento e infraestrutura diversos, especialmente aqueles com caminhos físicos separados até endpoints da AWS, fornece alta disponibilidade de conectividade.

Além da redundância física com várias conexões do AWS Direct Connect e vários túneis VPN (ou uma combinação de ambos), a implementação do roteamento dinâmico do Protocolo de Gateway da Borda (BGP) também é fundamental. O BGP dinâmico fornece redirecionamento automático do tráfego de um caminho para outro com base nas condições de rede em tempo real e nas políticas configuradas. Esse comportamento dinâmico é especialmente benéfico para manter a disponibilidade da rede e a continuidade do serviço em caso de falhas no link ou na rede. Ele seleciona rapidamente caminhos alternativos, aumentando a resiliência e a confiabilidade da rede.

Etapas da implementação

- Adquira conectividade de alta disponibilidade entre a AWS e seu ambiente on-premises.
  - Use várias conexões do AWS Direct Connect ou túneis VPN entre as redes privadas implantadas separadamente.
  - Use vários locais do AWS Direct Connect para gerar alta disponibilidade.
  - Se estiver usando várias Regiões da AWS, crie redundância em pelo menos duas delas.
- Sempre que possível, use o AWS Transit Gateway para a terminação de sua conexão VPN.
- Avalie os dispositivos do AWS Marketplace para a terminação das VPNs ou a extensão de sua SD-WAN para a AWS. Se você usa appliances do AWS Marketplace, implante instâncias redundantes em zonas de disponibilidade diferentes para alta disponibilidade.
- Forneça uma conexão redundante com o ambiente on-premises.
  - Você pode precisar de conexões redundantes com várias Regiões da AWS para atender às suas necessidades de disponibilidade.
  - Use o kit de ferramentas de resiliência do AWS Direct Connect para começar.

Recursos

Documentos relacionados:

- AWS Direct Connect Resiliency Recommendations
- Using Redundant Site-to-Site VPN Connections to Provide Failover
- Routing policies and BGP communities (“Políticas de roteamento e comunidades BGP”)
- Active/Active and Active/Passive Configurations in AWS Direct Connect
- Parceiro do APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure (Criação de uma infraestrutura de rede da AWS de várias VPCs escaláveis e seguras)
- Using redundant Site-to-Site VPN connections to provide failover
- Using the AWS Direct Connect Resiliency Toolkit to get started
- Endpoints da VPC e serviços de endpoint da VPC (AWS PrivateLink)
- O que é Amazon VPC?
- What is a transit gateway?
- What is AWS Site-to-Site VPN?
- Working with Direct Connect gateways

Vídeos relacionados:

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

REL02-BP03 Garantir contas de alocação de sub-rede IP para expansão e disponibilidade

Intervalos de endereços IP da Amazon VPC devem ser grandes o suficiente para acomodar os requisitos da workload, incluindo a futura expansão e alocação de endereços IP para sub-redes nas zonas de disponibilidade. Isso inclui load balancers, instâncias do EC2 e aplicativos baseados em contêiner.

Ao planejar sua topologia de rede, a primeira etapa é definir o espaço do endereço IP em si. Intervalos de endereços IP privados (seguindo as diretrizes RFC 1918) devem ser alocados para cada VPC. Atenda aos seguintes requisitos como parte desse processo:

- Permitir espaço de endereço IP para mais de uma VPC por Região.
- Em uma VPC, deixe espaço para várias sub-redes para cobrir várias zonas de disponibilidade.
- Considere deixar o espaço de bloco CIDR não utilizado em uma VPC para futura expansão.
- Verifique se há espaço de endereço IP para atender às necessidades de qualquer frota transitória de instâncias do Amazon EC2 que você use, como frotas spot para machine learning, clusters do Amazon EMR ou clusters do Amazon Redshift. Consideração semelhante deve ser dada aos clusters do Kubernetes, como o Amazon Elastic Kubernetes Service (Amazon EKS), pois cada pod do Kubernetes recebe um endereço roteável do bloco CIDR da VPC por padrão.
- Observe que os primeiros quatro endereços IP e o último endereço IP em cada bloco CIDR da sub-rede estão reservados e não estão disponíveis para seu uso.
- Observe que o bloco CIDR inicial da VPC alocado para sua VPC não pode ser alterado ou excluído, mas você pode adicionar blocos CIDR não sobrepostos à VPC. Os CIDRs IPv4 da sub-rede não podem ser alterados, mas os CIDRs IPv6 podem.
- O maior bloco CIDR de VPC possível é /16 e o menor é /28.
- Considere outras redes conectadas (VPC, on-premises ou outros provedores de nuvem) e garanta que o espaço de endereço IP não se sobreponha. Para mais informações, leia REL02-BP05 Aplicar intervalos de endereços IP privados não sobrepostos a todos os espaços de endereços privados onde estão conectados

Resultado desejado: uma sub-rede IP escalável pode ajudar você a se adaptar ao crescimento futuro e evitar desperdícios desnecessários.

Antipadrões comuns:

- deixar de considerar o crescimento futuro, levando a blocos CIDR muito pequenos que exigem reconfiguração e causando um possível tempo de inatividade.
- Estimar incorretamente quantos endereços IP um Elastic Load Balancer pode usar.
- Implantar muitos balanceadores de carga de alto tráfego nas mesmas sub-redes.
- Usar mecanismos de ajuste de escala automático automatizados sem monitorar o consumo de endereços IP.
- Definir intervalos CIDR excessivamente grandes muito além das expectativas de crescimento futuro, o que pode dificultar o emparelhamento com outras redes com intervalos de endereços sobrepostos.

Benefícios de estabelecer esta prática recomendada: Isso garante que você possa acomodar o crescimento das suas cargas de trabalho e continuar a fornecer disponibilidade à medida que elas se expandem.

Nível de risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

Planeje sua rede para acomodar crescimento, conformidade regulamentar e integração com outras pessoas. O crescimento pode ser subestimado, a conformidade regulamentar pode mudar e as aquisições ou conexões de rede privada podem ser difíceis de implementar sem o planejamento adequado.

- Selecione as Contas da AWS e regiões relevantes conforme seus requisitos de serviço, de latência, regulatórios e de recuperação de desastres (DR).
- Identifique suas necessidades para implantações regionais de VPC.
- Identifique o tamanho das VPCs.
  - Determine se você pretende implantar conectividade com várias VPCs.
    - O que é um Transit Gateway?
    - Conectividade com várias VPCs de região única
  - Determine se você precisa de rede segregada por requisitos normativos.
  - Crie VPCs com blocos CIDR de tamanho adequado para acomodar suas necessidades atuais e futuras.
    - Se você tiver projeções de crescimento desconhecidas, talvez prefira arriscar blocos CIDR maiores para reduzir a possibilidade de reconfiguração futura
  - Considere o uso de soluções de endereçamento IPv6 para sub-redes como parte de uma VPC de pilha dupla. O IPv6 é adequado para sub-redes privadas contendo frotas de instâncias ou contêineres efêmeros que, de outra forma, exigiriam um grande número de endereços IPv4.

Recursos

Práticas recomendadas relacionadas ao Well-Architected:

- REL02-BP05 Aplicar intervalos de endereços IP privados não sobrepostos a todos os espaços de endereços privados onde estão conectados

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- Conectividade com várias VPCs de região única
- O que é o Amazon VPC?
- IPv6 na AWS
- IPv6 on reference architectures
- Amazon Elastic Kubernetes Service launches IPv6 support

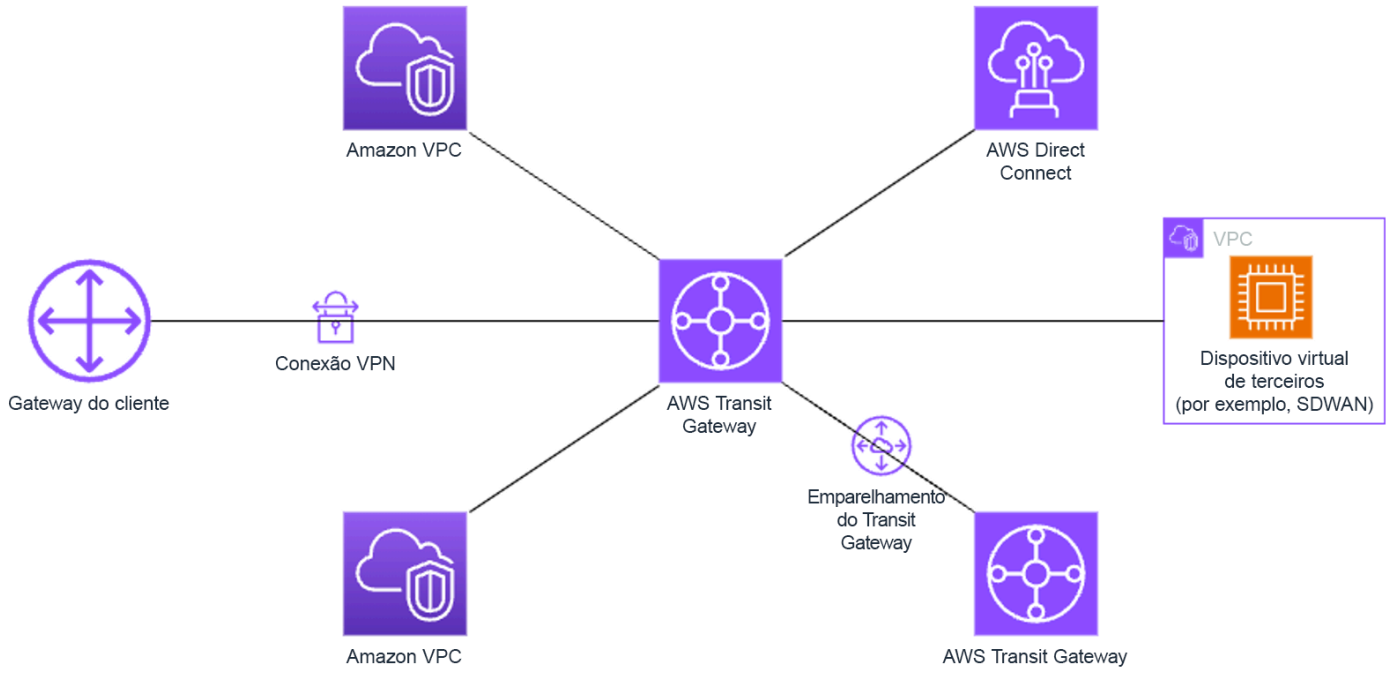
Vídeos relacionados:

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

REL02-BP04 Preferir topologias hub-and-spoke em vez da malha muitos para muitos

Ao conectar várias redes privadas, como nuvens privadas virtuais (VPCs) e redes on-premises, opte por uma topologia hub-and-spoke em vez de uma em malha. Diferentemente das topologias em malha, em que cada rede se conecta diretamente às outras e aumenta a complexidade e as despesas indiretas de gerenciamento, a arquitetura hub-and-spoke centraliza as conexões por meio de um único hub. Essa centralização simplifica a estrutura da rede e aprimora a operabilidade, a escalabilidade e o controle.

O AWS Transit Gateway é um serviço gerenciado, escalável e de alta disponibilidade projetado para a construção de redes hub-and-spoke na AWS. Ele serve como o hub central da rede que fornece segmentação de rede, roteamento centralizado e conexão simplificada com ambientes on-premises e de nuvem. A figura a seguir ilustra como você pode usar o AWS Transit Gateway para criar a topologia hub-and-spoke.



Antipadrões comuns:

- Você complica demais as políticas de roteamento em uma arquitetura hub-and-spoke, o que reduz a eficiência da rede e complica tanto a solução de problemas quanto o gerenciamento proativo.
- A segmentação insuficiente baseada em roteamento dentro do hub pode causar vulnerabilidades, o que possivelmente expõe a rede ao acesso não autorizado.
- Sem uma otimização cuidadosa, o tráfego roteado pelo hub pode gerar maiores custos de transferência de dados, especialmente para tráfego entre zonas de disponibilidade e regiões. Estratégias eficazes de gerenciamento de tráfego são essenciais para controlar as despesas.

Benefícios de estabelecer esta prática recomendada: à medida que o número de redes conectadas aumenta, mais desafiadores se tornam o gerenciamento e a expansão da conectividade em malha. O AWS Transit Gateway oferece um hub gerenciado escalável e confiável para construção e operação de topologias hub-and-spoke. Ao usar o AWS Transit Gateway, é possível estabelecer conexões e centralizar o roteamento de tráfego em várias redes.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

- Planeje sua rede.
- Crie o AWS Transit Gateway.
- Conecte as VPCs.
- Se necessário, crie conexões VPN ou gateways Direct Connect e associe-os ao Transit Gateway.
- Defina como o tráfego é direcionado entre as VPCs conectadas e outras conexões por meio da configuração das tabelas de rotas do Transit Gateway.
- Use o Amazon CloudWatch para monitorar e ajustar as configurações conforme necessário para otimizar a performance e os custos.

Recursos

Documentos relacionados:

- What Is a Transit Gateway?
- Construção de uma infraestrutura de rede da AWS escalável e segura com várias VPCs
- Building a global network using AWS Transit Gateway Inter-Region peering
- Opções de conectividade da Amazon Virtual Private Cloud
- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure

Vídeos relacionados:

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

REL02-BP05 Aplicar intervalos de endereços IP privados não sobrepostos a todos os espaços de endereços privados onde estão conectados

Os intervalos de endereços IP de cada uma das VPCs não devem se sobrepor quando emparelhados, conectados via Transit Gateway ou conectados por VPN. Evite conflitos de endereço IP entre uma VPC e ambientes on-premises ou com outros provedores de nuvem que você

usa. Você também deve ter uma maneira de alocar intervalos de endereços IP privados quando necessário. Um sistema de gerenciamento de endereços IP (IPAM) pode ajudar a automatizar isso.

Resultado desejado:

- Não há nenhum conflito de intervalo de endereços IP entre VPCs, ambientes on-premises ou outros provedores de nuvem.
- O gerenciamento adequado de endereços IP facilita a escalabilidade da infraestrutura de rede para atender ao crescimento e às mudanças nos requisitos de rede.

Antipadrões comuns:

- Usar o mesmo intervalo de IPs na VPC que você tem on-premises, na rede corporativa ou em outros provedores de nuvem.
- Não acompanhar os intervalos IPs das VPCs usadas para implantar suas workloads.
- Depender de processos manuais de gerenciamento de endereços IP, como planilhas.
- Superdimensionar ou subdimensionar blocos CIDR, o que resulta em desperdício de endereços IP ou espaço de endereços insuficiente para a workload.

Benefícios de estabelecer esta prática recomendada: o planejamento ativo da rede garantirá que você não tenha várias ocorrências do mesmo endereço IP nas redes interconectadas. Isso evita que problemas de roteamento ocorram em partes da carga de trabalho que usam os diferentes aplicativos.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

Use um IPAM, como o Amazon VPC IP Address Manager, para monitorar e gerenciar o uso de CIDR. Vários IPAMs estão disponíveis no AWS Marketplace. Avalie seu uso potencial na AWS, adicione intervalos de CIDR às VPCs existentes e crie VPCs para permitir um crescimento planejado no uso.

Etapas da implementação

- Colete o consumo atual de CIDR (por exemplo, VPCs e sub-redes).
  - Use as operações de API de serviço para coletar o consumo atual de CIDR.
  - Use o Amazon VPC IP Address Manager para descobrir recursos.

- Capture seu uso atual de sub-rede.
  - Use as operações de API de serviço para coletar sub-redes por VPC em cada região.
  - Use o Amazon VPC IP Address Manager para descobrir recursos.
- Registre o uso atual.
- Determine se você criou algum intervalo de IPs sobreposto.
- Calcule a capacidade não utilizada.
- Identifique intervalos de IP sobrepostos. Você pode migrar para um novo intervalo de endereços ou considerar o uso de técnicas, como o gateway NAT privado ou o AWS PrivateLink, se precisar conectar os intervalos sobrepostos.

Recursos

Práticas recomendadas relacionadas:

- Proteção de redes

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- Connecting Networks with Overlapping IP Ranges
- O que é Amazon VPC?
- O que é o IPAM?

Vídeos relacionados:

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

Arquitetura da carga de trabalho

Uma carga de trabalho confiável começa com as decisões iniciais de projeto que envolvem tanto o software quanto a infraestrutura. Suas escolhas de arquitetura afetarão o comportamento da sua workload em todos os seis pilares do Well-Architected. Para atingir a confiabilidade, há padrões específicos que você deve seguir.

As seções a seguir explicam as melhores práticas a serem usadas com esses padrões em relação à confiabilidade.

Tópicos

Projetar sua arquitetura de serviço da workload

Use uma Service-Oriented Architecture (SOA – Arquitetura orientada por serviços) ou uma arquitetura de microsserviços para criar cargas de trabalho altamente escaláveis e confiáveis. A SOA é a prática de tornar componentes de software reutilizáveis por meio de interfaces de serviço. A arquitetura de microsserviços vai além para tornar os componentes menores e mais simples.

As interfaces SOA usam padrões de comunicação comuns para que possam ser rapidamente incorporadas a novas cargas de trabalho. A SOA substituiu a prática de construção de arquiteturas monolíticas, que consistiam em unidades interdependentes e indivisíveis.

Na AWS, sempre usamos a SOA, mas agora adotamos a criação de nossos sistemas usando microsserviços. Embora os microsserviços tenham várias qualidades interessantes, o principal benefício para disponibilidade é que eles são menores e mais simples. Eles permitem diferenciar a disponibilidade exigida de diferentes serviços e, portanto, concentrar os investimentos mais especificamente nos microsserviços que têm as maiores necessidades de disponibilidade. Por exemplo, para entregar páginas de informações do produto em Amazon.com (“páginas de detalhes”), centenas de microsserviços são invocados para criar partes separadas da página. Embora haja alguns microsserviços que precisem estar disponíveis para fornecer o preço e os detalhes do produto, a grande maioria do conteúdo na página poderá simplesmente ser excluída se o serviço não estiver disponível. Mesmo itens como fotos e avaliações não são necessários para proporcionar uma experiência em que o cliente possa comprar um produto.

Práticas recomendadas

- REL03-BP01 Escolher como segmentar a workload
- REL03-BP02 Criar serviços enfocados em domínios e funcionalidades de negócios específicos
- REL03-BP03 Fornecer contratos de serviço por API

REL03-BP01 Escolher como segmentar a workload

A segmentação de workloads é importante ao determinar os requisitos de resiliência de sua aplicação. Uma arquitetura monolítica deve ser evitada sempre que possível. Em vez disso, considere cuidadosamente quais componentes da aplicação podem ser distribuídos em microsserviços. Dependendo dos requisitos de sua aplicação, isso pode acabar sendo uma combinação de uma arquitetura orientada a serviços (SOA) com microsserviços sempre que possível. Workloads com capacidade para serem do tipo sem estado têm maior chance de serem implantadas como microsserviços.

Resultado desejado: as workloads devem ser compatíveis, escaláveis e o mais vagamente agrupadas possível.

Ao tomar decisões sobre como segmentar uma workload, pondere os benefícios e as complexidades. O que é ideal para um novo produto a caminho do seu primeiro lançamento não se aplica a uma workload que foi criada para escalabilidade a partir das necessidades iniciais. Ao refatorar um monólito existente, você vai precisar considerar o quanto a aplicação vai oferecer um bom suporte a uma decomposição em direção à condição sem estado. A divisão dos serviços em pedaços menores permite que equipes pequenas e bem definidas os desenvolvam e gerenciem. No entanto, serviços menores podem introduzir complexidades que incluem maior latência potencial, depuração mais complexa e carga operacional aumentada.

Antipadrões comuns:

- O microsserviço Death Star é uma situação em que os componentes atômicos se tornam tão altamente interdependentes que a falha de um resulta em uma falha muito maior, o que torna os componentes tão rígidos e frágeis quanto um monólito.

Benefícios do estabelecimento desta prática:

- Mais segmentos específicos geram maior agilidade, flexibilidade organizacional e escalabilidade.
- Redução do impacto das interrupções do serviço.

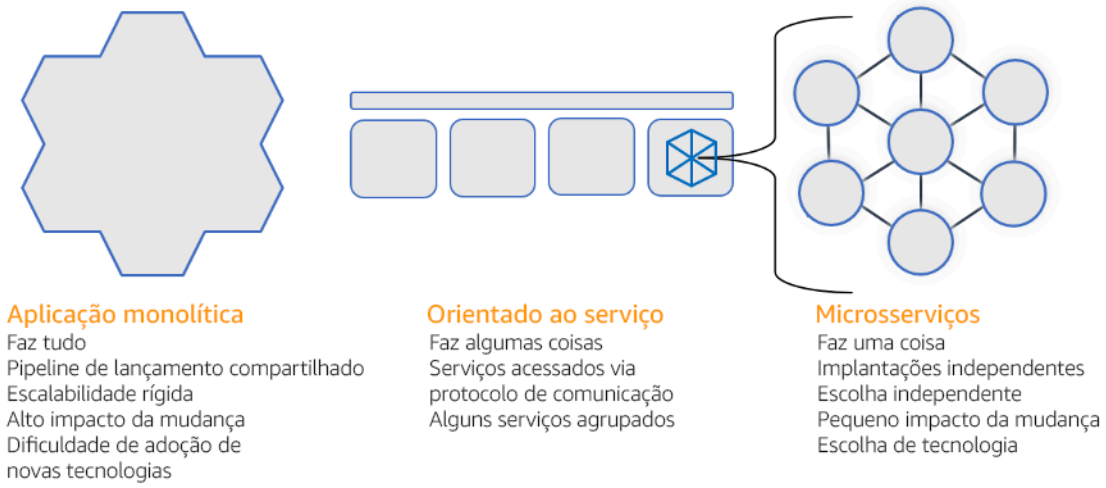
- Os componentes da aplicação podem ter requisitos de disponibilidade diferentes, aos quais uma segmentação mais atômica pode oferecer suporte.
- Responsabilidades bem definidas para as equipes que oferecem suporte à workload.

Nível de risco exposto se esta prática recomendada não for estabelecida: Alto

Orientação de implementação

Escolha o tipo de arquitetura com base no modo como você segmentará a workload. Escolha uma SOA ou arquitetura de microsserviços (ou, em alguns casos, uma arquitetura monolítica). Mesmo que você opte por começar com uma arquitetura monolítica, você deve garantir que ela seja modular e tenha a capacidade de evoluir para SOA ou microsserviços à medida que o produto escala com a adoção do usuário. A SOA e os microsserviços oferecem, respectivamente, segmentação menor, que é preferida como uma arquitetura moderna escalável e confiável, mas há compensações a serem consideradas, especialmente ao implantar uma arquitetura de microsserviços.

Uma compensação primária é que você agora tem uma arquitetura de computação distribuída que pode tornar mais difícil alcançar requisitos de latência do usuário final, e há complexidade adicional na depuração e no rastreamento de interações com o usuário. Use o AWS X-Ray para ajudar você a resolver esse problema. Outro efeito a ser considerado é o aumento da complexidade operacional à medida que você aumenta o número de aplicações que está gerenciando, o que requer a implantação de vários componentes de independência.



Arquiteturas monolítica, orientada a serviços e de microsserviços

Etapas da implementação

- Determine a arquitetura adequada para refatorar ou desenvolver sua aplicação. A SOA e os microsserviços oferecem respectivamente segmentação menor, que é preferida por ser uma arquitetura moderna escalável e confiável. A SOA pode ser o meio-termo ideal para alcançar uma segmentação menor e também evitar algumas das complexidades dos microsserviços. Para obter mais detalhes, consulte Compensações de microsserviços.
- Se sua carga de trabalho aceitá-la e sua organização puder sustentá-la, use uma arquitetura de microsserviços para obter a melhor agilidade e confiabilidade. Para obter mais detalhes, consulte Implementação de microsserviços na AWS.
- Considere seguir o padrão Strangler Fig para refatorar um monólito em componentes menores. Isso envolve a substituição gradual de componentes específicos da aplicação por novas aplicações e serviços. AWS Migration Hub Refactor Spaces atua como um ponto de partida para refatoração incremental. Para obter mais detalhes, consulte Migração simplificada de workloads on-premises herdadas usando um padrão strangler.
- A implementação de microsserviços pode exigir um mecanismo de descoberta de serviços para permitir que esses serviços distribuídos se comuniquem entre si. AWS App Mesh pode ser usado com arquiteturas orientadas por serviços para fornecer descoberta confiável e acesso a serviços. AWS Cloud Map também pode ser usado para descoberta dinâmica de serviços baseada em DNS.
- Se você estiver migrando de um monólito para SOA, Amazon MQ pode ajudar a eliminar a lacuna como um barramento de serviço ao reprojetar aplicações herdadas na nuvem.
- Para monólitos existentes com um único banco de dados compartilhado, escolha como reorganizar os dados em segmentos menores. Isso pode acontecer por unidade de negócios, padrão de acesso ou estrutura de dados. A esta altura no processo de refatoração, escolha se deseja prosseguir com um banco de dados relacional ou não relacional (NoSQL). Para obter mais detalhes, consulte De SQL para NoSQL.

Nível de esforço do plano de implementação: Alto

Recursos

Práticas recomendadas relacionadas:

- REL03-BP02 Criar serviços enfocados em domínios e funcionalidades de negócios específicos

Documentos relacionados:

- Amazon API Gateway: configurar uma API REST usando o OpenAPI
- O que é arquitetura orientada a serviços?
- Contexto delimitado (um padrão central no design orientado por domínio)
- Implementação de microsserviços na AWS
- Compensações de microsserviços
- Microsserviços - uma definição desse novo termo de arquitetura
- Microsserviços na AWS
- O que é o AWS App Mesh?

Exemplos relacionados:

- Workshop de modernização iterativa de aplicações

Vídeos relacionados:

- Delivering Excellence with Microservices on AWS (Entregando excelência com microsserviços na AWS)

REL03-BP02 Criar serviços enfocados em domínios e funcionalidades de negócios específicos

A arquitetura orientada a serviços (SOA) define serviços com funções bem delineadas que seguem as necessidades dos negócios. Os microsserviços usam modelos de domínio e contexto delimitado para traçar limites de serviço ao longo dos limites do contexto de negócios. O foco nos domínios de negócios e na funcionalidade ajuda as equipes a definir requisitos independentes de confiabilidade para seus serviços. Contextos delimitados isolam e encapsulam a lógica de negócios, permitindo que as equipes raciocinem melhor sobre como lidar com falhas.

Resultado desejado: Em conjunto, engenheiros e partes interessadas do negócio definem contextos delimitados e os usam para projetar sistemas como serviços que cumprem funções empresariais específicas. Essas equipes usam práticas estabelecidas, como Event Storming, para definir os requisitos. As novas aplicações são projetadas como serviços, limites bem definidos e acoplamento fraco. Os monólitos existentes são decompostos em contextos delimitados e os projetos de sistemas migram para arquiteturas SOA ou de microsserviços. Quando os monólitos são refatorados,

abordagens estabelecidas, como contextos de bolha e padrões de decomposição de monólitos, são aplicadas.

Os serviços orientados a domínios são executados como um ou mais processos que não compartilham o estado. Eles respondem de forma independente às flutuações na demanda e lidam com cenários de falha à luz dos requisitos específicos do domínio.

Antipadrões comuns:

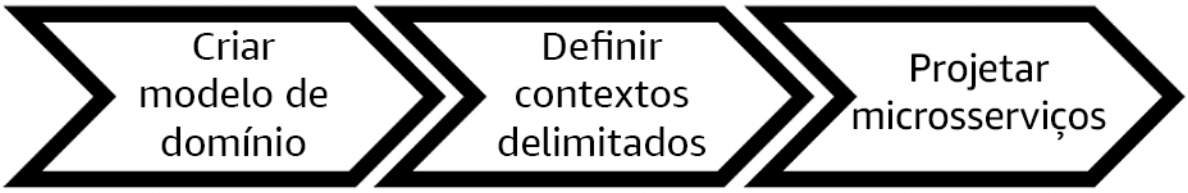
- As equipes são formadas em torno de domínios técnicos específicos, como UI e UX, middleware ou banco de dados, em vez de domínios empresariais específicos.
- As aplicações abrangem as responsabilidades do domínio. Serviços que abrangem contextos delimitados podem ser mais difíceis de manter, exigir maiores esforços de teste e que várias equipes de domínio participem das atualizações de software.
- As dependências de domínio, como as bibliotecas de entidades de domínio, são compartilhadas entre serviços, de forma que as alterações em um domínio de serviço exijam alterações em outros domínios de serviço.
- Os contratos de serviço e a lógica de negócios não expressam entidades em uma linguagem de domínio comum e consistente, ocasionando camadas de tradução que complicam os sistemas e aumentam os esforços de depuração.

Benefícios de estabelecer esta prática recomendada: As aplicações são projetadas como serviços independentes delimitados por domínios de negócios e usam uma linguagem comercial comum. Os serviços podem ser testados e implantados de forma independente. Os serviços atendem aos requisitos de resiliência específicos do domínio implementado.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

A decisão orientada por domínio (DDD) é a abordagem fundamental para projetar e criar software em torno de domínios empresariais. É útil trabalhar com uma framework existente ao criar serviços enfocados em domínios empresariais. Ao trabalhar com aplicações monolíticas existentes, você pode utilizar os padrões de decomposição que fornecem técnicas estabelecidas para modernizar aplicações em serviços.



Decisão orientada por domínio

Etapas da implementação

- As equipes podem realizar workshops de Event Storming a fim de identificar rapidamente eventos, comandos, agregados e domínios em um formato leve de notas adesivas.
- Depois que as entidades e as funções do domínio forem formadas em um contexto de domínio, você poderá dividir seu domínio em serviços usando contexto delimitado, em que entidades que compartilham recursos e atributos semelhantes são agrupadas. Com o modelo dividido em contextos, surge um modelo de como delimitar microsserviços.
  - Por exemplo, as entidades do site Amazon.com podem incluir pacote, entrega, programação, preço, desconto e moeda.
  - Pacote, entrega e cronograma são agrupados no contexto de envio, enquanto preço, desconto e moeda são agrupados no contexto de preços.
- Decompor monólitos em microsserviços descreve padrões para refatorar microsserviços. O uso de padrões para decomposição por capacidade comercial, subdomínio ou transação se alinha bem às abordagens orientadas por domínio.
- Técnicas táticas, como o contexto de bolha, permitem introduzir o DDD em aplicações existentes ou legadas sem reformulações antecipadas e compromissos totais com o DDD. Em uma abordagem de contexto de bolha, um pequeno contexto delimitado é estabelecido usando um mapeamento e coordenação de serviços, ou camada anticorrupção, que protege o modelo de domínio recém-definido de influências externas.

Depois que as equipes realizarem a análise de domínio e definirem entidades e contratos de serviço, elas podem utilizar os serviços da AWS para implementar o design orientado por domínio como serviços baseados em nuvem.

- Comece o desenvolvimento definindo testes que exercitem as regras de negócios de seu domínio. O desenvolvimento orientado por testes (TDD) e o desenvolvimento orientado por comportamento (BDD) ajudam as equipes a manter os serviços enfocados na solução de problemas de negócios.
- Selecione os serviços da AWS que mais bem atendam aos requisitos de domínio de sua empresa e arquitetura de microsserviços:
  - tecnologia sem servidor da AWS permite que sua equipe enfoque a lógica de domínio específica em vez de gerenciar servidores e infraestrutura.
  - Contêineres na AWS simplificam o gerenciamento de sua infraestrutura para que você possa enfocar nos requisitos de domínio.
  - Bancos de dados com propósito específico ajudam você a adequar seus requisitos de domínio ao tipo de banco de dados mais adequado.
- Criação de arquiteturas hexagonais na AWS descreve uma framework para criar lógica de negócios em serviços que funcionam retroativamente a partir de um domínio empresarial para atender aos requisitos funcionais e, depois, conectar adaptadores de integração. Os padrões que separam os detalhes da interface da lógica de negócios com serviços da AWS ajudam as equipes a enfocar na funcionalidade do domínio e melhorar a qualidade do software.

Recursos

Práticas recomendadas relacionadas:

- REL03-BP01 Escolher como segmentar a workload
- REL03-BP03 Fornecer contratos de serviço por API

Documentos relacionados:

- AWS Microsserviços
- Implementação de microsserviços na AWS
- How to break a Monolith into Microservices (Como dividir um monólito em microsserviços)
- Getting Started with DDD when Surrounded by Legacy Systems (Conceitos básicos do DDD quando cercado por sistemas herdados)
- Domain-Driven Design: Tackling Complexity in the Heart of Software (Design orientado por domínio: como lidar com a complexidade no núcleo do software)
- Criação de arquiteturas hexagonais na AWS

- Decompor monólitos em microsserviços
- Event Storming
- Mensagens entre contextos delimitados
- Microsserviços
- Desenvolvimento orientado por testes
- Desenvolvimento orientado pelo comportamento

Exemplos relacionados:

- Workshop nativo da nuvem corporativa
- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop) (Como projetar microsserviços nativos em nuvem na AWS (do DDD/EventStormingWorkshop))

Ferramentas relacionadas:

- Bancos de dados da Nuvem AWS
- Tecnologia sem servidor na AWS
- Contêineres na AWS

REL03-BP03 Fornecer contratos de serviço por API

Os contratos de serviço são acordos documentados entre produtores e consumidores de API estabelecidos em uma definição de API legível por máquina. Uma estratégia de versionamento de contrato permite que os consumidores continuem usando a API existente e migrem suas aplicações para uma API mais recente quando estiverem prontos. A implantação do produtor pode acontecer a qualquer momento, desde que o contrato seja cumprido. A equipe de serviços pode usar a pilha de tecnologia de sua preferência para cumprir o contrato de API.

Resultado desejado:

Antipadrões comuns: As aplicações criadas com arquiteturas orientadas a serviços ou microsserviços podem operar de forma independente e, ao mesmo tempo, ter uma dependência de runtime integrada. As alterações implantadas em um consumidor ou produtor de API não interrompem a estabilidade do sistema geral quando os dois lados seguem um contrato de API comum. Os componentes que se comunicam por meio de APIs de serviço podem realizar

lançamentos funcionais independentes, atualizações para dependências de runtime ou fazer failover em um site de recuperação de desastres (DR) com pouco ou nenhum impacto entre si. Além disso, serviços diferentes são capazes de escalar de forma independente a absorção da demanda de recursos sem exigir que outros serviços escalem simultaneamente.

- Criar APIs de serviço sem esquemas altamente tipificados. Isso ocasiona APIs que não podem ser usadas para gerar vinculações de API e payloads que não possam ser validadas de maneira programática.
- Não adotar uma estratégia de versionamento, o que força os consumidores de API a atualizarem e lançarem ou falharem com a evolução dos contratos de serviço.
- Mensagens de erro que vazam detalhes da implementação do serviço subjacente em vez de descreverem falhas de integração no contexto e no idioma do domínio.
- Não usar contratos de API para desenvolver casos de teste e simular implementações de API para permitir testes independentes dos componentes do serviço.

Benefícios de estabelecer esta prática recomendada: Sistemas distribuídos compostos por componentes que se comunicam por meio de contratos de serviço de API podem aumentar a confiabilidade. Os desenvolvedores podem detectar possíveis problemas no início do processo de desenvolvimento com a verificação de tipo durante a compilação a fim de verificar se as solicitações e as respostas seguem o contrato da API e se os campos obrigatórios estão presentes. Os contratos de API oferecem uma interface clara de autodocumentação de APIs e oferecem melhor interoperabilidade entre diferentes sistemas e linguagens de programação.

Nível de risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

Depois de identificar os domínios de negócios e determinar a segmentação da workload, você pode desenvolver suas APIs de serviço. Primeiro, defina contratos de serviço legíveis por máquina para APIs e, depois, implemente uma estratégia de versionamento de API. Quando estiver pronto para integrar serviços em protocolos comuns, como REST, GraphQL ou eventos assíncronos, você poderá incorporar serviços da AWS à sua arquitetura para integrar seus componentes com contratos de API altamente tipificados.

Serviços da AWS para contratos de API de serviços

Incorpore serviços da AWS, incluindo Amazon API Gateway, o AWS AppSynce o Amazon EventBridge à sua arquitetura para usar contratos de serviço de API em sua aplicação. O Amazon

API Gateway ajuda você a se integrar diretamente a serviços nativos da AWS e outros serviços da web. O API Gateway é compatível com a especificação da OpenAPI e versionamento. O AWS AppSync é um endpoint GraphQL gerenciado que você configura definindo um esquema GraphQL para definir uma interface de serviço para consultas, mutações e assinaturas. O Amazon EventBridge usa esquemas de eventos para definir eventos e gerar associações de código para seus eventos.

Etapas da implementação

- Primeiro, defina um contrato para sua API. Um contrato expressará os recursos de uma API, bem como definirá objetos e campos de dados altamente tipificados para a entrada e a saída da API.
- Ao configurar APIs no API Gateway, você pode importar e exportar especificações da OpenAPI para seus endpoints.
  - A importação de uma definição de OpenAPI simplifica a criação de sua API e pode ser integrada a ferramentas de infraestrutura como código da AWS, como o AWS Serverless Application Model e o AWS Cloud Development Kit (AWS CDK).
  - A exportação de uma definição de API simplifica a integração a ferramentas de teste de API e oferece ao consumidor de serviços uma especificação de integração.
- Você pode definir e gerenciar APIs do GraphQL com o AWS AppSync definindo um arquivo de esquema GraphQL para gerar sua interface de contrato e simplificar a interação com modelos REST complexos, várias tabelas de banco de dados ou serviços legados.
- Projetos do AWS Amplify integrados ao AWS AppSync geram arquivos de consulta JavaScript altamente tipificados para uso em sua aplicação, bem como uma biblioteca cliente do AWS AppSync GraphQL para tabelas do Amazon DynamoDB .
- Quando você consome eventos de serviço do Amazon EventBridge, eles seguem os esquemas já existentes no registro do esquema ou os definidos com a especificação da OpenAPI. Com um esquema definido no registro, também é possível gerar vinculações de cliente a partir do contrato de esquema para integrar seu código aos eventos.
- Estender ou realizar o versionamento de sua API. Estender uma API é uma opção mais simples ao adicionar campos que podem ser configurados com campos opcionais ou valores padrão para campos obrigatórios.
  - Contratos baseados em JSON para protocolos, como REST e GraphQL, podem ser uma boa opção para a extensão do contrato.
  - Contratos baseados em XML para protocolos, como SOAP, devem ser testados com consumidores de serviços para determinar a viabilidade da extensão do contrato.

- Ao realizar o versionamento de uma API, considere implementar o controle de versão por procuração em que uma fachada é usada para oferecer compatibilidade com versões para que a lógica possa ser mantida em uma única base de código.
  - Com o API Gateway, você pode usar mapeamentos de solicitações e respostas para simplificar a absorção de alterações no contrato estabelecendo uma fachada para fornecer valores padrão para novos campos ou para retirar os campos removidos de uma solicitação ou resposta. Com essa abordagem, o serviço subjacente pode manter uma única base de código.

Recursos

Práticas recomendadas relacionadas:

- REL03-BP01 Escolher como segmentar a workload
- REL03-BP02 Criar serviços enfocados em domínios e funcionalidades de negócios específicos
- REL04-BP02 Implementar dependências com acoplamento fraco
- REL05-BP03 Controlar e limitar as chamadas de repetição
- REL05-BP05 Definir tempos limite do cliente

Documentos relacionados:

- O que é uma API (interface de programação de aplicações)?
- Implementação de microsserviços na AWS
- Compensações de microsserviços
- Microsserviços - uma definição desse novo termo de arquitetura
- Microsserviços na AWS
- Trabalhar com extensões do API Gateway para OpenAPI
- Especificação da OpenAPI
- GraphQL: esquemas e tipos
- Vinculações de código do Amazon EventBridge

Exemplos relacionados:

- Amazon API Gateway: configurar uma API REST usando o OpenAPI
- Amazon API Gateway para a aplicação CRUD Amazon DynamoDB usando OpenAPI

- Padrões modernos de integração de aplicações em uma era sem servidor: integração do serviço API Gateway
- Implementar o versionamento baseado em cabeçalho do API Gateway com Amazon CloudFront
- AWS AppSync: criar uma aplicação cliente

Vídeos relacionados:

- Usar a OpenAPI no AWS SAM para gerenciar o API Gateway

Ferramentas relacionadas:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge

Projete as interações em um sistema distribuído para evitar falhas

Os sistemas distribuídos dependem das redes de comunicação para interconectar componentes, como servidores ou serviços. Sua carga de trabalho deve operar de forma confiável, apesar da perda de dados ou da latência nessas redes. Os componentes do sistema distribuído devem operar sem afetar negativamente outros componentes ou a carga de trabalho. Essas melhores práticas evitam falhas e melhoram o Mean Time Between Failures (MTBF – Tempo médio entre falhas).

Práticas recomendadas

- REL04-BP01 Identificar qual tipo de sistema distribuído é necessário
- REL04-BP02 Implementar dependências com acoplamento fraco
- REL04-BP03 Fazer um trabalho constante
- REL04-BP04 Fazer com que todas as respostas sejam idempotentes

REL04-BP01 Identificar qual tipo de sistema distribuído é necessário

Os sistemas distribuídos podem ser síncronos, assíncronos ou em lote. Os sistemas síncronos devem processar solicitações o mais rápido possível e se comunicar entre si fazendo chamadas síncronas de solicitação e resposta usando protocolos HTTP/S, REST ou de chamada de procedimento remoto (RPC). Os sistemas assíncronos se comunicam entre si trocando dados

de forma assíncrona por meio de um serviço intermediário sem acoplar sistemas individuais. Os sistemas em lote recebem um grande volume de dados de entrada, executam processos de dados automatizados sem intervenção humana e geram dados de saída.

Resultado desejado: criar uma workload que interaja efetivamente com dependências síncronas, assíncronas e em lote.

Antipadrões comuns:

- A workload espera indefinidamente por uma resposta de suas dependências, o que pode fazer com que os clientes da workload esgotem o tempo limite, sem saber se a solicitação foi recebida.
- A workload usa uma cadeia de sistemas dependentes que chamam um ao outro de forma síncrona. Para que toda a cadeia tenha êxito, isso exige primeiro que cada sistema esteja disponível e consiga processar uma solicitação, possivelmente fragilizando o comportamento e a disponibilidade geral.
- A workload comunica-se com as dependências de forma assíncrona e depende do conceito de entrega de mensagens garantida exatamente uma vez, quando muitas vezes ainda é possível receber mensagens duplicadas.
- A workload não usa ferramentas adequadas de agendamento em lote e permite a execução simultânea do mesmo trabalho em lotes.

Benefícios de estabelecer esta prática recomendada: é comum que determinada workload implemente um ou mais estilos de comunicação entre síncrono, assíncrono e em lote. Essa prática recomendada ajuda você a identificar as diferentes vantagens e desvantagens associadas a cada estilo de comunicação para tornar a workload capaz de tolerar interrupções em qualquer uma das dependências.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: alto

Orientação para a implementação

As seções a seguir contêm diretrizes de implementação gerais e específicas de cada tipo de dependência.

Orientação geral

- Certifique-se de que os objetivos de nível de serviço (SLOs) de performance e confiabilidade que suas dependências oferecem atendam aos requisitos de performance e confiabilidade da workload.

- Use os serviços de observabilidade da AWS para monitorar os tempos de resposta e as taxas de erro e garantir que a dependência esteja fornecendo serviços nos níveis necessários para a workload.
- Identifique os possíveis desafios que a workload pode enfrentar ao se comunicar com as dependências. Os sistemas distribuídos apresentam uma série de desafios que podem aumentar a complexidade arquitetônica, a carga operacional e o custo. Os desafios comuns são: latência, interrupções na rede, perda de dados, escalabilidade e atraso na replicação de dados.
- Implemente o tratamento de erros e o registro em log consistentes para ajudar a solucionar problemas quando a dependência tiver problemas.

Dependência síncrona

Nas comunicações síncronas, a workload envia uma solicitação para a dependência e bloqueia a operação à espera de uma resposta. Quando a dependência recebe a solicitação, ela tenta tratá-la o mais rápido possível e envia uma resposta de volta à workload. Um desafio significativo da comunicação síncrona é que ela causa o acoplamento temporal, o que exige que a workload e as respectivas dependências estejam disponíveis ao mesmo tempo. Quando a workload precisar se comunicar de forma síncrona com as dependências, pense na seguinte orientação:

- Sua workload não deve depender de várias dependências síncronas para realizar uma única função. Essa cadeia de dependências aumenta a fragilidade geral porque todas as dependências no caminho precisam estar disponíveis para que a solicitação seja concluída com êxito.
- Quando uma dependência não estiver íntegra ou estiver indisponível, determine suas estratégias de tratamento de erros e de novas tentativas. Evite usar comportamento bimodal. O comportamento bimodal ocorre quando a workload exibe um comportamento diferente nos modos normal e de falha. Para ter mais detalhes sobre o comportamento bimodal, consulte REL11-BP05 Usar estabilidade estática para evitar o comportamento bimodal.
- Lembre-se de que antecipar-se à falha é melhor do que fazer a workload esperar. Por exemplo, o Guia do desenvolvedor do AWS Lambda descreve como lidar com novas tentativas e falhas ao invocar funções do Lambda.
- Defina tempos limite quando a workload chamar sua dependência. Essa técnica evita esperas muito longas ou indefinidas por uma resposta. Para ler uma discussão útil sobre esse problema, consulte Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications.
- Minimize o número de chamadas feitas da workload para a dependência para atender a uma única solicitação. Ter chamadas interativas entre elas aumenta o acoplamento e a latência.

Dependência assíncrona

Para dissociar temporariamente a workload de sua dependência, elas devem se comunicar de forma assíncrona. Usando uma abordagem assíncrona, a workload pode continuar com qualquer outro processamento sem precisar esperar que a dependência, ou cadeia de dependências, envie uma resposta.

Quando a workload precisar se comunicar de forma assíncrona com a dependência, pense na seguinte orientação:

- Determine se deseja usar mensagens ou streaming de eventos com base no caso de uso e requisitos. O sistema de mensagens permite que a workload se comunique com a dependência enviando e recebendo mensagens por meio de um agente de mensagens. O streaming de eventos permite que a workload e a dependência usem um serviço de streaming para publicar e assinar eventos, entregues como fluxos contínuos de dados, que precisam ser processados o mais rápido possível.

- O Sistema de mensagens e o streaming de eventos gerenciam as mensagens de forma diferente, então você precisa tomar decisões sobre concessão com base em:
  - Prioridade da mensagem: os agentes de mensagens podem processar mensagens de alta prioridade antes das mensagens normais. No streaming de eventos, todas as mensagens têm a mesma prioridade.
  - Consumo de mensagens: os agentes de mensagens garantem que os consumidores recebam a mensagem. Os consumidores de streaming de eventos devem acompanhar a última mensagem que leram.
  - Ordenação de mensagens: com o sistema de mensagens, o recebimento de mensagens na ordem exata em que são enviadas não é garantido, a menos que você use uma abordagem de primeiro a entrar, primeiro a sair (FIFO). O streaming de eventos sempre preserva a ordem na qual os dados foram produzidos.
  - Exclusão da mensagem: com o sistema de mensagens, o consumidor deve excluir a mensagem após processá-la. O serviço de streaming de eventos anexa a mensagem a um fluxo e permanece lá até que o período de retenção da mensagem expire. Essa política de exclusão torna o streaming de eventos adequado para reproduzir mensagens.
- Defina como a workload sabe quando a dependência conclui o trabalho. Por exemplo, quando a workload invoca uma função do Lambda de forma assíncrona, o Lambda coloca o evento em uma fila e exibe uma resposta de êxito sem informações adicionais. Após a conclusão do

processamento, a função do Lambda pode enviar o resultado a um destino configurável com base no sucesso ou na falha.

- Crie a workload para lidar com mensagens duplicadas aproveitando a idempotência. Idempotência significa que os resultados da workload não mudam, mesmo que ela seja gerada mais de uma vez para a mesma mensagem. É importante ressaltar que os serviços de sistema de mensagens ou streaming reenviarão uma mensagem se ocorrer uma falha na rede ou se uma confirmação não tiver sido recebida.
- Se a workload não receber uma resposta da dependência, ela precisará reenviar a solicitação. Pense em limitar o número de novas tentativas para preservar a CPU, a memória e os recursos de rede da workload para lidar com outras solicitações. A documentação do AWS Lambda mostra como lidar com erros para invocação assíncrona.
- Utilize as ferramentas adequadas de observabilidade, depuração e rastreamento para gerenciar e operar a comunicação assíncrona da workload com a dependência. É possível usar o Amazon CloudWatch para monitorar os serviços sistema de mensagens e streaming de eventos. Você também pode instrumentar a workload com o AWS X-Ray para ter insights rapidamente e solucionar problemas.

Dependência em lote

Os sistemas em lote utilizam dados de entrada, iniciam uma série de trabalhos para processá-los e produzem alguns dados de saída, sem intervenção manual. Dependendo do tamanho dos dados, os trabalhos podem ser executados de minutos a, em alguns casos, vários dias. Quando a workload se comunica com a dependência em lote, pense na seguinte orientação:

- Defina a janela de tempo em que a workload deve executar o trabalho em lote. A workload pode configurar um padrão de recorrência para invocar um sistema em lote, por exemplo, a cada hora ou no final de cada mês.
- Determine a localização da entrada de dados e da saída de dados processados. Escolha um serviço de armazenamento, como Amazon Simple Storage Services (Amazon S3), Amazon Elastic File System (Amazon EFS) e Amazon FSx for Lustre, que permita que a workload leia e grave arquivos em grande escala.
- Se a workload precisar invocar vários trabalhos em lote, você poderá utilizar o AWS Step Functions para simplificar a orquestração de trabalhos em lote executados na AWS ou no ambiente on-premises. Este projeto de exemplo demonstra a orquestração de trabalhos em lote usando o Step Functions, o AWS Batche o Lambda.

- Monitore trabalhos em lote para procurar anormalidades, como um trabalho que leva mais tempo do que deveria para ser concluído. Você pode usar ferramentas, como o CloudWatch Container Insights, para monitorar os ambientes e os trabalhos do AWS Batch. Nesse caso, a workload impediria o início do próximo trabalho e informaria a equipe relevante sobre a exceção.

Recursos

Documentos relacionados:

- Nuvem AWS Operations: monitoramento e observabilidade
- A Amazon Builders' Library: desafios com sistemas distribuídos
- REL11-BP05 Usar estabilidade estática para evitar o comportamento bimodal
- Guia do desenvolvedor do AWS Lambda: Error handling and automatic retries in AWS Lambda
- Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications
- Sistema de mensagens da AWS
- O que são dados em streaming?
- Guia do desenvolvedor do AWS Lambda: Invocação assíncrona
- Perguntas frequentes sobre o Amazon Simple Queue Service: Filas FIFO
- Guia do desenvolvedor do Amazon Kinesis Data Streams: Handling Duplicate Records
- Guia do desenvolvedor do Amazon Simple Queue Service: Available CloudWatch metrics for Amazon SQS
- Guia do desenvolvedor do Amazon Kinesis Data Streams: Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch
- Guia do desenvolvedor do AWS X-Ray: AWS X-Ray concepts
- Exemplos da AWS no GitHub AWS: Step functions Complex Orchestrator App
- Guia do usuário do AWS Batch: AWS Batch CloudWatch Container Insights

Vídeos relacionados:

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

Ferramentas relacionadas:

- Amazon CloudWatch

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

REL04-BP02 Implementar dependências com acoplamento fraco

As dependências, como sistemas de enfileiramento, sistemas de streaming, fluxos de trabalho e load balancers, têm acoplamento fraco. O baixo acoplamento ajuda a isolar o comportamento de um componente de outros componentes que dependem dele, aumentando a resiliência e a agilidade.

Em sistemas fortemente acoplados, as mudanças em um componente podem exigir mudanças em outros componentes que dependem dele, o que resulta em desempenho degradado em todos eles. O acoplamento fraco interrompe essa dependência de forma que os componentes dependentes só precisem conhecer a interface versionada e publicada. A implementação de um baixo acoplamento entre dependências isola uma falha em uma dependência para não afetar a outra.

O acoplamento fraco permite modificar o código ou adicionar recursos a um componente, minimizando o risco para outros componentes que dependem dele. Ele também permite resiliência detalhada em nível de componente, caso em que é possível aumentar a escala horizontalmente ou até mesmo alterar a implementação subjacente da dependência.

Para melhorar ainda mais a resiliência por meio do baixo acoplamento, torne as interações de componentes assíncronas sempre que possível. Esse modelo é adequado para qualquer interação que não precise de uma resposta imediata e em que uma confirmação de que uma solicitação foi registrada será suficiente. Envolve um componente que gera eventos e outro que os consome. Os dois componentes não se integram por meio de interação direta ponto a ponto, mas geralmente por meio de uma camada de armazenamento durável intermediária, como uma fila do Amazon SQS, uma plataforma de dados de streaming, como o AWS Step Functions, ou o Amazon Kinesis.

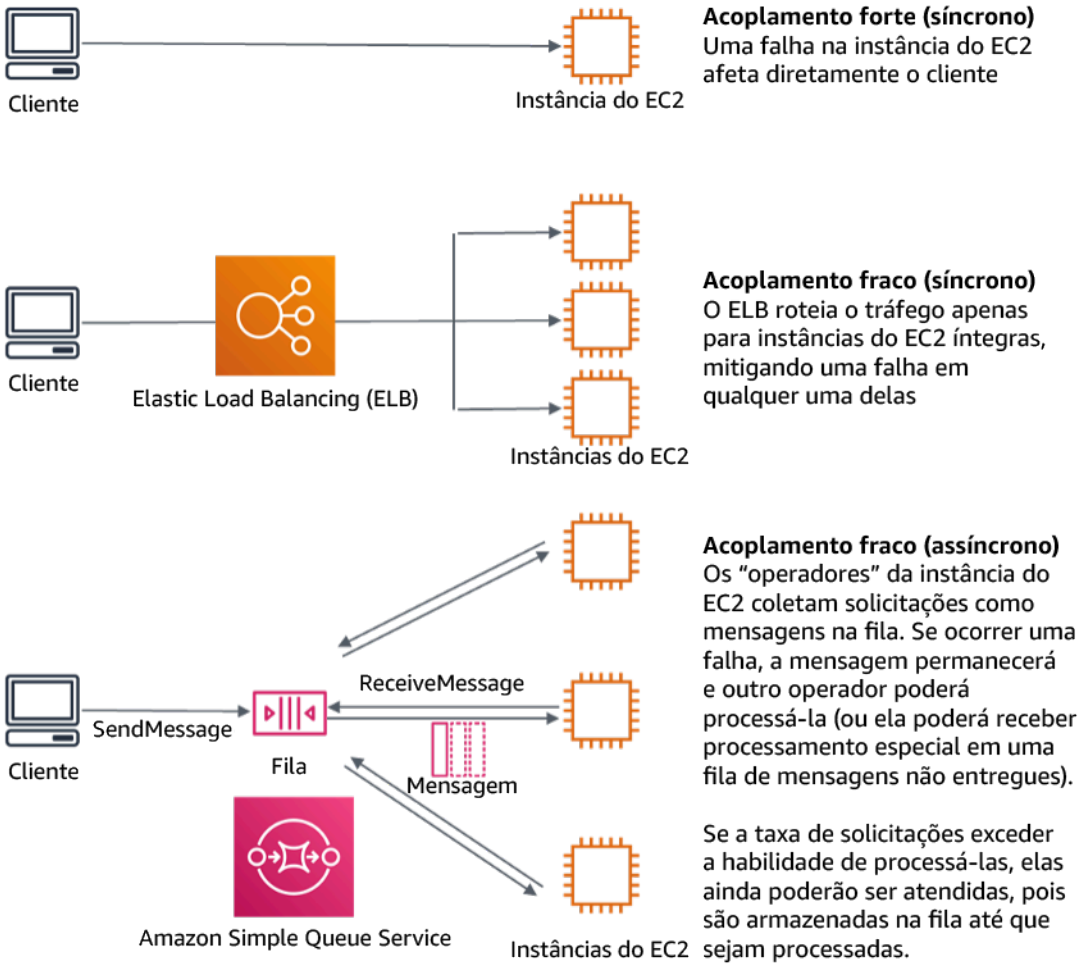


Figura 4: Dependências como sistemas de enfileiramento e load balancers têm baixo acoplamento

Filas do Amazon SQS e Elastic Load Balancers são apenas duas maneiras de adicionar uma camada intermediária para baixo acoplamento. Arquiteturas orientadas a eventos também podem ser criadas na Nuvem AWS usando o Amazon EventBridge, que pode abstrair clientes (produtores de eventos) dos serviços dos quais eles dependem (consumidores de eventos). O Amazon Simple Notification Service (Amazon SNS) é uma solução eficaz quando você precisa de mensagens de alto throughput, baseadas em push e de muitos para muitos. Usando tópicos do Amazon SNS, seus sistemas de editores podem enviar mensagens para um grande número de endpoints assinantes para processamento paralelo.

Embora as filas ofereçam várias vantagens, na maioria dos sistemas complexos em tempo real, as solicitações mais antigas do que um tempo limite (geralmente segundos) devem ser consideradas obsoletas (o cliente desistiu e não está mais esperando por uma resposta) e não processadas. Dessa forma, as solicitações mais recentes (e provavelmente ainda válidas) podem ser processadas.

Resultado desejado: a implementação de dependências com acoplamento fraco permite minimizar a área de superfície da falha para o nível de componente, o que ajuda a diagnosticar e resolver problemas. Também simplifica os ciclos de desenvolvimento, permitindo que as equipes implementem mudanças em um nível modular sem impactar o desempenho de outros componentes que dependem delas. Essa abordagem fornece a capacidade de aumentar a escala horizontalmente em nível de componente com base nas necessidades dos recursos, bem como na utilização de um componente que contribui para a redução de custos.

Antipadrões comuns:

- Implantar uma workload monolítica.
- Invocar diretamente as APIs entre níveis de workload sem recurso de failover ou processamento assíncrono da solicitação.
- Acoplamento forte usando dados compartilhados. Sistemas com acoplamento fraco devem evitar o compartilhamento de dados por meio de bancos de dados compartilhados ou outras formas de armazenamento de dados fortemente acoplado, o que pode reintroduzir o acoplamento forte e impedir a escalabilidade.
- Ignorar a contrapressão. A workload deve ter a capacidade de diminuir ou interromper a entrada de dados quando um componente não puder processá-los na mesma velocidade.

Benefícios do estabelecimento dessa prática recomendada: o acoplamento fraco ajuda a isolar o comportamento de um componente de outros componentes que dependem dele, aumentando a resiliência e a agilidade. A falha em um componente é isolada dos demais.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: alto

Orientações para a implementação

Implemente dependências com acoplamento fraco. Existem várias soluções que permitem criar aplicações com acoplamento fraco. Isso inclui serviços para implementar filas totalmente gerenciadas, fluxos de trabalho automatizados, reação a eventos e APIs, entre outros, que podem ajudar a isolar o comportamento de componentes de outros componentes e, dessa forma, aumentar a resiliência e a agilidade.

- Criar arquiteturas orientadas a eventos: o Amazon EventBridge ajuda a criar arquiteturas orientadas a eventos com acoplamento fraco e distribuídas.
- Implementar filas em sistemas distribuídos: é possível usar o Amazon Simple Queue Service (Amazon SQS) para integrar e desacoplar sistemas distribuídos.

- Conteinerizar componentes como microsserviços: os microsserviços permitem que as equipes criem aplicações constituídas de pequenos componentes independentes que se comunicam por meio de APIs bem definidas. O Amazon Elastic Container Service (Amazon ECS) e o Amazon Elastic Kubernetes Service (Amazon EKS) podem ajudar você a começar a usar contêineres mais rapidamente.
- Gerenciar fluxos de trabalho com o Step Functions: o Step Functions ajuda você a coordenar vários serviços da AWS em fluxos de trabalho flexíveis.
- Utilizar as arquiteturas de mensagens de publicação e assinatura (pub/sub): o Amazon Simple Notification Service (Amazon SNS) fornece a entrega de mensagens dos publicadores aos assinantes (também conhecidos como produtores e consumidores).

Etapas da implementação

- Os componentes em uma arquitetura orientada a eventos são iniciados por eventos. Eventos são ações que ocorrem em um sistema, como um usuário que adiciona um item a um carrinho. Quando uma ação é bem-sucedida, é gerado um evento que aciona o próximo componente do sistema.
  - Building Event-driven Applications with Amazon EventBridge
  - AWS re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge
- Os sistemas de mensagens distribuídos têm três partes principais que precisam ser implementadas para uma arquitetura baseada em fila. Eles incluem componentes do sistema distribuído, a fila usada para desacoplamento (distribuída em servidores do Amazon SQS) e as mensagens na fila. Um sistema típico tem produtores que iniciam a mensagem na fila e o consumidor que recebe a mensagem da fila. A fila armazena as mensagens em vários servidores do Amazon SQS para redundância.
  - Basic Amazon SQS architecture
  - Send Messages Between Distributed Applications with Amazon Simple Queue Service
- Os microsserviços, quando bem utilizados, melhoram a capacidade de manutenção e aumentam a escalabilidade, pois os componentes com acoplamento fraco são gerenciados por equipes independentes. Também permitem o isolamento de comportamentos em um único componente em caso de mudanças.
  - Implementação de microsserviços na AWS
  - Let's Architect! Architecting microservices with containers

- Com o AWS Step Functions é possível criar aplicações distribuídas, automatizar processos, orquestrar microsserviços, entre outras coisas. A orquestração de vários componentes em um fluxo de trabalho automatizado permite desacoplar as dependências na aplicação.
  - Create a Serverless Workflow with AWS Step Functions and AWS Lambda
  - Conceitos básicos do AWS Step Functions

Recursos

Documentos relacionados:

- Amazon EC2: Ensuring Idempotency
- A Amazon Builders' Library: desafios com sistemas distribuídos
- A Amazon Builders' Library: confiabilidade, trabalho constante e uma boa xícara de café
- O que é o Amazon EventBridge?
- O que é o Amazon Simple Queue Service?
- Break up with your monolith
- Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS
- Basic Amazon SQS architecture
- Arquitetura baseada em fila

Vídeos relacionados:

- AWS New York Summit 2019: Introduction to event-driven architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (inclui acoplamento fraco, trabalho constante, estabilidade estática)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda
- AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge
- AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices

REL04-BP03 Fazer um trabalho constante

Os sistemas podem falhar quando há alterações grandes e rápidas na carga. Por exemplo, se a sua workload está realizando uma verificação de integridade que monitora a integridade de milhares de servidores, ela deve sempre enviar a carga útil com o mesmo tamanho (um snapshot completo do estado atual). Se houver uma falha em todos os servidores ou se não houver falha alguma, o sistema de verificação de integridade realizará um trabalho constante sem alterações grandes e rápidas.

Por exemplo, se o sistema de verificação de integridade estiver monitorando 100.000 servidores, a carga nele será nominal a uma taxa de falha do servidor normalmente leve. No entanto, se um evento importante deixar metade desses servidores com problemas de integridade, o sistema de verificação de integridade ficará sobrecarregado tentando atualizar os sistemas de notificação e comunicar o estado com seus clientes. Portanto, em vez disso, o sistema de verificação de integridade deve enviar o snapshot completo do estado atual a cada vez. Os estados da integridade de 100.000 servidores, cada um representado por um bit, seriam apenas uma carga útil de 12,5 KB. independentemente de nenhum servidor ou falhar, ou todos eles falharem, o sistema de verificação de integridade está realizando um trabalho constante, e alterações grandes e rápidas não são uma ameaça para a estabilidade do sistema. Na verdade, é assim que o Amazon Route 53 lida com as verificações de integridade de endpoints (como endereços IP) para determinar como os usuários finais são roteados para eles.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Baixo

Orientações para a implementação

- Faça um trabalho constante para que os sistemas não falhem quando houver mudanças rápidas e grandes na carga.
- Implemente dependências com acoplamento fraco. As dependências, como sistemas de enfileiramento, sistemas de streaming, fluxos de trabalho e load balancers, têm acoplamento fraco. O baixo acoplamento ajuda a isolar o comportamento de um componente de outros componentes que dependem dele, aumentando a resiliência e a agilidade.
  - A Amazon Builders' Library: confiabilidade, trabalho constante e uma boa xícara de café
  - AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (inclui trabalho constante)
    - Para o exemplo de um sistema de verificação de integridade que monitora 100 mil servidores, crie as workloads de modo que os tamanhos da carga útil permaneçam constantes, seja qual for o número de êxitos ou falhas.

Recursos

Documentos relacionados:

- Amazon EC2: como garantir a idempotência
- A Amazon Builders' Library: desafios com sistemas distribuídos
- A Amazon Builders' Library: confiabilidade, trabalho constante e uma boa xícara de café

Vídeos relacionados:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (inclui trabalho constante)
- AWS re:Invent 2018: Close Loops & Opening Minds: How to Take Control of Systems, Big & Small ARC337 (inclui acoplamento fraco, trabalho constante, estabilidade estática)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP04 Fazer com que todas as respostas sejam idempotentes

Um serviço idempotente garante que cada solicitação seja concluída exatamente uma vez, de modo que fazer várias solicitações idênticas tem o mesmo efeito de uma única solicitação. Um serviço idempotente facilita para um cliente implementar novas tentativas sem o receio de que uma solicitação seja processada erroneamente várias vezes. Para fazer isso, os clientes podem emitir solicitações de API com um token de idempotência. O mesmo token é usado sempre que a solicitação é repetida. Uma API de serviço idempotente usa o token para retornar uma resposta idêntica à resposta que foi retornada na primeira vez que a solicitação foi concluída.

Em um sistema distribuído, é fácil executar uma ação no máximo uma vez (o cliente faz apenas uma solicitação) ou pelo menos uma vez (continue solicitando até o cliente receber a confirmação do sucesso). Porém, é difícil garantir que uma ação seja idempotente, o que significa que ela é executada exatamente uma vez, de modo que fazer várias solicitações idênticas tenha o mesmo efeito de uma única solicitação. Usando tokens de idempotência em APIs, os serviços podem receber uma solicitação mutante uma vez ou mais sem a criação de registros duplicados nem efeitos colaterais.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- Faça com que todas as respostas sejam idempotentes. Um serviço idempotente garante que cada solicitação seja concluída exatamente uma vez, de modo que fazer várias solicitações idênticas tem o mesmo efeito de uma única solicitação.
  - Os clientes podem emitir solicitações de API com um token de idempotência. O mesmo token é usado sempre que a solicitação é repetida. Uma API de serviço idempotente usa o token para retornar uma resposta idêntica à resposta que foi retornada na primeira vez que a solicitação foi concluída.
    - Amazon EC2: como garantir a idempotência

Recursos

Documentos relacionados:

- Amazon EC2: como garantir a idempotência
- A Amazon Builders' Library: desafios com sistemas distribuídos
- A Amazon Builders' Library: confiabilidade, trabalho constante e uma boa xícara de café

Vídeos relacionados:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops & Opening Minds: How to Take Control of Systems, Big & Small ARC337 (inclui acoplamento fraco, trabalho constante, estabilidade estática)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

Projetar as interações em um sistema distribuído para mitigar ou resistir a falhas

Os sistemas distribuídos dependem de redes de comunicação para interconectar componentes (como servidores ou serviços). Sua carga de trabalho deve operar de forma confiável, apesar da perda de dados ou da latência nessas redes. Os componentes do sistema distribuído devem operar sem afetar negativamente outros componentes ou a carga de trabalho. Essas melhores práticas permitem que as cargas de trabalho resistam a tensões ou falhas, recuperem-se mais rapidamente

delas e reduzam o impacto de tais prejuízos. Como resultado, o Mean Time To Recovery (MTTR – Tempo médio até a recuperação) é melhorado.

Essas melhores práticas evitam falhas e melhoram o Mean Time Between Failures (MTBF – Tempo médio entre falhas).

Práticas recomendadas

- REL05-BP01 Implementar uma degradação simples para transformar dependências rígidas aplicáveis em dependências flexíveis
- REL05-BP02 Controlar a utilização de solicitações
- REL05-BP03 Controlar e limitar as chamadas de repetição
- REL05-BP04 Antecipar-se à falha e filas limitadas
- REL05-BP05 Definir tempos limite do cliente
- REL05-BP06 Criar serviços sem estado sempre que possível
- REL05-BP07 Implementar medidas emergenciais

REL05-BP01 Implementar uma degradação simples para transformar dependências rígidas aplicáveis em dependências flexíveis

Os componentes da aplicação devem continuar desempenhando sua função principal mesmo que as dependências se tornem indisponíveis. Eles podem estar fornecendo dados um pouco obsoletos, dados alternativos ou até mesmo nenhum dado. Isso garante que o funcionamento geral do sistema seja minimamente impedido por falhas localizadas e, ao mesmo tempo, ofereça o valor empresarial central.

Resultado desejado: Quando as dependências de um componente não estão íntegras, o próprio componente ainda pode funcionar, embora de maneira prejudicada. Os modos de falha dos componentes devem ser vistos como operação normal. Os fluxos de trabalho devem ser projetados de forma que essas falhas não ocasionem à falha total ou, pelo menos, a estados previsíveis e recuperáveis.

Antipadrões comuns:

- Não identificar a principal funcionalidade empresarial necessária. Não testar se os componentes estão funcionando mesmo durante falhas de dependência.

- Não fornecer dados sobre erros ou quando apenas uma das várias dependências não está disponível e resultados parciais ainda podem ser retornados.
- Criar um estado inconsistente quando uma transação falha parcialmente.
- Não ter uma forma alternativa de acessar um armazenamento de parâmetros central.
- Invalidar ou esvaziar o estado local como resultado de uma falha na atualização sem levar em conta as consequências de fazer isso.

Benefícios de estabelecer esta prática recomendada: A degradação gradual melhora a disponibilidade do sistema como um todo e mantém as funções mais importantes em execução mesmo durante falhas.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

A implementação de uma degradação gradual ajuda a minimizar o impacto das falhas de dependência na função do componente. Preferencialmente, um componente detecta falhas de dependência e as contorna de uma maneira que afete minimamente outros componentes ou clientes.

Arquitetar para uma degradação gradual significa considerar possíveis modos de falha durante o projeto de dependência. Para cada modo de falha, tenha uma maneira de fornecer a maior parte ou pelo menos a funcionalidade mais crítica do componente para chamadores ou clientes. Essas considerações podem se tornar requisitos adicionais que podem ser testados e verificados. Preferencialmente, um componente é capaz de realizar sua função principal de maneira aceitável, mesmo quando uma ou várias dependências falhem.

Trata-se tanto de uma discussão empresarial quanto técnica. Todos os requisitos empresariais são importantes e devem ser atendidos, se possível. No entanto, ainda faz sentido perguntar o que deve acontecer quando nem todos eles podem ser cumpridos. Um sistema pode ser projetado para estar disponível e ser consistente, mas em circunstâncias em que um requisito deve ser descartado, qual deles é mais importante? Para o processamento de pagamentos, pode ser a consistência. Para uma aplicação em tempo real, pode ser a disponibilidade. Para um site voltado para o cliente, a resposta pode depender das expectativas do cliente.

O que isso significa depende dos requisitos do componente e do que deve ser considerado sua função principal. Por exemplo:

- Um site de comércio eletrônico pode exibir dados de vários sistemas diferentes, como recomendações personalizadas, produtos mais bem classificados e status dos pedidos dos

clientes na página de pouso. Quando um sistema upstream falha, ainda faz sentido exibir todo o resto em vez de mostrar uma página de erro para um cliente.

- Um componente que executa gravações em lote ainda poderá continuar processando um lote se ocorrer uma falha em uma das operações individuais. Deve ser simples implementar um mecanismo de repetição. Isso pode ser feito retornando informações sobre quais operações foram bem-sucedidas, quais falharam e por que falharam para o chamador, ou colocando solicitações com falha em uma fila de mensagens não entregues para implementar repetições assíncronas. As informações sobre operações com falha também devem ser registradas em log.
- Um sistema que processa transações deve verificar se todas ou nenhuma atualização individual foi executada. Para transações distribuídas, o padrão saga pode ser usado para reverter operações anteriores caso ocorra uma falha em uma operação posterior da mesma transação. Aqui, a função principal é manter a consistência.
- Sistemas essenciais devem ser capazes de lidar com dependências não correspondentes em tempo hábil. Nesses casos, o padrão do disjuntor pode ser usado. Quando as respostas de uma dependência começam a atingir o tempo limite, o sistema pode mudar para um estado fechado em que nenhuma chamada adicional é realizada.
- Uma aplicação pode ler parâmetros de um armazenamento de parâmetros. Pode ser útil criar imagens de contêiner com um conjunto padrão de parâmetros e usá-las caso o armazenamento de parâmetros não esteja disponível.

Observe que as vias percorridas em caso de falha do componente precisam ser testadas e devem ser significativamente mais simples do que a via principal. Geralmente, estratégias alternativas devem ser evitadas.

Etapas da implementação

Identifique dependências externas e internas. Leve em conta quais tipos de falhas podem ocorrer nelas. Pense em maneiras de minimizar o impacto negativo nos sistemas upstream e downstream e nos clientes durante essas falhas.

Veja a seguir uma lista de dependências e como degradar normalmente quando elas falham:

1. Falha parcial das dependências: Um componente pode fazer várias solicitações para sistemas downstream, como várias solicitações para um sistema ou uma solicitação para vários sistemas cada. Dependendo do contexto empresarial, diferentes maneiras de lidar com isso podem ser apropriadas (para obter mais detalhes, consulte exemplos anteriores em Orientações de implementação).

2. Um sistema downstream não consegue processar solicitações devido à alta carga: Se as solicitações para um sistema downstream falharem de modo consistente, não faz sentido continuar tentando. Isso pode criar carga adicional em um sistema já sobrecarregado e dificultar a recuperação. O padrão do disjuntor pode ser utilizado aqui, que monitora as chamadas com falha para um sistema downstream. Se ocorrer uma falha em um grande número de chamadas, ele deixará de enviar mais solicitações para o sistema downstream e só ocasionalmente permitirá que as chamadas passem para testar se o sistema downstream está disponível novamente.
3. Um armazenamento de parâmetros não está disponível: Para transformar um armazenamento de parâmetros, é possível usar o armazenamento em cache flexível de dependências ou padrões razoáveis incluídos nas imagens do contêiner ou da máquina. Observe que esses padrões precisam ser mantidos atualizados e incluídos nos pacotes de testes.
4. Um serviço de monitoramento ou outra dependência não funcional não está disponível: Se um componente não conseguir enviar logs, métricas ou rastreamentos de forma intermitente para um serviço de monitoramento central, geralmente é melhor continuar executando as funções empresariais normalmente. Não registrar em log nem enviar métricas silenciosamente por um longo período geralmente não é aceitável. Além disso, alguns casos de uso podem exigir entradas de auditoria completas para atender aos requisitos de conformidade.
5. Uma instância primária de um banco de dados relacional pode não estar disponível: Amazon Relational Database Service, como quase todos os bancos de dados relacionais, só pode ter uma instância primária de gravador. Isso cria um único ponto de falha para workloads de gravação e dificulta o ajuste de escala. Isso pode ser parcialmente reduzido com o uso de uma configuração Multi-AZ para alta disponibilidade ou da tecnologia sem servidor da Amazon Aurora para melhor ajuste de escala. Para requisitos de disponibilidade muito altos, pode fazer sentido não confiar no gravador principal. Para consultas que são somente leitura, podem ser usadas réplicas de leitura, que fornecem redundância e a capacidade de aumentar a escala horizontalmente, não apenas para verticalmente. As gravações podem ser armazenadas em buffer, por exemplo, em uma fila do Amazon Simple Queue Service, para que as solicitações de gravação dos clientes ainda possam ser aceitas mesmo que a primária esteja temporariamente indisponível.

Recursos

Documentos relacionados:

- Amazon API Gateway: controlar a utilização das solicitações de API para um melhor throughput
- CircuitBreaker (resume “Circuit Breaker” do livro “Release It!”)
- Repetições de erros e recuo exponencial na AWS

- Michael Nygard “Release It! Design and Deploy Production-Ready Software”
- A Amazon Builders’ Library: evitar fallback em sistemas distribuídos
- A Amazon Builders’ Library: evitar backlogs de fila insuperáveis
- A Amazon Builders’ Library:desafios e estratégias de armazenamento em cache
- A Amazon Builders’ Library: tempos limite, novas tentativas e recuo com tremulação

Vídeos relacionados:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders’ Library (DOP328) (Repetição, recuo e jitter: AWS re:Invent 2019: Introdução à Amazon Builders’ Library (DOP328)

Exemplos relacionados:

- Laboratório do Well-Architected: nível 300: implementação de verificações de integridade e do gerenciamento de dependências para melhorar a confiabilidade

REL05-BP02 Controlar a utilização de solicitações

Controle a utilização das solicitações para reduzir o esgotamento de recursos devido a aumentos inesperados na demanda. Solicitações abaixo das taxas de controle de utilização são processadas, enquanto aquelas acima do limite definido são rejeitadas com uma mensagem de retorno indicando que o uso da solicitação foi controlado.

Resultado desejado: Grandes picos de volume, sejam causados por aumentos repentinos de tráfego de clientes, ataques de inundação ou tempestades de novas tentativas, são reduzidos pelo controle de utilização de solicitações, permitindo que as workloads continuem com o processamento normal do volume de solicitações compatível.

Antipadrões comuns:

- Os controles de utilização de endpoint da API não são implementados ou são mantidos em valores padrão sem considerar os volumes esperados.
- Não há teste de carregamento nem limites de controle de utilização dos endpoints da API.
- Controlar a utilização de taxas de solicitações sem considerar o tamanho ou a complexidade da solicitação.

- Testar as taxas máximas de solicitação ou o tamanho máximo da solicitação, mas não testar os dois juntos.
- Os recursos não são provisionados nos mesmos limites estabelecidos nos testes.
- Os planos de uso não foram configurados nem considerados para consumidores de API de aplicação para aplicação (A2A).
- Os consumidores da fila que escalam horizontalmente não têm as configurações máximas de simultaneidade configuradas.
- A limitação de taxas por endereço IP não foi implementada.

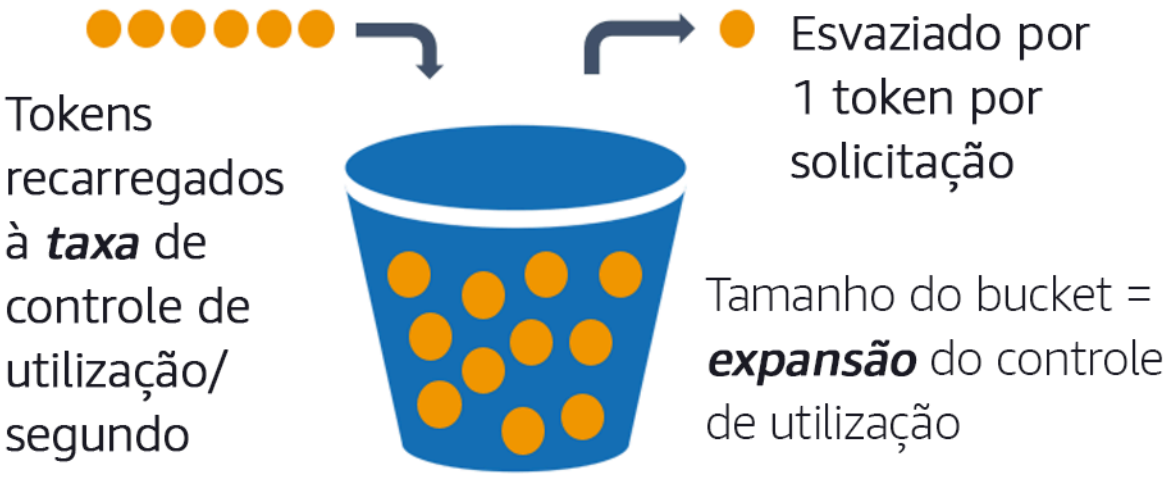
Benefícios de estabelecer esta prática recomendada: As workloads que definem limites de controle de utilização podem operar normalmente e processar a carga de solicitações aceitas com êxito em picos de volume inesperados. Os picos repentinos ou contínuos de solicitações para APIs e filas têm controle de utilização e não esgotam os recursos de processamento de solicitações. Os limites de taxas controlam a utilização de solicitantes individuais para que grandes volumes de tráfego de um único endereço IP ou consumidor de API não esgotem os recursos e afetem outros consumidores.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

Os serviços devem ser projetados para processar uma capacidade conhecida de solicitações; essa capacidade pode ser estabelecida por meio de testes de carga. Se as taxas de chegada de solicitações excederem os limites, a resposta apropriada sinalizará que uma solicitação teve controle de utilização. Isso permite que o consumidor resolva o erro e tente novamente mais tarde.

Quando seu serviço exigir uma implementação de controle de utilização, considere implementar o algoritmo de bucket de token, em que um token é contabilizado para uma solicitação. Os tokens são recarregados a uma taxa de controle de utilização por segundo e esvaziados de forma assíncrona por meio de um token por solicitação.



O algoritmo do bucket de token.

O Amazon API Gateway implementa o algoritmo do bucket de token de acordo com os limites da conta e da região e pode ser configurado por cliente com planos de uso. Além disso, o Amazon Simple Queue Service (Amazon SQS) e o Amazon Kinesis podem armazenar solicitações em buffer para suavizar a taxa de solicitações e permitir taxas de controle de utilização mais altas para solicitações que podem ser atendidas. Por fim, você pode implementar a limitação de taxa com o AWS WAF para controlar a utilização de consumidores de API específicos que geram uma carga excepcionalmente alta.

Etapas da implementação

É possível configurar o API Gateway com limites de controle de utilização para suas APIs e retornar erros "`429 Muitas solicitações`" quando os limites são excedidos. Você pode usar o AWS WAF com seus endpoints do AWS AppSync e do API Gateway para habilitar o limite de taxa por endereço IP. Além disso, se seu sistema tolerar o processamento assíncrono, será possível colocar mensagens em uma fila ou em um fluxo para acelerar as respostas aos clientes do serviço, o que permite que você atinja taxas de controle de utilização mais altas.

Com o processamento assíncrono, ao configurar o Amazon SQS como fonte de eventos para o AWS Lambda, é possível configurar a simultaneidade máxima para evitar que altas taxas de eventos consumam a cota de execução simultânea da conta disponível necessária para outros serviços em sua workload ou conta.

Embora o API Gateway ofereça uma implementação gerenciada do bucket de token, em casos em que não é possível usar o API Gateway, você pode utilizar as implementações de código aberto

específicas da linguagem (veja exemplos relacionados em Recursos) do bucket de token para seus serviços.

- Entenda e configure limites de controle de utilização do API Gateway em nível de conta por região, API por estágio e chave de API por nível do plano de uso.
- Aplique regras de controle de utilização de taxas do AWS WAF para endpoints do API Gateway e do AWS AppSync a fim de se proteger contra inundações e bloquear IPs mal-intencionados. As regras de controle de utilização de taxas também podem ser configuradas em chaves de API do AWS AppSync para consumidores A2A.
- Decida se você precisa de mais controle de limitação do que limitação de taxas para APIs do AWS AppSync e, em caso afirmativo, configure um API Gateway na frente do seu endpoint do AWS AppSync.
- Quando filas do Amazon SQS são configuradas como gatilhos para os consumidores da fila do Lambda, defina a simultaneidade máxima como um valor que processe o suficiente para atender aos seus objetivos de nível de serviço, mas não consuma limites de simultaneidade que afetem outras funções do Lambda. Considere definir a simultaneidade reservada em outras funções do Lambda na mesma conta e região ao consumir filas com o Lambda.
- Use o API Gateway com integrações de serviços nativos para Amazon SQS ou Kinesis para armazenar solicitações em buffer.
- Se você não puder usar o API Gateway, consulte bibliotecas específicas de linguagens para implementar o algoritmo do bucket de token para sua workload. Confira a seção de exemplos e faça sua própria pesquisa para encontrar uma biblioteca adequada.
- Teste os limites que você planeja definir ou permitir que sejam aumentados e documente os limites testados.
- Não aumente os limites além do que você estabeleceu nos testes. Ao aumentar um limite, verifique se os recursos provisionados já são equivalentes ou maiores do que os dos cenários de teste antes de aplicar o aumento.

Recursos

Práticas recomendadas relacionadas:

- REL04-BP03 Fazer um trabalho constante
- REL05-BP03 Controlar e limitar as chamadas de repetição

Documentos relacionados:

- Amazon API Gateway: controlar a utilização das solicitações de API para um melhor throughput
- AWS WAF: instrução de regra baseada em taxas
- Introdução da máxima simultaneidade do AWS Lambda ao usar o Amazon SQS como fonte de eventos
- AWS Lambda: simultaneidade máxima

Exemplos relacionados:

- As três regras mais importantes baseadas em taxas do AWS WAF
- Java Bucket4j
- Bucket de tokens do Python
- Bucket de tokens do Node
- Limitação da taxa de segmentação do sistema .NET

Vídeos relacionados:

- Implementing GraphQL API security best practices with AWS AppSync (Implementação das práticas recomendadas de segurança da API GraphQL com AWS AppSync)

Ferramentas relacionadas:

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

REL05-BP03 Controlar e limitar as chamadas de repetição

Use o recuo exponencial para repetir as solicitações em intervalos progressivamente maiores entre cada nova repetição. Introduza o jitter entre as repetições para tornar os intervalos de repetição aleatórios. Limite o número máximo de repetições.

Resultado desejado: Os componentes típicos em um sistema de software distribuído incluem servidores, load balancers, bancos de dados e servidores DNS. Durante a operação normal, esses componentes podem responder a solicitações com erros temporários ou limitados, além de erros que seriam persistentes, independentemente de repetições. Quando os clientes fazem solicitações aos serviços, elas consomem recursos, incluindo memória, threads, conexões, portas ou quaisquer outros recursos limitados. Controlar e limitar as repetições é uma estratégia para liberar e minimizar o consumo de recursos para que os componentes do sistema sob pressão não fiquem sobrecarregados.

Quando as solicitações do cliente atingem o tempo limite ou recebem respostas de erro, ele deve determinar se deve ou não tentar novamente. Se tentar novamente, ele o fará com um recuo exponencial com jitter e um valor máximo de repetição. Como resultado, os serviços e os processos de back-end recebem alívio da carga e do tempo de recuperação automática, ocasionando uma recuperação mais rápida e atendimento bem-sucedido das solicitações.

Antipadrões comuns:

- Implementar repetições sem adicionar recuo exponencial, jitter e valores máximos de repetição. O recuo e o jitter ajudam a evitar picos artificiais de tráfego devido a repetições coordenadas involuntariamente em intervalos comuns.
- Implementar repetições sem testar seus efeitos ou presumir que repetições já estejam incorporadas a um SDK sem testar cenários de repetição.
- Não entender os códigos de erro publicados das dependências, ocasionando a repetição de todos os erros, inclusive aqueles com uma causa clara que indica falta de permissão, erro de configuração ou outra condição que, previsivelmente, não será resolvida sem intervenção manual.
- Não abordar práticas de observabilidade, incluindo monitoramento e alertas sobre falhas repetidas de serviço para que os problemas subjacentes sejam divulgados e possam ser resolvidos.
- Desenvolver mecanismos de repetição personalizados quando os recursos de repetição integrados ou de terceiros são suficientes.
- Tentar novamente em várias camadas da pilha de aplicações de uma forma que agrava as tentativas de repetição, consumindo ainda mais recursos em uma tempestade de repetições. Entenda como esses erros afetam sua aplicação, as dependências nas quais você confia e implemente repetições em apenas um nível.
- Repetir chamadas de serviço que não são idempotentes, causando efeitos colaterais inesperados, como resultados duplicados.

Benefícios de estabelecer esta prática recomendada: As repetições ajudam os clientes a obter os resultados desejados quando as solicitações falham, mas também consomem mais tempo do servidor para obter as respostas bem-sucedidas que eles desejam. Quando as falhas são raras ou transitórias, as repetições funcionam bem. Quando as falhas são causadas pela sobrecarga de recursos, as repetições podem piorar as coisas. Adicionar um recuo exponencial com jitter às repetições do cliente permite que os servidores se recuperem quando as falhas são causadas pela sobrecarga de recursos. O jitter evita o alinhamento das solicitações em picos, e o recuo diminui o escalonamento de carga causado pela adição de repetições à carga normal da solicitação. Por fim, é importante configurar um número máximo de repetições ou o tempo decorrido para evitar a criação de backlogs que produzam falhas metaestáveis.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

Controle e limite as chamadas de repetição. Use o recuo exponencial para tentar novamente após intervalos progressivamente mais longos. Introduza jitter para tornar esses intervalos de repetição aleatórios e limite o número máximo de repetições.

Alguns AWS SDKs implementam repetições e recuo exponencial por padrão. Use essas implementações integradas da AWS quando aplicável em sua workload. Implemente uma lógica semelhante em sua workload ao chamar serviços que sejam idempotentes e em que repetições melhorem a disponibilidade do cliente. Decida quais são os tempos limite e quando parar de tentar novamente com base no seu caso de uso. Crie e exercite cenários de teste para esses casos de uso de repetições.

Etapas da implementação

- Determine a camada ideal em sua pilha de aplicações para implementar repetições para os serviços dos quais sua aplicação depende.
- Conheça os SDKs existentes que implementam estratégias comprovadas de repetição com retrocesso exponencial e jitter para a linguagem de sua escolha e dê preferência a esses SDKs em vez de escrever suas próprias implementações de repetição.
- Verifique se os serviços são idempotentes antes de implementar repetições. Depois que as repetições forem implementadas, elas devem ser testadas e exercitadas regularmente na produção.

- Ao chamar APIs de serviço da AWS, use os AWS SDKs e o AWS CLI e entenda as opções de configuração de repetições. Determine se os padrões funcionam para seu caso de uso, teste e ajuste conforme necessário.

Recursos

Práticas recomendadas relacionadas:

- REL04-BP04 Fazer com que todas as respostas sejam idempotentes
- REL05-BP02 Controlar a utilização de solicitações
- REL05-BP04 Antecipar-se à falha e filas limitadas
- REL05-BP05 Definir tempos limite do cliente
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas

Documentos relacionados:

- Repetições de erros e recuo exponencial na AWS
- A Amazon Builders' Library: tempos limite, novas tentativas e recuo com tremulação
- Recuo exponencial e jitter
- Tornar as tentativas seguras com APIs idempotentes

Exemplos relacionados:

- Repetição Spring
- Repetição Resilience4j

Vídeos relacionados:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328) (Repetição, recuo e jitter: AWS re:Invent 2019: Introdução à biblioteca de criadores da Amazon (DOP328)

Ferramentas relacionadas:

- AWS SDKs e ferramentas: comportamento de repetição

- AWS Command Line Interface: repetições da AWS CLI

REL05-BP04 Antecipar-se à falha e filas limitadas

Quando um serviço não consegue responder com êxito a uma solicitação, antecipe-se à falha. Isso permite a liberação dos recursos associados a uma solicitação e possibilita que o serviço se recupere se estiver ficando sem recursos. Antecipar-se à falha é um padrão de design de software bem estabelecido que pode ser utilizado para criar workloads altamente confiáveis na nuvem. As filas também correspondem a um padrão de integração empresarial bem estabelecido que pode facilitar o carregamento e permitir que os clientes liberem recursos quando o processamento assíncrono pode ser tolerado. Quando um serviço consegue responder com êxito em condições normais, mas falha quando a taxa de solicitações é muito alta, use uma fila para armazenar solicitações em buffer. No entanto, não permita a formação de backlogs de filas longas que possam ocasionar o processamento de solicitações antigas das quais um cliente já desistiu.

Resultado desejado: Quando os sistemas enfrentam contenção de recursos, tempos limite, exceções ou falhas de causa desconhecida que tornam os objetivos de nível de serviço inatingíveis, as estratégias de antecipação a falhas permitem uma recuperação mais rápida do sistema. Sistemas que precisam absorver picos de tráfego e acomodar o processamento assíncrono podem melhorar a confiabilidade ao permitir que os clientes liberem solicitações rapidamente usando filas para armazenar solicitações em buffer para serviços de back-end. Ao armazenar solicitações em filas, estratégias de gerenciamento de filas são implementadas para evitar backlogs intransponíveis.

Antipadrões comuns:

- Implementar filas de mensagens, mas não configurar filas de mensagens não entregues (DLQ) ou alarmes em volumes DLQ para detectar quando um sistema está em falha.
- Não medir a idade das mensagens em uma fila, uma medida de latência para entender quando os consumidores da fila estão ficando para trás ou cometendo erros, ocasionando repetições.
- Não limpar mensagens pendentes de uma fila, quando não há utilidade em processar essas mensagens se a necessidade empresarial deixar de existir.
- Configurar filas do tipo “first in first out” (FIFO) quando filas do tipo “last in first out” (LIFO) atenderia melhor às necessidades do cliente, por exemplo, quando a ordenação rigorosa não é necessária e o processamento de backlog está atrasando todas as solicitações novas e urgentes, ocasionando violação dos níveis de serviço de todos os clientes.
- Expor filas internas aos clientes em vez de expor APIs que gerenciem a entrada de trabalho e coloquem as solicitações em filas internas.

- Combinar muitos tipos de solicitações de trabalho em uma única fila, o que pode agravar as condições de backlog ao distribuir a demanda de recursos entre os tipos de solicitação.
- Processar solicitações complexas e simples na mesma fila, apesar da necessidade de monitoramento, tempos limite e alocação de recursos diferentes.
- Não validar entradas ou usar afirmações para implementar mecanismos de antecipação à falha em software que agreguem exceções a componentes de nível superior que podem lidar com erros sem problemas.
- Não remover recursos com defeito do roteamento de solicitações, principalmente quando as falhas estão emitindo êxitos e falhas em decorrência de travamento e reinicialização, falha de dependência intermitente, capacidade reduzida ou perda de pacotes de rede.

Benefícios de estabelecer esta prática recomendada: Sistemas que se antecipam à falha são mais fáceis de depurar e corrigir e geralmente expõem problemas de codificação e configuração antes que as versões sejam publicadas em produção. Os sistemas que incorporam estratégias eficazes de filas oferecem maior resiliência e confiabilidade a picos de tráfego e às condições intermitentes de falha do sistema.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

As estratégias de antecipação à falha podem ser codificadas em soluções de software e configuradas em infraestrutura. Além de se anteciparem à falha, as filas são uma técnica arquitetônica simples, mas poderosa, para dissociar os componentes do sistema e facilitar o carregamento. O Amazon CloudWatch oferece recursos para monitorar e alertar sobre falhas. Quando se sabe que um sistema está falhando, estratégias de mitigação podem ser invocadas, inclusive evitar recursos afetados. Quando os sistemas implementam filas com o Amazon SQS e outras tecnologias de fila para facilitar o carregamento, eles devem considerar como gerenciar os backlogs de filas, bem como as falhas no consumo de mensagens.

Etapas da implementação

- Implemente afirmações programáticas ou métricas específicas em seu software e use-as para alertar explicitamente sobre problemas do sistema. O Amazon CloudWatch ajuda você a criar métricas e alarmes com base no padrão de log da aplicação e na instrumentação do SDK.
- Use métricas e alarmes do CloudWatch para eliminar recursos danificados que estão aumentando a latência no processamento ou falhando repetidamente no processamento das solicitações.

- Use o processamento assíncrono criando APIs para aceitar e anexar solicitações às filas internas usando o Amazon SQS e, depois, responder ao cliente que produz a mensagem com uma mensagem de êxito para que o cliente possa liberar recursos e prosseguir com outros trabalhos enquanto os consumidores da fila de back-end processam as solicitações.
- Avalie e monitore a latência do processamento da fila produzindo uma métrica do CloudWatch sempre que retirar uma mensagem de uma fila, comparando o momento presente com o carimbo de data/hora da mensagem.
- Quando falhas impedem o processamento bem-sucedido de mensagens ou geram picos de tráfego em volumes que não podem ser processados de acordo com acordos de serviço, deixe de lado o tráfego antigo ou excedente para uma fila de transbordamento. Isso permite o processamento prioritário de trabalhos novos e antigos, quando há capacidade disponível. Essa técnica é uma aproximação do processamento LIFO e permite o processamento normal do sistema para todos os novos trabalhos.
- Use filas de mensagens não entregues ou de redirecionamento para mover mensagens que não podem ser processadas do backlog para um local que possa ser pesquisado e resolvido posteriormente.
- Tente novamente ou, quando possível, elimine as mensagens antigas comparando o momento presente com o carimbo de data/hora da mensagem e descartando as mensagens que não são mais relevantes para o cliente solicitante.

Recursos

Práticas recomendadas relacionadas:

- REL04-BP02 Implementar dependências com acoplamento fraco
- REL05-BP02 Controlar a utilização de solicitações
- REL05-BP03 Controlar e limitar as chamadas de repetição
- REL06-BP02 Definir e calcular as métricas (agregação)
- REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema

Documentos relacionados:

- Evitar backlogs de fila intransponíveis
- Falha rápida
- Como evitar um aumento do backlog de mensagens na minha fila do Amazon SQS?

- Elastic Load Balancing: mudança de zona
- Controlador de recuperação de aplicações Amazon Route 53: controle de roteamento para failover de tráfego

Exemplos relacionados:

- Padrões de integração empresarial: canal de mensagens não entregues

Vídeos relacionados:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications (re:Invent 2022: operar aplicações Multi-AZ altamente disponíveis)

Ferramentas relacionadas:

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

REL05-BP05 Definir tempos limite do cliente

Defina tempos limite adequados para conexões e solicitações, verifique-os sistematicamente e não confie nos valores padrão, pois eles não estão cientes das especificações da workload.

Resultado desejado: Os tempos limite do cliente devem considerar o custo para o cliente, o servidor e a workload associados à espera por solicitações que levam um tempo anormal para serem concluídas. Como não é possível saber a causa exata de nenhum tempo limite, os clientes devem usar o conhecimento dos serviços para desenvolver expectativas de causas prováveis e prazos apropriados.

As conexões do cliente atingem o tempo limite com base nos valores configurados. Depois de encontrar um tempo limite, os clientes tomam a decisão de recuar e tentar novamente ou abrir um disjuntor. Esses padrões evitam a emissão de solicitações que podem exacerbar uma condição de erro subjacente.

Antipadrões comuns:

- Não estar ciente dos tempos limite do sistema ou dos tempos limite padrão.
- Não estar ciente do tempo normal de conclusão da solicitação.
- Não estar ciente das possíveis causas das solicitações levarem muito tempo para serem concluídas ou dos custos de performance do cliente, do serviço ou da workload associados à espera por essas conclusões.
- Não estar ciente da probabilidade de uma rede danificada fazer com que uma solicitação falhe somente quando o tempo limite é atingido e dos custos para a performance do cliente e da workload por não adotar um tempo limite mais curto.
- Não testar cenários de tempo limite tanto para conexões quanto para solicitações.
- Definir tempos limite muito altos, o que pode resultar em longos tempos de espera e aumentar a utilização de recursos.
- Definir tempos limite muito baixos, gerando falhas artificiais.
- Ignorar padrões para lidar com erros de tempo limite para chamadas remotas, como disjuntores e novas tentativas.
- Não considerar o monitoramento de taxas de erro de chamadas de serviço, objetivos de nível de serviço para latência e valores atípicos de latência. Essas métricas podem fornecer informações sobre tempos limite agressivos ou permissivos.

Benefícios de estabelecer esta prática recomendada: Os tempos limite de chamadas remotas são configurados e os sistemas são projetados para lidar com os tempos limite normalmente, de forma que os recursos sejam conservados quando as chamadas remotas respondem de forma anormalmente lenta e os erros de tempo limite são tratados normalmente pelos clientes do serviço.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

Defina um tempo limite de conexão e um tempo limite de solicitação em qualquer chamada de dependência de serviço e, geralmente, em qualquer chamada entre processos. Muitas frameworks oferecem recursos de tempo limite integrados, mas tenha cuidado, pois algumas têm valores padrão infinitos ou superiores ao aceitável para seus objetivos de serviço. Um valor muito alto reduz a utilidade do tempo limite porque os recursos continuam a ser consumidos enquanto o cliente aguarda o decorrer do tempo limite. Um valor muito baixo pode gerar maior tráfego no back-end e maior latência, porque muitas solicitações são repetidas. Em alguns casos, isso pode levar a interrupções completas porque todas as solicitações estão sendo repetidas.

Considere o seguinte ao determinar as estratégias de tempo limite:

- As solicitações podem levar mais tempo do que o normal para serem processadas devido ao conteúdo, a deficiências em um serviço de destino ou a uma falha na partição de rede.
- Solicitações com conteúdo anormalmente caro podem consumir recursos desnecessários do servidor e do cliente. Nesse caso, reduzir o tempo limite dessas solicitações e não tentar novamente pode preservar os recursos. Os serviços também devem se proteger de conteúdo anormalmente caro com limitações e tempos limite do servidor.
- Solicitações que demoram muito devido a uma falha no serviço podem expirar e ser repetidas. Deve-se considerar os custos do serviço para a solicitação e a nova tentativa, mas se a causa for uma deficiência localizada, uma nova tentativa provavelmente não será cara e reduzirá o consumo de recursos do cliente. O tempo limite também pode liberar recursos do servidor, dependendo da natureza da deficiência.
- Solicitações que demoram muito para serem concluídas porque a solicitação ou a resposta não foi entregue pela rede podem expirar e ser repetidas. Como a solicitação ou a resposta não foi entregue, a falha teria sido o resultado, independentemente da duração do tempo limite. Nesse caso, o tempo limite não liberará recursos do servidor, mas liberará recursos do cliente e melhorará a performance da workload.

Aproveite os padrões de design bem estabelecidos, como novas tentativas e disjuntores, para lidar com os tempos de espera de forma eficiente e oferecer compatibilidade com abordagens de antecipação à falha. AWS SDKs e a AWS CLI permitem a configuração dos tempos limite de conexão e solicitação e as repetições com recuo exponencial e instabilidade. As funções do AWS Lambda são compatíveis com a configuração de tempos limite e com o AWS Step Functions, você pode criar disjuntores de uso de pouco código que utilizam integrações pré-incorporadas com serviços da AWS e SDKs. O AWS App Mesh Envoy oferece recursos de tempo limite e disjuntor.

Etapas da implementação

- Configure tempos limite em chamadas de serviço remoto e utilize os recursos de tempo limite de linguagem integrados ou as bibliotecas de tempo limite de código aberto.
- Quando sua workload fizer chamadas com um AWS SDK, revise a documentação para saber a configuração de tempo limite específica da linguagem.
  - Python
  - PHP
  - .NET

  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- Ao usar AWS SDKs ou comandos da AWS CLI em sua workload, configure os valores de tempo limite padrão definindo os padrões de configuração da AWS de `connectTimeoutInMillis` e `tlsNegotiationTimeoutInMillis`.
- Aplique opções de linha de comando `cli-connect-timeout` e `cli-read-timeout` para controlar comandos únicos da AWS CLI para serviços da AWS.
- Monitore o tempo limite de chamadas de serviço remoto e defina alarmes para erros persistentes para que você possa lidar proativamente com cenários de erro.
- Implemente métricas do CloudWatch e detecção de anomalias do CloudWatch em taxas de erro de chamada, objetivos de nível de serviço para latência e valores atípicos de latência para fornecer informações sobre o gerenciamento de tempos limite excessivamente agressivos ou permissivos.
- Configure tempos limite em funções do Lambda.
- Os clientes do API Gateway devem implementar suas próprias repetições ao lidar com os tempos limite. O API Gateway é compatível com um tempo limite de integração de 50 milissegundos a 29 segundos para integrações posteriores e não tenta novamente quando as solicitações de integração atingem o tempo limite.
- Implemente o padrão de disjuntor para evitar fazer chamadas remotas quando o tempo limite está prestes a ser atingido. Abra o circuito para evitar falhas nas chamadas e feche-o quando as chamadas estiverem respondendo normalmente.
- Para workloads baseadas em contêineres, analise os recursos do App Mesh Envoy para utilizar os tempos limite e os disjuntores integrados.
- Use o AWS Step Functions para criar disjuntores de pouco uso de código para chamadas de serviço remoto, especialmente ao chamar AWS SDKs nativos e integrações do Step Functions compatíveis para simplificar sua workload.

Recursos

Práticas recomendadas relacionadas:

- REL05-BP03 Controlar e limitar as chamadas de repetição

- REL05-BP04 Antecipar-se à falha e filas limitadas
- REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema

Documentos relacionados:

- AWS SDK: repetições e tempos limite
- A Amazon Builders' Library: tempos limite, novas tentativas e recuo com tremulação
- Cotas do Amazon API Gateway e notas importantes
- AWS Command Line Interface: opções de linha de comando
- AWS SDK for Java 2.x: configurar tempos limite de API
- AWS Botocore usando o objeto de configuração e a referência de configuração
- AWS SDK for .NET: repetições e tempos limite
- AWS Lambda: configurar as opções de função do Lambda

Exemplos relacionados:

- Usar o padrão do disjuntor com o AWS Step Functions e o Amazon DynamoDB
- Martin Fowler: CircuitBreaker

Ferramentas relacionadas:

- AWS SDKs
- As funções do AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 Criar serviços sem estado sempre que possível

Os sistemas não devem exigir estado ou devem descarregar o estado de modo que não haja dependência entre solicitações de clientes diferentes em relação aos dados armazenados localmente no disco ou na memória. Isso permite que os servidores sejam substituídos quando necessário sem prejudicar a disponibilidade.

Quando os usuários ou serviços interagem com um aplicativo, eles geralmente executam uma série de interações que formam uma sessão. Uma sessão são dados exclusivos para usuários que persistem entre solicitações enquanto usam o aplicativo. Um aplicativo sem estado é um aplicativo que não precisa de conhecimento de interações anteriores e não armazena informações da sessão.

Depois de projetados para serem sem estado, você pode usar serviços de computação com tecnologia sem servidor, como o AWS Lambda ou o AWS Fargate.

Além da substituição do servidor, outro benefício dos aplicativos sem estado é que eles podem escalar horizontalmente, pois qualquer um dos recursos de computação disponíveis (como instâncias do EC2 e funções do AWS Lambda) pode atender a qualquer solicitação.

Benefícios de estabelecer esta desta prática recomendada: os sistemas projetados para serem sem estado são mais adaptáveis à escalabilidade horizontal, possibilitando a adição ou remoção de capacidade com base na flutuação do tráfego e da demanda. Eles também são inerentemente resilientes a falhas e oferecem flexibilidade e agilidade no desenvolvimento de aplicações.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

Crie aplicações sem estado. As aplicações sem estado permitem a escalabilidade horizontal e são tolerantes a falhas de um nó individual. Analise e compreenda os componentes da aplicação que mantêm estado dentro da arquitetura. Isso ajuda você a avaliar o impacto potencial da transição para um design sem estado. Uma arquitetura sem estado dissocia os dados de usuários e descarrega os dados de sessões. Isso oferece a flexibilidade de escalar cada componente de forma independente para atender às diferentes demandas de workload e otimizar a utilização de recursos.

Etapas da implementação

- Identifique e compreenda os componentes com estado na aplicação.
- Dissocie os dados, separando e gerenciando os dados de usuários da lógica principal da aplicação.
  - O Amazon Cognito consegue dissociar os dados de usuários do código da aplicação usando recursos, como bancos de identidades, grupos de usuários e Amazon Cognito Sync.
  - Você pode usar o AWS Secrets Manager para dissociar os dados de usuários, armazenando segredos em um local seguro e centralizado. Isso significa que o código da aplicação não precisa armazenar segredos, o que a torna mais segura.

    - Pense em usar o Amazon S3 para armazenar dados grandes e não estruturados, como imagens e documentos. Sua aplicação poderá recuperar esses dados quando necessário, eliminando a necessidade de armazená-los na memória.
    - Use o Amazon DynamoDB para armazenar informações como perfis de usuário. Sua aplicação poderá consultar esses dados praticamente em tempo real.
- Descarregue os dados de sessões em um banco de dados, cache ou arquivos externos.
    - Amazon ElastiCache, Amazon DynamoDB, Amazon Elastic File System (Amazon EFS) e Amazon MemoryDB for Redis são exemplos de serviços da AWS que você pode usar para descarregar os dados de sessões.
- Crie uma arquitetura sem estado depois de identificar quais dados de estado e de usuários precisam ser mantidos com sua solução de armazenamento preferida.

Recursos

Práticas recomendadas relacionadas:

- REL11-BP03 Automatizar a reparação em todas as camadas

Documentos relacionados:

- A Amazon Builders' Library: evitar fallback em sistemas distribuídos
- A Amazon Builders' Library: evitar backlogs de fila insuperáveis
- A Amazon Builders' Library:desafios e estratégias de cache
- Práticas para a camada da Web sem estado na AWS

REL05-BP07 Implementar medidas emergenciais

Medidas emergenciais são processos rápidos que podem atenuar o impacto da disponibilidade na workload.

As medidas emergenciais funcionam com a desativação, o controle de utilização ou a alteração do comportamento dos componentes ou das dependências com o uso de mecanismos conhecidos e testados. Isso pode aliviar as deficiências da workload decorrentes da exaustão dos recursos provocada por aumentos inesperados na demanda e reduzir o impacto de falhas em componentes não essenciais da workload.

Resultado desejado: ao implementar medidas de emergência, é possível estabelecer processos bem conhecidos para manter a disponibilidade dos componentes essenciais na workload. A workload deve se degradar normalmente e continuar desempenhando suas funções essenciais aos negócios durante a ativação de uma medida emergencial. Para obter mais detalhes sobre a degradação simples, consulte REL05-BP01 Implementar uma degradação simples para transformar dependências rígidas aplicáveis em dependências flexíveis.

Antipadrões comuns:

- A falha de dependências não essenciais afeta a disponibilidade da workload principal.
- Não testar ou verificar o comportamento dos componentes essenciais durante a deterioração de componentes não essenciais.
- Não há critérios claros e determinísticos definidos para ativação ou desativação de uma medida emergencial.

Benefícios do estabelecimento desta prática recomendada: a implementação de medidas emergenciais pode melhorar a disponibilidade dos componentes essenciais na workload fornecendo aos resolvedores processos estabelecidos para responder a picos inesperados na demanda ou a falhas de dependências não essenciais.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

- Identifique os componentes essenciais na workload.
- Projete e arquitete os componentes essenciais na workload para resistirem à falha de componentes não essenciais.
- Conduza testes para validar o comportamento dos componentes essenciais durante a falha de componentes não essenciais.
- Defina e monitore métricas ou acionadores relevantes para iniciar procedimentos de medida emergencial.
- Defina os procedimentos (manuais ou automatizados) que compõem a medida emergencial.

Etapas da implementação

- Identificar os componentes essenciais aos negócios na workload.

  - Cada componente técnico na workload deve ser mapeado para a função de negócios relevante e classificado como essencial ou não essencial. Para obter exemplos de funcionalidades essenciais e não essenciais na Amazon, consulte Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second.
  - Essa é uma decisão técnica e de negócios e varia de acordo com a organização e a workload.
- Projete e arquitete os componentes essenciais na workload para resistirem à falha de componentes não essenciais.
  - Durante a análise de dependências, considere todos os possíveis modos de falha e verifique se os mecanismos de medida emergencial fornecem a funcionalidade essencial aos componentes subsequentes.
- Conduza testes para validar o comportamento dos componentes essenciais durante a ativação das medidas emergenciais.
  - Evite o comportamento bimodal. Para obter mais detalhes, consulte REL11-BP05 Usar estabilidade estática para evitar o comportamento bimodal.
- Defina, monitore e emita alertas sobre as métricas relevantes para iniciar o procedimento de medida emergencial.
  - A descoberta das métricas certas a serem monitoradas depende da workload. Alguns exemplos de métricas são a latência ou o número de solicitações com falha feitas para uma dependência.
- Defina os procedimentos, manuais ou automatizados, que compõem a medida emergencial.
  - Isso pode incluir mecanismos como descarte de carga, controle de utilização de solicitações ou implementação de degradação simples.

Recursos

Práticas recomendadas relacionadas:

- REL05-BP01 Implementar uma degradação simples para transformar dependências rígidas aplicáveis em dependências flexíveis
- REL05-BP02 Controlar a utilização de solicitações
- REL11-BP05 Usar estabilidade estática para evitar o comportamento bimodal

Documentos relacionados:

- Automatizar uma implantação prática e sem intervenção manual

- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

Vídeos relacionados:

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

Gerenciamento de alterações

As alterações na carga de trabalho ou no respectivo ambiente devem ser previstas e acomodadas para alcançar uma operação confiável da carga de trabalho. As alterações incluem aquelas impostas à sua carga de trabalho, como picos na demanda, bem como aquelas internas, como implantações de recursos e patches de segurança.

As seções a seguir explicam as melhores práticas para o gerenciamento de alterações:

Tópicos

Monitorar os recursos da workload

Os logs e as métricas são uma ferramenta poderosa para saber a integridade das suas cargas de trabalho. Você pode configurar sua carga de trabalho para monitorar logs e métricas e enviar notificações quando os limites forem ultrapassados ou em caso de eventos importantes. O monitoramento permite que sua carga de trabalho reconheça quando limites de baixa performance são ultrapassados ou ocorrem falhas, para que ela possa se recuperar automaticamente em resposta.

O monitoramento é essencial para garantir que você esteja cumprindo seus requisitos de disponibilidade. Seu monitoramento precisa detectar falhas de modo eficaz. O pior modo de falha é a falha “silenciosa”, em que a funcionalidade não está mais ativa, mas não há como detectar isso a não ser indiretamente. Seus clientes sabem antes de você. Alertá-lo quando ocorrem problemas é um dos principais motivos para monitorar. Seus alertas devem ser desassociados dos sistemas o máximo possível. Se a interrupção no serviço não permitir que você receba alertas, o período de interrupção será maior.

Na AWS, instrumentamos nossas aplicações em vários níveis. Registramos latência, taxas de erros e disponibilidade para cada solicitação, para todas as dependências e para as principais operações no processo. Registramos métricas de operação bem-sucedida também. Isso nos permite ver problemas iminentes antes que ocorram. Não consideramos apenas a latência média. Nós nos concentramos ainda mais em exceções de latência, como 99,9 e 99,99 percentil. Isso ocorre porque,

se uma solicitação de 1.000 ou 10.000 for lenta, isso ainda será uma experiência ruim. Também, embora sua média possa ser aceitável, se uma a cada 100 das suas solicitações causar latência extrema, ele acabará se tornando um problema à medida que seu tráfego aumenta.

O monitoramento na AWS consiste em quatro fases distintas:

1. Geração – Monitorar todos os componentes da carga de trabalho
2. Agregação – Definir e calcular métricas
3. Processamento e alarmes em tempo real – Enviar notificações e automatizar respostas
4. Armazenamento e estudo analítico

Práticas recomendadas

- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP02 Definir e calcular as métricas (agregação)
- REL06-BP03 Envie notificações (processamento e emissão de alarmes em tempo real)
- REL06-BP04 Automatizar respostas (processamento e emissão de alarmes em tempo real)
- REL06-BP05 Análises
- REL06-BP06 Realizar revisões regularmente
- REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema

REL06-BP01 Monitorar todos os componentes da workload (geração)

monitore os componentes da carga de trabalho com o Amazon CloudWatch ou ferramentas de terceiros. Monitore os serviços da AWS com o painel do AWS Health.

Todos os componentes da carga de trabalho devem ser monitorados, incluindo front-end, lógica de negócios e níveis de armazenamento. Defina as principais métricas, descreva como extraí-las dos logs (se necessário) e defina limites de ativação para eventos de alarme correspondentes. Certifique-se de que as métricas sejam relevantes para os indicadores-chave de performance (KPIs) da workload e use métricas e logs para identificar os primeiros sinais de alerta de degradação do serviço. Por exemplo, uma métrica relacionada a resultados de negócios, como o número de pedidos processados com êxito por minuto, pode indicar problemas de workload mais rapidamente do que uma métrica técnica, como a utilização da CPU. Use o painel do AWS Health para uma visualização personalizada da performance e da disponibilidade dos serviços da AWS subjacentes aos recursos da AWS.

O monitoramento na nuvem oferece novas oportunidades. A maioria dos provedores de nuvem desenvolveu ganchos personalizáveis e pode entregar insights para ajudar você a monitorar várias camadas da workload. Serviços da AWS, como o Amazon CloudWatch, aplicam algoritmos estatísticos e de machine learning para analisar continuamente métricas de sistemas e de aplicações, determinam linhas de base normais e detectam anomalias com intervenção mínima do usuário. Os algoritmos de detecção de anomalias consideram a sazonalidade e as mudanças de tendência das métricas.

A AWS disponibiliza uma abundância de informações de monitoramento e de log para consumo, que podem ser usadas para definir métricas específicas de workload, processos de alteração sob demanda e adotar técnicas de machine learning, independentemente da experiência em ML.

Além disso, monitore todos os seus endpoints externos para garantir que eles sejam independentes de sua implementação de base. Este monitoramento ativo pode ser feito com transações sintéticas (às vezes chamadas de canários de usuário, mas que não devem ser confundido com implantações canário) que executam periodicamente um número de tarefas comuns que correspondem às ações realizadas pelos clientes da workload. Mantenha estas tarefas de curta duração e certifique-se de não sobrecarregar a workload durante o teste. O Amazon CloudWatch Synthetics permite criar canários sintéticos para monitorar seus endpoints e APIs. Você também pode combinar os nós sintéticos do cliente canário com o console do AWS X-Ray para identificar quais canários sintéticos estão enfrentando problemas com erros, falhas ou taxas de controle de utilização para o período selecionado.

Resultado desejado:

Coletar e usar métricas críticas de todos os componentes da workload para garantir sua confiabilidade e a experiência ideal do usuário. Detectar que uma workload não está alcançando resultados de negócios permite que você declare rapidamente um desastre e se recupere de um incidente.

Antipadrões comuns:

- Monitorar apenas as interfaces externas com sua carga de trabalho.
- Não gerar métricas específicas de workload e confiar apenas nas métricas fornecidas pelos serviços da AWS usados pela sua workload.
- Usar apenas métricas técnicas na workload e não monitorar nenhuma métrica relacionada a KPIs não técnicos para os quais a workload contribui.
- Depender do tráfego de produção e de verificações de integridade simples para monitorar e avaliar o estado da workload.

Benefícios do estabelecimento dessa prática recomendada: O monitoramento em todos os níveis da workload permite prever e resolver problemas mais rapidamente nos componentes que a compõem.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

1. Habilite o registro em log quando disponível. Os dados de monitoramento devem ser obtidos de todos os componentes das workloads. Ative o registro em log adicional, como os logs de acesso do S3, e habilite sua workload para registrar dados específicos da workload. Colete métricas para médias de CPU, E/S de rede e E/S de disco de serviços como o Amazon ECS, o Amazon EKS, o Amazon EC2, o Elastic Load Balancing, o AWS Auto Scaling e o Amazon EMR. Perceber Serviços da AWS que publicam métricas do CloudWatch para uma lista dos serviços da AWS que publicam métricas do CloudWatch.
2. Revise todas as métricas padrão e explore quaisquer lacunas na coleta de dados. Cada serviço gera métricas padrão. A coleta de métricas padrão permite que você entenda melhor as dependências entre os componentes da workload e como a confiabilidade e a performance destes componentes a afetam. Você também pode criar e publicar suas próprias métricas para CloudWatch usando o AWS CLI ou uma API. Isso
3. Avalie todas as métricas para decidir quais alertar para cada serviço da AWS na sua workload. Você pode escolher selecionar um subconjunto de métricas que tenha um grande impacto na confiabilidade da workload. Focar em métricas e limites críticos permite refinar o número de alertas de emergência e pode ajudar a minimizar falso-positivos.
4. Defina alertas e o processo de recuperação para a workload depois que o alerta for acionado. A definição de alertas permite que você notifique, escalone e siga rapidamente as etapas necessárias para se recuperar de um incidente e atender ao seu objetivo de tempo de recuperação (RTO) prescrito. Você pode usar o alarmes do Amazon CloudWatch para invocar fluxos de trabalho automatizados e iniciar procedimentos de recuperação com base em limites definidos.
5. Explore o uso de transações sintéticas para coletar dados relevantes sobre o estado das workloads. O monitoramento sintético segue as mesmas rotas e realiza as mesmas ações que um cliente, possibilitado que você verifique continuamente a experiência do cliente, mesmo quando não há tráfego de clientes nas workloads. Ao usar transações sintéticas, você pode descobrir problemas antes que seus clientes o façam.

Recursos

Práticas recomendadas relacionadas:

- REL11-BP03 Automatizar a reparação em todas as camadas

Documentos relacionados:

- Conceitos básicos do painel do AWS Health: integridade da sua conta
- Serviços da AWS que publicam métricas do CloudWatch
- Logs de acesso para o Network Load Balancer
- Logs de acesso para seu application load balancer
- Acessar o Amazon CloudWatch Logs para o AWS Lambda
- Registro em log de acesso ao servidor do Amazon S3
- Habilite logs de acesso para o Classic Load Balancer
- Exportação de dados de log para o Amazon S3
- Instalação do agente do CloudWatch em uma instância do Amazon EC2
- Publicar métricas personalizadas
- Uso de painéis do Amazon CloudWatch
- Uso de métricas do Amazon CloudWatch
- Uso de canários (Amazon CloudWatch Synthetics)
- O que é o Amazon CloudWatch Logs?

Guias do usuário:

- Criação de uma trilha
- Monitoramento de métricas de memória e de disco para instâncias do Linux do Amazon EC2
- Uso do CloudWatch Logs com instâncias de contêiner
- Logs de fluxo da VPC
- O que é o Amazon DevOps Guru?
- O que é o AWS X-Ray?

Blogs relacionados:

- Depuração com o Amazon CloudWatch Synthetics e o AWS X-Ray

Exemplos e workshops relacionados:

- Laboratórios do AWS Well-Architected: excelência operacional: monitoramento de dependência
- A Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- Workshop de observabilidade

REL06-BP02 Definir e calcular as métricas (agregação)

Armazene os dados de log e aplique filtros quando necessário para calcular métricas, como contagens de um evento de log específico ou latência calculada com base na data e hora dos eventos de log.

O Amazon CloudWatch e o Amazon S3 funcionam como camadas primárias de agregação e armazenamento. Para alguns serviços, como o AWS Auto Scaling e o Elastic Load Balancing, métricas padrão são fornecidas para carga de CPU ou latência média de solicitação em um cluster ou uma instância. Para serviços de streaming, como o VPC Flow Logs e o AWS CloudTrail, dados de evento são encaminhados ao CloudWatch Logs, e você precisa definir e aplicar filtros de métricas para extraí-las dos dados do evento. Isso fornece dados de séries temporais, que podem servir como entradas para alarmes do CloudWatch que você define para acionar alertas.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

- Defina e calcule as métricas (agregação). Armazene os dados de log e aplique filtros quando necessário para calcular métricas como contagens de um evento de log específico ou latência calculada com base na data e hora dos eventos de log
  - Os filtros de métrica definem os termos e os padrões a serem procurados nos dados de log à medida que são enviados para o CloudWatch Logs. O CloudWatch Logs usa esses filtros para transformar dados de log em métricas numéricas do CloudWatch, que você pode representar graficamente ou para as quais pode definir um alarme.
    - Pesquisa e filtragem de dados de log
  - Use um terceiro confiável para agregar logs.
    - Siga as instruções do terceiro. A maioria dos produtos de terceiros integra-se ao CloudWatch e ao Amazon S3.

- Alguns serviços da AWS podem publicar logs diretamente no Amazon S3. Se seu principal requisito de logs for o armazenamento no Amazon S3, você poderá facilmente fazer com que o serviço que produz os logs os envie diretamente ao Amazon S3 sem configurar uma infraestrutura adicional.
  - Envie logs diretamente ao Amazon S3

Recursos

Documentos relacionados:

- Consultas de exemplo do Amazon CloudWatch Logs Insights
- Depuração com o Amazon CloudWatch Synthetics e o AWS X-Ray
- Um workshop de observabilidade
- Pesquisa e filtragem de dados de log
- Envie logs diretamente ao Amazon S3
- A Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional

REL06-BP03 Envie notificações (processamento e emissão de alarmes em tempo real)

Quando as organizações detectam possíveis problemas, elas enviam notificações e alertas em tempo real para a equipe e os sistemas apropriados para responder de forma rápida e eficaz a esses problemas.

Resultado desejado: respostas rápidas a eventos operacionais são possíveis por meio da configuração de alarmes relevantes com base nas métricas de serviços e aplicações. Quando os limites do alarme são violados, o pessoal e os sistemas apropriados são notificados para que possam resolver os problemas subjacentes.

Antipadrões comuns:

- Configuração de alarmes com um limite excessivamente alto, resultando em falha no envio de notificações vitais.
- Configurar alarmes com um limite muito baixo, ocasionando inatividade diante de alertas importantes devido ao ruído de notificações excessivas.
- Não atualizar os alarmes e seu limite quando o uso muda.

- Para alarmes mais bem abordados por meio de ações automatizadas, enviar a notificação ao pessoal em vez de gerar a ação automatizada gera o envio excessivo de notificações.

Benefícios de estabelecer esta prática recomendada: O envio de notificações e alertas em tempo real para o pessoal e os sistemas apropriados permite a detecção precoce de problemas e respostas rápidas aos incidentes operacionais.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

As workloads devem ser equipadas com processamento e alarmes em tempo real para melhorar a detectabilidade de problemas que possam afetar a disponibilidade da aplicação e servir como gatilhos para respostas automatizadas. As organizações podem realizar processamento e alarmes em tempo real criando alertas com métricas definidas para receber notificações sempre que eventos significativos ocorrerem ou quando uma métrica ultrapassar um limite.

Amazon CloudWatch permite criar alarmes de métricas e compostos usando alarmes do CloudWatch com base em limite estático, detecção de anomalias e outros critérios. Para obter mais detalhes sobre os tipos de alarme que você pode configurar usando o CloudWatch, consulte a seção de alarmes da documentação do CloudWatch.

É possível criar visualizações personalizadas de métricas e alertas dos recursos da AWS para as equipes usando painéis do CloudWatch. As páginas iniciais personalizáveis no console do CloudWatch permitem que você monitore seus recursos em uma única visualização em várias regiões.

Os alarmes podem realizar uma ou mais ações, como enviar uma notificação a um tópico do Amazon SNS, realizando uma ação do Amazon EC2 ou do Amazon EC2 Auto Scaling , ou criando um OpsItem ou incidente no AWS Systems Manager.

O Amazon CloudWatch usa o Amazon SNS para enviar notificações quando o alarme muda de estado, fornecendo a entrega de mensagens dos publicadores (produtores) para os assinantes (consumidores). Para obter mais detalhes sobre como configurar notificações do Amazon SNS, consulte Configuring Amazon SNS.

O CloudWatch envia eventos do EventBridge segurança sempre que um alarme do CloudWatch é criado, atualizado, excluído ou o estado muda. É possível usar o EventBridge com esses eventos para criar regras que realizam ações, como enviar uma notificação sempre que o estado de

um alarme mudar ou acionar eventos automaticamente na conta usando o Systems Manager Automation.

Quando você deve usar o EventBridge ou o Amazon SNS?

Tanto o EventBridge quanto o Amazon SNS podem ser usados para desenvolver aplicações orientadas a eventos, e sua escolha dependerá de suas necessidades específicas.

O Amazon EventBridge é recomendado quando você deseja criar uma aplicação que reaja a eventos de suas próprias aplicações, aplicações SaaS e serviços da AWS. O EventBridge é o único serviço baseado em eventos que se integra diretamente com parceiros SaaS de terceiros. O EventBridge também ingere automaticamente eventos de mais de 200 serviços da AWS sem exigir que os desenvolvedores criem recursos em suas contas.

O EventBridge usa uma estrutura definida baseada em JSON para eventos e ajuda você a criar regras que são aplicadas em todo o corpo do evento para selecionar eventos a serem encaminhados a um destino. O EventBridge no momento é compatível com mais de vinte serviços da AWS como destinos, incluindo AWS Lambda, o Amazon SQS, Amazon SNS, Amazon Kinesis Data Streamse o Amazon Data Firehose.

O Amazon SNS é recomendado para aplicações que precisam de alta distribuição (milhares ou milhões de endpoints). Um padrão comum que vemos é que os clientes usam o Amazon SNS como destino para a regra a fim de filtrar os eventos de que precisam e distribuí-los para vários endpoints.

As mensagens não são estruturadas e podem estar em qualquer formato. O Amazon SNS permite o encaminhamento de mensagens a seis tipos diferentes de destinos, incluindo Lambda, Amazon SQS, endpoints HTTP/S, SMS, push de dispositivos móveis e e-mail. A latência típica do Amazon SNS é inferior a 30 milissegundos. Uma ampla variedade de serviços da AWS envia ao Amazon SNS mensagens configurando o serviço para fazer isso (mais de trinta, incluindo o Amazon EC2, o Amazon S3e o Amazon RDS).

Etapas da implementação

1. Crie um alarme usando alarmes do Amazon CloudWatch.
    a. Um alarme de métrica monitora uma única métrica do CloudWatch ou uma expressão dependente de métricas do CloudWatch. O alarme inicia uma ou mais ações com base no valor da métrica ou expressão em comparação com um limite em vários intervalos de tempo. A ação pode consistir em enviar uma notificação a um tópico do Amazon SNS, realizando uma ação do Amazon EC2 ou do Amazon EC2 Auto Scaling , ou criando um OpsItem ou incidente no AWS Systems Manager.

   b. Um alarme composto consiste em uma expressão de regra que considera as condições de alarme de outros alarmes que você criou. O alarme composto só entra no estado de alarme se todas as condições da regra forem atendidas. Os alarmes especificados na expressão da regra de um alarme composto podem incluir alarmes de métricas e outros alarmes compostos. Alarmes compostos podem enviar notificações do Amazon SNS quando o estado muda e podem criar Systems Manager OpsItems ou incidentes quando entram no estado de alarme, mas não conseguem realizar ações do Amazon EC2 ou do Auto Scaling.

2. Configure o Notificações do Amazon SNS. Ao criar um alarme do CloudWatch, é possível incluir um tópico do Amazon SNS para enviar uma notificação quando o alarme mudar de estado.
3. Crie regras no EventBridge que corresponde aos alarmes do CloudWatch especificados. Cada regra é compatível com vários destinos, incluindo funções do Lambda. Por exemplo, você pode definir um alarme que é iniciado quando o espaço disponível em disco está acabando, o que aciona uma função do Lambda por meio de uma regra do EventBridge para limpar o espaço. Para obter mais detalhes sobre destinos do EventBridge, consulte EventBridge targets.

Recursos

Práticas recomendadas relacionadas ao Well-Architected:

- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP02 Definir e calcular as métricas (agregação)
- REL12-BP01 Usar playbooks para investigar falhas

Documentos relacionados:

- Amazon CloudWatch
- CloudWatch Logs insights
- Using Amazon CloudWatch alarms
- Using Amazon CloudWatch dashboards
- Using Amazon CloudWatch metrics (Uso de métricas do Amazon CloudWatch)
- Setting up Amazon SNS notifications
- detecção de anomalias do CloudWatch
- CloudWatch Logs data protection
- Amazon EventBridge

- Amazon Simple Notification Service

Vídeos relacionados:

- reinvent 2022 observability videos (Vídeos sobre observabilidade do AWS re:Invent 2022)
- AWS re:Invent 2022 - Observability best practices at Amazon (AWS re:Invent 2022: práticas recomendadas de observabilidade na Amazon)

Exemplos relacionados:

- Um workshop de observabilidade
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

REL06-BP04 Automatizar respostas (processamento e emissão de alarmes em tempo real)

Use a automação para executar uma ação quando um evento é detectado, por exemplo, para substituir componentes com falha.

O processamento automatizado de alarmes em tempo real é implementado para que os sistemas possam tomar medidas corretivas rapidamente e tentar evitar falhas ou degradação dos serviços quando os alarmes são acionados. As respostas automatizadas a alarmes podem incluir a substituição de componentes com falha, o ajuste da capacidade computacional, o redirecionamento do tráfego para hosts, zonas de disponibilidade ou outras regiões íntegras e a notificação dos operadores.

Resultado desejado: os alarmes em tempo real são identificados, e o processamento automatizado dos alarmes é configurado para invocar as ações apropriadas realizadas para manter os objetivos de nível de serviço e os acordos de serviço (SLAs). A automação pode variar de atividades de autorrecuperação de componentes individuais a failover de todo o site.

Antipadrões comuns:

- Não ter um inventário ou catálogo claro dos principais alarmes em tempo real.
- Não haver respostas automatizadas para alarmes essenciais (por exemplo, quando a computação está quase esgotada, ocorre o ajuste de escala automático).
- Usar ações contraditórias de resposta a alarmes.

- Não haver procedimentos operacionais padrão (SOPs) para os operadores seguirem ao receberem notificações de alerta.
- Não monitorar as alterações da configuração, pois alterações não detectadas podem causar tempo de inatividade nas workloads.
- Não haver uma estratégia para desfazer alterações não intencionais da configuração.

Benefícios do estabelecimento desta prática recomendada: a automatização do processamento de alarmes pode melhorar a resiliência do sistema. O sistema executa ações corretivas automaticamente, reduzindo as atividades manuais que permitem intervenções humanas sujeitas a erros. A workload opera, atende às metas de disponibilidade e reduz a interrupção do serviço.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

Para gerenciar alertas com eficiência e automatizar as respectivas respostas, categorize os alertas com base em sua criticidade e impacto, documente os procedimentos de resposta e planeje as respostas antes das tarefas de classificação.

Identifique tarefas que exigem ações específicas (geralmente detalhadas em runbooks) e examine todos os runbooks e manuais para determinar as tarefas que podem ser automatizadas. Geralmente, se for possível definir ações, elas poderão ser automatizadas. Se não for possível automatizar as ações, documente as etapas manuais em um SOP e treine os operadores a respeito. Conteste continuamente os processos manuais em busca de oportunidades de automação em que seja possível estabelecer e manter um plano para automatizar as respostas a alertas.

Etapas da implementação

1. Criar um inventário de alarmes: para obter uma lista de todos os alarmes, é possível utilizar a AWS CLI usando o comando do Amazon CloudWatch `describe-alarms`. Dependendo de quantos alarmes você configurou, talvez seja necessário usar paginação para recuperar um subconjunto de alarmes para cada chamada ou, alternativamente, utilizar o AWS SDK para obter os alarmes usando uma chamada de API.
2. Documentar todas as ações do alarme: atualize um runbook com todos os alarmes e as respectivas ações, independentemente de serem manuais ou automatizados. O AWS Systems Managerfornece runbooks predefinidos. Para obter mais informações sobre runbooks, consulte Working with runbooks. Para obter detalhes sobre como visualizar o conteúdo do runbook, consulte View runbook content.

3. Configurar e gerenciar ações de alarmes: para qualquer um dos alarmes que exijam uma ação, especifique a ação automatizada usando o SDK do CloudWatch. Por exemplo, é possível alterar o estado das instâncias do Amazon EC2 automaticamente com base em um alarme do CloudWatch criando e ativando ações em um alarme ou desativando ações em um alarme.

   Também é possível usar o Amazon EventBridge para responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicações ou alterações de recursos. Você pode criar regras para indicar em quais eventos tem interesse e as ações a serem executadas quando um evento corresponder a uma regra. As ações que podem ser iniciadas automaticamente incluem invocar um perfil do AWS Lambda, invocar o `Run Command` do Amazon EC2, transmitir o evento para o Amazon Kinesis Data Streams e visualizar o documento Automatizar o Amazon EC2 usando o EventBridge.

4. Procedimentos operacionais padrão (SOPs): com base nos componentes da aplicação, o AWS Resilience Hub recomenda vários modelos de SOP. É possível usar esses SOPs para documentar todos os processos que um operador deve seguir caso um alerta seja emitido. Também é possível estruturar um SOP com base nas recomendações do Resilience Hub, caso em que você precisa de uma aplicação do Resilience Hub com uma política de resiliência associada, bem como de uma avaliação histórica de resiliência em relação a essa aplicação. As recomendações para o SOP são produzidas pela avaliação de resiliência.

   O Resilience Hub trabalha com o Systems Manager para automatizar as etapas dos SOPs, fornecendo vários documentos do SSM que podem ser usados como base para esses SOPs. Por exemplo, o Resilience Hub pode recomendar um SOP para adicionar espaço em disco com base em um documento de automação existente do SSM.

5. Realizar ações automatizadas usando o Amazon DevOps Guru: é possível usar o Amazon DevOps Guru para monitorar automaticamente recursos da aplicação em busca de comportamento anômalo e fornecer recomendações direcionadas para acelerar o tempo de identificação e de correção de problemas. Com o DevOps Guru, é possível monitorar fluxos de dados operacionais quase em tempo real de várias fontes, incluindo as métricas do Amazon CloudWatch, o AWS Config, o AWS CloudFormation e o AWS X-Ray. Também é possível usar o DevOps Guru para criar OpsItems no OpsCenter e enviar eventos ao EventBridge para automação adicional.

Recursos

Práticas recomendadas relacionadas:

- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP02 Definir e calcular as métricas (agregação)
- REL06-BP03 Envie notificações (processamento e emissão de alarmes em tempo real)
- REL08-BP01 Usar runbooks para atividades padrão, como implantação

Documentos relacionados:

- AWS Systems Manager Automation
- Creating an EventBridge Rule That Triggers on an Event from an AWS Resource
- Um workshop de observabilidade
- A Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- What is Amazon DevOps Guru?
- Trabalhar com documentos de automação (playbooks)

Vídeos relacionados:

- AWS re:Invent 2022: Práticas recomendadas de observabilidade na Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction to AWS Resilience Hub
- Criar sistemas de tickets personalizados para notificações do Amazon DevOps Guru
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru

Exemplos relacionados:

- Reliability Workshops (Workshops sobre confiabilidade)
- Workshop do Amazon CloudWatch e do Systems Manager

REL06-BP05 Análises

colete arquivos de log e históricos de métricas e analise-os para obter tendências mais abrangentes e informações sobre a carga de trabalho.

O Amazon CloudWatch Logs oferece suporte a uma linguagem de consulta simples, mas poderosa que você pode usar para analisar dados de log. O Amazon CloudWatch Logs também

oferece suporte a assinaturas que permitem que os dados fluam perfeitamente ao Amazon S3, onde você pode usar o ou o Amazon Athena para consultar esses dados. Ele oferece suporte a consultas em uma grande variedade de formatos. Perceber Formatos de dados e SerDes compatíveis no guia do usuário do Amazon Athena para obter mais informações. Para análise de conjuntos enormes de arquivos de log, você pode executar um cluster do Amazon EMR para executar análises em escala de petabytes.

Existem várias ferramentas fornecidas por parceiros da AWS e por terceiros que permitem agregação, processamento, armazenamento e estudo analítico. Essas ferramentas incluem New Relic, Splunk, Loggly, Logstash, CloudHealth e Nagios. Porém, a geração fora dos registros do aplicativo e do sistema é única para cada provedor de nuvem e costuma ser única para cada serviço.

Uma parte do processo de monitoramento que costuma ser negligenciada é o gerenciamento de dados. Você precisa determinar os requisitos de retenção para monitorar os dados e então aplicar as políticas de ciclo de vida de acordo. O Amazon S3 oferece suporte ao gerenciamento de ciclo de vida no nível do bucket do S3. Esse gerenciamento de ciclo de vida pode ser aplicado de modo diferente a diferentes caminhos no bucket. Mais perto do fim do ciclo de vida, você pode fazer a transição dos dados ao Amazon S3 Glacier para armazenamento de longo prazo e posterior expiração após o fim do período de retenção. A classe de armazenamento S3 Intelligent-Tiering foi projetada para otimizar custos movendo automaticamente dados para o nível de acesso mais econômico, sem impacto na performance ou sobrecarga operacional.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- O CloudWatch Logs Insights permite pesquisar e analisar dinamicamente seus dados de log no Amazon CloudWatch Logs.
  - Análise de dados de log com o CloudWatch Logs Insights
  - Consultas de exemplo do Amazon CloudWatch Logs Insights
- Use o Amazon CloudWatch Logs para enviar logs para o Amazon S3, onde você pode usar o Amazon Athena para consultar dados.
  - Como analiso meus logs de acesso ao servidor do Amazon S3 usando o Athena?
    - Crie uma política de ciclo de vida do S3 para o bucket de logs de acesso ao seu servidor. Configure a política de ciclo de vida para remover periodicamente os arquivos de log. Esse procedimento reduz a quantidade de dados que o Athena analisa em cada consulta.
      - Como faço para criar uma política de ciclo de vida de um bucket do S3?

Recursos

Documentos relacionados:

- Consultas de exemplo do Amazon CloudWatch Logs Insights
- Análise de dados de log com o CloudWatch Logs Insights
- Depuração com o Amazon CloudWatch Synthetics e o AWS X-Ray
- Como faço para criar uma política de ciclo de vida de um bucket do S3?
- Como analiso meus logs de acesso ao servidor do Amazon S3 usando o Athena?
- Um workshop de observabilidade
- A Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional

REL06-BP06 Realizar revisões regularmente

Revise frequentemente a implementação do monitoramento da workload e atualize-a com base em eventos e alterações significativos.

O monitoramento eficaz é orientado pelas principais métricas de negócios. Certifique-se de que essas métricas sejam acomodadas em sua carga de trabalho à medida que as prioridades de negócios mudam.

Auditar seu monitoramento ajuda a garantir que você saiba quando um aplicativo está atingindo as respectivas metas de disponibilidade. A análise da causa raiz requer a capacidade de descobrir o que aconteceu quando ocorreram falhas. A AWS fornece serviços que permitem acompanhar o estado dos seus serviços durante um incidente:

- Amazon CloudWatch Logs: você pode armazenar seus logs nesse serviço e inspecionar seu conteúdo.
- Amazon CloudWatch Logs Insights: é um serviço totalmente gerenciado que permite analisar logs massivos em segundos. Ele oferece consultas e visualizações rápidas e interativas.
- AWS Config: você pode ver qual infraestrutura da AWS estava em uso em diferentes momentos.
- AWS CloudTrail: você pode ver quais APIs da AWS foram invocadas, a que horas e por qual entidade principal.

Na AWS, realizamos uma reunião semanal para revisar a performance operacional e para compartilhar aprendizados entre as equipes. Como há tantas equipes na AWS, criamos A roda para

escolher aleatoriamente uma carga de trabalho para revisão. Estabelecer um ritmo regular para análises de performance operacional e compartilhamento de conhecimento aprimora sua capacidade de obter uma performance superior de suas equipes operacionais.

Antipadrões comuns:

- Coletar apenas as métricas padrão.
- Definir uma estratégia de monitoramento e nunca revisá-la.
- Não analisar o monitoramento quando alterações importantes são implantadas.

Benefícios do estabelecimento dessa prática recomendada: A revisão regular do monitoramento permite a antecipação de possíveis problemas, em vez de reagir a notificações quando um problema previsto realmente ocorrer.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- Crie vários painéis para a workload. Você deve ter um painel superior com as principais métricas de negócios e as métricas técnicas identificadas como as mais relevantes à integridade projetada da carga de trabalho conforme a variação do uso. Você também deve ter painéis para vários níveis e dependências da aplicação que podem ser inspecionados.
  - Uso de painéis do Amazon CloudWatch
- Programe e realize revisões regulares dos painéis da workload. Realize uma inspeção regular dos painéis. Você pode ter graus diferentes de profundidade para a inspeção.
  - Inspecione as tendências nas métricas. Compare os valores das métricas com os valores históricos para ver se há tendências que possam indicar algo que precise de investigação. Exemplos disso incluem: aumento da latência, diminuição da função principal de negócios e aumento das respostas a falhas.
  - Verifique se há exceções ou anomalias nas suas métricas. As médias ou os valores medianos podem mascarar as exceções e as anomalias. Examine os valores mais altos e mais baixos durante o período e investigue as causas das pontuações extremas. À medida que você continua a eliminar essas causas, a redução da definição de extremo permite melhorar cada vez mais a consistência da performance da workload.
  - Procure mudanças bruscas no comportamento. Uma mudança imediata na quantidade ou na direção de uma métrica pode indicar que houve uma alteração na aplicação ou fatores externos aos quais você talvez precise adicionar outras métricas para acompanhar.

Recursos

Documentos relacionados:

- Consultas de exemplo do Amazon CloudWatch Logs Insights
- Depuração com o Amazon CloudWatch Synthetics e o AWS X-Ray
- Um workshop de observabilidade
- A Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- Uso de painéis do Amazon CloudWatch

REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema

Rastreie as solicitações à medida que elas são processadas por meio de componentes de serviço para que as equipes de produto possam analisar e depurar problemas com maior facilidade e melhorar a performance.

Resultado desejado: As workloads com rastreamento abrangente em todos os componentes são fáceis de depurar, o que melhora o tempo médio até a resolução (MTTR) de erros e latência simplificando a descoberta da causa raiz. O rastreamento completo reduz o tempo necessário para descobrir os componentes afetados e detalhar as causas raiz dos erros ou da latência.

Antipadrões comuns:

- O rastreamento é usado para alguns componentes, mas não para todos. Por exemplo, sem rastrear o AWS Lambda, as equipes podem não entender claramente a latência causada por partidas a frio em uma workload com picos.
- Canários sintéticos ou monitoramento de usuário real (RUM) não são configurados com rastreamento. Sem canários ou RUM, a telemetria de interação com o cliente é omitida da análise de rastreamento, gerando um perfil de performance incompleto.
- As workloads híbridas incluem ferramentas de rastreamento nativas da nuvem e de terceiros, mas ainda não foram tomadas medidas eletivas e integram totalmente uma única solução de rastreamento. Com base na solução de rastreamento escolhida, os SDKs de rastreamento nativos de nuvem devem ser usados para instrumentar componentes que não são nativos de nuvem ou ferramentas de terceiros devem ser configuradas para ingerir a telemetria de rastreamento nativa de nuvem.

Benefícios de estabelecer esta prática recomendada: Quando as equipes de desenvolvimento são alertadas sobre problemas, elas podem ter uma visão completa das interações dos componentes do sistema, incluindo a correlação componente por componente com registros em log, performance e falhas. Como o rastreamento facilita a identificação visual das causas raiz, menos tempo é gasto na investigação delas. As equipes que entendem detalhadamente as interações dos componentes tomam decisões melhores e mais rápidas ao resolver problemas. Decisões como quando invocar o failover de recuperação de desastres (DR) ou onde melhor implementar estratégias de autorrecuperação podem ser aprimoradas com a análise de rastreamentos de sistemas e aumentar a satisfação do cliente com seus serviços.

Nível de risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

As equipes que operam aplicações distribuídas podem usar ferramentas de rastreamento para estabelecer um identificador de correlação, coletar rastreamentos de solicitações e criar mapas de serviço dos componentes conectados. Todos os componentes da aplicação devem ser incluídos nos rastreamentos de solicitações, incluindo clientes de serviços, gateways de middleware e barramentos de eventos, componentes computacionais e armazenamento, incluindo armazenamentos de chave-valor e bancos de dados. Inclua canários sintéticos e monitoramento de usuários reais em sua configuração de rastreamento completo a fim de medir as interações remotas com clientes e a latência, para que você possa avaliar com precisão a performance de seus sistemas em relação aos seus objetivos e acordos de serviço.

Você pode usar o AWS X-Ray e os serviços de instrumentação de monitoramento de aplicações do Amazon CloudWatch para oferecer uma visão completa das solicitações à medida que elas passam por sua aplicação. O X-Ray coleta a telemetria da aplicação e permite que você a visualize e filtre em payloads, funções, rastreamentos, serviços, APIs e pode ser ativada para componentes do sistema com pouco ou nenhum código. O monitoramento de aplicações do CloudWatch inclui o ServiceLens para integrar seus rastreamentos a métricas, logs e alarmes. O monitoramento de aplicações do CloudWatch também inclui sintéticos a fim de monitorar seus endpoints e APIs, bem como monitoramento de usuários reais para instrumentar seus clientes de aplicações web.

Etapas da implementação

- Use o AWS X-Ray em todos os serviços nativos compatíveis, como Amazon S3, AWS Lambda e Amazon API Gateway. Esses serviços da AWS permitem ao X-Ray alternar a configuração usando a infraestrutura como código, AWS SDKs ou o AWS Management Console.

- Aplicações de instrumentos AWS Distro for Open Telemetry e X-Ray ou agentes de coleta de terceiros.
- Revise o Guia do desenvolvedor do AWS X-Ray para implementação específica da linguagem de programação. Essas seções da documentação detalham como instrumentar solicitações HTTP, consultas SQL e outros processos específicos de sua linguagem de programação de aplicações.
- Use o rastreamento do X-Ray para Canários sintéticos do Amazon CloudWatch e os Amazon CloudWatch RUM para analisar o caminho da solicitação de seu cliente de usuário final por meio de sua infraestrutura downstream da AWS.
- Configure métricas e alarmes do CloudWatch com base na integridade dos recursos e na telemetria canário para que as equipes sejam alertadas sobre problemas rapidamente e, depois, possam se aprofundar em rastreamentos e mapas de serviços com o ServiceLens.
- Habilite a integração do X-Ray a ferramentas de rastreamento de terceiros, como Datadog, New Relicou Dynatrace se você estiver usando ferramentas de terceiros para sua solução de rastreamento principal.

Recursos

Práticas recomendadas relacionadas:

- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas

Documentos relacionados:

- O que é o AWS X-Ray?
- Amazon CloudWatch: Monitoramento de aplicações
- Depuração com o Amazon CloudWatch Synthetics e o AWS X-Ray
- A Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- Integrar o AWS X-Ray a outros serviços da AWS
- AWS Distro for OpenTelemetry e AWS X-Ray
- Amazon CloudWatch: uso do monitoramento sintético
- Amazon CloudWatch: usar o CloudWatch RUM
- Configurar o canário sintético do Amazon CloudWatch e o alarme do Amazon CloudWatch
- Disponibilidade e além: compreensão e melhoria da resiliência de sistemas distribuídos na AWS

Exemplos relacionados:

- Um workshop de observabilidade

Vídeos relacionados:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts (re:Invent 2022: Como monitorar aplicações em várias contas)
- How to Monitor your AWS Applications (Como monitorar suas aplicações da AWS)

Ferramentas relacionadas:

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

Projetar a workload de modo que ela se adapte às alterações na demanda

Uma carga de trabalho escalável oferece elasticidade para adicionar ou remover recursos automaticamente para que atendam melhor à demanda atual a qualquer momento.

Práticas recomendadas

- REL07-BP01 Usar a automação ao obter ou escalar recursos
- REL07-BP02 Obter recursos após a detecção de danos em uma workload
- REL07-BP03 Obter recursos após a detecção de que mais recursos são necessários para uma workload
- REL07-BP04 Fazer o teste de carga da sua workload

REL07-BP01 Usar a automação ao obter ou escalar recursos

Ao substituir recursos danificados ou escalar sua workload, automatize o processo por meio dos serviços gerenciados pela AWS, como o Amazon S3 e o AWS Auto Scaling. Você também pode usar ferramentas de terceiros e os AWS SDKs para automatizar a escalabilidade.

Os serviços gerenciados pela AWS incluem o Amazon S3, o Amazon CloudFront, o AWS Auto Scaling, o AWS Lambda, o Amazon DynamoDB, o AWS Fargate e o Amazon Route 53.

O AWS Auto Scaling permite detectar e substituir instâncias danificadas. Ele também permite criar planos de escalabilidade para recursos, incluindo instâncias e frotas Spot do Amazon EC2 , tarefas do Amazon ECS tabelas e índices do Amazon DynamoDB e réplicas do Amazon Aurora .

Ao escalar instâncias do EC2, certifique-se de usar várias zonas de disponibilidade (de preferência, pelo menos três) e adicione ou remova capacidade para manter o equilíbrio entre essas zonas de disponibilidade. Tarefas do ECS ou pods do Kubernetes (ao usar o Amazon Elastic Kubernetes Service) também devem ser distribuídos em várias zonas de disponibilidade.

Ao usar o AWS Lambda, as instâncias são escaladas automaticamente. Sempre que uma notificação de evento é recebida para sua função, o AWS Lambda localiza rapidamente a capacidade livre dentro de sua frota de computação e executa seu código até a simultaneidade alocada. Você precisa se certificar de que a simultaneidade necessária esteja configurada no Lambda específico e no seu Service Quotas.

O Amazon S3 escala automaticamente para lidar com altas taxas de solicitação. Por exemplo, seu aplicativo pode atingir pelo menos 3.500 solicitações PUT/COPY/POST/DELETE ou 5.500 solicitações GET/HEAD por segundo por prefixo em um bucket. Não há limites para o número de prefixos em um bucket. Você pode aumentar a performance de leitura ou gravação paralelizando as leituras. Por exemplo, se você criar 10 prefixos em um bucket do Amazon S3 para paralelizar leituras, poderá escalar sua performance de leitura para 55 mil solicitações de leitura por segundo.

Configure e use o Amazon CloudFront ou uma rede de entrega de conteúdo (CDN) confiável. Uma CDN pode fornecer tempos mais rápidos de resposta ao usuário final e atender às solicitações de conteúdo do cache, reduzindo a necessidade de escalar a workload.

Antipadrões comuns:

- Implementar grupos de Auto Scaling para autorreparação, mas não implementar elasticidade.
- Usar a escalabilidade automática para responder a grandes aumentos no tráfego.
- Implantar aplicativos altamente com estado, eliminando a opção de elasticidade.

Benefícios do estabelecimento dessa prática recomendada: A automação elimina a possibilidade de erros manuais na implantação e no descomissionamento de recursos. A automação remove o risco de custos excedentes e de negação de serviço decorrentes da lentidão na resposta às necessidades de implantação ou de descomissionamento.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

- Configure e use o AWS Auto Scaling. Ele monitora seus aplicativos e ajusta automaticamente a capacidade para manter uma performance estável e previsível com o menor custo possível. Ao usar o AWS Auto Scaling, você pode configurar a escalabilidade da aplicação para vários recursos em diversos serviços.
  - O que é o AWS Auto Scaling?
    - Configure o Auto Scaling nas instâncias do Amazon EC2 e frotas spot, nas tarefas do Amazon ECS, nas tabelas e índices do Amazon DynamoDB, nas réplicas do Amazon Aurora e nos dispositivos do AWS Marketplace, conforme aplicável.
      - Gerenciamento da capacidade de throughput de modo automático com o DynamoDB Auto Scaling
        - Use as operações de API de serviço para especificar alarmes, políticas de escalabilidade e tempos de aquecimento e de resfriamento.
- Use o Elastic Load Balancing. Os load balancers podem distribuir a carga por caminho ou por conectividade de rede.
  - O que é o Elastic Load Balancing?
    - O Application Load Balancers pode distribuir a carga por caminho.
      - O que é um Application Load Balancer?
        - Configure um Application Load Balancer para distribuir o tráfego para diferentes workloads com base no caminho sob o nome de domínio.
        - É possível usar os Application Load Balancers para distribuir as cargas de maneira integrada ao AWS Auto Scaling para gerenciar a demanda.
          - Uso de um balanceador de carga com um grupo de Auto Scaling
    - Os Network Load Balancers podem distribuir a carga por conexão.
      - O que é um Network Load Balancer?
        - Configure um Network Load Balancer para distribuir o tráfego para cargas de trabalho diferentes por meio do TCP ou para ter um conjunto constante de endereços IP para a carga de trabalho.
        - É possível usar os Network Load Balancers para distribuir as cargas de maneira integrada ao AWS Auto Scaling para gerenciar a demanda.

- Use um provedor DNS altamente disponível. Nomes DNS permitem que os usuários insiram nomes, em vez de endereço IP, para acessar suas workloads e distribuem essas informações a um escopo definido, em geral, globalmente para usuários da workload.
  - Use o Amazon Route 53 ou um provedor DNS confiável.
    - O que é o Amazon Route 53?
  - Use o Route 53 para gerenciar as distribuições e os balanceadores de carga do CloudFront.
    - Determine os domínios e subdomínios que serão gerenciados.
    - Crie conjuntos de registros adequados com os registros ALIAS ou CNAME.
      - Trabalhando com registros
- Use a rede global da AWS para otimizar o caminho dos usuários às aplicações. O AWS Global Accelerator monitora continuamente a integridade dos endpoints da aplicação e redireciona o tráfego para endpoints íntegros em menos de 30 segundos.
  - O AWS Global Accelerator é um serviço que melhora a disponibilidade e a performance das aplicações com usuários locais ou globais. Ele fornece endereços IP estáticos que atuam como um ponto de entrada fixo para os endpoints da aplicação em uma ou várias Regiões da AWS, como os Application Load Balancers, os Network Load Balancers ou as instâncias do Amazon EC2.
    - O que é o AWS Global Accelerator?
- Configure e use o Amazon CloudFront ou uma rede de entrega de conteúdo (CDN) confiável. Uma rede de entrega de conteúdo pode fornecer tempos mais rápidos de resposta ao usuário final e atender às solicitações de conteúdo que podem causar escalabilidade desnecessária das suas workloads.
  - O que é o Amazon CloudFront?
    - Configure as distribuições do Amazon CloudFront para suas workloads ou use uma CDN de terceiros.
      - Você pode limitar o acesso às workloads para que elas sejam acessíveis somente pelo CloudFront usando os intervalos de IPs para o CloudFront nos seus grupos de segurança ou suas políticas de acesso de endpoint.

Recursos

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudá-lo a criar soluções de computação automatizadas

- AWS Auto Scaling: como funcionam os planos de escalabilidade
- AWS Marketplace: produtos que podem ser usados com Auto Scaling
- Gerenciamento da capacidade de throughput de modo automático com o DynamoDB Auto Scaling
- Uso de um balanceador de carga com um grupo de Auto Scaling
- O que é o AWS Global Accelerator?
- O que é o Amazon EC2 Auto Scaling?
- O que é o AWS Auto Scaling?
- O que é o Amazon CloudFront?
- O que é o Amazon Route 53?
- O que é o Elastic Load Balancing?
- O que é um Network Load Balancer?
- O que é um Application Load Balancer?
- Trabalhando com registros

REL07-BP02 Obter recursos após a detecção de danos em uma workload

Escale recursos de modo reativo quando necessário, se a disponibilidade for afetada, para restaurar a disponibilidade da carga de trabalho.

Primeiro, você deve configurar as verificações de integridade e os critérios nessas verificações para indicar quando a disponibilidade é afetada pela falta de recursos. Notifique o pessoal apropriado para escalar manualmente o recurso ou inicie a automação para escalá-lo automaticamente.

A escala pode ser ajustada manualmente para a workload (por exemplo, alterando o número de instâncias do EC2 em um grupo do Auto Scaling ou modificando o throughput de uma tabela do DynamoDB por meio do AWS Management Console ou da AWS CLI). No entanto, a automação deve ser usada sempre que possível (consulte Usar automação ao obter ou escalar recursos).

Resultado desejado: as atividades de ajuste de escala (de forma automática ou manual) são iniciadas para restaurar a disponibilidade após a detecção de uma falha ou de uma degradação da experiência do cliente.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

Implemente a observabilidade e o monitoramento em todos os componentes da workload para monitorar a experiência do cliente e detectar falhas. Defina os procedimentos, manuais ou automatizados, que escalam os recursos necessários. Para obter mais informações, consulte REL11-BP01 Monitorar todos os componentes da workload para detectar falhas.

Etapas da implementação

- Definir os procedimentos, manuais ou automatizados, que escalam os recursos necessários.
  - Os procedimentos de ajuste de escala dependem de como os diferentes componentes da workload são projetados.
  - Eles também variam dependendo da tecnologia subjacente utilizada.
    - Os componentes que usam o AWS Auto Scaling podem utilizar planos de ajuste de escala para configurar um conjunto de instruções para escalar os recursos. Se você trabalha com o AWS CloudFormation ou adiciona tags aos recursos da AWS, poderá configurar planos de ajuste de escala para diferentes conjuntos de recursos por aplicação. O Auto Scaling fornece recomendações para estratégias de ajuste de escala personalizadas para cada recurso. Depois que o plano de ajuste de escala for criado, o Auto Scaling combinará os métodos de ajuste de escala dinâmica e preditiva para compatibilidade com a estratégia de ajuste de escala. Para obter mais detalhes, consulte Como funcionam os planos de escalabilidade.
    - O Amazon EC2 Auto Scaling verifica se você tem o número correto de instâncias do Amazon EC2 disponíveis para processar a carga da aplicação. Você cria coleções de instâncias do EC2, chamadas de grupos do Auto Scaling. É possível especificar o número mínimo e máximo de instâncias em cada grupo do Auto Scaling, e o Amazon EC2 Auto Scaling garantirá que o grupo nunca fique abaixo ou acima desses limites. Para obter mais detalhes, consulte O que é o Amazon EC2 Auto Scaling?
    - O ajuste de escala automático do Amazon DynamoDB usa o serviço Application Auto Scaling para ajustar dinamicamente a capacidade de throughput provisionado por você, em resposta a padrões de tráfego reais. Isso permite que uma tabela ou um índice secundário global aumente a capacidade provisionada de leitura e gravação para sustentar aumentos repentinos no tráfego, sem controle de utilização. Para obter mais detalhes, consulte Gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB.

Recursos

Práticas recomendadas relacionadas:

- REL07-BP01 Usar a automação ao obter ou escalar recursos
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas

Documentos relacionados:

- AWS Auto Scaling: Como funcionam os planos de escalabilidade
- Gerenciar a capacidade de throughput automaticamente com o Auto Scaling do DynamoDB
- O que é o Amazon EC2 Auto Scaling?

REL07-BP03 Obter recursos após a detecção de que mais recursos são necessários para uma workload

Escale os recursos proativamente para atender à demanda e evitar impacto na disponibilidade.

Muitos serviços da AWS são escalados automaticamente para atender à demanda. Se estiver usando instâncias do Amazon EC2 ou clusters do Amazon ECS, você poderá configurar a escalabilidade automática deles para que ocorra com base nas métricas de uso que correspondam à demanda da workload. Para o Amazon EC2, a utilização média da CPU, a contagem de solicitações do load balancer ou a largura de banda da rede podem ser usadas para expandir (ou reduzir) instâncias do EC2. Para o Amazon ECS, a utilização média da CPU, a contagem de solicitações do balanceador de carga e a utilização da memória podem ser usadas para aumentar (ou reduzir) a escala horizontalmente de tarefas do ECS. Ao usar o Target Auto Scaling na AWS, o Autoscaler atua como um termostato doméstico, adicionando ou removendo recursos para manter o valor pretendido (por exemplo, 70% de utilização da CPU) que você especificar.

O AWS Auto Scaling também pode fazer o Auto Scaling preditivo, que usa machine learning para analisar a carga de trabalho histórica de cada recurso e prevê regularmente a carga futura para os próximos dois dias.

A Lei de Little ajuda a calcular quantas instâncias de computação (instâncias do EC2, funções simultâneas do Lambda etc.) são necessárias.

$B = \lambda W$

L = número de instâncias (ou simultaneidade média no sistema)

λ = taxa média na qual as solicitações chegam (requisição por segundo)

W = tempo médio que cada solicitação gasta no sistema (s)

Por exemplo, a 100 rps, se cada solicitação demorar 0,5 segundos para ser processada, você precisará de 50 instâncias para acompanhar a demanda.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- Obtenha recursos após a detecção de que mais recursos são necessários para uma workload. Escale os recursos proativamente para atender à demanda e evitar impacto na disponibilidade.
  - Calcule quantos recursos de computação serão necessários (simultaneidade de computação) para processar uma determinada taxa de solicitações.
    - Histórias sobre a Lei de Little
  - Quando você tiver um padrão histórico de uso, configure a escalabilidade programada para a escalabilidade automática do Amazon EC2.
    - Escalabilidade programada para o Amazon EC2 Auto Scaling
  - Use a escalabilidade preditiva da AWS.
    - Escalabilidade preditiva para o EC2 com Machine Learning

Recursos

Documentos relacionados:

- AWS Auto Scaling: como funcionam os planos de escalabilidade
- AWS Marketplace: produtos que podem ser usados com Auto Scaling
- Gerenciamento da capacidade de throughput de modo automático com o DynamoDB Auto Scaling
- Escalabilidade preditiva para o EC2 com Machine Learning
- Escalabilidade programada para o Amazon EC2 Auto Scaling
- Histórias sobre a Lei de Little
- O que é o Amazon EC2 Auto Scaling?

REL07-BP04 Fazer o teste de carga da sua workload

Adote uma metodologia de teste de carga para avaliar se a ação de escalabilidade atende aos requisitos da carga de trabalho.

É importante realizar testes de carga sustentada. Os testes de carga devem descobrir o ponto de interrupção e testar a performance da workload. A AWS facilita a configuração de ambientes de teste temporários que modelam a escala de sua workload de produção. Na nuvem, você pode criar um ambiente de teste em escala de produção sob demanda, concluir seus testes e descomissionar os recursos. Como você paga somente pelo ambiente de teste quando está em execução, é possível simular seu ambiente ativo por uma fração do custo dos testes on-premises.

Os testes de carga em produção também devem ser considerados como parte dos dias de jogos em que o sistema de produção é destacado, durante horas de menor utilização do cliente, com todo o pessoal disponível para interpretar os resultados e resolver os problemas que surgirem.

Antipadrões comuns:

- Executar testes de carga em implantações que não têm a mesma configuração da sua produção.
- Executar testes de carga apenas em componentes individuais da carga de trabalho, e não nela toda.
- Executar testes de carga com um subconjunto de solicitações, e não com um conjunto representativo de solicitações reais.
- Executar testes de carga para um pequeno fator de segurança acima da carga esperada.

Benefícios de estabelecer esta prática recomendada: Você sabe quais componentes em sua arquitetura falham sob carga e pode identificar as métricas que devem ser observadas para indicar que você está se aproximando dessa carga a tempo de resolver o problema, evitando o impacto dessa falha.

Nível de risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

- Realize testes de carga para identificar qual aspecto da workload indica que é necessário adicionar ou remover capacidade. Os testes de carga devem ter tráfego representativo semelhante ao que você recebe na produção. Aumente a carga enquanto observa as métricas que você preparou para determinar aquelas que indicam quando é necessário adicionar ou remover recursos.
  - Teste de carga distribuída na AWS: simular milhares de usuários conectados
    - Identifique a combinação de solicitações. Você pode ter diversas combinações de solicitações, portanto, deve examinar vários períodos ao identificar a combinação de tráfego.

- Implemente um direcionador de carga. Você pode usar um código personalizado, um código aberto ou um software comercial para implementar um direcionador de carga.
- Faça o teste de carga inicialmente com uma pequena capacidade. Você vê alguns efeitos imediatos ao direcionar a carga para uma capacidade menor, possivelmente tão pequena quanto uma instância ou um contêiner.
- Faça o teste de carga com uma capacidade maior. Os efeitos serão diferentes em uma carga distribuída, portanto, você deve testar o mais próximo possível de um ambiente de produto.

Recursos

Documentos relacionados:

- Teste de carga distribuída na AWS: simular milhares de usuários conectados
- Load testing applications

Vídeos relacionados:

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

Implementar alterações

Alterações controladas são necessárias para implantar novas funcionalidades e garantir que as cargas de trabalho e o ambiente operacional estejam executando software conhecido e corrigido corretamente. Se essas alterações forem descontroladas, será difícil prever o efeito ou resolver problemas decorrentes delas.

Práticas recomendadas

- REL08-BP01 Usar runbooks para atividades padrão, como implantação
- REL08-BP02 Integrar testes funcionais como parte da sua implantação
- REL08-BP03 Integrar testes de resiliência como parte da sua implantação
- REL08-BP04 Implantar usando infraestrutura imutável
- REL08-BP05 Implantar alterações com automação

REL08-BP01 Usar runbooks para atividades padrão, como implantação

Os runbooks são os procedimentos predefinidos para alcançar um resultado específico. Use-os para executar atividades padrão, sejam elas feitas manualmente ou automaticamente. Os exemplos incluem a implantação de uma workload, a aplicação de patches a ela ou a realização de modificações de DNS.

Por exemplo, coloque processos em vigor para garantir a segurança de reversão durante implantações. Garantir que você possa reverter uma implantação sem qualquer interrupção para seus clientes é essencial para tornar um serviço confiável.

Para procedimentos de runbooks, comece com um processo manual efetivo válido, implemente-o em código e acione-o para ser executado automaticamente quando adequado.

Mesmo para cargas de trabalho sofisticadas altamente automatizadas, os runbooks ainda são úteis para organizar dias de jogos ou atender a requisitos rigorosos de relatórios e auditoria.

Observe que playbooks são usados em resposta a incidentes específicos, e runbooks são usados para alcançar resultados específicos. Muitas vezes, os runbooks são para atividades de rotina, enquanto os playbooks são usados para responder a eventos que não são rotineiras.

Antipadrões comuns:

- Executar alterações não planejadas na configuração em produção.
- Ignorar as etapas do seu plano para agilizar a implantação, resultando em falha na implantação.
- Fazer alterações sem testar a inversão delas.

Benefícios do estabelecimento desta prática recomendada: O planejamento eficaz da alteração aumenta sua capacidade de executá-la com êxito, porque você está ciente de todos os sistemas afetados. A validação da alteração em ambientes de teste aumenta sua confiança.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

- Documente os procedimentos em runbooks para permitir respostas consistentes e rápidas a eventos bem conhecidos.
  - AWS Well-Architected Framework: conceitos: runbook

- Use o princípio de infraestrutura como código para definir sua infraestrutura. Ao usar o AWS CloudFormation (ou um terceiro confiável) para definir a infraestrutura, você poderá usar o software de controle de versão para controlar as versões e acompanhar as alterações.
  - Use o AWS CloudFormation (ou um provedor confiável de terceiros) para definir sua infraestrutura.
    - O que é o AWS CloudFormation?
  - Use bons princípios de design de software para criar modelos exclusivos e desacoplados.
    - Determine as permissões, os modelos e as partes responsáveis pela implementação.
      - Controle de acesso com o AWS Identity and Access Management
    - Use o controle de origem, como o AWS CodeCommit ou uma ferramenta confiável de terceiros, para controle de versão.
      - O que é o AWS CodeCommit?

Recursos

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudá-lo a criar soluções de implantação automatizada
- AWS Marketplace: produtos que podem ser usados para automatizar suas implantações
- AWS Well-Architected Framework: conceitos: runbook
- O que é o AWS CloudFormation?
- O que é o AWS CodeCommit?

Exemplos relacionados:

- Automatização de operações com playbooks e runbooks

REL08-BP02 Integrar testes funcionais como parte da sua implantação

Os testes funcionais são executados como parte da implantação automatizada. Se os critérios de êxito não forem atendidos, o pipeline será interrompido ou revertido. Esses testes são executados em um ambiente de pré-produção, que é preparado antes da produção no pipeline. Idealmente, isso é feito como parte de um pipeline de implantação.

Resultado desejado: você usa a automação para realizar testes funcionais, e os dados de teste associados reduzem a duração e as despesas dos testes e melhoram a precisão dos resultados.

Você integra testes funcionais como parte do processo de implantação, o que ajuda a automatizar os canais de lançamento para atualizações rápidas e confiáveis de aplicações e infraestrutura.

Antipadrões comuns:

- Você realiza testes manualmente fora do pipeline de implantação.
- Você ignora as etapas de teste na automação por meio de fluxos de trabalho manuais de emergência.
- Você não segue os planos e os processos de teste estabelecidos em favor de cronogramas acelerados.

Benefícios de estabelecer esta desta prática recomendada: os testes funcionais confirmam que o sistema opera de acordo com os requisitos especificados. Ele é usado para verificar de forma consistente a ordem de funcionamento pretendida dos componentes, como interfaces de usuário, APIs, bancos de dados e código-fonte. Quando você examina esses componentes do sistema, os testes funcionais verificam se cada recurso se comporta conforme o esperado, o que protege as expectativas do usuário e a integridade do software. Integre testes funcionais como parte da implantação regular e use a automação para implantar todas as mudanças, o que reduz a possibilidade de introdução de erros humanos.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: alto

Orientação para a implementação

Integre testes funcionais como parte da implantação. Os testes funcionais são executados como parte da implantação automatizada. Se os critérios de sucesso não forem atendidos, o pipeline será interrompido ou revertido. O AWS CodePipeline fornece um canal de entrega contínua para testes automatizados, o que permite que os testadores automatizem todo o processo de teste e implantação. Ele se integra aos serviços da AWS, como o AWS CodeBuild e o AWS CodeDeploy, para automatizar as fases de compilação, teste e implantação do ciclo de vida de desenvolvimento de software.

Etapas da implementação

- Configure o pipeline: configure os estágios de origem, criação, teste e implantação usando o console do AWS CodePipeline ou a AWS Command Line Interface (CLI).

- Defina a fonte: com o AWS CodePipeline, é possível recuperar automaticamente o código-fonte de sistemas de controle de versão, como o GitHub, o AWS CodeCommit ou o Bitbucket, que verifica se o código mais recente é sempre usado para testes.
- Automatize compilações e testes: o AWS CodeBuild pode criar e testar automaticamente o código e gerar relatórios de teste. Ele comporta frameworks de teste conhecidos, como JUnit, NUnit e TestNG.
- Implante o código: depois que o código for criado e testado, o AWS CodeDeploy poderá implantá-lo no ambiente de teste, incluindo instâncias do Amazon EC2, funções do AWS Lambda ou servidores on-premises.
- Monitore os pipelines: o AWS CodePipeline pode acompanhar o progresso do pipeline e o status de cada estágio. Você também pode usar verificações de qualidade para bloquear o pipeline de acordo com o status de execução do teste. Também pode receber notificações sobre qualquer falha ou conclusão do estágio do pipeline.

Recursos

Documentos relacionados:

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

REL08-BP03 Integrar testes de resiliência como parte da sua implantação

Integre os testes de resiliência introduzindo falhas conscientemente no sistema para medir a capacidade em caso de cenários disruptivos. Os testes de resiliência são diferentes dos testes de unidade e de função que geralmente são integrados aos ciclos de implantação, pois se concentram na identificação de falhas imprevistas no sistema. Embora seja seguro começar com a integração dos testes de resiliência na pré-produção, defina uma meta para implementar esses testes na produção como parte dos dias de jogos.

Resultado desejado: o teste de resiliência ajuda a aumentar a confiança na capacidade do sistema de resistir à degradação na produção. Os experimentos identificam pontos fracos que podem causar falha, o que ajuda a melhorar o sistema para mitigar falhas e degradações de forma automática e eficiente.

Antipadrões comuns:

- Falta de observabilidade e monitoramento nos processos de implantação.
- Dependência em pessoas para resolver falhas do sistema.
- Mecanismos de análise de baixa qualidade.
- Foco em problemas conhecidos de um sistema e ausência de experimentação para identificar quaisquer problemas desconhecidos.
- Identificação de falhas, mas sem resolução.
- Nenhuma documentação de descobertas e runbooks.

Benefícios de estabelecer essas práticas recomendadas: os testes de resiliência integrados às implantações ajudam a identificar problemas desconhecidos no sistema que, de outra forma, passariam despercebidos, o que pode causar inatividade na produção. A identificação de problemas desconhecidos em um sistema ajuda você a documentar descobertas, integrar testes ao processo de CI/CD e criar runbooks, o que simplifica a mitigação por meio de mecanismos eficientes e reproduzíveis.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

As formas de teste de resiliência mais comuns que podem ser integradas às implantações do sistema são a recuperação de desastres e a engenharia do caos.

- Inclua atualizações nos planos de recuperação de desastres e nos procedimentos operacionais padrão (SOPs) em qualquer implantação significativa.
- Integre testes de confiabilidade aos canais de implantação automatizados. Serviços como o AWS Resilience Hub podem ser integrados ao pipeline de CI/CD para estabelecer avaliações contínuas de resiliência que são analisadas automaticamente como parte de cada implantação.
- Defina as aplicações no AWS Resilience Hub. As avaliações de resiliência geram trechos de código que ajudam você a criar procedimentos de recuperação como documentos do AWS Systems Manager para as aplicações e fornecem uma lista de monitores e alarmes recomendados do Amazon CloudWatch.
- Depois que os planos de DR e SOPs forem atualizados, conclua os testes de recuperação de desastres para verificar se eles são eficazes. O teste de recuperação de desastres ajuda a determinar se você pode restaurar o sistema após um evento e retornar às operações

normais. Você pode simular várias estratégias de recuperação de desastres e identificar se seu planejamento é suficiente para atender às necessidades de disponibilidade. As estratégias comuns de recuperação de desastres incluem backup e restauração, luz piloto, espera fria, espera quente, standby a quente e ativo-ativo, e todas elas diferem em custo e complexidade. Antes do teste de recuperação de desastres, recomendamos que você defina o objetivo de tempo de recuperação (RTO) e o objetivo de ponto de recuperação (RPO) para simplificar a escolha da estratégia a ser simulada. A AWS oferece ferramentas de recuperação de desastres como o AWS Elastic Disaster Recovery para ajudar você a começar a planejar e testar.

- Os experimentos de engenharia do caos introduzem interrupções no sistema, como interrupções na rede e falhas no serviço. Ao simular com falhas controladas, você pode descobrir as vulnerabilidades do sistema e, ao mesmo tempo, conter os impactos das falhas injetadas. Assim como as outras estratégias, execute simulações de falhas controladas em ambientes de não produção usando serviços, como o AWS Fault Injection Service, para ganhar confiança antes da implantação na produção.

Recursos

Documentos relacionados:

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

Vídeos relacionados:

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

REL08-BP04 Implantar usando infraestrutura imutável

A infraestrutura imutável é um modelo que não requer atualizações, patches de segurança ou que alterações na configuração ocorram no local nas workloads de produção. Quando uma alteração é necessária, a arquitetura é criada em uma nova infraestrutura e implantada na produção.

Siga uma estratégia de implantação de infraestrutura imutável para aumentar a confiabilidade, a consistência e a reprodutibilidade nas implantações de workload.

Resultado desejado: com a infraestrutura imutável, nenhuma modificação no local é permitida para executar recursos de infraestrutura em uma workload. Em vez disso, quando uma alteração é necessária, um novo conjunto de recursos atualizados da infraestrutura que contém todas as alterações necessárias é implantado paralelamente aos recursos existentes. Essa implantação é validada automaticamente e, se bem-sucedida, o tráfego é gradualmente transferido para o novo conjunto de recursos.

Essa estratégia de implantação aplica-se a atualizações de software, patches de segurança, alterações na infraestrutura, atualizações de configuração e atualizações de aplicações, entre outros.

Antipadrões comuns:

- Implementação de mudanças no local em recursos da infraestrutura em execução.

Benefícios do estabelecimento desta prática recomendada:

- Maior consistência entre os ambientes: como não há diferenças nos recursos de infraestrutura entre os ambientes, a consistência aumenta e os testes são simplificados.
- Redução dos desvios da configuração: ao substituir os recursos da infraestrutura por uma configuração conhecida e controlada por versão, a infraestrutura é definida para um estado conhecido, testado e confiável, o que evita desvios de configuração.
- Implantações atômicas confiáveis: as implantações são concluídas com sucesso ou, do contrário, nada muda, aumentando a consistência e a confiabilidade no processo de implantação.
- Implantações simplificadas: as implantações são simplificadas porque não precisam oferecer suporte a atualizações. As atualizações são apenas novas implantações.
- Implantações mais seguras com processos de reversão e recuperação rápidos: as implantações são mais seguras porque a versão de trabalho anterior não é alterada. Você pode reverter para ele se forem detectados erros.

- Procedimento de segurança aprimorado: quando não se permitem alterações na infraestrutura, os mecanismos de acesso remoto (como o SSH) podem ser desativados. Isso reduz o vetor de ataque, melhorando o procedimento de segurança da organização.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

Automação

Ao definir uma estratégia de implantação de infraestrutura imutável, é recomendável usar a automação o máximo possível para aumentar a reprodutibilidade e minimizar a possibilidade de erro humano. Para obter mais detalhes, consulte REL08-BP05 Implantar alterações com automação e Automating safe, hands-off deployments.

Com a infraestrutura como código (IaC), as etapas de provisionamento, orquestração e implantação da infraestrutura são definidas de forma programática, descritiva e declarativa e armazenadas em um sistema de controle de origem. O uso da infraestrutura como código simplifica a automatização da implantação da infraestrutura e ajuda a obter a imutabilidade da infraestrutura.

Padrões de implantação

Quando uma mudança na workload é necessária, a estratégia de implantação da infraestrutura imutável exige que um novo conjunto de recursos da infraestrutura seja implantado, incluindo todas as alterações necessárias. É importante que esse novo conjunto de recursos siga um padrão de implantação que minimize o impacto sobre o usuário. Há duas estratégias principais para essa implantação:

Implantação canário: prática de direcionar um pequeno número de clientes para a nova versão, geralmente em execução em uma única instância de serviço (o canário). Em seguida, você examina profundamente todas as alterações de comportamento ou erros gerados. Você poderá remover o tráfego da implantação canário se encontrar problemas críticos e enviar os usuários de volta para a versão anterior. Se a implantação for bem-sucedida, será possível continuar a implantação a uma velocidade desejada e monitorar as alterações em busca de erros até a implantação estar concluída. O AWS CodeDeploy pode ser configurado com uma configuração de implantação que permitirá uma implantação canário.

Implantação azul/verde: é semelhante à implantação canário, exceto que uma frota completa da aplicação é implantada em paralelo. Você alterna as implantações entre as duas pilhas (azul e verde). Novamente, é possível enviar o tráfego para a nova versão e voltar para a versão antiga

se houver problemas na implantação. Normalmente, todo o tráfego é alternado de uma vez. No entanto, também é possível usar frações do tráfego para cada versão para aumentar a adoção da nova versão usando os recursos de roteamento de DNS ponderado do Amazon Route 53. O AWS CodeDeploy e o AWS Elastic Beanstalk podem ser definidos com uma configuração de implantação que permitirá uma implantação azul/verde.

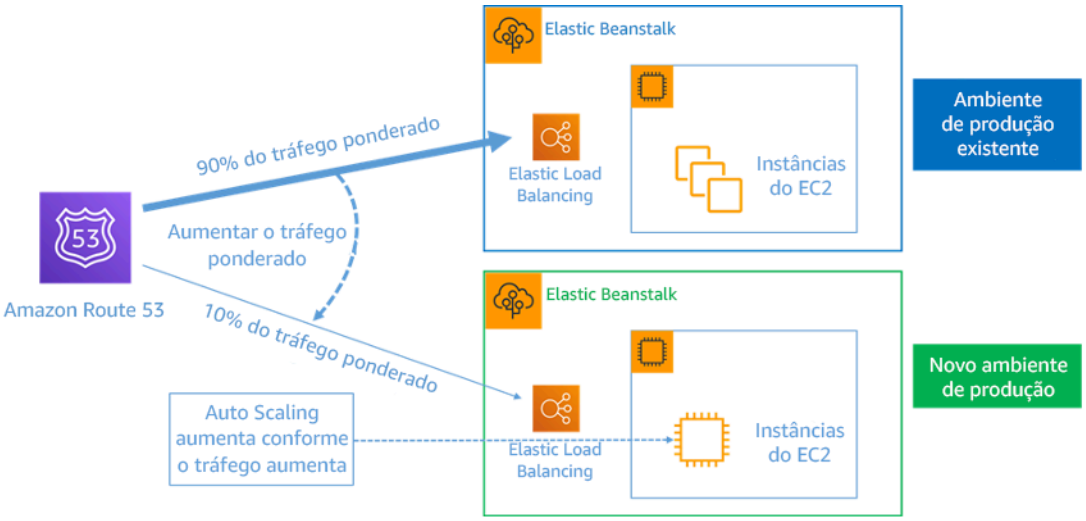


Figura 8: Implantação azul/verde com o AWS Elastic Beanstalk e o Amazon Route 53

Detecção de desvios

Define-se desvio como qualquer alteração que faça com que um recurso da infraestrutura tenha um estado ou uma configuração diferente do esperado. Alterações não gerenciadas da configuração, sejam de que tipo for, são contrárias ao conceito de infraestrutura imutável e devem ser detectadas e corrigidas para que a implementação da infraestrutura imutável seja bem-sucedida.

Etapas da implementação

- Proibir a modificação no local dos recursos de infraestrutura em execução.
  - É possível usar o AWS Identity and Access Management (IAM) para especificar quem ou o que pode acessar serviços e recursos na AWS, gerenciar as permissões refinadas centralmente e analisar o acesso para refinar as permissões em toda a AWS.
- Automatize a implantação dos recursos da infraestrutura para aumentar a reprodutibilidade e minimizar a possibilidade de erro humano.
  - Conforme descrito no whitepaper Introdução ao DevOps na AWS, a automação é uma referência dos serviços da AWS e é compatível com todos os serviços, recursos e ofertas.
  - Pré-fabricar a imagem de máquina da Amazon (AMI) pode acelerar o tempo para iniciá-la. O EC2 Image Builderé um serviço da AWS totalmente gerenciado que ajuda você a

automatizar a criação, a manutenção, a validação, o compartilhamento e a implantação de AMIs personalizadas, seguras e atualizadas para Linux ou Windows.

- Alguns dos serviços compatíveis com automação são:
  - O AWS Elastic Beanstalké um serviço para implantar e escalar rapidamente aplicações web desenvolvidas com Java, .NET, PHP, Node.js, Python, Ruby, Go e Docker em servidores conhecidos, como Apache, NGINX, Passenger e IIS.
  - O AWS Proton ajuda as equipes da plataforma a se conectarem e coordenarem todas as diferentes ferramentas de que as equipes de desenvolvimento precisam para provisionamento de infraestrutura, implantações de código, monitoramento e atualizações. O AWS Proton permite o provisionamento de infraestrutura como código automatizada e a implantação de aplicações sem servidor e baseadas em contêineres.
- A utilização da infraestrutura como código facilita a automatização da implantação da infraestrutura e ajuda a obter a imutabilidade da infraestrutura. A AWS fornece serviços que permitem a criação, a implantação e a manutenção da infraestrutura de forma programática, descritiva e declarativa.
  - O AWS CloudFormation ajuda os desenvolvedores a criar recursos da AWS de forma ordenada e previsível. Os recursos são escritos em arquivos de texto usando o formato JSON ou YAML. Os modelos exigem sintaxe e estrutura específicas que dependem dos tipos de recurso que estão sendo criados e gerenciados. Você cria os recursos em JSON ou YAML com qualquer editor de código, como o AWS Cloud9, e os insere em um sistema de controle de versão, e o CloudFormation cria os serviços especificados de maneira segura e repetível.
  - O AWS Serverless Application Model (AWS SAM) é uma estrutura de código aberto que você pode usar para criar aplicações sem servidor na AWS. O AWS SAM integra-se a outros serviços da AWS e é uma extensão do AWS CloudFormation.
  - O AWS Cloud Development Kit (AWS CDK) é um framework de desenvolvimento de software de código aberto para modelar e provisionar recursos da aplicação em nuvem usando linguagens de programação conhecidas. É possível usar o AWS CDK para modelar a infraestrutura de aplicações usando TypeScript, Python, Java e .NET. O AWS CDK usa o AWS CloudFormation em segundo plano para provisionar recursos de forma segura e repetível.
  - O AWS Cloud Control API apresenta um conjunto comum de APIs de criação, leitura, atualização, exclusão e lista (CRUDL) para ajudar os desenvolvedores a gerenciar a infraestrutura em nuvem de forma fácil e consistente. As APIs comuns do Cloud Control API permitem que os desenvolvedores gerenciem de maneira uniforme o ciclo de vida de serviços da AWS e de terceiros.

- Implemente padrões de implantação que minimizem o impacto no usuário.
  - Implantações canário:
    - Configurar uma implantação de versão canary do API Gateway
    - Create a pipeline with canary deployments for Amazon ECS using AWS App Mesh
  - Implantações azul/verde: o whitepaper Blue/Green Deployments on AWS descreve exemplos de técnicas para implementar estratégias de implantação azul/verde.
- Detecte variações da configuração ou do estado. Para obter mais detalhes, consulteDetectar alterações de configuração não gerenciadas em pilhas e recursos.

Recursos

Práticas recomendadas relacionadas:

- REL08-BP05 Implantar alterações com automação

Documentos relacionados:

- Automatizar uma implantação prática e sem intervenção manual
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- Infraestrutura como código
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

Vídeos relacionados:

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

REL08-BP05 Implantar alterações com automação

As implantações e a aplicação de patches são automatizadas para eliminar o impacto negativo.

As alterações nos sistemas de produção são uma das maiores áreas de risco para muitas organizações. Consideramos as implantações um problema de primeira classe a ser resolvido junto com os problemas de negócio que o software aborda. Atualmente, isso significa usar a automação nas operações sempre que for viável, incluindo testar e implantar alterações, adicionar ou remover capacidade e migrar dados.

Resultado desejado: você incorpora segurança de implantação automatizada ao processo de lançamento com testes extensivos de pré-produção, reversões automáticas e implantações de produção escalonadas. Essa automação minimiza o impacto potencial na produção causado por falhas nas implantações, e os desenvolvedores não precisam mais monitorar ativamente as implantações na produção.

Antipadrões comuns:

- Você realiza alterações manuais.
- Ignora etapas na automação por meio de fluxos de trabalho manuais de emergência.
- Não segue os planos e os processos estabelecidos em favor de cronogramas acelerados.
- Executa implantações subsequentes rápidas sem permitir o tempo de incorporação.

Benefícios de estabelecer esta prática recomendada: ao usar a automação para implantar todas as alterações, você elimina a possibilidade de introdução de erro humano e fornece a capacidade de testar antes de alterar a produção. A execução desse processo antes do início da produção verifica se os planos estão concluídos. Além disso, a reversão automática no processo de liberação pode identificar problemas de produção e retornar a workload ao estado operacional anterior.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: médio

Orientações para a implementação

Automatize o pipeline de implantação. Os pipelines de implantação permitem invocar testes automatizados e detecção de anomalias. Além disso, eles interrompem o pipeline em uma determinada etapa antes da implantação em produção ou revertem automaticamente uma alteração. Uma parte integral disso é a adoção da cultura de integração contínua e entrega/implantação contínua (CI/CD), em que uma confirmação ou uma alteração de código passa por vários estágios automatizados, desde os estágios de compilação e teste até a implantação em ambientes de produção.

Embora a sabedoria convencional sugira que você mantenha as pessoas informadas para os procedimentos operacionais mais difíceis, sugerimos automatizar esses procedimentos exatamente por isso.

Etapas da implementação

Você pode automatizar as implantações para remover as operações manuais seguindo estas etapas:

- Configure um repositório de código para armazenar o código com segurança: use o AWS CodeCommit para criar um repositório seguro baseado em Git.
- Configure um serviço de integração contínua para compilar o código-fonte, executar testes e criar artefatos de implantação: para configurar um projeto de compilação para essa finalidade, consulte Getting started with AWS CodeBuild using the console.
- Configure um serviço de implantação que automatize as implantações de aplicações e gerencie a complexidade das atualizações de aplicações sem depender de implantações manuais propensas a erros: o AWS CodeDeploy automatiza as implantações de software em uma variedade de serviços de computação, como Amazon EC2, AWS Fargate, AWS Lambda e os servidores on-premises. Para configurar essas etapas, consulte Getting started with CodeDeploy.
- Configure um serviço de entrega contínua que automatize seus canais de lançamento para atualizações mais rápidas e confiáveis de aplicações e infraestrutura: pense em usar o AWS CodePipeline para ajudar a automatizar os canais de lançamento. Para ter mais detalhes, consulte CodePipeline tutorials.

Recursos

Práticas recomendadas relacionadas:

- OPS05-BP04 Usar sistemas de gerenciamento de compilação e de implantação
- OPS05-BP10 Automatizar totalmente a integração e a implantação
- OPS06-BP02 Testar as implantações
- OPS06-BP04 Automatizar os testes e a reversão

Documentos relacionados:

- Continuous Delivery of Nested AWS CloudFormation Stacks Using AWS CodePipeline
- Complete CI/CD with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline
- Parceiro da APN: parceiros que podem ajudá-lo a criar soluções de implantação automatizada
- AWS Marketplace: produtos que podem ser usados para automatizar suas implantações
- Automate chat messages with webhooks.
- A Amazon Builders' Library: garantindo a segurança da reversão durante as implantações
- A Amazon Builders' Library: acelerando com a entrega contínua

- What Is AWS CodePipeline?
- What Is CodeDeploy?
- AWS Systems Manager Patch Manager
- What is Amazon SES?
- What is Amazon Simple Notification Service?

Vídeos relacionados:

- AWS Summit 2019: CI/CD on AWS

Gerenciamento de falhas

> Falhas são inevitáveis e tudo acabará falhando ao longo do tempo: de roteadores a discos rígidos, de sistemas operacionais a unidades de memória corrompendo pacotes TCP, de erros transitórios a falhas permanentes. Isso é uma certeza, independentemente de você usar o hardware da mais alta qualidade ou os componentes de menor custo - Werner Vogels, diretor de tecnologia da Amazon.com

Falhas de componentes de hardware de baixo nível são algo a ser eliminado todos os dias em um datacenter no local. Na nuvem, no entanto, você deve estar protegido contra a maioria desses tipos de falhas. Por exemplo, volumes do Amazon EBS são colocados em uma zona de disponibilidade específica onde são replicados automaticamente para proteger você contra falhas de um único componente. Todos os volumes do EBS foram projetados para oferecer 99,999% de disponibilidade. Os objetos do Amazon S3 são armazenados em no mínimo três Zonas de disponibilidade, fornecendo 99,999% de durabilidade de objetos durante um determinado ano. Independentemente do seu provedor de nuvem, há o potencial de falhas para afetar sua carga de trabalho. Portanto, tome medidas para implementar a resiliência se precisar que sua workload seja confiável.

Um pré-requisito para aplicar as melhores práticas discutidas aqui é que você deve garantir que as pessoas que projetam, implementam e operam suas cargas de trabalho estejam cientes dos objetivos de negócios e das metas de confiabilidade para alcançá-los. Essas pessoas devem estar cientes e treinadas para esses requisitos de confiabilidade.

As seções a seguir explicam as melhores práticas para gerenciar falhas para evitar impacto na carga de trabalho.

Tópicos

Fazer o backup de dados

Faça backup de dados, aplicativos e configurações para atender aos requisitos de objetivos de tempo de recuperação (RTO) e objetivos de ponto de recuperação (RPO).

Práticas recomendadas

- REL09-BP01 Identificar e fazer backup de todos os dados que precisam de backup ou reproduzir os dados das fontes
- REL09-BP02 Proteger e criptografar backups
- REL09-BP03 Realizar backup de dados automaticamente
- REL09-BP04 Realizar a recuperação periódica dos dados para verificar a integridade e os processos de backup

REL09-BP01 Identificar e fazer backup de todos os dados que precisam de backup ou reproduzir os dados das fontes

Compreenda e use os recursos de backup dos serviços e recursos de dados usados pela workload. A maioria dos serviços oferece recursos para fazer backup dos dados da workload.

Resultado desejado: as fontes de dados foram identificadas e classificadas com base na criticidade. Depois, estabeleça uma estratégia de recuperação de dados com base no RPO. A estratégia envolve fazer backup dessas fontes de dados ou poder reproduzir dados de outras fontes. Em caso de perda de dados, a estratégia implementada permite a recuperação ou reprodução de dados dentro do RPO e RTO definidos.

Fase de maturidade da nuvem: básica

Antipadrões comuns:

- Não estar ciente de todas as fontes de dados para a workload e sua criticidade.
- Não fazer backups de fontes de dados essenciais.
- Fazer backups apenas de algumas fontes de dados sem usar a criticidade como critério.
- Não ter um RPO definido ou a frequência de backup não atender ao RPO.
- Não avaliar a necessidade de um backup ou se os dados podem ser reproduzidos de outras fontes.

Benefícios do estabelecimento dessa prática recomendada: identificar os locais onde os backups são necessários e implementar um mecanismo para criar backups ou poder reproduzir os dados de uma fonte externa melhora a capacidade de restaurar e recuperar dados durante uma interrupção.

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto

Orientações para a implementação

Todos os armazenamentos de dados da AWS oferecem recursos de backup. Serviços como o Amazon RDS e o Amazon DynamoDB oferecem suporte adicional ao backup automatizado que permite a recuperação a um ponto anterior no tempo (PITR), permitindo restaurar um backup a qualquer momento até cinco minutos ou menos, antes da hora atual. Muitos serviços da AWS permitem copiar backups para outra Região da AWS. O AWS Backup é uma ferramenta que permite centralizar e automatizar a proteção de dados nos serviços da AWS. O AWS Elastic Disaster Recovery permite copiar workloads de servidor completas e manter uma proteção de dados contínua de um ambiente on-premises, entre zonas de disponibilidade ou entre regiões, com um objetivo de ponto de recuperação (RPO) medido em segundos.

O Amazon S3 pode ser usado como um destino de backup para fontes de dados autogerenciadas e gerenciadas pela AWS. Os serviços da AWS, como o Amazon EBS, o Amazon RDS e o Amazon DynamoDB, têm recursos integrados para criar backups. É possível também usar um software de backup de terceiros.

E possível fazer backup de dados on-premises na Nuvem AWS usando o AWS Storage Gateway ou o AWS DataSync. Os buckets do Amazon S3 podem ser usados para armazenar esses dados na AWS. O Amazon S3 oferece vários níveis de armazenamento, como Amazon S3 Glacier ou S3 Glacier Deep Archive para reduzir os custos do armazenamento de dados.

Você pode atender às necessidades de recuperação de dados reproduzindo os dados de outras fontes. Por exemplo, os nós de réplica do Amazon ElastiCache ou as réplicas de leitura do Amazon RDS poderiam ser usadas para reproduzir dados caso os primários sejam perdidos. Em casos em que fontes como essa podem ser usadas para atender ao objetivo de ponto de recuperação (RPO) e objetivo de tempo de recuperação (RTO), pode ser que você não precise de um backup. Outro exemplo, se estiver trabalhando com o Amazon EMR, pode não ser necessário fazer backup do armazenamento de dados HDFS, contanto que você possa reproduzir os dados no Amazon EMR pelo Amazon S3.

Ao selecionar uma estratégia de backup, considere o tempo necessário para recuperar os dados. Ele depende do tipo de backup (no caso de uma estratégia de backup) ou da complexidade do mecanismo de reprodução de dados. O tempo deve estar dentro do RTO para a workload.

Etapas da implementação

1. Identifique todas as fontes de dados para a workload. Os dados podem ser armazenados em vários recursos, como bancos de dados, volumes, sistemas de arquivos, sistemas de registro e armazenamento de objetos. Consulte a seção Recursos para encontrar Documentos relacionados sobre diferentes serviços da AWS onde os dados são armazenados e o recurso de backup que esses serviços fornecem.
2. Classifique as fontes de dados com base na criticidade. Diferentes conjuntos de dados terão diferentes níveis de criticidade para uma workload e, portanto, diferentes requisitos de resiliência. Por exemplo, alguns dados podem ser críticos e exigir um RPO próximo de zero, enquanto outros dados podem ser menos críticos e tolerar um RPO mais alto e a perda de alguns dados. Da mesma forma, diferentes conjuntos de dados também podem ter diferentes requisitos de RTO.
3. Use a AWS ou serviços de terceiros para criar backups dos dados. O AWS Backup é um serviço gerenciado que permite criar backups de várias fontes de dados na AWS. O AWS Elastic Disaster Recovery lida com a replicação de dados automáticos de subsegundos em uma Região da AWS. A maioria dos serviços da AWS também possui recursos nativos para criar backups. O AWS Marketplace tem muitas soluções que também fornecem esses recursos. Consulte os Recursos listados abaixo para obter informações sobre como criar backups de dados de vários serviços da AWS.
4. Para dados sem backup, estabeleça um mecanismo de reprodução de dados. Você pode optar por não fazer backup dos dados que podem ser reproduzidos de outras fontes por vários motivos. Às vezes, pode ser mais barato reproduzir dados de fontes se necessário, em vez de criar um backup, pois pode haver um custo associado ao armazenamento de backups. Outro exemplo é quando a restauração de um backup demora mais do que a reprodução dos dados das fontes, resultando em uma violação no RTO. Nestas situações, considere concessões e estabeleça um processo bem definido de como os dados podem ser reproduzidos dessas fontes quando a recuperação de dados for necessária. Por exemplo, se você carregou dados do Amazon S3 para um data warehouse (como o Amazon Redshift) ou para um cluster MapReduce (como o Amazon EMR) para analisá-los, esse é um exemplo de dados que podem ser reproduzidos de outras fontes. Desde que os resultados dessas análises sejam armazenados em algum lugar ou reproduzíveis, você não sofreria uma perda de dados devido a uma falha no data warehouse ou no cluster do MapReduce. Outros exemplos que podem ser reproduzidos de origens incluem caches (como o Amazon ElastiCache) ou réplicas de leitura do RDS.
5. Estabeleça uma frequência para fazer backup de dados. A criação de backups de fontes de dados é um processo periódico, e a frequência deve depender do RPO.

Nível de esforço do plano de implementação: moderado

Recursos

Práticas recomendadas relacionadas:

REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados

REL13-BP02 Usar estratégias de recuperação definidas para cumprir os objetivos de recuperação

Documentos relacionados:

- O que é o Reachability Analyzer?
- What is AWS DataSync? (O que é o AWS Data Sync?)
- What is Volume Gateway? (O que é o Gateway de Volumes?)
- Parceiro do APN: parceiros que podem ajudar com o backup
- AWS Marketplace: products that can be used for backup (AWS Marketplace: produtos que podem ser usados para backup)
- Snapshots do Amazon EBS
- Backing Up Amazon EFS (Fazer backup do Amazon EFS)
- Backing up Amazon FSx for Windows File Server (Fazer backup do Amazon FSx para Windows File Server)
- Backup e restauração para o ElastiCache for Redis
- Creating a DB Cluster Snapshot in Neptune (Criar um snapshot do cluster de banco de dados no Neptune)
- Criar um snapshot do banco de dados
- Creating an EventBridge Rule That Triggers on a Schedule (Criar uma regra do EventBridge que é acionada de acordo com uma programação)
- Replicação entre regiões com o Amazon S3
- EFS para EFS AWS Backup
- Exporting Log Data to Amazon S3 (Exportação de dados de log para o Amazon S3)
- Gerenciamento do ciclo de vida de objetos
- Usar backup e restauração sob demanda para o DynamoDB
- Recuperação pontual para DynamoDB
- Criação de snapshots de índices no Amazon OpenSearch Service

- O que é o Reachability Analyzer?

Vídeos relacionados:

- AWS re:Invent 2021: Backup, disaster recovery, and ransomware protection with AWS (Backup, recuperação de desastres e proteção contra ransomware com a AWS)
- AWS Backup Demo: Cross-Account and Cross-Region Backup (Demonstração: Backup entre contas e entre regiões)
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

Exemplos relacionados:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3 (Laboratório do Well-Architected: implementação da replicação bidirecional entre regiões (CRR) para o Amazon S3)
- Laboratório do Well-Architected: teste de backup e restauração de dados
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload (Laboratório do Well-Architected: backup e restauração com failback para workload do Analytics)
- Well-Architected Lab - Disaster Recovery - Backup and Restore (Laboratório do Well-Architected: recuperação de desastres: backup e restauração)

REL09-BP02 Proteger e criptografar backups

Controle e detecte o acesso a backups usando autenticação e autorização. Use a criptografia para prevenir e detectar se a integridade dos dados de backups está comprometida.

Antipadrões comuns:

- Ter o mesmo acesso aos backups e à automação de restauração que os dados.
- Não criptografar seus backups.

Benefícios do estabelecimento dessa prática recomendada: a proteção dos backups impede a violação dos dados, e a criptografia dos dados impede o acesso a eles se forem expostos por engano.

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto

Orientações para a implementação

Controle e detecte o acesso a backups usando autenticação e autorização, como o AWS Identity and Access Management (IAM). Use a criptografia para prevenir e detectar se a integridade dos dados de backups está comprometida.

O Amazon S3 oferece suporte a vários métodos de criptografia de dados ociosos. Ao usar a criptografia do lado do servidor, o Amazon S3 aceita os objetos como dados não criptografados e, depois, criptografa-os ao armazená-los. Ao usar a criptografia do lado do cliente, a aplicação da workload é responsável por criptografar os dados antes de serem enviados ao Amazon S3. Ambos os métodos permitem que você use o AWS Key Management Service (AWS KMS) para criar e armazenar a chave de dados, ou você pode fornecer sua própria chave, pela qual você é responsável. Usando o AWS KMS, você pode definir políticas usando o IAM sobre quem pode e não pode acessar suas chaves de dados e dados descriptografados.

Para o Amazon RDS, se você tiver optado por criptografar seus bancos de dados, seus backups também serão criptografados. Os backups do DynamoDB sempre são criptografados. Ao usar o AWS Elastic Disaster Recovery, todos os dados em trânsito e em repouso são criptografados. Com o Elastic Disaster Recovery, os dados em repouso podem ser criptografados usando a chave de criptografia de volume padrão do Amazon EBS ou uma chave personalizada gerenciada pelo cliente.

Etapas da implementação

1. Use a criptografia em cada um dos seus armazenamentos de dados. Se os dados de origem forem criptografados, o backup também será.
   - Use criptografia no Amazon RDS. Você pode configurar a criptografia em repouso usando o AWS Key Management Service ao criar uma instância do RDS.
   - Use criptografia em volumes do Amazon EBS. Você pode configurar a criptografia padrão ou especificar uma chave exclusiva após a criação do volume.
   - Use a criptografia do Amazon DynamoDB necessária. O DynamoDB criptografa todos os dados em repouso. Você pode usar uma chave do AWS KMS de propriedade da AWS ou uma chave do KMS gerenciada pela AWS, especificando uma chave armazenada na sua conta.
   - Criptografe seus dados armazenados no Amazon EFS. Configure a criptografia ao criar seu sistema de arquivos.
   - Configure a criptografia nas regiões de origem e de destino. Você pode configurar a criptografia em repouso no Amazon S3 usando as chaves armazenadas no KMS, mas as chaves são específicas da região. Você pode especificar as chaves de destino ao configurar a replicação.

- Escolha se deseja usar a critptografia do Amazon EBS para o Elastic Disaster Recovery padrão ou personalizada. Essa opção criptografa os dados em repouso replicados nos discos da sub-rede da área de preparação e os discos replicados.

2. Implemente permissões de privilégio mínimo para acessar seus backups. Siga as práticas recomendadas para limitar o acesso aos backups, aos snapshots e às réplicas de acordo com as práticas recomendadas de segurança.

Recursos

Documentos relacionados:

- AWS Marketplace: products that can be used for backup (AWS Marketplace: produtos que podem ser usados para backup)
- Criptografia do Amazon EBS
- Amazon S3: proteção de dados usando criptografia
- Replicar objetos criados com criptografia do lado do servidor (SSE) usando chaves do AWS KMS
- Criptografia em repouso do DynamoDB
- Criptografar recursos do Amazon RDS
- Encrypting Data and Metadata in Amazon EFS (Criptografia de dados e metadados no Amazon EFS)
- Encryption for Backups in AWS (Criptografia para backups na AWS)
- Gerenciamento de tabelas criptografadas
- Pilar Segurança: AWS Well-Architected Framework
- O que é o AWS Elastic Disaster Recovery?

Exemplos relacionados:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3 (Laboratório do Well-Architected: implementação da replicação bidirecional entre regiões (CRR) para o Amazon S3)

REL09-BP03 Realizar backup de dados automaticamente

Configure os backups para serem feitos automaticamente com base em uma programação periódica informada pelo objetivo de ponto de recuperação (RPO) ou de acordo com alterações no conjunto de dados. É necessário fazer frequentemente o backup automático de conjuntos de dados críticos com requisitos de baixa perda de dados, enquanto o backup de dados menos críticos, em que alguma perda é aceitável, pode ser feito com menos frequência.

Resultado desejado: um processo automatizado que cria backups de fontes de dados em uma frequência estabelecida.

Antipadrões comuns:

- Fazer backups manualmente.
- Usar recursos que têm o recurso de backup, mas não incluir o backup em sua automação.

Benefícios do estabelecimento dessa prática recomendada: a automação de backups garante que eles sejam feitos regularmente com base no RPO, e alerta você caso isso não ocorra.

Nível de risco exposto se esta prática recomendada não é estabelecida: médio

Orientações para a implementação

É possível usar o AWS Backup para criar backups de dados automatizados de várias fontes de dados da AWS. É possível fazer backup das instâncias do Amazon RDS quase continuamente a cada cinco minutos e dos objetos do Amazon S3 a cada quinze minutos, proporcionando recuperação a um ponto anterior no tempo (PITR) para um momento específico no histórico de backup. Para outras fontes de dados da AWS, como volumes do Amazon EBS, tabelas do Amazon DynamoDB ou sistemas de arquivos do Amazon FSx, o AWS Backup pode executar backup automatizado de hora em hora. Esses serviços também oferecem recursos de backup nativos. Os serviços da AWS que oferecem backup automatizado com recuperação a um ponto anterior no tempo incluem Amazon DynamoDB, Amazon RDS e Amazon Keyspaces (para Apache Cassandra). Eles podem ser restaurados a um momento específico no histórico do backup. A maioria dos outros serviços de armazenamento de dados da AWS permite programar backups periódicos, até de hora em hora.

O Amazon RDS e o Amazon DynamoDB oferecem backup contínuo com recuperação a um ponto anterior no tempo. O versionamento do Amazon S3, quando habilitado, é automático. O Amazon

Data Lifecycle Manager pode ser usado para automatizar a criação, a cópia e a exclusão de snapshots do Amazon EBS. Ele também pode automatizar a criação, a cópia, a suspensão e o cancelamento do registro de imagens de máquina da Amazon (AMIs) com base no Amazon EBS e seus snapshots subjacentes do Amazon EBS.

O AWS Elastic Disaster Recovery fornece replicação contínua no nível de bloco do ambiente de origem (on-premises ou a AWS) para a região de recuperação de destino. Os snapshots do Amazon EBS de um ponto anterior no tempo são criados automaticamente e gerenciados pelo serviço.

Para obter uma visão centralizada da automação e do histórico de backups, o AWS Backup oferece uma solução de backup totalmente gerenciada e baseada em políticas. Ele centraliza e automatiza o backup de dados em vários serviços da AWS, na nuvem e on-premises, usando o AWS Storage Gateway.

Além do versionamento, o Amazon S3 oferece replicação. Todo o bucket do S3 pode ser replicado automaticamente para outro bucket na mesma Região da AWS ou em uma diferente.

Etapas da implementação

1. Identifique fontes de dados que estão sendo copiados manualmente. Para obter mais detalhes, consulte REL09-BP01 Identificar e fazer backup de todos os dados que precisam de backup ou reproduzir os dados das fontes.
2. Determine o RPO da workload. Para obter mais detalhes, consulte REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados.
3. Use uma solução de backup automático ou um serviço gerenciado. O AWS Backup é um serviço totalmente gerenciado que facilita centralizar e automatizar a proteção de dados nos serviços da AWS, na nuvem e no ambiente on-premises. O uso de planos no AWS Backup permite a criação de regras que definem os recursos para backup e a frequência com que esses backups devem ser criados. A frequência deve ser informada pelo RPO estabelecido na Etapa 2. Para obter orientações práticas sobre como criar backups automáticos usando o AWS Backup, consulte Testing Backup and Restore of Data (Teste de backup e restauração de dados). A maioria dos serviços da AWS que armazenam dados oferecem recursos de backup nativos. Por exemplo, o RDS pode ser aproveitado para backups automatizados com recuperação a um ponto anterior no tempo (PITR).
4. Para fontes de dados incompatíveis com uma solução de backup automatizado ou serviço gerenciado, como fontes de dados ou filas de mensagens on-premises, considere usar uma solução confiável de terceiros para criar backups automatizados. Como alternativa, você pode criar automação para fazer isso usando a AWS CLI ou os SDKs. Você pode usar o AWS Lambda

Functions ou o AWS Step Functions para definir a lógica envolvida na criação de um backup de dados e o Amazon EventBridge para executá-la em uma frequência baseada no RPO.

Nível de esforço do plano de implementação: baixo

Recursos

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar com o backup
- AWS Marketplace: products that can be used for backup (AWS Marketplace: produtos que podem ser usados para backup)
- Creating an EventBridge Rule That Triggers on a Schedule (Criar uma regra do EventBridge que é acionada de acordo com uma programação)
- O que é o AWS Backup?
- O que é o Reachability Analyzer?
- O que é o AWS Elastic Disaster Recovery?

Vídeos relacionados:

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

Exemplos relacionados:

- Laboratório do Well-Architected: teste de backup e restauração de dados

REL09-BP04 Realizar a recuperação periódica dos dados para verificar a integridade e os processos de backup

Execute um teste de recuperação para confirmar se a implementação do processo de backup atende aos seus objetivos de tempo de recuperação (RTO) e de ponto de recuperação (RPO).

Resultado desejado: os dados de backups são recuperados periodicamente usando mecanismos bem definidos para garantir que a recuperação seja possível dentro do objetivo de tempo de recuperação (RTO) estabelecido para a workload. Verifique se a restauração de um backup resulta

em um recurso contendo os dados originais sem que estejam corrompidos ou inacessíveis e que a perda de dados esteja dentro do objetivo de ponto de recuperação (RPO).

Antipadrões comuns:

- Restaurar um backup, mas não consultar ou recuperar os dados para garantir que a restauração seja útil.
- Presumir a existência de um backup.
- Presumir que o backup de um sistema esteja totalmente operacional e que os dados possam ser recuperados dele.
- Presumir que o tempo para recuperar ou restaurar dados de um backup esteja dentro do RTO para a workload.
- Presumir que os dados contidos no backup estejam dentro do RPO para a workload.
- Restaurar ad hoc, sem usar um runbook, ou o lado de fora de um procedimento automatizado estabelecido.

Benefícios do estabelecimento dessa prática recomendada: testar a recuperação dos backups garante que os dados possam ser restaurados quando necessário, sem a preocupação de que possam estar ausentes ou corrompidos. Os testes também garantem que a restauração e a recuperação sejam possíveis de acordo com o RTO da workload e qualquer perda de dados se enquadre no RPO da workload.

Nível de risco exposto se esta prática recomendada não é estabelecida: médio

Orientações para a implementação

Testar a capacidade de backup e de restauração aumenta a confiança na aptidão de realizar essas ações durante uma interrupção. Restaure periodicamente os backups em um novo local e execute testes para verificar a integridade dos dados. Alguns testes comuns que devem ser realizados são verificar se todos os dados estão disponíveis, não corrompidos, acessíveis e garantir que toda a perda de dados se enquadre no RPO da workload. Eles também podem ajudar a verificar se os mecanismos de recuperação são rápidos o suficiente para acomodar o RTO da workload.

Ao usar a AWS, você pode criar um ambiente de teste e restaurar os backups para avaliar os recursos de RTO e RPO e executar testes de conteúdo e integridade dos dados.

Além disso, o Amazon RDS e o Amazon DynamoDB permitem a recuperação point-in-time (PITR). Ao usar o backup contínuo, você pode restaurar o conjunto de dados para o estado em que estava em uma data e hora especificadas.

Se todos os dados estiverem disponíveis, não corrompidos, acessíveis e qualquer perda de dados está de acordo com o RPO da workload. Eles também podem ajudar a verificar se os mecanismos de recuperação são rápidos o suficiente para acomodar o RTO da workload.

O AWS Elastic Disaster Recovery oferece snapshots contínuos de recuperação a um ponto anterior no tempo de volumes do Amazon EBS. Como os servidores de origem são replicados, os estados de um ponto anterior no tempo são registrados ao longo do tempo com base na política configurada. O Elastic Disaster Recovery ajuda a verificar a integridade desses snapshots iniciando instâncias para fins de teste e simulação sem redirecionar o tráfego.

Etapas da implementação

1. Identifique fontes de dados das quais está sendo feito backup no momento e onde esses backups estão sendo armazenados. Para obter orientações de implementação, consulte REL09-BP01 Identificar e fazer backup de todos os dados que precisam de backup ou reproduzir os dados das fontes.
2. Estabeleça critérios para a validação de dados a cada fonte de dados. Diferentes tipos de dados terão propriedades distintas que podem exigir mecanismos de validação diferentes. Considere como validar esses dados antes de se sentir confiante em usá-los na produção. Algumas maneiras comuns de validar dados são o uso de dados e propriedades de backup, como tipo de dados, formato, soma de verificação, tamanho ou uma combinação deles com lógica de validação personalizada. Por exemplo, pode ser uma comparação dos valores de soma de verificação entre o recurso restaurado e a fonte de dados no momento em que o backup foi criado.
3. Estabeleça o RTO e o RPO para restaurar os dados com base na criticidade deles. Para obter orientações de implementação, consulte REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados.
4. Avalie sua capacidade de recuperação. Revise sua estratégia de backup e de restauração para entender se ela pode cumprir o RTO e o RPO e ajuste a estratégia conforme necessário. Usando o AWS Resilience Hub, você pode executar uma avaliação da workload. Essa avaliação analisa a configuração da aplicação em relação à política de resiliência e relata se as metas de RTO e RPO podem ser cumpridas.
5. Faça uma restauração de teste com os processos atualmente estabelecidos usados na produção para restauração de dados. Esses processos dependem de como foi feito o backup da fonte de

dados original, do formato e do local de armazenamento do próprio backup ou se os dados são reproduzidos de outras fontes. Por exemplo, se você estiver usando um serviço gerenciado, como o AWS Backup, isso pode ser tão simples quanto restaurar o backup em um novo recurso. Se você usar o AWS Elastic Disaster Recovery, poderá iniciar uma simulação de recuperação.

6. Valide a recuperação de dados do recurso restaurado com base nos critérios estabelecidos anteriormente para validação dos dados. Os dados restaurados e recuperados contêm o registro ou item mais recente no momento do backup? Esses dados se enquadram no RPO da workload?
7. Calcule o tempo necessário para a restauração e recuperação e compare-o com o RTO estabelecido. Esse processo se enquadra no RTO da workload? Por exemplo, compare o carimbo de data/hora em que o processo de restauração foi iniciado e que a validação da recuperação foi concluída para calcular quanto tempo esse processo leva. Todas as chamadas de API da AWS têm carimbo de data/hora e essas informações estão disponíveis no AWS CloudTrail. Embora essas informações possam fornecer detalhes sobre o início do processo de restauração, o carimbo final de data/hora da conclusão da validação deve ser registrado pela lógica de validação. Se você usar um processo automático, serviços como o Amazon DynamoDB poderão ser usados para armazenar essas informações. Além disso, muitos serviços da AWS oferecem um histórico de eventos que fornece informações sobre a data e hora em que determinadas ações ocorreram. No AWS Backup, as ações de backup e restauração são chamadas de trabalhos, e esses trabalhos contêm informações de data e hora como parte dos metadados, que podem ser usados para calcular o tempo necessário para restauração e recuperação.
8. Informe as partes interessadas se a validação de dados falhar ou se o tempo necessário para restauração e recuperação exceder o RTO estabelecido para a workload. Ao implementar a automação para fazer isso, como neste laboratório, serviços como o Amazon Simple Notification Service (Amazon SNS) podem ser usados para enviar notificações por push como e-mail ou SMS às partes interessadas. Essas mensagens também podem ser publicadas em aplicações de mensagens, como o Amazon Chime, o Slack ou o Microsoft Teams ou usadas para criar tarefas como OpsItems usando o AWS Systems Manager OpsCenter.
9. Automatize esse processo para ser executado periodicamente. Por exemplo, serviços como o AWS Lambda ou uma máquina de estado no AWS Step Functions podem ser usados para automatizar os processos de restauração e recuperação, e é possível usar o Amazon EventBridge para acionar esse fluxo de trabalho de automação periodicamente, conforme mostrado no diagrama de arquitetura abaixo. Saiba como Automatizar a validação da recuperação de dados com o AWS Backup. Além disso, este laboratório do Well-Architected fornece uma experiência prática de como automatizar várias das etapas aqui.

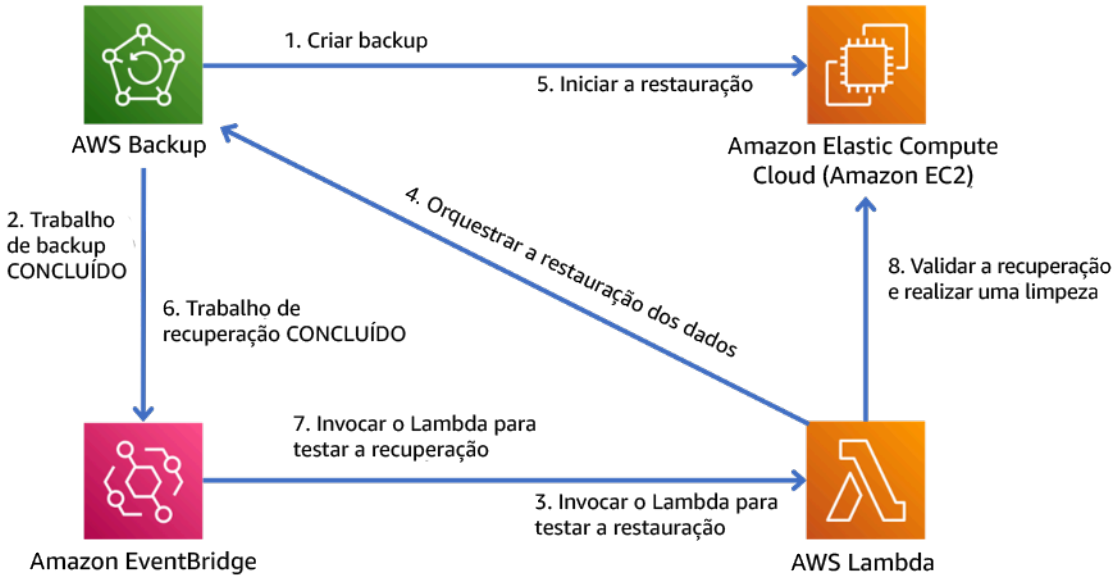


Figura 9. Um processo de backup e restauração automatizado

Nível de esforço do plano de implementação: moderado a alto, dependendo da complexidade dos critérios de validação.

Recursos

Documentos relacionados:

- Automate data recovery validation with AWS Backup (Automatizar validação de recuperação de dados com o AWS Backup)
- Parceiro do APN: parceiros que podem ajudar com o backup
- AWS Marketplace: products that can be used for backup (AWS Marketplace: produtos que podem ser usados para backup)
- Creating an EventBridge Rule That Triggers on a Schedule (Criar uma regra do EventBridge que é acionada de acordo com uma programação)
- Usar backup e restauração sob demanda para o DynamoDB
- O que é o AWS Backup?
- O que é o Reachability Analyzer?
- O que é o Reachability Analyzer?
- AWS Elastic Disaster Recovery

Exemplos relacionados:

- Laboratório do Well-Architected: teste de backup e restauração de dados

Use o isolamento de falhas para proteger a carga de trabalho

Os limites isolados de falhas restringem o efeito de uma falha em uma carga de trabalho a um número controlado de componentes. A falha não afeta os componentes fora do limite. Ao usar vários limites isolados de falhas, você pode restringir o impacto sobre sua carga de trabalho.

Práticas recomendadas

- REL10-BP01 Implantar a workload em vários locais
- REL10-BP02 Escolher os locais apropriados para sua implantação de vários locais
- REL10-BP03 Automatizar a recuperação de componentes restritos a um único local
- REL10-BP04 Usar arquiteturas de anteparo para limitar o escopo de impacto

REL10-BP01 Implantar a workload em vários locais

Distribua os dados e os recursos da workload por várias zonas de disponibilidade ou por Regiões da AWS, quando necessário. A diversidade dos locais pode variar conforme a necessidade.

Um dos princípios fundamentais do design de serviço na AWS é evitar pontos únicos de falha em infraestrutura física subjacente. Isso nos motiva a criar software e sistemas que usam várias zonas de disponibilidade e são resilientes à falha de uma única zona. De modo similar, os sistemas são criados para serem resilientes à falha de um único nó de computação, volume de armazenamento ou instância de um banco de dados. Ao criar um sistema que dependa de componentes redundantes, é importante garantir que os componentes operem de modo independente e, no caso de Regiões da AWS, de modo autônomo. Os benefícios obtidos com cálculos teóricos de disponibilidade com componentes redundantes só serão válidos se isso for verdadeiro.

Zonas de disponibilidade (AZ)

As Regiões da AWS são compostas de várias zonas de disponibilidade projetadas para serem independentes umas das outras. Cada zona de disponibilidade é separada por uma grande distância física de outras zonas para evitar cenários de falha correlacionados devido a riscos ambientais, como incêndios, enchentes e tornados. Cada zona de disponibilidade tem uma infraestrutura física independente: conexões dedicadas à rede elétrica, fontes de alimentação de reserva independentes, serviços mecânicos independentes e conectividade de rede independente dentro e além da zona de disponibilidade. O design limita as falhas em qualquer um desses sistemas apenas à AZ afetada.

Apesar de estarem geograficamente separadas, as zonas de disponibilidade estão localizadas na mesma área regional, permitindo uma rede de alto throughput e baixa latência. Toda a Região da AWS (em todas as zonas de disponibilidade, consistindo em vários datacenters fisicamente independentes) pode ser tratada como um único destino de implantação lógica para a workload, incluindo a capacidade de replicar dados de forma síncrona (por exemplo, entre bancos de dados). Assim, você pode usar as zonas de disponibilidade em uma configuração ativa/ativa ou ativa/em espera.

As zonas de disponibilidade são independentes e, portanto, a disponibilidade da workload aumenta quando ela é projetada para usar várias zonas. Alguns serviços da AWS (incluindo o plano de dados da instância do Amazon EC2) são implantados como serviços estritamente zonais, compartilhando o destino com a zona de disponibilidade em que estão. No entanto, as instâncias do Amazon EC2 nas outras AZs não serão afetadas e continuarão funcionando. Da mesma forma, se uma falha em uma zona de disponibilidade fizer com que um banco de dados do Amazon Aurora falhe, uma instância do Aurora de réplica de leitura em uma AZ não afetada poderá ser promovida automaticamente para primária. Entretanto, os serviços da AWS regionais (como o Amazon DynamoDB) usam várias zonas de disponibilidade em uma configuração ativa/ativa para atingir as metas de design de disponibilidade para aquele serviço, sem a necessidade de configurar o posicionamento da AZ.

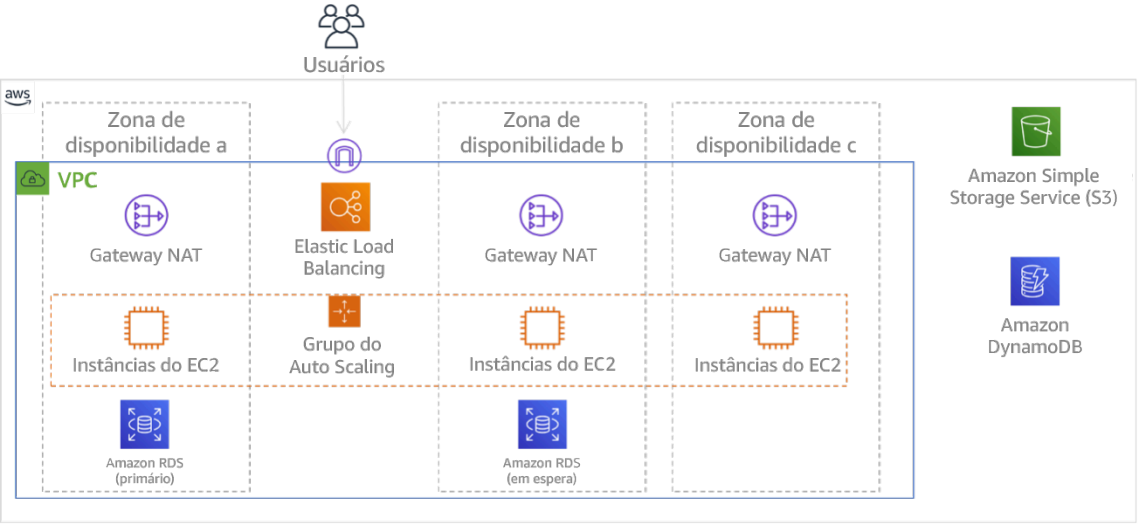


Figura 9: arquitetura multicamadas implantada em três Zonas de disponibilidade. Observe que o Amazon S3 e o Amazon DynamoDB são sempre multi-AZ automaticamente. O ELB também é implantado em todas as três zonas.

Embora os ambientes de gerenciamento da AWS costumem permitir o gerenciamento de recursos dentro de toda a região (várias zonas de disponibilidade), determinados ambientes (incluindo o Amazon EC2 e o Amazon EBS) podem filtrar os resultados para uma única zona de disponibilidade.

Quando isso é feito, a solicitação é processada apenas na zona de disponibilidade especificada, o que reduz a exposição a interrupções em outras zonas de disponibilidade. Veja um exemplo da AWS CLI que ilustra como obter informações da instância do Amazon EC2 apenas da zona de disponibilidade us-east-2c:

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

Zonas locais da AWS

Zonas locais da AWS atuam de forma semelhante às zonas de disponibilidade nas suas respectivas Região da AWS, pois elas podem ser selecionadas como um local de posicionamento para recursos zonais da AWS, como sub-redes e instâncias do EC2. O que as torna especiais é que elas estão localizadas não na Região da AWS associada, mas perto de grandes centros populacionais, industriais e de TI onde não existe nenhuma Região da AWS atualmente. No entanto, elas ainda mantêm uma conexão segura e de alta largura de banda entre as workloads locais na zona local e as executadas na Região da AWS. Você deve usar as zonas locais da AWS para implantar workloads mais perto dos seus usuários para requisitos de baixa latência.

Amazon Global Edge Network

A Amazon Global Edge Network consiste em locais da borda em cidades ao redor do mundo. O Amazon CloudFront usa essa rede para entregar conteúdo aos usuários finais com menor latência. O AWS Global Accelerator permite criar endpoints de workload nesses locais da borda para fornecer integração à rede global da AWS próxima aos seus usuários. O Amazon API Gateway habilita endpoints de API otimizados para borda usando uma distribuição do CloudFront para facilitar o acesso do cliente por meio do local da borda mais próximo.

Regiões da AWS

As Regiões da AWS foram projetadas para serem autônomas. Portanto, para usar uma abordagem multirregional, você pode implantar cópias dedicadas de serviços em cada região.

Uma abordagem multirregional é comum para estratégias de recuperação de desastres atenderem aos objetivos de recuperação quando ocorrem eventos pontuais de grande escala. Perceber Planejar para a recuperação de desastres (DR) para obter mais informações sobre essas estratégias. No entanto, aqui focaremos na disponibilidade, que busca entregar um objetivo de tempo de atividade médio ao longo do tempo. Para objetivos de alta disponibilidade, geralmente uma arquitetura multirregional será projetada para ser ativa/ativa, onde cada cópia de serviço (nas suas respectivas regiões) está ativa (atendimento a solicitações).

Recomendação

Os objetivos de disponibilidade para a maioria das workloads podem ser cumpridos usando uma estratégia multi-AZ em uma única Região da AWS. Considere arquiteturas multirregionais somente quando as workloads tiverem requisitos de disponibilidade extrema ou outros objetivos de negócios que exijam uma arquitetura multirregional.

A AWS oferece a capacidade de operar serviços entre regiões. Por exemplo, a AWS fornece replicação contínua e assíncrona de dados usando replicação do Amazon Simple Storage Service (Amazon S3), réplicas de leitura do Amazon RDS (incluindo réplicas de leitura do Aurora) e tabelas globais do Amazon DynamoDB. Com a replicação contínua, as versões dos dados estão disponíveis para uso quase imediato em cada uma das suas regiões ativas.

Ao usar o AWS CloudFormation, você pode definir a infraestrutura e implantá-la de forma consistente em todas as Contas da AWS e Regiões da AWS. O AWS CloudFormation StackSets estende essa funcionalidade, permitindo que criar, atualizar ou excluir pilhas do AWS CloudFormation em várias contas e regiões com uma única operação. Para implantações de instância do Amazon EC2, uma imagem de máquina da Amazon (AMI) é usada para fornecer informações como configuração de hardware e software instalado. É possível implementar um pipeline do construtor de imagem do Amazon EC2 que cria as AMIs necessárias e as copia para as regiões ativas. Isso garante que essas AMIs de referência (golden) tenham o necessário para implantar e expandir a workload em cada nova região.

Para rotear o tráfego, o Amazon Route 53 e o AWS Global Accelerator permitem a definição de políticas que determinam os usuários que vão para cada endpoint regional ativo. Com o Global Accelerator, você define uma discagem de tráfego para controlar a porcentagem do tráfego que é direcionado para cada endpoint da aplicação. O Route 53 é compatível com a abordagem de porcentagem e com várias outras políticas disponíveis, incluindo as baseadas em geoproximidade e latência. O Global Accelerator aproveita automaticamente a extensa rede de servidores de borda da AWS para integrar o tráfego à estrutura da rede da AWS o mais rápido possível, resultando em menores latências de solicitação.

Todos esses recursos operam de forma a preservar a autonomia de cada região. Há poucas exceções a essa abordagem, incluindo nossos serviços que fornecem entrega global de borda (como o Amazon CloudFront e o Amazon Route 53), juntamente com o ambiente de gerenciamento para o serviço AWS Identity and Access Management (IAM). A maioria dos serviços opera totalmente dentro de uma única região.

Datacenter no local

Para workloads executadas em um datacenter on-premises, arquitete uma experiência híbrida quando possível. O AWS Direct Connect fornece uma conexão de rede dedicada entre o local e a AWS, permitindo que você execute em ambos.

Outra opção é executar a infraestrutura e os serviços da AWS on-premises usando o AWS Outposts. O AWS Outposts é um serviço totalmente gerenciado que estende a infraestrutura da AWS, os serviços da AWS, as APIs e as ferramentas para o seu datacenter. A mesma infraestrutura de hardware usada na Nuvem AWS é instalada no seu datacenter. O AWS Outposts é então conectados à Região da AWS mais próxima. Em seguida, você pode usar AWS Outposts para oferecer suporte a cargas de trabalho com baixa latência ou requisitos de processamento de dados locais.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

- Use várias zonas de disponibilidade e Regiões da AWS. Distribua os dados e os recursos da workload por várias zonas de disponibilidade ou por Regiões da AWS, quando necessário. A diversidade dos locais pode variar conforme a necessidade.
  - Os serviços regionais são inerentemente implantados nas zonas de disponibilidade.
    - Isso inclui o Amazon S3, o Amazon DynamoDB e o AWS Lambda (quando não conectados a uma VPC).
  - Implemente suas cargas de trabalho baseadas em contêiner, instância e função em várias zonas de disponibilidade. Use datastores multizona, incluindo caches. Use os recursos do EC2 Auto Scaling, o posicionamento de tarefas do ECS, a configuração da função do AWS Lambda ao executá-lo na sua VPC e clusters do ElastiCache.
    - Use sub-redes que estão em zonas de disponibilidade separadas ao implantar grupos de Auto Scaling.
      - Exemplo: distribuição de instâncias entre zonas de disponibilidade
      - Estratégias de posicionamento de tarefas do Amazon ECS
      - Configuração de uma função do AWS Lambda para acessar recursos em uma Amazon VPC
      - Escolha de regiões e zonas de disponibilidade
    - Use sub-redes que estão em zonas de disponibilidade separadas ao implantar grupos de Auto Scaling.
      - Exemplo: distribuição de instâncias entre zonas de disponibilidade

    - Use os parâmetros de posicionamento de tarefas do ECS, especificando grupos de sub-rede do banco de dados.
        - Estratégias de posicionamento de tarefas do Amazon ECS
    - Use sub-redes em várias zonas de disponibilidade ao configurar uma função para executar na sua VPC.
        - Configuração de uma função do AWS Lambda para acessar recursos em uma Amazon VPC
    - Use várias zonas de disponibilidade com os clusters do ElastiCache.
        - Escolha de regiões e zonas de disponibilidade
- Se a workload precisar ser implantada em várias regiões, escolha uma estratégia multirregional. A maioria das necessidades de confiabilidade pode ser atendida em uma única Região da AWS usando uma estratégia de várias zonas de disponibilidade. Use uma estratégia multirregional quando necessário para atender às suas demandas empresariais.
    - AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
        - O backup para outra Região da AWS pode servir como mais uma camada visando garantir que os dados estejam disponíveis quando necessário.
        - Algumas workloads têm requisitos regulamentares que exigem o uso de uma estratégia multirregional.
- Avalie o AWS Outposts para a workload. Se a carga de trabalho exigir baixa latência do datacenter no local ou tiver requisitos de processamento de dados locais. Em seguida, execute a infraestrutura e os serviços da AWS on-premises usando o AWS Outposts
    - O que é o AWS Outposts?
- Determine se as zonas locais da AWS ajudam você a fornecer serviços aos usuários. Se você tiver requisitos de baixa latência, veja se as zonas locais da AWS estão próximas dos seus usuários. Se estiverem, use-as para implantar as workloads mais próximas desses usuários.
    - Perguntas frequentes sobre zonas locais da AWS

Recursos

Documentos relacionados:

- Infraestrutura global da AWS
- Perguntas frequentes sobre zonas locais da AWS
- Estratégias de posicionamento de tarefas do Amazon ECS

- Escolha de regiões e zonas de disponibilidade
- Exemplo: distribuição de instâncias entre zonas de disponibilidade
- Tabelas globais: replicação em várias regiões com o DynamoDB
- Uso de bancos de dados globais do Amazon Aurora
- Série de blogs sobre a criação de uma aplicação multirregional com os serviços da AWS
- O que é o AWS Outposts?

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure (NET339)

REL10-BP02 Escolher os locais apropriados para sua implantação de vários locais

Resultado desejado

Para alta disponibilidade, sempre (que possível) implante os componentes da workload em várias zonas de disponibilidade (AZs), conforme mostrado na figura 10. Para workloads com requisitos de resiliência extrema, avalie cuidadosamente as opções para uma arquitetura multirregional.

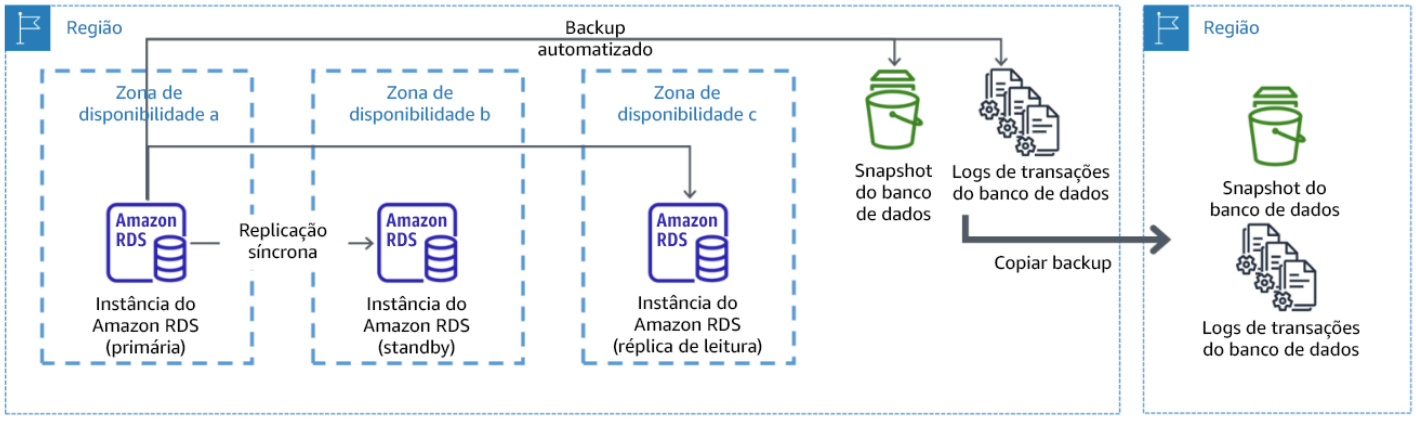


Figura 10: Uma implantação de banco de dados multi-AZ resiliente com backup para outra região da AWS

Antipadrões comuns:

- Projetar uma arquitetura multirregional quando uma arquitetura multi-AZ seria suficiente para atender aos requisitos.
- Não contabilizar as dependências entre os componentes da aplicação caso os requisitos de resiliência e de vários locais forem diferentes entre esses componentes.

Benefícios do estabelecimento desta prática recomendada:

Para resiliência, você deve usar uma abordagem que construa camadas de defesa. Uma camada protege contra interrupções menores e mais comuns criando uma arquitetura altamente disponível usando várias AZs. Outra camada de defesa destina-se a proteger contra eventos raros, como desastres naturais generalizados e interrupções em nível regional. Essa segunda camada envolve arquitetar a aplicação para abranger várias Regiões da AWS.

- A diferença entre as disponibilidades de 99,5% e 99,99% é superior a 3,5 horas por mês. A disponibilidade esperada de uma workload só pode atingir “quatro noves” se estiver em várias AZs.
- Ao executar a workload em várias AZs, é possível isolar falhas de energia, refrigeração, rede e a maioria dos desastres naturais, como incêndio e inundação.
- A implementação de uma estratégia multirregional para a workload ajuda a protegê-la contra desastres naturais generalizados, que afetam uma grande área geográfica de um país, ou falhas técnicas de escopo regional. Esteja ciente de que a implementação de uma arquitetura multirregional pode ser complexa e, geralmente, não é necessária para a maioria das workloads.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

Para um evento de desastre baseado em interrupção ou perda parcial de uma zona de disponibilidade, implementar uma workload altamente disponível em várias zonas de disponibilidade em uma única Região da AWS ajuda a atenuar os desastres naturais e técnicos. Cada Região da AWS é composta por várias zonas de disponibilidade, cada uma isolada de falhas nas outras zonas e separadas por uma distância significativa. No entanto, para um evento de desastre que inclua o risco de perder vários componentes da zona de disponibilidade, distantes umas das outras de forma significativa, deve-se implementar opções de recuperação de desastres para atenuar as falhas de escopo regional. Para workloads que exigem resiliência extrema (infraestrutura crítica, aplicações

relacionados à integridade, infraestrutura do sistema financeiro etc.), pode ser necessária uma estratégia multirregional.

Etapas da implementação

1. Avalie a workload e determine se as necessidades de resiliência podem ser atendidas por uma abordagem multi-AZ (Região da AWS única) ou se elas requerem uma abordagem multirregional. A implementação de uma arquitetura multirregional para atender a esses requisitos introduzirá complexidade adicional, portanto, considere cuidadosamente seu caso de uso e seus requisitos. Os requisitos de resiliência quase sempre podem ser atendidos usando uma única Região da AWS. Considere os seguintes requisitos possíveis ao determinar a necessidade de usar várias regiões:
   a. Recuperação de desastres (DR): para um evento de desastre baseado em interrupção ou perda parcial de uma zona de disponibilidade, implementar uma workload altamente disponível em várias zonas de disponibilidade em uma única Região da AWS ajuda a atenuar os desastres naturais e técnicos. Para um evento de desastre que inclua o risco de perder vários componentes da zona de disponibilidade, distantes umas das outras de forma significativa, deve-se implementar recuperação de desastres multirregional para atenuar os desastres naturais ou as falhas técnicas de escopo regional.
   b. Alta disponibilidade (AD): é possível usar uma arquitetura multirregional (usando várias AZs em cada região) para alcançar uma disponibilidade superior a quatro noves (> 99,99%).
   c. Localização de pilhas: ao implantar uma workload para um público global, é possível implantar pilhas localizadas em diferentes Regiões da AWS para atender o público nessas regiões. A localização pode incluir idioma, moeda e tipos de dados armazenados.
   d. Proximidade aos usuários: ao implantar uma workload para um público global, é possível reduzir a latência implantando pilhas em Regiões da AWS perto de onde os usuários finais estão.
   e. Residência de dados: algumas workloads estão sujeitas a requisitos de residência de dados, em que os dados de determinados usuários devem permanecer dentro das fronteiras de um país específico. Com base na regulamentação em questão, você pode optar por implantar uma pilha inteira ou apenas os dados na Região da AWS dentro dessas fronteiras.
2. Veja alguns exemplos de funcionalidade multi-AZ fornecida pelos serviços da AWS:
   a. Para proteger workloads usando o EC2 ou o ECS, implante um Elastic Load Balancer na frente dos recursos de computação. Em seguida, o Elastic Load Balancing fornece a solução para detectar instâncias em zonas com problemas de integridade e rotear o tráfego para as íntegras.
      i. Conceitos básicos do Application Load Balancers

   ii. Conceitos básicos do Network Load Balancers

b. Em caso de instâncias do EC2 executando software comercial pronto para uso que não oferece suporte ao balanceamento de carga, é possível obter uma forma de tolerância a falhas implementando uma metodologia de recuperação de desastre multi-AZ.

   i. the section called “REL13-BP02 Usar estratégias de recuperação definidas para cumprir os objetivos de recuperação”

c. Para tarefas do Amazon ECS, implante seu serviço uniformemente em três AZs para alcançar um equilíbrio entre disponibilidade e custo.

   i. Práticas recomendadas de disponibilidade do Amazon ECS | Contêineres

d. Para os que não são Aurora Amazon RDS, você pode escolher multi-AZ como uma opção de configuração. Em caso de falha da instância de banco de dados primário, o Amazon RDS promove automaticamente um banco de dados em espera para receber o tráfego em outra zona de disponibilidade. Também é possível criar réplicas de leitura multirregionais para melhorar a resiliência.

   i. Implantações multi-AZ do Amazon RDS

   ii. Criação de uma réplica de leitura em uma Região da AWS diferente

3. Veja alguns exemplos de funcionalidade multirregional fornecida pelos serviços da AWS:

a. Para workloads do Amazon S3, em que a disponibilidade multi-AZ é fornecida automaticamente pelo serviço, considere os pontos de acesso multirregionais se for necessária uma implantação multirregional.

   i. Pontos de acesso multirregionais no Amazon S3

b. Para tabelas do DynamoDB, em que a disponibilidade multi-AZ é fornecida automaticamente pelo serviço, é possível converter tabelas existentes em tabelas globais para aproveitar várias regiões.

   i. Conversão de tabelas de região única do Amazon DynamoDB em tabelas globais

c. Se a workload for liderada pelo Application Load Balancers ou pelo Network Load Balancers, use o AWS Global Accelerator para melhorar a disponibilidade da aplicação direcionando o tráfego para várias regiões que contenham endpoints íntegros.

   i. Endpoints para aceleradores padrão no AWS Global Accelerator – AWS Global Accelerator (amazon.com)

d. Para aplicações que utilizam o AWS EventBridge, considere os barramentos entre regiões para encaminhar eventos para outras regiões selecionadas.

   i. Envio e recebimento de eventos do Amazon EventBridge entre Regiões da AWS

e. Para bancos de dados do Amazon Aurora, considere os bancos de dados globais do Aurora, que abrangem várias regiões da AWS. Os clusters existentes também podem ser modificados para adicionar novas regiões.
   i. Conceitos básicos dos bancos de dados globais do Amazon Aurora
f. Se a workload incluir chaves de criptografia do AWS Key Management Service (AWS KMS), considere se as chaves multirregionais são apropriadas para a aplicação.
   i. Chaves multirregionais no AWS KMS
g. Para recursos de outros serviços da AWS, consulte Série sobre a criação de uma aplicação multirregional com os serviços da AWS

Nível de esforço para o plano de implementação: Moderado a alto

Recursos

Documentos relacionados:

- Série sobre a criação de uma aplicação multirregional com os serviços da AWS
- Arquitetura de recuperação de desastres (DR) na AWS, parte IV: multissite ativo-ativo
- Infraestrutura global da AWS
- Perguntas frequentes sobre zonas locais da AWS
- Arquitetura de recuperação de desastres (DR) na AWS, parte I: estratégias de recuperação na nuvem
- Recuperação de desastres é diferente na nuvem
- Tabelas globais: replicação em várias regiões com o DynamoDB

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Auth0: arquitetura multirregional de alta disponibilidade que escala a até mais de 1,5 bilhão de logins por mês com failover automático

Exemplos relacionados:

- Arquitetura de recuperação de desastres (DR) na AWS, parte I: estratégias de recuperação na nuvem
- DTCC alcança resiliência muito além do que conseguem em ambiente on-premises
- Expedia Group usa uma arquitetura multirregional e de várias zonas de disponibilidade com um serviço de DNS proprietário para adicionar resiliência às aplicações
- Uber: recuperação de desastres para Kafka multirregional
- Netflix: ativo-ativo para resiliência multirregional
- Como criamos residência de dados para o Atlassian Cloud
- Intuit TurboTax executa em duas regiões

REL10-BP03 Automatizar a recuperação de componentes restritos a um único local

Se os componentes da workload só puderem ser executados em uma única zona de disponibilidade ou datacenter on-premises, você deverá implementar capacidade suficiente para fazer uma recompilação completa da workload em conformidade com os objetivos de recuperação definidos.

Nível de risco exposto se esta prática recomendada não é estabelecida: médio

Orientações para a implementação

Se a melhor prática para implantar a carga de trabalho em vários locais não for possível devido a restrições tecnológicas, você deverá implementar um caminho alternativo para a resiliência. Você deve automatizar a capacidade de recriar a infraestrutura necessária, reimplantar aplicativos e recriar os dados necessários para esses casos.

Por exemplo, o Amazon EMR executa todos os nós de um determinado cluster na mesma zona de disponibilidade, pois a execução de um cluster na mesma zona melhora a performance dos fluxos de trabalho, pois fornece uma taxa de acesso a dados mais alta. Se esse componente for necessário para a resiliência da workload, você deverá ter uma maneira de reimplantar o cluster e seus dados. Além disso, para o Amazon EMR, você deve provisionar redundância de maneiras diferentes de usar o Multi-AZ. Você pode provisionar vários nós. Usando o EMR File System (EMRFS), os dados no EMR podem ser armazenados no Amazon S3, que, por sua vez, podem ser replicados em várias zonas de disponibilidade ou Regiões da AWS.

Da mesma forma, o Amazon Redshift, por padrão, provisiona o cluster em uma zona de disponibilidade escolhida aleatoriamente dentro da Região da AWS selecionada. Todos os nós de cluster são provisionados na mesma zona.

Para workloads com estado baseadas em servidor e implantadas em um datacenter on-premises, é possível usar o AWS Elastic Disaster Recovery para proteger as workloads na AWS. Se você já estiver hospedado na AWS, poderá usar o Elastic Disaster Recovery para proteger a workload em uma zona de disponibilidade ou região alternativa. O Elastic Disaster Recovery usa a replicação contínua no nível de bloco para uma área de preparação leve visando fornecer recuperação rápida e confiável de aplicações on-premises e na nuvem.

Etapas da implementação

1. Implemente a autorreparação. Quando possível, use a escalabilidade automática para implantar instâncias ou contêineres. Quando não for possível, use a recuperação automática de instâncias do EC2 ou implemente a automação de autorreparação com base nos eventos de ciclo de vida do contêiner do Amazon EC2 ou do ECS.
    - Use os grupos do Amazon EC2 Auto Scaling para workloads de contêiner e instâncias que não têm requisitos para um endereço IP de instância única, endereço IP privado, endereço IP elástico e metadados da instância.
        - Os dados do usuário do modelo de execução podem ser usados para implementar uma automação que pode recuperar automaticamente a maioria das workloads.
    - Use a recuperação automática de instâncias do Amazon EC2 para workloads que exijam um único endereço de ID da instância, endereço IP privado, endereço IP elástico e metadados da instância.
        - A recuperação automática enviará alertas de status de recuperação para um tópico do SNS quando a falha na instância for detectada.
    - Use os eventos do ciclo de vida da instância do Amazon EC2 ou os eventos do Amazon ECS para automatizar a autorreparação, em que não seja possível usar a escalabilidade automática ou a recuperação do EC2.
        - Use os eventos para chamar a automação que recuperará seu componente de acordo com a lógica do processo necessária.
    - Proteja workloads com estado que são limitadas a um único local usando o AWS Elastic Disaster Recovery.

Recursos

Documentos relacionados:

- Eventos do Amazon ECS
- Ganchos do ciclo de vida do Amazon EC2 Auto Scaling
- Recupere sua instância.
- Escalabilidade automática do serviço
- O que é o Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP04 Usar arquiteturas de anteparo para limitar o escopo de impacto

Implemente arquiteturas de anteparo (também chamadas de arquiteturas baseadas em células) para restringir o efeito ou a falha em uma workload a um número limitado de componentes.

Resultado desejado: uma arquitetura baseada em células usa várias instâncias isoladas de uma workload, em que cada instância é considerada uma célula. Cada célula é independente, não compartilha o estado com outras células e processa um subconjunto das solicitações gerais da workload. Isso reduz o possível impacto de uma falha, como uma atualização de software incorreta, a uma célula individual e às solicitações que ela está processando. Se uma workload usa 10 células para atender a 100 solicitações, quando ocorre uma falha, 90% das solicitações gerais não seriam afetadas pela falha.

Antipadrões comuns:

- Permitir que as células cresçam sem limites.
- Aplicar implantações ou atualizações de código a todas as células ao mesmo tempo.
- Compartilhar o estado ou os componentes entre as células (com a exceção da camada do roteador).
- Adicionar negócios complexos ou rotear lógica para a camada do roteador.
- Não minimizar as interações entre as células.

Benefícios do estabelecimento dessa prática recomendada: com arquiteturas baseadas em células, muitos tipos comuns de falhas são contidas na própria célula, fornecendo isolamento de falhas

adicional. Esses limites de falhas podem fornecer resiliência com relação a tipos de falha que, de outra maneira, seriam difíceis de conter, como implantações de código sem êxito ou solicitações corrompidas ou que acionam um modo de falha específico (também conhecidas como solicitações com conteúdo malicioso).

Orientação de implementação

Em um navio, as anteparas garantem que uma ruptura no casco seja contida em uma seção do casco. Em sistemas complexos, esse padrão costuma ser replicado para permitir o isolamento de falhas. Os limites isolados de falhas restringem o efeito de uma falha em uma workload a um número controlado de componentes. A falha não afeta os componentes fora do limite. Ao usar vários limites isolados de falhas, você pode restringir o impacto sobre sua carga de trabalho. Na AWS, os clientes podem usar várias zonas de disponibilidade e regiões para fornecer o isolamento de falhas, mas o conceito do isolamento de falhas também pode ser estendido à arquitetura da workload.

A workload geral é composta por células particionadas por uma chave de partição. Essa chave precisa se alinhar à granularidade do serviço, ou da maneira natural que a workload de um serviço pode ser subdividida em interações mínimas entre células. Exemplos de chaves de partição são ID de cliente, ID de recurso ou qualquer outro parâmetro facilmente acessível na maioria das chamadas de API. Uma camada de roteamento de célula distribui solicitações a células individuais com base na chave de partição e apresenta um único endpoint aos clientes.

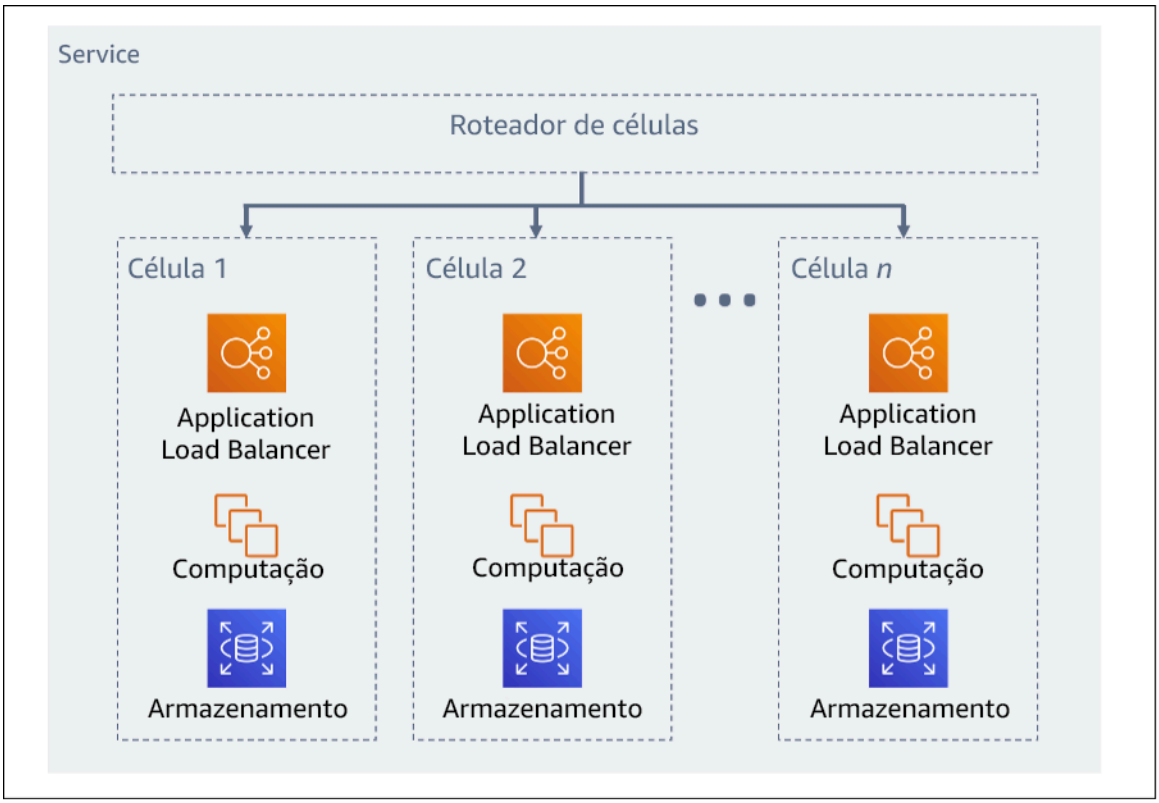


Figura 11: arquitetura baseada em células

Etapas da implementação

Ao projetar uma arquitetura baseada em células, há várias considerações de design a levar em conta:

1. Chave de partição: deve-se dedicar uma consideração especial ao escolher a chave de partição.
    - Ela precisa se alinhar à granularidade do serviço, ou da maneira natural que a workload de um serviço pode ser subdividida em interações mínimas entre células. Dentre os exemplos estão o `ID de cliente` ou `ID de recurso`.
    - A chave de partição deve estar disponível em todas as solicitações, seja diretamente ou de uma maneira que possa ser facilmente inferida de forma determinística por outros parâmetros.
2. Mapeamento de células persistentes: os serviços de upstream só devem interagir com uma única célula pelo ciclo de vida dos recursos.
    - Dependendo da workload, uma estratégia de migração de células pode ser necessária para migrar os dados de uma célula para outra. Um possível cenário de quando é necessário fazer uma migração de célula seria quando um usuário ou recurso específico na workload fica grande demais e exige uma célula dedicada.
    - As células não devem compartilhar estado ou componentes entre si.
    - Consequentemente, as interações entre as células devem ser evitadas e mantidas no mínimo, já que elas podem criar dependências entre as células e, assim, reduzir as melhorias do isolamento de falhas.
3. Camada do roteador: a camada do roteador é um componente compartilhado entre células e, portanto, não pode seguir a mesma estratégia de compartimentalização das células.
    - É recomendável que a camada do roteador distribua as solicitações para células individuais usando um algoritmo de mapeamento de partição de maneira computacionalmente eficiente, como combinando funções de hash criptográficas e aritmética modular para mapear chaves de partição a células.
    - Para evitar impactos em várias células, a camada de roteamento deve permanecer o mais simples e horizontalmente escalável possível, o que exige evitar uma lógica empresarial complexa nessa camada. Isso tem o benefício adicional de facilitar a compreensão de seu comportamento esperado em todos os momentos, permitindo uma capacidade de testes completa. Conforme explicado por Colm MacCárthaigh em Reliability, constant work, and a good cup of coffee (Confiabilidade, trabalho constante e uma boa xícara de café), designs

simples e padrões de trabalho constantes produzem sistemas confiáveis e reduzem a antifragilidade.

4. Tamanho da célula: as células devem ter um tamanho máximo e não devem ter permissão para crescer além disso.
   - O tamanho máximo deve ser identificado com a realização de testes completos, até que os pontos de ruptura sejam atingidos e as margens operacionais seguras sejam estabelecidas. Para obter mais detalhes sobre como implementar práticas de testes, consulte REL07-BP04 Fazer o teste de carga da sua workload
   - A workload geral deve crescer com a adição de mais células, permitindo que a workload seja escalada com aumentos na demanda.
5. Estratégias de várias zonas de disponibilidade ou várias regiões: várias camadas de resiliência devem ser utilizadas para a proteção contra diferentes domínios de falha.
   - Para resiliência, você deve usar uma abordagem que crie camadas de defesa. Uma camada protege contra interrupções menores e mais comuns criando uma arquitetura altamente disponível usando várias AZs. Outra camada de defesa destina-se a proteger contra eventos raros, como desastres naturais generalizados e interrupções em nível regional. Essa segunda camada envolve arquitetar a aplicação para abranger várias Regiões da AWS. A implementação de uma estratégia multirregional para a workload ajuda a protegê-la contra desastres naturais generalizados, que afetam uma grande área geográfica de um país, ou falhas técnicas de escopo regional. Esteja ciente de que a implementação de uma arquitetura multirregional pode ser complexa e, geralmente, não é necessária para a maioria das workloads. Para obter mais detalhes, consulte REL10-BP02 Escolher os locais apropriados para sua implantação de vários locais.
6. Implantação de código: uma estratégia de implantação de código escalonada deve ter preferência com relação à implantação de alterações de código em todas as células ao mesmo tempo.
   - Isso ajudará a reduzir a possibilidade de falhas em várias células devido a uma implantação incorreta ou a erro humano. Para obter mais detalhes, consulte Automating safe, hands-off deployment (Automatizar uma implantação prática e segura).

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto

Recursos

Práticas recomendadas relacionadas:

- REL07-BP04 Fazer o teste de carga da sua workload

- REL10-BP02 Escolher os locais apropriados para sua implantação de vários locais

Documentos relacionados:

- Reliability, constant work, and a good cup of coffee (Confiabilidade, trabalho constante e uma boa xícara de café)
- AWS and Compartmentalization (AWS e compartimentalização)
- isolamento de carga de trabalho usando fragmentos aleatórios
- Automating safe, hands-off deployment (Automatizar uma implantação prática e segura)

Vídeos relacionados:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: fechar ciclos e abrir mentes: como controlar sistemas, sejam grandes ou pequenos)
- AWS re:Invent 2018: como a AWS reduz o impacto das falhas (ARC338)
- Fragmentação aleatória: AWS re:Invent 2019: apresentação da Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021: tudo falha o tempo todo: como criar o design visando a resiliência

Exemplos relacionados:

- Laboratório do Well-Architected: isolamento de falhas com fragmentação aleatória

Projete a workload para resistir às falhas de componentes

As cargas de trabalho que exigem alta disponibilidade e baixo Tempo médio até a recuperação (MTTR) devem ser projetadas visando a resiliência.

Práticas recomendadas

- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP02 Failover para recursos íntegros
- REL11-BP03 Automatizar a reparação em todas as camadas

- REL11-BP04 Confiar no plano de dados e não no ambiente de gerenciamento durante a recuperação
- REL11-BP05 Usar estabilidade estática para evitar o comportamento bimodal
- REL11-BP06 Enviar notificações quando os eventos afetarem a disponibilidade
- REL11-BP07 Arquitetar o produto para cumprir as metas de disponibilidade e os acordos de nível de serviço (SLAs) de tempo de atividade

REL11-BP01 Monitorar todos os componentes da workload para detectar falhas

Monitore constantemente a integridade da workload para que você e seus sistemas automatizados detectem falhas ou degradações assim que elas ocorrerem. Monitore os indicadores-chave de performance (KPIs) com base no valor empresarial.

Todos os mecanismos de reparo e recuperação devem começar com a capacidade de detectar problemas rapidamente. As falhas técnicas devem ser detectadas primeiro para que possam ser resolvidas. No entanto, a disponibilidade é baseada na capacidade da workload de entregar valor empresarial, portanto, os indicadores-chave de performance (KPIs) que medem isso precisam fazer parte da sua estratégia de detecção e remediação.

Resultado desejado: Os componentes essenciais de uma workload são monitorados de forma independente para detectar e alertar sobre falhas quando e onde elas acontecem.

Antipadrões comuns:

- Nenhum alarme foi configurado, portanto as interrupções ocorrem sem notificação.
- Os alarmes existem, mas com limites que não permitem um tempo adequado para reação.
- As métricas não são coletadas com frequência suficiente para atender ao objetivo de tempo de recuperação (RTO)
- Somente as interfaces da workload voltadas para o cliente são monitoradas ativamente.
- Coleta apenas das métricas técnicas, não das métricas de função de negócios.
- Não há métricas que medem a experiência do usuário da workload.
- São criados monitores em excesso.

Benefícios de estabelecer esta prática recomendada: O monitoramento adequado de todas as camadas permite reduzir o tempo de recuperação ao reduzir o tempo de detecção.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

Identifique todas as workloads que serão analisadas para monitoramento. Depois de identificar todos os componentes da workload que precisarão ser monitorados, você precisará determinar o intervalo de monitoramento. O intervalo de monitoramento terá um impacto direto na rapidez com que a recuperação pode ser iniciada com base no tempo necessário para detectar uma falha. O tempo médio de detecção (MTTD) é a quantidade de tempo entre a ocorrência de uma falha e o início das operações de reparo. A lista de serviços deve ser extensa e completa.

O monitoramento deve abranger todas as camadas da pilha de aplicações, incluindo aplicação, plataforma, infraestrutura e rede.

Sua estratégia de monitoramento deve considerar o impacto de falhas cinzentas. Para obter mais detalhes sobre falhas cinzentas, consulte Falhas cinzentas no whitepaper Advanced Multi-AZ Resilience Patterns (Padrões avançados de resiliência Multi-AZ).

Etapas da implementação

- O intervalo de monitoramento depende da rapidez com que você precisa fazer a recuperação. O tempo de recuperação é determinado pelo tempo necessário para a recuperação. Desse modo, você deve considerar esse tempo e o objetivo de tempo de recuperação (RTO) para determinar a frequência da coleta.
- Configure o monitoramento detalhado de componentes e serviços gerenciados.
  - Determine se o monitoramento detalhado para instâncias do EC2 e Auto Scaling são necessários. O monitoramento detalhado fornece métricas de intervalo de um minuto, e o monitoramento padrão fornece métricas de intervalo de cinco minutos.
  - Determine se o monitoramento aprimorado para RDS é necessário. O monitoramento aprimorado usa um agente nas instâncias do RDS para obter informações úteis sobre processos ou threads diferentes.
  - Determine os requisitos de monitoramento de componentes essenciais sem servidor para Lambda, o API Gateway, o Amazon EKS, o Amazon ECS, e todos os tipos de balanceadores de carga.
  - Determine os requisitos de monitoramento dos componentes de armazenamento para Amazon S3, o Amazon FSx, o Amazon EFSe o Amazon EBS.

- Crie métricas personalizadas para medir os indicadores-chave de performance (KPIs) do negócio. As workloads implementam as principais funções empresariais, que devem ser usadas como KPIs para ajudar a identificar quando ocorre um problema indireto.
- Utilize os canários de usuário para monitorar a experiência do usuário e verificar se há falhas. O teste de transação sintética (também conhecido como “teste canário”, que não deve ser confundido com “implantações canário”), que pode executar e simular o comportamento do cliente, está entre os processos de teste mais importantes. Execute esses testes constantemente nos endpoints da workload de diversos locais remotos.
- Crie métricas personalizadas que rastreiem a experiência do usuário. Se você puder estabelecer instrumentos de medição da experiência do cliente, conseguirá determinar o momento de degradação da experiência do consumidor.
- Defina alarmes para detectar quando uma parte da workload não estiver funcionando corretamente e indicar quando deve ser feita a escalabilidade automática dos recursos. Os alarmes podem ser exibidos visualmente em painéis, enviar alertas pelo Amazon SNS ou por e-mail e trabalhar com o Auto Scaling para aumentar ou reduzir a escala dos recursos da workload.
- Crie painéis para visualizar suas métricas. É possível usar os painéis para ver as tendências, os casos atípicos e outros indicadores de possíveis problemas ou para obter uma indicação de problemas a serem investigados.
- Crie monitoramento de rastreamento distribuído para seus serviços. Com o monitoramento distribuído, você compreende como está a performance de sua aplicação e seus serviços subjacentes para identificar e solucionar a causa principal de problemas e erros de performance.
- Crie painéis de sistemas de monitoramento (usando CloudWatch ou X-Ray) e coleta de dados em uma Região e conta separadas.
- Crie integração para o monitoramento do Amazon Health Aware para permitir a visibilidade de monitoramento de recursos da AWS que possam ter degradações. Para workloads essenciais aos negócios, essa solução fornece acesso a alertas proativos e em tempo real para serviços da AWS.

Recursos

Práticas recomendadas relacionadas:

- Definição de disponibilidade
- REL11-BP06 Enviar notificações quando os eventos afetarem a disponibilidade

Documentos relacionados:

- Amazon CloudWatch Synthetics permite criar canários de usuário
- Habilitar ou desabilitar o monitoramento detalhado da instância
- Monitoramento aprimorado
- Monitoramento de grupos e instâncias do Auto Scaling usando o Amazon CloudWatch
- Publicar métricas personalizadas
- Uso dos alarmes do Amazon CloudWatch
- Uso de painéis do CloudWatch
- Uso de painéis do CloudWatch entre regiões e contas
- Uso do rastreamento do X-Ray entre regiões e contas
- Compreensão da disponibilidade
- Implementação do Amazon Health Aware (AHA)

Vídeos relacionados:

- Mitigating gray failures (Mitigando falhas cinzentas)

Exemplos relacionados:

- Laboratório do Well-Architected: nível 300: implementação de verificações de integridade e do gerenciamento de dependências para melhorar a confiabilidade
- Um workshop de observabilidade: explore o X-Ray

Ferramentas relacionadas:

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 Failover para recursos íntegros

Se ocorrer uma falha no recurso, os recursos íntegros deverão continuar atendendo às solicitações. Para falhas de localização (como zona de disponibilidade ou Região da AWS), garanta que você tenha sistemas implementados para realizar failover para recursos íntegros em locais sem problemas.

Ao projetar um serviço, distribua a carga entre recursos, zonas de disponibilidade ou regiões. Portanto, a falha ou a deficiência de um recurso individual podem ser atenuadas transferindo o tráfego para os recursos íntegros restantes. Pense em como os serviços são descobertos e encaminhados em caso de falha.

Projete seus serviços pensando na recuperação de falhas. Na AWS, projetamos os serviços para minimizar o tempo para recuperação de falhas e o impacto sobre os dados. Nossos serviços usam principalmente datastores que reconhecem solicitações apenas após serem armazenadas de modo durável entre várias réplicas em uma região. Eles são criados para usar isolamento com base em células e usar o isolamento de falhas fornecido por zonas de disponibilidade. Usamos automação amplamente em nossos procedimentos operacionais. Também otimizamos nossa funcionalidade de substituir e reiniciar para a recuperação rápida de interrupções.

Os padrões e os designs que permitem o failover variam para cada serviço de plataforma da AWS. Muitos serviços gerenciados nativos da AWS são nativamente várias zonas de disponibilidade (como o Lambda ou o API Gateway). Outros serviços da AWS (como EC2 e EKS) exigem designs específicos de práticas recomendadas para oferecer compatibilidade com o failover de recursos ou armazenamento de dados entre AZs.

O monitoramento deve ser configurado para conferir se o recurso de failover está íntegro, acompanhar o andamento do failover dos recursos e monitorar a recuperação do processo empresarial.

Resultado desejado: Os sistemas são capazes de usar novos recursos de forma automática ou manual para se recuperarem da degradação.

Antipadrões comuns:

- Planejar o fracasso não faz parte da fase de planejamento e design.
- O RTO e o RPO não são estabelecidos.
- Monitoramento insuficiente para detectar falhas nos recursos.
- Isolamento adequado dos domínios de falha.
- O failover multirregional não é considerado.
- A detecção de falhas é sensível ou agressiva demais ao decidir realizar o failover.
- Não testar nem validar o design de failover.
- Executar a automação de autorreparação sem notificar que a reparação era necessária.

- Falta de um período de amortecimento a fim de evitar falhas muito precoces.

Benefícios de estabelecer esta prática recomendada: Você pode criar sistemas mais resilientes que mantenham a confiabilidade em caso de falhas, degradando-se normalmente e se recuperando com rapidez.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

Os serviços da AWS, como o Elastic Load Balancing e o Amazon EC2 Auto Scaling, ajudam a distribuir a carga entre recursos e zonas de disponibilidade. Portanto, a falha de um recurso individual (como uma instância do EC2) ou o comprometimento de uma zona de disponibilidade podem ser atenuados desviando o tráfego para os recursos íntegros restantes.

Para workloads multirregionais, os projetos são mais complicados. Por exemplo, réplicas de leitura entre regiões permitem que você implante os dados em várias Regiões da AWS. No entanto, o failover ainda é necessário para promover a réplica de leitura como primária e direcionar seu tráfego para o novo endpoint. O Amazon Route 53, o Route 53, o Route 53 ARC, o CloudFront e o AWS Global Accelerator podem ajudar a direcionar o tráfego entre Regiões da AWS.

Serviços da AWS, como Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge ou Amazon DynamoDB, são implantados automaticamente em várias zonas de disponibilidade pela AWS. Em caso de falha, esses serviços da AWS direcionam automaticamente o tráfego para locais íntegros. Os dados são armazenados de forma redundante em várias zonas de disponibilidade e permanecem disponíveis.

Para o Amazon RDS, o Amazon Aurora, o Amazon Redshift, o Amazon EKS ou o Amazon ECS, Multi-AZ é uma opção de configuração. A AWS poderá direcionar o tráfego para a instância íntegra se o failover for iniciado. Essa ação de failover pode ser realizada pela AWS ou conforme exigido pelo cliente.

Para instâncias do Amazon EC2, o Amazon Redshift, tarefas do Amazon ECS ou pods do Amazon EKS, você escolhe em quais zonas de disponibilidade implantar. Para alguns designs, o Elastic Load Balancing fornece a solução para detectar instâncias em zonas não íntegras e direcionar o tráfego para as zonas íntegras. O Elastic Load Balancing também pode rotear o tráfego para componentes em seu datacenter on-premises.

Para o failover de tráfego em várias regiões, o redirecionamento pode utilizar o Amazon Route 53, o Route 53 ARC, o AWS Global Accelerator, o Route 53 Private DNS for VPCs ou o CloudFront para oferecer uma maneira de definir domínios da Internet e atribuir políticas de roteamento, incluindo verificações de integridade, para rotear o tráfego para regiões íntegras. O AWS Global Accelerator fornece endereços IP estáticos que atuam como um ponto de entrada fixo para sua aplicação e, depois, são roteados para os endpoints nas Regiões da AWS de sua escolha, usando a rede global da AWS em vez da Internet com o objetivo de melhorar o desempenho e a confiabilidade.

Etapas da implementação

- Crie designs de failover para todas as aplicações e serviços apropriados. Isole cada componente da arquitetura e crie designs de failover que atendam ao RTO e ao RPO de cada componente.
- Configure ambientes inferiores (como desenvolvimento ou teste) com todos os serviços necessários para ter um plano de failover. Implemente as soluções usando a infraestrutura como código (IaC) para garantir a repetibilidade.
- Configure um local de recuperação, como uma segunda região, para implementar e testar os designs de failover. Se necessário, os recursos para testes podem ser configurados temporariamente para limitar os custos adicionais.
- Determine quais planos de failover são automatizados pela AWS, quais podem ser automatizados por um processo de DevOps e quais podem ser manuais. Documente e avalie o RTO e o RPO de cada serviço.
- Crie um manual de failover e inclua todas as etapas para realizar o failover de cada recurso, aplicação e serviço.
- Crie um manual de failback e inclua todas as etapas de failback (com tempo) de cada recurso, aplicação e serviço.
- Crie um plano para iniciar e ensaiar o manual. Use simulações e testes de caos para testar a automação e as etapas do manual.
- Para danos na localização (como zona de disponibilidade ou Região da AWS), garanta que você tenha sistemas implementados para realizar failover para recursos íntegros em locais sem problemas. Confira a cota, os níveis de ajuste de escala automático e os recursos em execução antes do teste de failover.

Recursos

Práticas recomendadas relacionadas ao Well-Architected:

- REL13 – Plano para DR
- REL10 – Usar o isolamento de falhas para proteger a workload

Documentos relacionados:

- Setting RTO and RPO Targets
- Set up Route 53 ARC with application loadbalancers
- Failover using Route 53 Weighted routing
- DR with Route 53 ARC
- EC2 with autoscaling
- EC2 Deployments - Multi-AZ
- ECS Deployments - Multi-AZ
- Switch traffic using Route 53 ARC
- Lambda with an Application Load Balancer and Failover
- ACM Replication and Failover
- Parameter Store Replication and Failover
- ECR cross region replication and Failover
- Secrets manager cross region replication configuration
- Enable cross region replication for EFS and Failover
- EFS Cross Region Replication and Failover
- Networking Failover
- S3 Endpoint failover using MRAP
- Create cross region replication for S3
- Failover Regional API Gateway with Route 53 ARC
- Failover using multi-region global accelerator
- Failover with DRS
- Creating Disaster Recovery Mechanisms Using Amazon Route 53

Exemplos relacionados:

- Recuperação de desastres na AWS

- Recuperação elástica de desastres na AWS

REL11-BP03 Automatizar a reparação em todas as camadas

Após a detecção de uma falha, use recursos automatizados para executar ações de correção. As degradações podem ser corrigidas automaticamente por meio de mecanismos internos de serviço ou exigir que os recursos sejam reiniciados ou removidos por meio de ações de remediação.

Para aplicações autogerenciadas e reparação entre regiões, os projetos de recuperação e os processos de recuperação automatizados podem ser extraídos de práticas recomendadas existentes.

A capacidade de reiniciar ou remover um recurso é uma ferramenta importante para corrigir falhas. Uma prática recomendada é deixar os serviços sem estado sempre que possível. Isso evita a perda de dados ou a disponibilidade na reinicialização do recurso. Na nuvem, você pode (e geralmente deve) substituir todo o recurso (por exemplo, uma instância de computação ou função sem servidor) como parte da reinicialização. A reinicialização em si é uma maneira simples e confiável de se recuperar de falhas. Muitos tipos diferentes de falhas ocorrem em cargas de trabalho. As falhas podem ocorrer em hardware, software, comunicações e operações.

Reiniciar ou tentar novamente também se aplica às solicitações de rede. Aplique a mesma abordagem de recuperação tanto a um tempo limite de rede quanto a uma falha de dependência em que a dependência retorna um erro. Ambos os eventos têm um efeito similar sobre o sistema, assim, em vez de tentar tornar qualquer um dos eventos um caso especial, aplique uma estratégia similar de nova tentativa limitada com recuo e variação exponenciais. A capacidade de reiniciar é um mecanismo de recuperação presente na computação orientada para a recuperação e arquiteturas de cluster de alta disponibilidade.

Resultado desejado: Ações automatizadas são executadas para corrigir a detecção de uma falha.

Antipadrões comuns:

- Provisionamento de recursos sem dimensionamento automático.
- Implantação de aplicações em instâncias ou contêineres individualmente.
- Implantação de aplicações que não podem ser implantadas em vários locais sem usar a recuperação automática.
- Reparação manual de aplicações que não são reparadas por meio do ajuste de escala automático e recuperação automática.

- Sem automação para failover nos bancos de dados.
- Não há métodos automatizados para redirecionar o tráfego para novos endpoints.
- Sem replicação de armazenamento.

Benefícios de estabelecer esta prática recomendada: A reparação automatizada pode reduzir seu tempo médio de recuperação e melhorar sua disponibilidade.

Nível de risco exposto se esta prática recomendada não for estabelecida: alto

Orientação para implementação

Os designs para Amazon EKS ou outros serviços do Kubernetes devem incluir conjuntos mínimos e máximos de réplicas ou com estado e tamanho mínimo do cluster e do grupo de nós. Esses mecanismos fornecem uma quantidade mínima de recursos de processamento continuamente disponíveis e, ao mesmo tempo, remediam automaticamente quaisquer falhas usando o ambiente de gerenciamento do Kubernetes.

Os padrões de design que são acessados por meio de um balanceador de carga usando clusters de computação devem utilizar grupos Auto Scaling. O Elastic Load Balancing (ELB) distribui automaticamente o tráfego de entrada da aplicação entre vários destinos e dispositivos virtuais em uma ou mais Zonas de Disponibilidade (AZs).

Designs baseados em computação em cluster que não usam balanceamento de carga devem ter seu tamanho projetado para a perda de pelo menos um nó. Isso permitirá que o serviço se mantenha funcionando em uma capacidade potencialmente reduzida enquanto recupera um novo nó. Entre os exemplos de serviço estão Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK e Amazon OpenSearch Service. Muitos desses serviços podem ser projetados com recursos adicionais de recuperação automática. Algumas tecnologias de cluster devem gerar um alerta sobre a perda de um nó, acionando um fluxo de trabalho automático ou manual para recriar um novo nó. Esse fluxo de trabalho pode ser automatizado usando o AWS Systems Manager para corrigir problemas rapidamente.

É possível usar o Amazon EventBridge para monitorar e filtrar eventos, como alarmes do CloudWatch ou alterações no estado de outros serviços da AWS. Com base nas informações do evento, ele pode então invocar AWS Lambda, Systems Manager Automation ou outros destinos para executar uma lógica de remediação personalizada na workload. O Amazon EC2 Auto Scaling pode ser configurado para verificar a integridade da instância EC2. Se a instância estiver em qualquer

estado que não seja em execução, ou se o status do sistema for prejudicado, o Amazon EC2 Auto Scaling considerará que essa instância não é íntegra e executará uma instância de substituição. Para substituições em grande escala (como a perda de uma Zona de Disponibilidade inteira), a estabilidade estática é preferida para alta disponibilidade.

Etapas da implementação

- Use grupos do Auto Scaling para implantar camadas em uma workload. O Auto Scaling pode executar a autorreparação em aplicações sem estado e adicionar e remover capacidade.
- Para instâncias computacionais mencionadas anteriormente, use balanceamento de carga e escolha o tipo apropriado de balanceador de carga.
- Considere a recuperação para Amazon RDS. Com instâncias em espera, configure para failover automático para a instância em espera. Para a réplica de leitura do Amazon RDS, é necessário um fluxo de trabalho automatizado para tornar uma réplica de leitura primária.
- Implemente recuperação automática em instâncias do EC2 que têm aplicações implantadas que não podem estar em vários locais e podem tolerar a reinicialização em caso de falhas. É possível usar a recuperação automática para substituir o hardware com falha e reiniciar a instância quando a aplicação não puder ser implantada em vários locais. Os metadados da instância e os endereços IP associados são mantidos, bem como os volumes do EBS e pontos de montagem no Amazon Elastic File System ou sistemas de arquivos para Lustre e Windows. Com o uso do AWS OpsWorks, é possível configurar a autorreparação das instâncias do EC2 no nível da camada.
- Implemente a recuperação automatizada usando o AWS Step Functions e o AWS Lambda quando não for possível usar o ajuste de escala automático ou a recuperação automática, ou quando a recuperação automática falhar. Quando não for possível usar o ajuste de escala automático e a recuperação automática ou quando a recuperação automática falhar, você poderá automatizar a reparação usando o AWS Step Functions e o AWS Lambda.
- O Amazon EventBridge pode ser usado para monitorar e filtrar eventos como alarmes do CloudWatch ou mudanças de estado em outros serviços da AWS. Com base nas informações do evento, ele pode invocar o AWS Lambda (ou outros destinos) para executar a lógica de correção personalizada na workload.

Recursos

Práticas recomendadas relacionadas:

- Definição de disponibilidade

- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas

Documentos relacionados:

- Como funciona o AWS Auto Scaling
- Recuperação automática do Amazon EC2
- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- O que é o Amazon FSx for Lustre?
- O que é o Amazon FSx for Windows File Server?
- AWS OpsWorks: como usar a correção automática para substituir instâncias com falha
- O que é o AWS Step Functions?
- O que é o AWS Lambda?
- O que é o Amazon EventBridge?
- Uso dos alarmes do Amazon CloudWatch
- Failover do Amazon RDS
- SSM - Automação do Systems Manager
- Práticas recomendadas de arquitetura resiliente

Vídeos relacionados:

- Provisionar e escalar automaticamente o OpenSearch Service
- Failover automático do Amazon RDS

Exemplos relacionados:

- Workshop sobre Auto Scaling
- Workshop de failover do Amazon RDS

Ferramentas relacionadas:

- CloudWatch

- CloudWatch X-Ray

REL11-BP04 Confiar no plano de dados e não no ambiente de gerenciamento durante a recuperação

Os ambientes de gerenciamento fornecem as APIs administrativas usadas para criar, ler e descrever, atualizar, excluir e listar recursos (CRUDL), enquanto os planos de dados lidam com o tráfego diário de serviços. Ao implementar respostas de recuperação ou mitigação a eventos potencialmente impactantes na resiliência, concentre-se em usar um número mínimo de operações do ambiente de gerenciamento para recuperar, redimensionar, restaurar, reparar ou realizar o failover do serviço. A ação do plano de dados deve substituir qualquer atividade durante esses eventos de degradação.

Por exemplo, estas são ações do ambiente de gerenciamento: iniciar uma nova instância de computação, criar armazenamento em bloco e descrever serviços de fila. Quando você executa instâncias de computação, o ambiente de gerenciamento precisa realizar várias tarefas, como encontrar um host físico com capacidade, alocar interfaces de rede, preparar volumes de armazenamento em blocos locais, gerar credenciais e adicionar regras de segurança. Os ambientes de gerenciamento tendem a ser uma orquestração complicada.

Resultado desejado: Quando um recurso entra em um estado comprometido, o sistema é capaz de se recuperar automática ou manualmente, transferindo o tráfego de recursos danificados para recursos saudáveis.

Antipadrões comuns:

- Dependência da alteração dos registros DNS para redirecionar o tráfego.
- Dependência das operações de escalonamento do ambiente de gerenciamento para substituir componentes danificados devido a recursos insuficientemente provisionados.
- Dependência de ações de ambiente de gerenciamento abrangentes, com vários serviços e várias APIs para remediar qualquer categoria de deficiência.

Benefícios de estabelecer esta prática recomendada: O aumento da taxa de sucesso da remediação automatizada pode reduzir seu tempo médio de recuperação e melhorar a disponibilidade da workload.

Nível de risco exposto se esta prática recomendada não for estabelecida: médio: para certos tipos de degradação do serviço, os ambientes de gerenciamento são afetados. As dependências

do uso extensivo do ambiente de gerenciamento para remediação podem aumentar o tempo de recuperação (RTO) e o tempo médio de recuperação (MTTR).

Orientação para implementação

Para limitar as ações do plano de dados, avalie cada serviço quanto às ações necessárias para restaurar o serviço.

Use o Amazon Route 53 Application Recovery Controller para mudar o tráfego de DNS. Esses recursos monitoram continuamente a capacidade da aplicação de se recuperar de falhas, permitindo que você controle a recuperação da aplicação em várias Regiões da AWS, Zonas de Disponibilidade e ambientes on-premises.

As políticas de roteamento do Route 53 usam o ambiente de gerenciamento. Portanto, não confie nele para recuperação. Os planos de dados do Route 53 respondem as consultas ao DNS, além de realizarem e avaliarem verificações de integridade. Eles são distribuídos globalmente e projetados para um Acordo de Serviço (SLA) de 100% de disponibilidade.

As APIs e consoles de gerenciamento do Route 53 usados para criar, atualizar e excluir recursos do Route 53 são executados em ambientes de gerenciamento projetados para priorizar a consistência e a durabilidade necessária para gerenciar o DNS. Para que isso aconteça, os ambientes de gerenciamento estão localizados em uma única região: Leste dos EUA (Norte da Virgínia). Embora ambos os sistemas sejam construídos para serem muito confiáveis, os ambientes de gerenciamento não estão incluídos no SLA. Pode ser que ocorram raros eventos onde o design resiliente do plano de dados permita que ele mantenha a disponibilidade, enquanto os ambientes de gerenciamento não. Para mecanismos de recuperação de desastres e failover, use funções de plano de dados para fornecer a melhor confiabilidade possível.

Para o Amazon EC2, use designs de estabilidade estática para limitar as ações do ambiente de gerenciamento. As ações do ambiente de gerenciamento incluem a ampliação de recursos individualmente ou usando grupos do Auto Scaling (ASG). Para obter os níveis mais altos de resiliência, provisione capacidade suficiente no cluster usado para failover. Se essa capacidade precisar ser limitada, defina valores no sistema geral completo para definir com segurança o tráfego total que atinge o conjunto limitado de recursos.

Para serviços como Amazon DynamoDB, Amazon API Gateway, balanceadores de carga e AWS Lambda sem servidor, o uso aproveita o plano de dados. No entanto, criar novas funções, balanceadores de carga, gateways de API ou tabelas de DynamoDB é uma ação do ambiente de gerenciamento e deve ser concluída antes da degradação, como preparação para um evento e

ensaio das ações de failover. Para o Amazon RDS, as ações do plano de dados permitem o acesso aos dados.

Para obter mais informações sobre planos de dados, ambientes de gerenciamento e como a AWS cria serviços para atender metas de alta disponibilidade, consulte Estabilidade estática usando Zonas de disponibilidade.

Entenda quais operações estão no plano de dados e quais estão no ambiente de gerenciamento.

Etapas da implementação

Para cada workload que precisa ser restaurada após um evento de degradação, avalie o runbook de failover, o projeto de alta disponibilidade, o projeto de recuperação automática ou o plano de restauração de recursos de HA. Identifique cada ação que pode ser considerada uma ação do ambiente de gerenciamento.

Considere alterar a ação de gerenciamento para uma ação do plano de dados:

- Auto Scaling (ambiente de gerenciamento) em comparação com recursos do Amazon EC2 pré-escalados (plano de dados)
- Migre para Lambda e seus métodos de escalabilidade (plano de dados) ou Amazon EC2 e ASG (ambiente de gerenciamento)
- Avalie qualquer projeto usando o Kubernetes e a natureza das ações do ambiente de gerenciamento. Adicionar pods é uma ação do plano de dados no Kubernetes. As ações devem se limitar à adição de pods e não adição de nós. O uso de nós com provisionamento excessivo é o método preferido para limitar as ações do ambiente de gerenciamento

Considere abordagens alternativas que permitam que as ações do plano de dados afetem a mesma remediação.

- Alteração de registro Route 53 (ambiente de gerenciamento) ou Route 53 ARC (plano de dados)
- Verificações de integridade do Route 53 para atualizações mais automatizadas

Se o serviço for essencial, considere alguns serviços em uma região secundária para permitir mais ações no ambiente de gerenciamento e no plano de dados em uma região não afetada.

- Amazon EC2 Auto Scaling ou Amazon EKS em uma região primária em comparação com Amazon EC2 Auto Scaling ou Amazon EKS em uma região secundária e roteando o tráfego para a região secundária (ação do ambiente de gerenciamento).

- Faça uma réplica de leitura na primária secundária ou tente a mesma ação na região primária (ação do ambiente de gerenciamento).

Recursos

Práticas recomendadas relacionadas:

- Definição de disponibilidade
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudar na automação da sua tolerância a falhas
- AWS Marketplace: produtos que podem ser usados para tolerância a falhas
- Amazon Builders' Library: evite a sobrecarga em sistemas distribuídos colocando o menor serviço no controle
- API do Amazon DynamoDB (ambiente de gerenciamento e plano de dados)
- Execuções do AWS Lambda (divididas entre o ambiente de gerenciamento e o plano de dados)
- Plano de dados do AWS Elemental MediaStore
- Criação de aplicações altamente resilientes usando o Amazon Route 53 Application Recovery Controller, parte 1: pilha de região única
- Criação de aplicações altamente resilientes usando o Amazon Route 53 Application Recovery Controller, parte 2: pilha multirregional
- Criação de mecanismos de recuperação de desastres usando o Amazon Route 53
- O que é o Route 53 Application Recovery Controller?
- Ambiente de gerenciamento e plano de dados do Kubernetes

Vídeos relacionados:

- Back to Basics - Using Static Stability (De volta ao básico: uso da estabilidade estática)
- Building resilient multi-site workloads using AWS global services (Criação de workloads resilientes em vários sites usando serviços globais da AWS)

Exemplos relacionados:

- Introdução ao Amazon Route 53 Application Recovery Controller
- Amazon Builders' Library: evite a sobrecarga em sistemas distribuídos colocando o menor serviço no controle
- Criação de aplicações altamente resilientes usando o Amazon Route 53 Application Recovery Controller, parte 1: pilha de região única
- Criação de aplicações altamente resilientes usando o Amazon Route 53 Application Recovery Controller, parte 2: pilha multirregional
- Estabilidade estática usando Zonas de disponibilidade

Ferramentas relacionadas:

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 Usar estabilidade estática para evitar o comportamento bimodal

As workloads devem ser estaticamente estáveis e operar somente em um único modo normal. O comportamento bimodal ocorre quando a workload exibe um comportamento diferente nos modos normal e de falha.

Por exemplo, você pode tentar se recuperar de uma falha na zona de disponibilidade iniciando novas instâncias em uma zona de disponibilidade diferente. Isso pode resultar em uma resposta bimodal durante um modo de falha. Em vez disso, você deve criar workloads que sejam estaticamente estáveis e que operem em apenas um modo. Neste exemplo, essas instâncias deveriam ter sido provisionadas na segunda zona de disponibilidade antes da falha. Esse design de estabilidade estática verifica se a workload opera somente em um único modo.

Resultado desejado: As workloads não apresentam comportamento bimodal durante os modos normal e de falha.

Antipadrões comuns:

- Supor que os recursos sempre possam ser provisionados, independentemente do escopo da falha.
- Tentar adquirir recursos dinamicamente durante uma falha.
- Não provisionar recursos adequados entre zonas ou regiões até que ocorra uma falha.

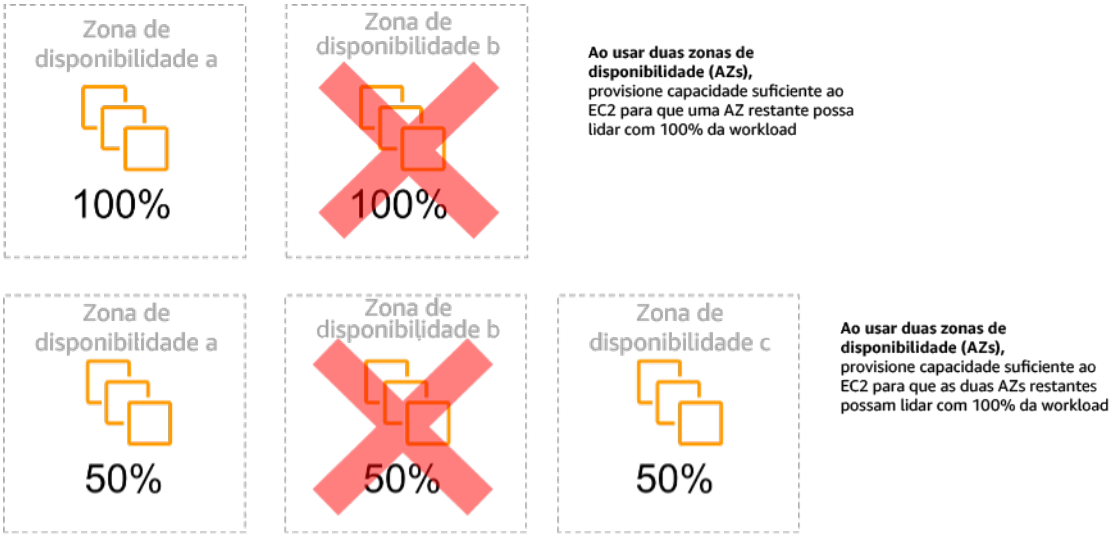
- Pensar em projetos estáticos estáveis somente para recursos computacionais.

Benefícios de estabelecer esta prática recomendada: As workloads executadas com projetos estaticamente estáveis podem ter resultados previsíveis durante eventos normais e de falha.

Nível de risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

O comportamento bimodal ocorre quando a workload apresenta um comportamento diferente nos modos normal e de falha (por exemplo, depender da inicialização de novas instâncias se uma zona de disponibilidade falhar). Um exemplo de comportamento bimodal é quando projetos estáveis do Amazon EC2 provisionam instâncias suficientes em cada zona de disponibilidade para lidar com a carga da workload se uma AZ for removida. A integridade do Elastic Load Balancing ou do Amazon Route 53 conferiria a possibilidade de isolar a carga das instâncias prejudicadas. Depois que o tráfego for deslocado, use o AWS Auto Scaling para substituir de forma assíncrona instâncias da zona com falha e executá-las nas zonas íntegras. A estabilidade estática para implantação de computação (como instâncias ou contêineres do EC2) resulta na mais alta confiabilidade.



Estabilidade estática de instâncias do EC2 em várias zonas de disponibilidade

Isso deve ser comparado ao custo desse modelo e ao valor comercial de manter a workload em todos os casos de resiliência. É mais barato provisionar menos capacidade computacional e depender da inicialização de novas instâncias em caso de falha. No entanto, para falhas em grande escala (como um dano regional ou da zona de disponibilidade), essa abordagem é menos eficaz porque depende tanto de um plano operacional quanto de recursos suficientes disponíveis nas zonas ou regiões não afetadas.

A solução também deve comparar a confiabilidade com os custos necessários para a workload. As arquiteturas de estabilidade estática se aplicam a uma variedade de arquiteturas, incluindo instâncias de computação espalhadas por zonas de disponibilidade, projetos de réplicas de leitura de banco de dados, projetos de cluster do Kubernetes (Amazon EKS) e arquiteturas de failover de várias regiões.

Também é possível implementar um design mais estável estaticamente usando mais recursos em cada zona. Ao adicionar mais zonas, você reduz a quantidade de computação adicional necessária para a estabilidade estática.

Um exemplo de comportamento bimodal seria um tempo limite de rede que poderia fazer com que um sistema tentasse atualizar seu próprio estado de configuração por completo. Isso adicionaria uma carga inesperada a outro componente e poderia fazê-lo falhar, resultando em outras consequências inesperadas. Esse ciclo de feedback negativo afeta a disponibilidade da workload. Em vez disso, você pode criar sistemas estaticamente estáveis e operar em apenas um modo. Um design estático estável faria um trabalho constante e sempre atualizaria o estado da configuração em um ritmo fixo. Quando uma chamada falha, a workload usa o valor previamente armazenado em cache e inicia um alarme.

Outro exemplo de comportamento bimodal é permitir que os clientes ignorem o cache da workload em caso de falhas. Essa pode parecer uma solução que acomoda as necessidades do cliente, mas pode alterar significativamente as demandas da workload e provavelmente resultar em falhas.

Avalie workloads importantes para determinar quais workloads exigem esse tipo de projeto de resiliência. Para as que são consideradas críticas, cada componente da aplicação deve ser revisado. Exemplos de tipos de serviço que exigem avaliações de estabilidade estática são:

- Computação: Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- Bancos de dados: Amazon Redshift, Amazon RDS, Amazon Aurora
- Armazenamento: Amazon S3 (Zona única), Amazon EFS (montagens), Amazon FSx (montagens)
- Balanceadores de carga: Em determinados projetos

Etapas da implementação

- Crie sistemas que sejam estaticamente estáveis e que operem em apenas um modo. Nesse caso, provisione instâncias suficientes em cada zona de disponibilidade ou região para lidar com a capacidade da workload se uma zona de disponibilidade ou região for removida. Uma variedade de serviços pode ser usada para roteamento a recursos íntegros, como:
  - Roteamento de DNS entre regiões

  - Roteamento de várias regiões do Amazon S3
  - AWS Global Accelerator
  - Amazon Route 53 Application Recovery Controller
- Configure réplicas de leitura do banco de dados para contabilizar a perda de uma única instância principal ou de uma réplica de leitura. Se o tráfego estiver sendo servido por réplicas de leitura, a quantidade em cada zona de disponibilidade e cada região deve ser igual à necessidade geral em caso de falha na zona ou região.
- Configure dados importantes no armazenamento do Amazon S3, projetado para ser estaticamente estável para dados armazenados em caso de falha na zona de disponibilidade. Se Se a classe de armazenamento Amazon S3 One Zone-IA for usada, ela não deverá ser considerada estaticamente estável, pois a perda dessa zona minimiza o acesso a esses dados armazenados.
- Às vezes, os balanceadores de carga são configurados incorretamente ou intencionalmente para atender a uma zona de disponibilidade específica. Nesse caso, o design estaticamente estável pode envolver a distribuição de uma workload entre várias zonas de disponibilidade em um design mais complexo. O design original pode ser usado para reduzir o tráfego entre zonas por motivos de segurança, latência ou custo.

Recursos

Práticas recomendadas relacionadas ao Well-Architected:

- Definição de disponibilidade
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP04 Confiar no plano de dados e não no ambiente de gerenciamento durante a recuperação

Documentos relacionados:

- Minimizar dependências em um plano de recuperação de desastres
- A Amazon Builders' Library: estabilidade estática usando zonas de disponibilidade
- Limites de isolamento de falhas
- Estabilidade estática usando Zonas de disponibilidade
- Multi-AZ do Amazon RDS
- Minimizar dependências em um plano de recuperação de desastres

- Roteamento de DNS entre regiões
- Roteamento de várias regiões do Amazon S3
- AWS Global Accelerator
- Route 53 ARC
- Zona única do Amazon S3
- Balanceamento de carga entre zonas

Vídeos relacionados:

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

Exemplos relacionados:

- A Amazon Builders’ Library: estabilidade estática usando zonas de disponibilidade

REL11-BP06 Enviar notificações quando os eventos afetarem a disponibilidade

As notificações são enviadas após a detecção de limites violados, mesmo que o evento causado pelo problema tenha sido resolvido automaticamente.

A correção automatizada permite que a workload seja confiável. No entanto, ele também pode obscurecer problemas subjacentes que precisam ser resolvidos. Implemente eventos e monitoramento apropriados para que você possa detectar padrões de problemas, incluindo aqueles abordados pela correção automática, para que você possa resolver problemas de causa-raiz.

Os sistemas resilientes são projetados para que os eventos de degradação sejam comunicados imediatamente às equipes apropriadas. Essas notificações devem ser enviadas por meio de um ou vários canais de comunicação.

Resultado desejado: Os alertas são enviados imediatamente às equipes de operações quando os limites são violados. Esses alertas podem incluir taxas de erro, latência ou outras métricas importantes de indicadores-chave de performance (KPI), permitindo que esses problemas sejam resolvidos o mais rápido possível e o impacto do usuário seja evitado ou minimizado.

Antipadrões comuns:

- Enviar muitos alarmes.
- Enviar alarmes que não resultam em ações concretas.
- Definir limites de alarme muito altos (supersensíveis) ou muito baixos (subsensíveis).
- Não enviar alarmes para dependências externas.
- Não considerar falhas cinzentas ao projetar monitoramento e alarmes.
- Executar a automação da correção, mas sem notificar a equipe apropriada de que a correção era necessária.

Benefícios de estabelecer esta prática recomendada: As notificações de recuperação alertam as equipes operacionais e empresariais sobre as degradações do serviço, para que possam reagir imediatamente a fim de minimizar o tempo médio de detecção (MTTD) e o tempo médio de reparo (MTTR). As notificações de eventos de recuperação também garantem que você não ignore problemas que ocorrem com pouca frequência.

Nível de risco exposto se esta prática recomendada não for estabelecida: Médio. A falha na implementação de mecanismos adequados de monitoramento e notificação de eventos pode resultar em falha na detecção de padrões de problemas, incluindo aqueles resolvidos pela recuperação automática. Uma equipe só será informada da degradação do sistema quando os usuários entrarem em contato com o atendimento ao cliente ou por acaso.

Orientações para a implementação

Ao definir uma estratégia de monitoramento, um alarme acionado é um evento comum. Esse evento provavelmente conteria um identificador para o alarme, o estado do alarme (como `EM ALARME` ou `OK`) e detalhes do que o desencadeou. Em muitos casos, deve-se detectar um evento de alarme e enviar uma notificação por e-mail. Este é um exemplo de uma ação em um alarme. A notificação do alarme é fundamental para a observabilidade, pois informa às pessoas certas que há um problema. No entanto, quando a ação sobre eventos amadurece em sua solução de observabilidade, ela pode corrigir automaticamente o problema sem a necessidade de intervenção humana.

Depois que os alarmes de monitoramento de KPI forem estabelecidos, os alertas deverão ser enviados às equipes apropriadas quando os limites forem excedidos. Esses alertas também podem ser usados para acionar processos automatizados que tentarão remediar a degradação.

Para um monitoramento de limite mais complexo, considere alarmes compostos. Eles usam vários alarmes de monitoramento de KPI para criar um alerta com base na lógica operacional de negócios. Os alarmes do CloudWatch podem ser configurados para enviar e-mails ou registrar incidentes em

sistemas de rastreamento de incidentes de terceiros usando a integração com o Amazon SNS ou Amazon EventBridge.

Etapas para a implementação

Crie vários tipos de alarme com base na forma como as workloads são monitoradas, por exemplo:

- Os alarmes de aplicações são usados para detectar quando alguma parte da workload não está funcionando adequadamente.
- Alarmes de infraestrutura indicam quando escalar os recursos. Os alarmes podem ser exibidos visualmente em painéis, enviar alertas pelo Amazon SNS ou por e-mail e trabalhar com o Auto Scaling para aumentar ou reduzir a escala dos recursos da workload horizontalmente.
- Simples alarmes estáticos podem ser criados para monitorar quando uma métrica ultrapassa um limite estático durante um número específico de períodos de avaliação.
- Os alarmes compostos podem representar alarmes complexos de várias origens.
- Depois que o alarme for criado, crie eventos de notificação apropriados. Você pode invocar diretamente uma API do Amazon SNS para enviar notificações e vincular qualquer automação para correção ou comunicação.
- integre o Amazon Health Aware para permitir a visibilidade de monitoramento de recursos da AWS que possam ter degradações. Para workloads essenciais aos negócios, essa solução fornece acesso a alertas proativos e em tempo real para serviços da AWS.

Recursos

Práticas recomendadas relacionadas ao Well-Architected:

- Definição de disponibilidade

Documentos relacionados:

- Criar um alarme do CloudWatch com base em um limite estático
- O que é o Amazon EventBridge?
- O que é o Amazon Simple Notification Service?
- Publicar métricas personalizadas
- Uso dos alarmes do Amazon CloudWatch

- Amazon Health Aware (AHA)
- Setup CloudWatch Composite alarms
- What's new in AWS Observability at re:Invent 2022

Ferramentas relacionadas:

- CloudWatch
- CloudWatch X-Ray

REL11-BP07 Arquitetar o produto para cumprir as metas de disponibilidade e os acordos de nível de serviço (SLAs) de tempo de atividade

Arquitete o produto para cumprir as metas de disponibilidade e os acordos de nível de serviço (SLAs) de tempo de atividade. Se você publicar ou concordar de forma privada com as metas de disponibilidade ou SLAs de tempo de atividade, verifique se sua arquitetura e seus processos operacionais foram projetados para comportá-los.

Resultado desejado: cada aplicação tem uma meta definida com relação à disponibilidade e SLA para métricas de desempenho, o que pode ser monitorado e mantido para cumprir os resultados empresariais.

Antipadrões comuns:

- Planejar e implantar workloads sem definir SLAs.
- As métricas de SLA são definidas como altas sem justificativa ou requisitos empresariais.
- Definir SLAs sem considerar as dependências e o SLA subjacente.
- Os designs da aplicação são criados sem considerar o modelo de responsabilidade compartilhada para resiliência.

Benefícios do estabelecimento dessa prática recomendada: projetar aplicações com base nas principais metas de resiliência ajuda a cumprir os objetivos empresariais e atender às expectativas dos clientes. Esses objetivos ajudam a orientar o processo de design da aplicação que avalia diferentes tecnologias e considera as vantagens e desvantagens.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientação de implementação

Os designs da aplicação precisam levar em conta um conjunto de requisitos diversos que são derivados dos objetivos empresariais, operacionais e financeiros. Nos requisitos operacionais, as workloads precisam ter metas de métricas de resiliência específicas para que possam ser monitorados e comportados adequadamente. As métricas de resiliência não devem ser definidas nem derivadas depois de implantar a workload. Elas devem ser definidas durante a fase de design e ajudar a orientar as diversas decisões e concessões.

- Cada workload deve ter seu próprio conjunto de métricas de resiliência. Essas métricas podem ser diferentes de outras aplicações empresariais.
- Reduzir as dependências pode ter um impacto positivo na disponibilidade. Cada workload deve considerar suas dependências e seus SLAs. Em geral, escolha dependências com metas de disponibilidade iguais ou maiores que as metas da workload.
- Considere designs com acoplamento fraco para que a workload possa operar corretamente apesar do comprometimento da dependência, quando possível.
- Reduza as dependências do ambiente de gerenciamento, especialmente durante uma recuperação ou degradação. Avalie os designs estaticamente estáveis com relação às workloads essenciais à missão. Use a economia de recursos para aumentar a disponibilidade dessas dependências em uma workload.
- A capacidade de observação e a instrumentalização são críticas para cumprir os SLAs reduzindo o tempo médio de detecção (MTTD) e o tempo médio de reparo (MTTR).
- Falha menos frequente (MTBF mais longo), tempo de detecção de falhas mais curto (MTTD mais curto) e tempo de reparo mais curto (MTTR mais curto) são os três fatores usados para melhorar a disponibilidade em sistemas distribuídos.
- Estabelecer e cumprir métricas de resiliência para uma workload é fundamental para qualquer design eficaz. Esses designs devem levar em consideração as vantagens e desvantagens da complexidade de design, as dependências do serviço, o desempenho, a escalabilidade e os custos.

Etapas da implementação

- Analise e documente o design da workload considerando as seguintes questões:
  - Onde são usados ambientes de gerenciamento na workload?
  - Como a workload implementa tolerância a falhas?

  - Quais são os padrões de design para componentes de escalabilidade, escalabilidade automática, redundância e alta disponibilidade?
  - Quais são os requisitos para disponibilidade e consistência de dados?
  - Há considerações quanto à economia de recursos ou estabilidade estática de recursos?
  - Quais são as dependências do serviço?
- Defina métricas de SLA com base na arquitetura da workload enquanto trabalha com as partes interessadas. Considere os SLAs de todas as dependências usadas pela workload.
- Quando a meta de SLA for definida, otimize a arquitetura para cumprir o SLA.
- Quando for definido o design que cumprirá o SLA, implemente mudanças operacionais, automação do processo e runbooks que também terão como foco uma redução de MTTD e MTTR.
- Depois da implantação, monitore e informe sobre o SLA.

Recursos

Práticas recomendadas relacionadas:

- REL03-BP01 Escolher como segmentar a workload
- REL10-BP01 Implantar a workload em vários locais
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- REL11-BP03 Automatizar a reparação em todas as camadas
- REL12-BP05 Testar a resiliência por meio da engenharia do caos
- REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados
- Compreensão de integridade da workload

Documentos relacionados:

- Availability with redundancy (Disponibilidade com redundância)
- Pilar Confiabilidade: disponibilidade
- Measuring availability (Medição da disponibilidade)
- AWS Fault Isolation Boundaries (Limites de isolamento de falhas da AWS)
- Modelo de responsabilidade compartilhada para resiliência
- estabilidade estática usando Zonas de disponibilidade
- AWS Service Level Agreements (SLAs) (Acordos de Nível de Serviço (SLAs) da AWS)

- Guidance for Cell-based Architecture on AWS(Orientações para arquitetura baseada em células com a AWS)
- Infraestrutura da AWS
- Advanced Multi-AZ Resiliance Patterns whitepaper (Whitepaper de padrões de resiliência Multi-AZ avançados)

Serviços relacionados:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

Testar a confiabilidade

Depois de projetar sua carga de trabalho para resiliência à pressão da produção, o teste é a única maneira de garantir que ela opere conforme projetado e com a resiliência esperada.

Teste para validar se sua carga de trabalho atende aos requisitos funcionais e não funcionais, pois erros ou gargalos de performance podem afetar a confiabilidade de sua carga de trabalho. Teste a resiliência de sua carga de trabalho para ajudá-lo a encontrar erros latentes que apenas aparecem na produção. Exercite esses testes regularmente.

Práticas recomendadas

- REL12-BP01 Usar playbooks para investigar falhas
- REL12-BP02 Realizar análise pós-incidente
- REL12-BP03 Testar os requisitos funcionais
- REL12-BP04 Testar os requisitos de escalabilidade e performance
- REL12-BP05 Testar a resiliência por meio da engenharia do caos
- REL12-BP06 Realizar dias de jogo regularmente

REL12-BP01 Usar playbooks para investigar falhas

Faça a documentação do processo de investigação em playbooks para permitir respostas consistentes e rápidas em cenários de falha. Os playbooks são as etapas predefinidas executadas para identificar os fatores que contribuem para um cenário de falha. Os resultados de qualquer etapa

do processo são usados para determinar as próximas etapas a serem seguidas até que o problema seja identificado ou encaminhado.

O playbook é um planejamento proativo que você deve fazer para poder executar ações reativas com eficácia. Quando cenários de falha não cobertos pelo playbook forem encontrados na produção, resolva primeiro o problema (apague o fogo). Em seguida, volte e veja as etapas que você seguiu para resolver o problema e use-as para adicionar uma nova entrada no playbook.

Observe que playbooks são usados em resposta a incidentes específicos, enquanto runbooks são usados para alcançar resultados específicos. Muitas vezes, runbooks são usados para atividades de rotina e os playbooks são usados para responder a eventos que não são rotineiros.

Antipadrões comuns:

- Planejar a implantação de uma carga de trabalho sem conhecer os processos para diagnosticar problemas ou responder a incidentes.
- Decisões não planejadas de quais sistemas coletar logs e métricas ao investigar um evento.
- Não armazenar as métricas e os eventos pelo tempo suficiente para recuperar os dados.

Benefícios do estabelecimento desta prática recomendada: Capturar playbooks garante que os processos possam ser seguidos de forma consistente. A codificação dos seus playbooks limita a introdução de erros por atividades manuais. A automação dos playbooks reduz o tempo de resposta a um evento ao eliminar a necessidade de intervenção de membros da equipe ou ao fornecer a eles informações adicionais desde o início da intervenção.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

- Use playbooks para identificar problemas. Os manuais são processos documentados para investigar problemas. Faça a documentação dos processos em playbooks para permitir respostas consistentes e rápidas em cenários de falha. Os playbooks devem incluir as informações e as diretrizes necessárias para que uma pessoa com as devidas qualificações colete as informações aplicáveis, identifique possíveis fontes de falha, isole as falhas e determine os fatores contribuintes (realize uma análise pós-incidente).
  - Implemente playbooks como código. Execute suas operações como código ao criar scripts de seus playbooks para garantir a consistência e reduzir os erros causados por processos manuais. Os playbooks podem ser compostos por vários scripts representando as diferentes

etapas que podem ser necessárias para identificar os fatores que contribuem para um problema. As atividades do runbook podem ser acionadas ou executadas como parte das atividades do playbook, ou podem solicitar a execução de um playbook em resposta a eventos identificados.

- Automatizar playbooks operacionais com o AWS Systems Manager
- AWS Systems Manager Run Command
- AWS Systems Manager Automation
- O que é o AWS Lambda?
- O que é o Amazon EventBridge?
- Usar alarmes do Amazon CloudWatch

Recursos

Documentos relacionados:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- Automatizar playbooks operacionais com o AWS Systems Manager
- Usar alarmes do Amazon CloudWatch
- Uso de canários (Amazon CloudWatch Synthetics)
- O que é o Amazon EventBridge?
- O que é o AWS Lambda?

Exemplos relacionados:

- Automating operations with Playbooks and Runbooks (Automatização de operações com manuais e runbooks)

REL12-BP02 Realizar análise pós-incidente

Analise os eventos que afetam o cliente e identifique os fatores contribuintes e os itens de ação preventiva. Use essas informações para desenvolver mitigações para limitar ou evitar recorrência. Desenvolva procedimentos para respostas rápidas e eficazes. Comunique os fatores contribuintes e as ações corretivas conforme apropriado, de acordo com o público-alvo. Tenha um método para comunicar essas causas a outras pessoas, conforme necessário.

Avalie por que os testes existentes não encontraram o problema. Adicione testes para esse caso se os testes ainda não existirem.

Resultado desejado: suas equipes têm uma abordagem consistente e acordada para lidar com a análise pós-incidente. Um mecanismo é o processo de correção de erros (COE). O processo de COE ajuda as equipes a identificar, compreender e abordar as causas básicas dos incidentes, ao mesmo tempo em que cria mecanismos e barreiras de proteção para limitar a probabilidade do mesmo incidente ocorrer novamente.

Antipadrões comuns:

- Encontrar fatores contribuintes, mas não continuar buscando mais profundamente outros possíveis problemas e abordagens de mitigação.
- Identificar apenas as causas de erros humanos e não oferecer nenhum treinamento ou automação que possa evitar erros humanos.
- Concentrar-se em atribuir a culpa em vez de compreender a causa raiz, criando uma cultura de medo e impedindo a comunicação aberta.
- Não compartilhar insights, o que mantém as descobertas da análise de incidentes em um pequeno grupo e impede que outras pessoas se beneficiem das lições aprendidas.
- Não ter um mecanismo para capturar conhecimento institucional e, dessa forma, perder insights valiosos por não preservar as lições aprendidas na forma de práticas recomendadas atualizadas e resultando em incidentes repetidos com a mesma causa raiz ou similar.

Benefícios do estabelecimento dessa prática recomendada: a realização de análises pós-incidentes e o compartilhamento dos resultados permitem que outras workloads atenuem o risco caso tenham implementado os mesmos fatores contribuintes, além de possibilitar que elas implementem a mitigação ou a recuperação automatizada antes que ocorra um incidente.

Nível de exposição a riscos se esta prática recomendada não for estabelecida: alto

Orientações para a implementação

Uma boa análise pós-incidente oferece oportunidades para propor soluções comuns a problemas com padrões de arquitetura usados em outros locais nos sistemas.

A base do processo da COE é documentar e resolver problemas. É recomendável definir uma forma padronizada de documentar as causas raízes essenciais e garantir que elas sejam analisadas e

abordadas. Atribua uma propriedade clara ao processo de análise pós-incidente. Designe uma equipe ou uma pessoa responsável para supervisionar as investigações e o acompanhamento de incidentes.

Incentive uma cultura que se concentre no aprendizado e na melhoria, em vez de na atribuição de culpas. Enfatize que a meta é evitar futuros incidentes, não penalizar pessoas.

Desenvolva procedimentos bem definidos para conduzir análises pós-incidentes. Esses procedimentos devem descrever as etapas a serem seguidas, as informações a serem coletadas e as principais questões a serem abordadas durante a análise. Investigue os incidentes minuciosamente, indo além das causas imediatas para identificar as causas raízes e os fatores contribuintes. Use técnicas, como os cinco porquês, para se aprofundar nos problemas subjacentes.

Mantenha um repositório das lições aprendidas com as análises dos incidentes. Esse conhecimento institucional pode servir como referência para futuros incidentes e iniciativas de prevenção. Compartilhe descobertas e insights de análises pós-incidentes e considere realizar reuniões abertas sobre a revisão pós-incidente para discutir as lições aprendidas.

Etapas da implementação

- Ao conduzir a análise pós-incidente, verifique se o processo está livre de culpabilização. Isso permite que as pessoas envolvidas no incidente sejam imparciais com as ações corretivas propostas e promovam uma autoavaliação honesta e a colaboração entre as equipes.
- Defina uma forma padronizada de documentar problemas essenciais. Um exemplo de estrutura para esse documento é o seguinte:
  - O que aconteceu?
  - Qual foi o impacto nos clientes e em sua empresa?
  - Qual foi a causa raiz?
  - Quais dados você tem para apoiar isso?
    - Por exemplo, métricas e grafos
  - Quais foram as implicações críticas nos pilares, especialmente em relação à segurança?
    - Ao arquitetar workloads, você faz concessões entre os pilares com base no contexto da sua empresa. Essas decisões de negócios podem definir suas prioridades de engenharia. Você pode reduzir custos e assim diminuir a confiabilidade em ambientes de desenvolvimento, ou otimizar a confiabilidade e aumentar os custos para soluções importantes. A segurança é sempre prioritária, porque você precisa proteger seus clientes.
  - Quais lições você aprendeu?

    - Quais ações corretivas você está tomando?
        - Itens de ação
        - Itens relacionados
- Crie procedimentos operacionais padrão bem definidos para conduzir análises pós-incidentes.
- Configure um processo padronizado de relatórios de incidentes. Documente todos os incidentes de forma abrangente, incluindo o relatório inicial do incidente, logs, comunicações e ações tomadas durante o incidente.
- Lembre-se de que um incidente não exige uma interrupção. Pode ser uma quase falha ou um sistema que, embora esteja funcionando de forma inesperada, cumpre sua função de negócios.
- Melhore continuamente o processo de análise pós-incidente com base no feedback e nas lições aprendidas.
- Capture as principais descobertas em um sistema de gerenciamento de conhecimento e considere os padrões que devem ser adicionados aos guias de desenvolvedor ou às listas de verificação de pré-implantação.

Recursos

Documentos relacionados:

- Why you should develop a correction of error (COE)

Vídeos relacionados:

- Amazon's approach to failing successfully
- AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon

REL12-BP03 Testar os requisitos funcionais

Use técnicas como testes de unidade e de integração que validam a funcionalidade necessária.

Você obtém os melhores resultados quando esses testes são executados automaticamente como parte das ações de compilação e implantação. Por exemplo, usando o AWS CodePipeline, os desenvolvedores confirmam alterações em um repositório de origem onde o CodePipeline detecta automaticamente as alterações. Essas alterações são criadas e os testes são executados. Após a conclusão dos testes, o código criado é implantado nos servidores de preparação para testes.

No servidor de preparação, o CodePipeline executa mais testes, como testes de integração ou carga. Após a conclusão bem-sucedida desses testes, o CodePipeline implanta o código testado e aprovado nas instâncias de produção.

Além disso, a experiência mostra que o teste de transações sintéticas (também conhecido como teste canário, que não deve ser confundido com as implantações canário) que pode executar e simular o comportamento do cliente está entre os processos de teste mais importantes. Execute esses testes constantemente nos endpoints da carga de trabalho de diversos locais remotos. O Amazon CloudWatch Synthetics permite criar canários para monitorar seus endpoints e APIs.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

- Teste os requisitos funcionais. Esse procedimento inclui testes de unidade e de integração que validam a funcionalidade necessária.
  - Usar o CodePipeline com o AWS CodeBuild para testar código e executar builds
  - O AWS CodePipeline adiciona compatibilidade para testes de unidade e de integração personalizada com o AWS CodeBuild
  - Entrega contínua e integração contínua
  - Uso de canários (Amazon CloudWatch Synthetics)
  - Automação de teste de software

Recursos

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar na implementação de um pipeline de integração contínua
- O AWS CodePipeline adiciona compatibilidade para testes de unidade e de integração personalizada com o AWS CodeBuild
- AWS Marketplace: produtos que podem ser usados para integração contínua
- Entrega contínua e integração contínua
- Automação de teste de software
- Usar o CodePipeline com o AWS CodeBuild para testar código e executar builds
- Uso de canários (Amazon CloudWatch Synthetics)

REL12-BP04 Testar os requisitos de escalabilidade e performance

Use técnicas como o teste de carga para validar se a workload atende aos requisitos de escalabilidade e performance.

Na nuvem, você pode criar um ambiente de teste em escala de produção sob demanda para sua carga de trabalho. Se você executar esses testes na infraestrutura reduzida, deverá escalar os resultados observados para o que você acha que acontecerá na produção. Os testes de carga e performance também podem ser feitos na produção se você tiver cuidado para não afetar os usuários reais e marcar seus dados de teste para que eles não se sintam com dados reais do usuário e estatísticas de uso corrompidas ou relatórios de produção.

Com os testes, certifique-se de que seus recursos básicos, configurações de escalabilidade, cotas de serviço e design de resiliência operem conforme o esperado sob carga.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

- Teste os requisitos de escalabilidade e performance. Execute o teste de carga para validar se a carga de trabalho atende aos requisitos de escalabilidade e performance.
  - Distributed Load Testing on AWS: simulate thousands of connected users (Teste de carga distribuída na AWS: simular milhares de usuários conectados)
  - Apache JMeter
    - Implante seu aplicativo em um ambiente idêntico ao seu ambiente de produção e execute um teste de carga.
      - Use os conceitos de infraestrutura como código para criar um ambiente que seja o mais semelhante possível ao seu ambiente de produção.

Recursos

Documentos relacionados:

- Distributed Load Testing on AWS: simulate thousands of connected users (Teste de carga distribuída na AWS: simular milhares de usuários conectados)
- Apache JMeter

REL12-BP05 Testar a resiliência por meio da engenharia do caos

Execute testes de caos regularmente em ambientes que estão em produção, ou muito próximos de entrarem em produção, para entender como seu sistema responde a condições adversas.

Resultado desejado:

A resiliência da workload é regularmente verificada por meio da aplicação de engenharia de caos na forma de testes de injeção de falha ou injeção de carga inesperada, além de testes de resiliência que validam o comportamento conhecido esperado da workload durante um evento. Combine engenharia de caos e testes de resiliência para ter confiança de que sua workload poderá sobreviver à falha de componentes e se recuperar de interferências inesperadas com pouco ou nenhum impacto.

Antipadrões comuns:

- Projetar para resiliência, mas não verificar como a workload funciona como um todo quando ocorrem falhas.
- Nunca realizar testes sob condições reais e carga esperada.
- Não tratar seus testes como código nem mantê-los ao longo do ciclo de desenvolvimento.
- Não realizar testes de caos tanto como parte do pipeline de CI/CD quanto fora das implantações.
- Negar o uso de análises pós-incidentes passadas ao determinar quais falhas usar para realizar testes.

Benefícios do estabelecimento desta prática recomendada: A injeção de falhas para verificar a resiliência de uma workload permite que você obtenha confiança de que os procedimentos de recuperação de seu design resiliente vão funcionar em caso de falha real.

Nível de risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação de implementação

A engenharia de caos proporciona à sua equipe os recursos para injetar continuamente interferências (simulações) reais de maneira controlada no provedor de serviço, na infraestrutura, na workload e no componente, com pouco ou nenhum impacto para os clientes. Permite que as equipes aprendam com as falhas e observem, mensurem e aumentem a resiliência das workloads, além de validar o acionamento de alertas e a notificação das equipes em caso de evento.

Quando realizada continuamente, a engenharia de caos pode destacar deficiências nas workloads que, se não respondidas, podem afetar negativamente a disponibilidade e a operação.

> **Note**
>
> A engenharia do caos é a disciplina de experimentar um sistema distribuído para aumentar a confiança na capacidade do sistema de resistir a condições turbulentas na produção. – Princípios da engenharia do caos

Se um sistema é capaz de suportar essas interferências, os testes de caos devem ser mantidos como testes de regressão automatizados. Dessa forma, os testes de caos devem ser realizados como parte do ciclo de vida de desenvolvimento dos sistemas (SDLC) e como parte do pipeline de CI/CD.

Para garantir que sua workload pode sobreviver à falha de componentes, injete eventos reais como parte dos testes. Por exemplo, realize testes com perda de instâncias do Amazon EC2 ou failover da instância de banco de dados primária do Amazon RDS e verifique se a workload não é afetada (ou apenas minimamente afetada). Use uma combinação de falhas de componentes para simular eventos que podem ser causados por uma interferência em uma zona de disponibilidade.

Para falhas no nível da aplicação (como travamentos), você pode começar com fatores de estresse, como exaustão de memória e CPU.

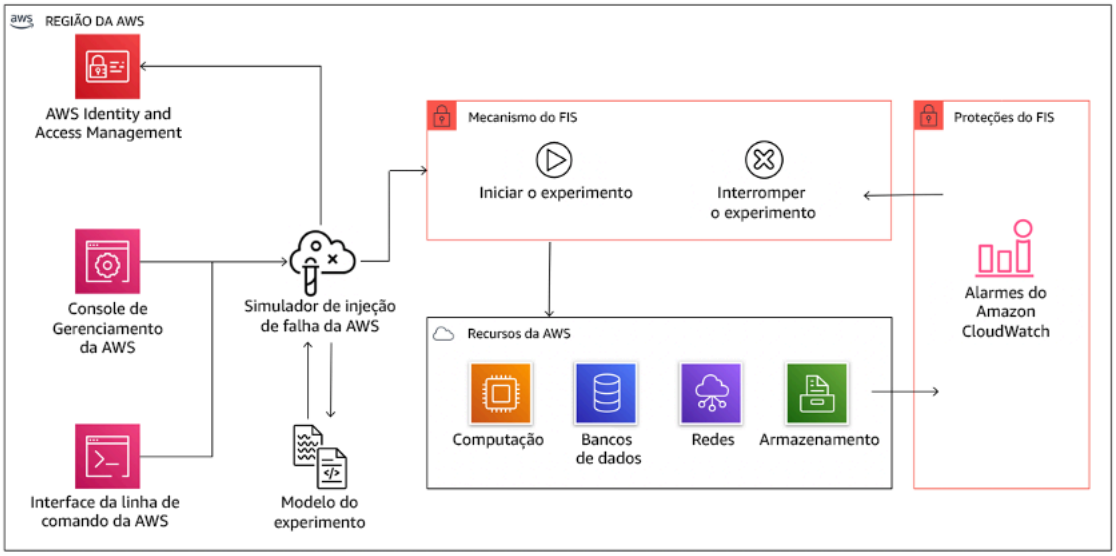
Para validar mecanismos de fallback ou failover para dependências externas devido a interferências intermitentes na rede, os componentes devem simular esse tipo de evento bloqueando o acesso aos provedores externos durante um período especificado, que pode variar de segundos a horas.

Outros modos de degradação podem levar a uma redução nas funcionalidades e a respostas lentas, muitas vezes levando a uma interrupção dos serviços. Essa degradação costuma resultar de um aumento na latência de serviços críticos e comunicação de rede não confiável (pacotes abandonados). Testes com essas falhas, incluindo efeitos de rede como latência, mensagens perdidas e falhas de DNS, podem incluir a incapacidade de resolver um nome, alcançar o serviço de DNS ou estabelecer conexões com serviços dependentes.

Ferramentas de engenharia de caos:

o AWS Fault Injection Service (AWS FIS) é um serviço totalmente gerenciado para a execução de experimentos de injeção de falha que podem ser usados como parte do pipeline de CD, ou fora do pipeline. O AWS FIS é uma boa opção para ser usado durante dias de jogo de engenharia de caos. Oferece suporte à introdução simultânea de falhas em diferentes tipos de recursos, incluindo Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service

(Amazon EKS) e Amazon RDS. Essas falhas incluem encerramento de recursos, failovers forçados, esgotamento de CPU ou memória, controle de utilização, latência e perda de pacotes. Por ser integrado a alarmes do Amazon CloudWatch, você pode definir condições de parada como barreiras de proteção para reverter um teste se causar impacto inesperado.



O AWS Fault Injection Service se integra a recursos da AWS para permitir a execução de experimentos de injeção de falha para as workloads.

Existem também várias opções de terceiros para experimentos de injeção de falhas. Elas incluem ferramentas de código aberto, como o Chaos Toolkit, Chaos Meshe aos Litmus Chaos, bem como opções comerciais como o Gremlin. Para expandir o escopo de falhas que podem ser injetadas na AWS, o AWS FIS integra-se ao Chaos Mesh e Litmus Chaos, possibilitando que você coordene fluxos de trabalho de injeção de falhas entre várias ferramentas. Por exemplo, você pode executar um teste de estresse na CPU de um pod usando falhas do Chaos Mesh ou Litmus enquanto encerra uma porcentagem selecionada aleatoriamente de nós de cluster usando ações de falha do AWS FIS.

Etapas da implementação

- Determine quais falhas usar para os testes.

  Avalie o design de sua workload quanto à resiliência. Tais designs (criados usando as práticas recomendadas do Well-Architected Framework) consideram riscos baseados em dependências críticas, eventos passados, problemas conhecidos e requisitos de conformidade. Liste cada elemento do design destinado a manter a resiliência e as falhas que foi projetado para mitigar. Para obter mais informações sobre a criação dessas listas, consulte o artigo técnico Análise de prontidão operacional que orienta você sobre como criar um processo para impedir a recorrência

de incidentes passados. O processo de modos de falhas e análises de efeitos (FMEA) proporciona um framework para realização de análise de falhas em nível de componente e como elas afetam a workload. O FMEA foi descrito em mais detalhes por Adrian Cockcroft em Failure Modes and Continuous Resilience (Modos de falhas e resiliência contínua).

- Atribua uma prioridade a cada falha.

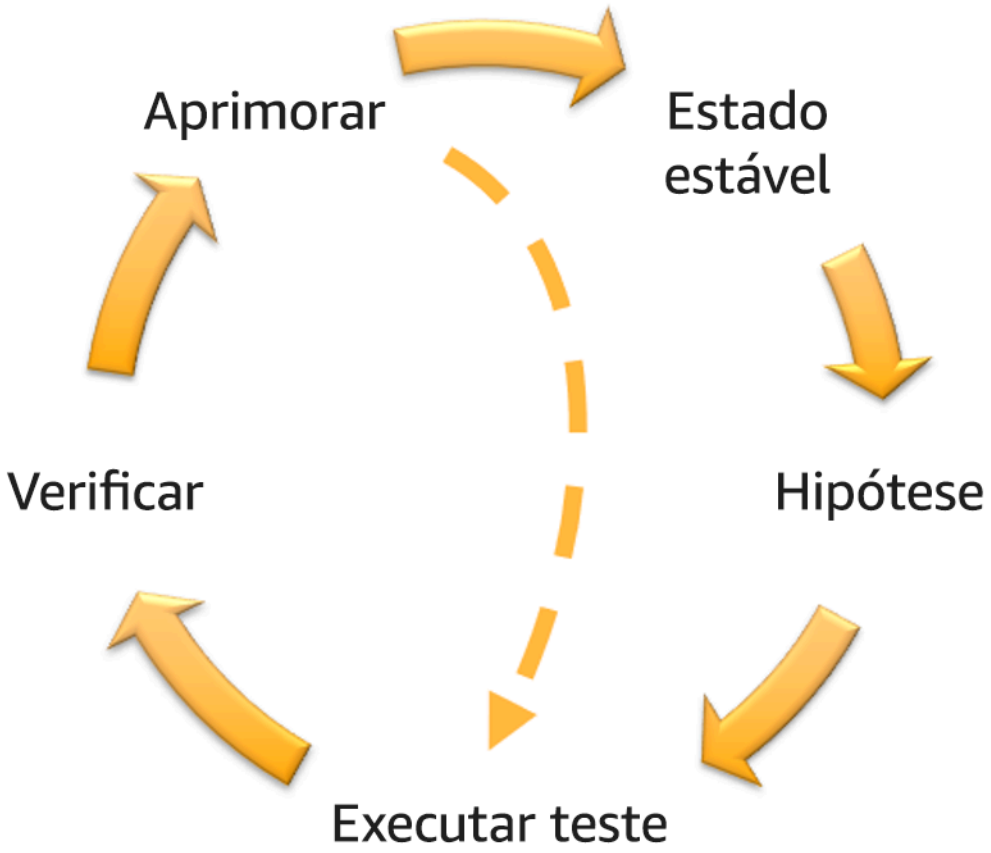
  Comece com uma categorização bruta, como alta, média e baixa. Para avaliar a prioridade, considere a frequência da falha e o impacto da falha na workload total.

  Ao considerar a frequência de determinada falha, analise os dados passados para essa workload sempre que disponíveis. Caso contrário, use os dados de outras workloads executadas em ambientes semelhantes.

  Ao considerar o impacto de determinada falha, em geral, quanto maior o escopo da falha, maior o impacto. Considere também o design e a finalidade da workload. Por exemplo, a capacidade de acessar os datastores de origem é essencial para uma workload que executa análise e transformação de dados. Nesse caso, priorize testes de falhas de acesso, além de acesso controlado e inserção de latência.

  Análises pós-incidente são boas fontes de dados para entender a frequência e o impacto dos modos de falha.

  Use a prioridade atribuída para determinar quais falhas escolher para testar primeiro e a sequência para desenvolver novos testes de injeção de falhas.
- Para cada teste realizado, siga o flywheel de engenharia de caos e resiliência contínua.



Flywheel de engenharia de caos e resiliência contínua, usando o método científico por Adrian Hornsby.

- Defina o estado estável como uma saída mensurável de uma workload que indica comportamento normal.

  Sua workload apresentará estado estável se estiver operando de maneira confiável e conforme o esperado. Portanto, valide a integridade da workload antes de definir o estado estável. O estado estável nem sempre significa que não há nenhum impacto à workload quando ocorre uma falha, já que determinada porcentagem de falhas pode estar dentro de limites aceitáveis. O estado estável é a linha de base que você vai observar durante o teste, o que vai destacar anomalias se a hipótese definida na próxima etapa não sair conforme o esperado.

  Por exemplo, um estado estável de um sistema de pagamentos pode ser definido como o processamento de 300 TPS com taxa de sucesso de 99% e tempo de ida e volta de 500 ms.

- Formule uma hipótese sobre como a workload vai reagir à falha.

Uma boa hipótese se baseia em como se espera que a workload mitigue a falha para manter o estado estável. A hipótese afirma que para determinado tipo de falha, o sistema ou a workload vai permanecer em estado estável, pois a workload foi projetada com mitigações específicas. O tipo específico de falhas e mitigações deve ser especificado na hipótese.

O modelo a seguir pode ser usado para a hipótese (mas outras palavras também são aceitáveis):

> **Note**
>
> Se *falha específica* ocorrer, a workload *nome da workload* vai *descrever os controles de mitigação* para manter *impacto da métrica de negócios ou técnica*.

Por exemplo:

- Se 20% dos nós no grupo de nós do Amazon EKS forem desativados, a API Transaction Create continuará atendendo ao 99.º percentil das solicitações em menos de 100 ms (estado estável). Os nós do Amazon EKS vão se recuperar em cinco minutos e os pods serão agendados e processarão o tráfego oito minutos depois do início do experimento. Os alertas serão acionados em três minutos.
- Se ocorrer uma única falha de instância do Amazon EC2, a verificação de integridade do Elastic Load Balancing do sistema de ordem vai fazer com que o Elastic Load Balancing envie solicitações apenas para as instâncias íntegras restantes, enquanto o Amazon EC2 Auto Scaling substitui a instância com falha, mantendo um aumento inferior a 0,01% na quantidade de erros no servidor (5xx) (estado estável).
- Se a instância de banco de dados primária do Amazon RDS falhar, a workload de coleta de dados da cadeia de suprimentos vai entrar em failover e se conectará à instância de banco de dados de espera do Amazon RDS para manter menos de um minuto de erros de leitura ou gravação de banco de dados (estado estável).

- Execute o teste injetando a falha.

Um teste deve, por padrão, ser seguro contra falhas e tolerado pela workload. Se você sabe que a workload vai falhar, não execute o teste. A engenharia de caos deve ser usada para encontrar incertezas conhecidas ou desconhecidas. Incertezas conhecidas são coisas que você conhece, mas não entende totalmente, enquanto incertezas desconhecidas são coisas que você não

conhece nem entende totalmente. Realizar testes em uma workload que você sabe que está quebrada não oferecerá novos insights. Seu teste deve ser cuidadosamente planejado, ter um escopo claro do impacto e fornecer um mecanismo de reversão que possa ser aplicado em caso de turbulência inesperada. Se sua devida diligência mostrar que a workload sobreviverá ao teste, prossiga com o teste. Há diversas opções para injetar as falhas. Para workloads na AWS, AWS FIS oferece diversas simulações de falhas predefinidas chamadas de ações. Você também pode definir ações personalizadas que são executadas no AWS FIS usando documentos do AWS Systems Manager.

Nós desencorajamos o uso de scripts personalizados para testes de caos, a menos que os scripts tenham os recursos para entender o estado atual da workload, sejam capazes de emitir logs e ofereçam mecanismos para rollbacks e condições de parada sempre que possível.

Um conjunto de ferramentas ou framework eficaz que ofereça suporte à engenharia de caos deve monitorar o estado atual de um experimento, emitir logs e fornecer mecanismos de rollback para oferecer suporte à execução controlada de um teste. Comece com um serviço estabelecido, como o AWS FIS, que permita que você realize testes com um escopo claramente definido e mecanismos de segurança que reverterão o teste se ele introduzir turbulência inesperada. Para conhecer uma ampla variedade de testes que usam o AWS FIS, consulte também o laboratório Aplicações resilientes e bem-arquitetadas com engenharia de caos. Além disso, o AWS Resilience Hub vai analisar sua workload e criar testes que você pode escolher para implementação e execução no AWS FIS.

> **Note**
>
> Para cada teste, entenda claramente o escopo e seu impacto. Recomendamos que as falhas sejam simuladas primeiro em um ambiente que não seja de produção, antes de serem executadas em produção.

Os testes devem ser executados em produção sob carga real usando implantações canário que ativam implantações de controle e experimentais no sistema, sempre que viável. A realização de testes durante horários fora de pico é uma boa prática para mitigar o impacto potencial durante o primeiro teste em produção. Além disso, se o uso de tráfego real de clientes for algo muito arriscado, você poderá executar testes usando tráfego sintético na infraestrutura de produção em implantações de controle e experimentais. Quando não for possível usar a produção, realize os testes em ambientes de pré-produção que sejam o mais parecido possível com produção.

Estabeleça e monitore barreiras de proteção para garantir que o teste não afete o tráfego de produção ou outros sistemas além dos limites aceitáveis. Estabeleça condições de parada para interromper um teste se ele atingir um limite definido de uma métrica de barreira de proteção. Isso deve incluir as métricas de estado estável da workload, bem como a métrica em relação aos componentes em que você está injetando a falha. A monitor sintético (também conhecido como canário de usuário) é uma métrica que geralmente deve ser incluída como proxy de usuário. Condições de parada do AWS FIS são compatíveis como parte do modelo de teste, permitindo até cinco condições de parada por modelo.

Um dos princípios de caos é minimizar o escopo do teste e seu impacto:

embora deva existir uma provisão para algum impacto negativo de curto prazo, é responsabilidade e obrigação do engenheiro de caos garantir que as perdas dos testes sejam minimizadas e contidas.

Um método para verificar o escopo e o impacto potencial é realizar o teste primeiro em um ambiente que não seja de produção, verificando se os limites para as condições de parada são ativados conforme o esperado durante o teste e se há observabilidade em vigor para identificar uma exceção, em vez de testar diretamente em produção.

Ao executar testes de injeção de falhas, verifique se todas as partes responsáveis estão bem informadas. Comunique-se com as equipes adequadas, como equipes de operações, equipes de confiabilidade do serviço e atendimento ao cliente, para avisá-las sobre quando os testes serão realizados e o que esperar. Ofereça a essas equipes ferramentas de comunicação para que informem os responsáveis pela execução do teste caso percebam algum efeito adverso.

Você deve restaurar a workload e seus sistemas subjacentes de volta para o estado íntegro original. Normalmente, o design resiliente da workload vai se autorrestaurar. No entanto, alguns designs de falhas ou testes malsucedidos podem deixar a workload em um estado de falha inesperado. Ao final do teste, você deverá estar ciente disso e restaurar a workload e os sistemas. Com o AWS FIS, você pode definir uma configuração de reversão (também chamada de ação posterior) nos parâmetros de ação. Uma ação posterior restaura o destino para o estado em que estava antes da execução da ação. Independentemente de serem automatizadas (como as que usam o AWS FIS) ou manuais, essas ações posteriores devem fazer parte de um playbook que descreve como detectar e lidar com falhas.

- Verifique a hipótese.

Princípios da engenharia do caos oferecem a seguinte orientação sobre como verificar o estado estável de sua workload:

Concentre-se na saída mensurável de um sistema, em vez de atributos internos do sistema. As medições dessa saída durante um curto período constituem um proxy do estado estável do sistema. A throughput total do sistema, as taxas de erros e os percentis de latência podem ser métricas de interesse que representam o comportamento do estado estável. Ao focar em padrões de comportamento sistêmicos durante os testes, a engenharia de caos verifica se o sistema de fato funciona em vez de tentar validar como ele funciona.

Nos dois exemplos anteriores, nós incluímos as métricas de estado estável de menos de 0,01% de aumento na quantidade de erros no servidor (5xx) e menos de um minuto de erros de leitura ou gravação de banco de dados.

Os erros 5xx são uma boa métrica, pois são consequência do modo de falha que um cliente da workload vai vivenciar diretamente. A medição dos erros do banco de dados é boa como consequência direta da falha, mas também deve ser complementada com uma medição de impacto para o cliente, como solicitações malsucedidas ou erros apresentados ao cliente. Além disso, inclua um monitor sintético (também conhecido como canário de usuário) em todas as APIs ou URIs acessadas pelo cliente da workload.

- Melhore o design da workload para agregar resiliência.

  Se o estado estável não tiver sido mantido, investigue como o design da workload pode ser melhorado para mitigar a falha, aplicando as práticas recomendadas do pilar Confiabilidade do AWS Well-Architected. Orientação e recursos adicionais podem ser encontrados na AWS Builder's Library, que contém artigos sobre como melhorar as verificações de integridade ou implantar repetições sem recuo no código de sua aplicação, entre outros.

  Depois de implementar essas mudanças, execute o teste novamente (mostrado pela linha pontilhada no flywheel de engenharia de caos) para determinar a eficácia. Se a etapa de verificação indicar que a hipótese é verdadeira, a workload estará em estado estável e o ciclo continuará.

- Execute testes regularmente.

Um teste de caos é um ciclo, e os testes devem ser realizados regularmente como parte da engenharia de caos. Depois que uma workload cumprir a hipótese do teste, o teste deverá ser automatizado para ser executado continuamente como parte de regressão do pipeline de CI/CD.

Para saber como fazer isso, consulte este blog sobre como executar testes do AWS FIS usando o AWS CodePipeline. Este laboratório sobre testes recorrentes do AWS FIS em um pipeline de CI/CD permite que você trabalhe de maneira prática.

Os testes de injeção de falhas também fazem parte dos dias de jogo (consulte REL12-BP06 Realizar dias de jogo regularmente). Os dias de jogo simulam uma falha ou um evento para verificar sistemas, processos e respostas das equipes. O objetivo é realmente executar as ações que a equipe executaria como se um evento excepcional acontecesse.

- Capture e armazene os resultados do teste.

  Os resultados da injeção de falhas devem ser capturados e persistidos. Inclua todos os dados necessários (como tempo, workload e condições) para poder analisar os resultados e as tendências do teste posteriormente. Exemplos de resultados podem incluir capturas de tela de painéis, despejos em CSV do banco de dados da métrica ou um registro manual dos eventos e das observações do teste. O registro do teste em log com o AWS FIS pode fazer parte dessa captura de dados.

Recursos

Práticas recomendadas relacionadas:

- REL08-BP03 Integrar testes de resiliência como parte da sua implantação
- REL13-BP03 Testar a implementação da recuperação de desastres para validá-la

Documentos relacionados:

- O que é o AWS Fault Injection Service?
- O que é o AWS Resilience Hub?
- Princípios da engenharia do caos
- Engenharia de caos: planejando seu primeiro teste
- Engenharia de resiliência: aprendendo a aceitar falhas
- Histórias sobre engenharia de caos
- Evitar fallback em sistemas distribuídos
- Implantação canário para testes de caos

Vídeos relacionados:

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316) (AWS re:Invent 2020: teste de resiliência usando engenharia de caos)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1) (AWS re:Invent 2019: melhoria da resiliência com engenharia de caos)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301) (AWS re:Invent 2019: execução da engenharia de caos em um universo de tecnologia sem servidor)

Exemplos relacionados:

- Laboratório do Well-Architected: nível 300: testes de resiliência do Amazon EC2, Amazon RDS e Amazon S3
- Laboratório Engenharia de caos na AWS
- Laboratório Aplicações resilientes e bem-arquitetadas com engenharia de caos
- Laboratório Caos em tecnologia sem servidor
- Laboratório Mensurar e aumentar a resiliência de sua aplicação com o AWS Resilience Hub

Ferramentas relacionadas:

- AWS Fault Injection Service
- AWS Marketplace: plataforma de engenharia de caos Gremlin
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP06 Realizar dias de jogo regularmente

Use os dias de jogo para praticar regularmente seus procedimentos de resposta a eventos e falhas o mais próximo possível da produção (inclusive em ambientes de produção) e com as pessoas que estarão envolvidas nos cenários de falha reais. Os dias de jogo aplicam medidas para garantir que os eventos de produção não afetem os usuários.

Os dias de jogo simulam uma falha ou evento para testar sistemas, processos e respostas das equipes. O objetivo é realmente executar as ações que a equipe executaria como se um evento

excepcional acontecesse. Isso ajudará a compreender onde as melhorias podem ser feitas e pode ajudar a desenvolver experiência organizacional ao lidar com eventos. Eles devem ser realizados regularmente para que a equipe desenvolva memória muscular sobre como responder.

Depois que o projeto de resiliência estiver em vigor e tiver sido testado em ambientes que não sejam de produção, um dia de jogo será a maneira de garantir que tudo funcione conforme o planejado na produção. Um dia de jogo, especialmente o primeiro, é uma atividade de "todos os funcionários" em que engenheiros e operações são informados quando isso acontecerá e o que ocorrerá. Há runbooks disponíveis. Os eventos simulados são executados, incluindo possíveis eventos de falha, nos sistemas de produção da maneira prescrita, e o impacto é avaliado. Se todos os sistemas operarem conforme projetado, a detecção e a recuperação automática ocorrerão com pouco ou nenhum impacto. No entanto, se houver impacto negativo, o teste será revertido e os problemas da workload serão corrigidos manualmente, se necessário (usando o runbook). Como os dias de jogos ocorrem na produção, todas as precauções devem ser tomadas para garantir que não haja impacto na disponibilidade dos clientes.

Antipadrões comuns:

- Documentar seus procedimentos, mas nunca os praticar.
- Não incluir os tomadores de decisão de negócios nos exercícios de teste.

Benefícios do estabelecimento desta prática recomendada: A realização frequente dos dias de jogo garante que toda a equipe siga e valide as políticas e os procedimentos apropriados quando ocorrer um incidente real.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- Programe os dias de jogo para praticar regularmente os runbooks e os manuais. Os dias de jogo devem incluir todas as pessoas envolvidas em um evento de produção: proprietário da empresa, equipe de desenvolvimento, equipe operacional e equipes de resposta a incidentes.
  - Execute os testes de carga ou de performance e, em seguida, execute a injeção de falha.
  - Procure por anomalias nos runbooks e oportunidades de praticar os playbooks.
    - Se você se desviar dos runbooks, refine-os ou corrija o comportamento. Se você praticar o playbook, identifique o runbook que deveria ter sido usado ou crie um novo.

Recursos

Documentos relacionados:

- O que é o AWS GameDay?

Vídeos relacionados:

- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)

Exemplos relacionados:

- Laboratórios do AWS Well-Architected: testes de resiliência

Planejar para a recuperação de desastres (DR)

Implementar backups e componentes redundantes de carga de trabalho é o ponto de partida da sua estratégia de DR. RTO e RPO são os seus objetivos para a restauração de sua workload. Defina-os de acordo com suas necessidades de negócios. Implemente uma estratégia para atender a esses objetivos, considerando os locais e a função dos recursos e dos dados da carga de trabalho. A probabilidade de interrupção e o custo de recuperação também são fatores principais que ajudam a determinar o valor empresarial de fornecer a recuperação de desastres para uma workload.

A disponibilidade e a recuperação de desastres dependem das mesmas práticas recomendadas, como o monitoramento de falhas, a implantação de vários locais e o failover automático. No entanto, a disponibilidade se concentra nos componentes da workload, enquanto a recuperação de desastres se concentra nas cópias distintas de toda a workload. A recuperação de desastres tem objetivos diferentes da disponibilidade, focalizando-se no tempo de recuperação após um desastre.

Práticas recomendadas

- REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados
- REL13-BP02 Usar estratégias de recuperação definidas para cumprir os objetivos de recuperação
- REL13-BP03 Testar a implementação da recuperação de desastres para validá-la
- REL13-BP04 Gerenciar o desvio de configuração para o local ou a região de DR
- REL13-BP05 Automatizar a recuperação

REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados

A carga de trabalho tem um Recovery Time Objective (RTO – Objetivo do tempo de recuperação) e um Recovery Point Objective (RPO – Objetivo do ponto de recuperação).

Recovery Time Objective (RTO – Objetivo do tempo de recuperação) é o atraso máximo aceitável entre a interrupção do serviço e sua restauração. Isso determina o que é considerado uma janela de tempo aceitável quando o serviço está indisponível.

Recovery Point Objective (RPO – Objetivo do ponto de recuperação) é o tempo máximo aceitável desde o último ponto de recuperação de dados. Isso determina o que é considerado uma perda aceitável de dados entre o último ponto de recuperação e a interrupção do serviço.

Os valores de RTO e RPO são considerações importantes ao selecionar uma estratégia de recuperação de desastres (DR) apropriada para a workload. Esses objetivos são determinados pelo negócio e, em seguida, usados pelas equipes técnicas para selecionar e implementar uma estratégia de DR.

Resultado desejado:

Cada workload tem um RTO e um RPO atribuídos, definidos com base no impacto empresarial. A workload é atribuída a uma camada predefinida com um RTO e um RPO associados, estabelecendo a disponibilidade do serviço e a perda aceitável de dados. Se isso não for possível, poderá ser atribuído sob medida por workload com a intenção de criar camadas posteriormente. O RTO e o RPO são usados como uma das principais considerações para a seleção da implementação de uma estratégia de recuperação de desastres para a workload. São considerações adicionais na escolha de uma estratégia de DR as restrições de custo, as dependências da workload e os requisitos operacionais.

Para o RTO, compreenda o impacto com base na duração de uma interrupção. É linear ou há implicações não lineares? (Por exemplo, após quatro horas, você desliga uma linha de produção até o início do próximo turno).

Uma matriz de recuperação de desastres, como a seguinte, pode ajudar você a compreender como a criticidade da workload se relaciona com os objetivos de recuperação. (Observe que os valores reais dos eixos X e Y devem ser personalizados de acordo com as necessidades da sua organização).

Matriz de recuperação de desastres						
		Objetivo do ponto de recuperação				
		< 1 minuto	< 1 hora	< 6 horas	< 1 dia	+ 1 dia
Objetivo do tempo de recuperação	< 10 minutos	Crítica	Crítica	Alto	Médio	Médio
	< 2 horas	Crítica	Alto	Médio	Médio	Baixo
	< 8 horas	Alto	Médio	Médio	Baixo	Baixo
	< 24 horas	Médio	Médio	Baixo	Baixo	Baixo
	+ de 24 horas	Médio	Baixo	Baixo	Baixo	Baixo

Figura 16: Matriz de recuperação de desastres

Antipadrões comuns:

- Objetivos de recuperação não definidos.
- Seleção de objetivos de recuperação arbitrários.
- Seleção de objetivos de recuperação que são muito permissivos e não atendem aos objetivos de negócios.
- Não compreender o impacto do tempo de inatividade e da perda de dados.
- Seleção de objetivos de recuperação irreais, como nenhum tempo para recuperação e nenhuma perda de dados, que podem não ser alcançáveis para a configuração da workload.
- Seleção de objetivos de recuperação mais rigorosos do que os objetivos de negócios reais. Isso força implementações de DR mais caras e complicadas do que as necessidades da workload.
- Seleção de objetivos de recuperação incompatíveis com os da workload dependente.
- Os objetivos de recuperação não consideram os requisitos regulamentares de conformidade.
- RTO e RPO definidos para uma workload, mas nunca testados.

Benefícios do estabelecimento dessa prática recomendada: Os objetivos de recuperação referentes a tempo e perda de dados são necessários para orientar a implementação da DR.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Alto

Orientações para a implementação

Para a workload, você deve compreender o impacto do tempo de inatividade e da perda de dados em seus negócios. O impacto geralmente aumenta com maior tempo de inatividade ou perda de dados, mas a forma desse crescimento pode diferir com base no tipo de workload. Por exemplo, pode ser que você consiga tolerar o tempo de inatividade por até uma hora com pouco impacto, mas depois disso o impacto aumenta rapidamente. O impacto nos negócios se manifesta de diversas formas, incluindo custo monetário (como perda de receita), confiança do cliente (e impacto na reputação), problemas operacionais (como folha de pagamento ausente ou diminuição na produtividade) e risco regulatório. Use as etapas a seguir para compreender esses impactos e defina o RTO e o RPO para sua workload.

Etapas da implementação

1. Determine as partes interessadas do negócio para a workload e interaja com eles para implementar essas etapas. Os objetivos de recuperação para uma workload são uma decisão de negócios. As equipes técnicas trabalham com as partes interessadas do negócio para usar esses objetivos para selecionar uma estratégia de DR.

> **Note**
>
> Para as etapas 2 e 3, você pode usar o the section called “Planilha de implementação”.

2. Reúna as informações necessárias para tomar uma decisão respondendo às perguntas abaixo.
3. Você tem categorias ou níveis de criticidade para o impacto da workload na sua organização?
   a. Se sim, atribua esta workload a uma categoria
   b. Se não, estabeleça estas categorias. Crie cinco ou menos categorias e refine o intervalo do seu objetivo de tempo de recuperação para cada uma delas. Os exemplos de categorias incluem: crítica, alta, média, baixa. Para entender como uma workloads é mapeada para uma categoria, considere se ela é de missão crítica, importante para os negócios ou não comercial.
   c. Defina o RTO e o RPO da workload com base na categoria. Sempre escolha uma categoria mais restrita (RTO e RPO mais baixos) do que os valores brutos calculados no começo desta etapa. Se isso resultar em uma mudança de valor inadequadamente grande, considere a criação de uma nova categoria.
4. Com base nessas respostas, atribua valores de RTO e RPO à workload. Isso pode ser feito diretamente ou atribuindo a workload a uma camada de serviço predefinida.

5. Documente o plano de recuperação de desastres (DRP) para esta workload, que faz parte do plano de continuidade de negócios (BCP) da sua organização, em um local acessível à equipe de workload e às partes interessadas
   a. Registre o RTO, o RPO e as informações usadas para determinar esses valores. Inclua a estratégia usada para avaliar o impacto da workload nos negócios.
   b. Registre outras métricas, além do RTO e do RPO que você está acompanhando ou planeja acompanhar, para os objetivos de recuperação de desastres
   c. Você adicionará detalhes da sua estratégia de DR e runbook a este plano ao criá-los.
6. Ao pesquisar a criticidade da workload em uma matriz, como a da figura 15, você pode começar a estabelecer camadas predefinidas de serviço estabelecidos para sua organização.
7. Após implementar uma estratégia de DR (ou uma prova de conceito para uma estratégia de DR) conforme the section called “REL13-BP02 Usar estratégias de recuperação definidas para cumprir os objetivos de recuperação”, teste a estratégia para determinar a capacidade de tempo de recuperação (RTC) e a capacidade de ponto de recuperação (RPC) reais da workload. Se elas não atenderem aos objetivos de recuperação de destino, trabalhe com as partes interessadas do negócio para ajustar esses objetivos ou faça alterações na estratégia de DR para atingir os objetivos de destino.

Perguntas principais

1. Qual é o tempo máximo que a workload pode ficar inativa antes que ocorra um impacto grave nos negócios?
   a. Determine o custo monetário (impacto financeiro direto) para o negócio por minuto se a workload for interrompida.
   b. Considere que o impacto nem sempre é linear. O impacto pode ser limitado no início e aumentar rapidamente após um ponto crítico.
2. Qual é a quantidade máxima de dados que podem ser perdidos antes que ocorra um impacto severo nos negócios?
   a. Considere esse valor para seu armazenamento de dados mais crítico. Identifique a respectiva criticidade para outros armazenamentos de dados.
   b. Os dados de workload podem ser recriados em caso de perda? Se isso for operacionalmente mais fácil do que fazer backup e restauração, escolha o RPO com base na criticidade dos dados de origem usados para recriar os dados da workload.

3. Quais são os objetivos de recuperação e as expectativas de disponibilidade das workloads das quais este depende (downstream) ou as workloads que dependem deste (upstream)?
   a. Escolha objetivos de recuperação que permitam que essa workload atenda aos requisitos das dependências upstream.
   b. Escolha objetivos de recuperação que possam ser alcançados com base nos recursos de recuperação das dependências downstream. Dependências downstream não críticas (aquelas que podem ser “contornadas”) podem ser excluídas. Ou trabalhe com dependências críticas downstream para melhorar os recursos de recuperação quando necessário.

Perguntas adicionais

Considere estas perguntas e como elas podem se aplicar a essa workload:

4. Você tem RTO e RPO diferentes dependendo do tipo de interrupção (região versus AZ etc.)?
5. Existe um momento específico (sazonalidade, eventos de vendas, lançamentos de produtos) em que seu RTO/RPO pode mudar? Se sim, quais são a medição e o limite de tempo diferentes?
6. Quantos clientes serão afetados se a workload for interrompida?
7. Qual será o impacto na reputação se a workload for interrompida?
8. Quais outros impactos operacionais poderão ocorrer se a workload for interrompida? Por exemplo, impacto na produtividade do funcionário se os sistemas de e-mail não estiverem disponíveis ou se os sistemas de folha de pagamento não puderem enviar transações.
9. Como o RTO e o RPO da workload se alinham à estratégia de DR da linha empresarial e organizacional?
10. Há obrigações contratuais internas para a prestação de um serviço? Há penalidades por não cumpri-las?
11. Quais são as restrições regulatórias ou de conformidade com os dados?

Planilha de implementação

Você pode usar esta planilha para as etapas 2 e 3 de implementação. É possível ajustar esta planilha para atender às suas necessidades específicas, como adicionar perguntas.

Etapa 2: Perguntas principais	Aplicável à workload?	RTO da workload	RPO da workload	Ajuste do RTO.	Ajuste do RPO.	Instruções
[1] tempo máximo em que a workload pode ficar inativa						medido com o tempo desde o início da interrupção da recuperação
[2] quantidade máxima de dados que podem ser perdidos						medido com o tempo desde o conjunto de dados bom mais recente restaurável
[3a] dependências upstream						insira os objetivos mais estritos de recuperação upstream
[3b] dependências downstream						insira os objetivos menos estritos de recuperação downstream
[3a] dependências upstream reconciliadas						Se o valor upstream for menor que os valores atuais e o valor downstream for maior,
[3b] dependências downstream reconciliadas						trabalhe com as dependências para fazer a reconciliação e insira os valores reconciliados aqui
[3] dependências						valores menores para atender às dependências upstream ou aumentá-las com base nas capacidades das dependências downstream
Etapa 2: Perguntas adicionais						Indique se a pergunta é aplicável. Se ela não for aplicável, ignore-a
RTO/RPO de base						Carregue os valores de RTO e de RPO acima para baixo, aqui
[4] tipo de interrupção	[ ]S/[ ]N					Insira os objetivos de recuperação para o tipo de evento com os requisitos mais estritos
[5] objetivos baseados em tempo específico	[ ]S/[ ]N					Insira os objetivos de recuperação para momentos com os requisitos mais estritos
[6] clientes interrompidos	[ ]S/[ ]N					Faça um gráfico dos clientes afetados como uma função de tempo de inatividade ou de perda de dados. Use isso para inserir o RTO e o RPO máximos permissíveis com base no impacto no cliente.
[7] impacto na reputação	[ ]S/[ ]N					Trabalhe com a empresa para determinar o RTO e o RPO máximos com base no impacto na reputação
[8] impacto operacional	[ ]S/[ ]N					Insira o RTO e o RPO máximos com base no impacto operacional
[9] alinhamento organizacional	[ ]S/[ ]N					Insira o RTO e o RPO máximos para workloads desse tipo de acordo com as necessidades da LOB e da organização
[10] obrigações contratuais	[ ]S/[ ]N					Insira o RTO e o RPO máximos com base nas obrigações contratuais
[11] conformidade normativa	[ ]S/[ ]N					Insira o RTO e o RPO máximos com base na conformidade normativa aplicável
alvo baseado em questões adicionais						Use o valor mínimo (valor mais estrito) das perguntas 4 a 11 e insira-o aqui
alvo ajustado						Se os objetivos na linha acima não puderem ser acomodados, trabalhe com as partes interessadas para flexibilizar as restrições e insira o novo mínimo aqui
RTO/RPO ajustado						Insira os valores do RPO/RTO de base ou ajuste o alvo, o que for menor
Etapa 3						
Mapear para categoria ou camada predefinida						Ajuste os dois valores para baixo (mais estritos) para que se alinhem com a camada mais próxima definida

Planilha

Nível de esforço para o plano de implementação: Baixo

Recursos

Práticas recomendadas relacionadas:

- the section called “REL09-BP04 Realizar a recuperação periódica dos dados para verificar a integridade e os processos de backup”
- the section called “REL13-BP02 Usar estratégias de recuperação definidas para cumprir os objetivos de recuperação”
- the section called “REL13-BP03 Testar a implementação da recuperação de desastres para validá-la”

Documentos relacionados:

- Blog de arquitetura da AWS: série de recuperação de desastres
- Recuperação de desastres de workloads na AWS: recuperação na nuvem (whitepaper da AWS)

- Gerenciamento de políticas de resiliência com o AWS Resilience Hub
- Parceiro do APN: parceiros que podem ajudar com a recuperação de desastres
- AWS Marketplace: produtos que podem ser usados para recuperação de desastres

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Recuperação de desastres de workloads na AWS

REL13-BP02 Usar estratégias de recuperação definidas para cumprir os objetivos de recuperação

Defina uma estratégia de recuperação de desastres (DR) que cumpra os objetivos de recuperação da workload. Escolha uma estratégia como backup e restauração, standby (ativo/passivo) ou ativo/ativo.

Resultado desejado: há uma estratégia de DR definida e implementada para cada workload, permitindo que ela atinja os objetivos de DR. As estratégias de DR entre workloads fazem uso de padrões reutilizáveis (como as estratégias descritas anteriormente).

Antipadrões comuns:

- Implementar procedimentos de recuperação inconsistentes para workloads com objetivos de DR semelhantes.
- Deixar que a estratégia de DR seja implementada ad hoc quando ocorrer um desastre.
- Não ter um plano para a recuperação de desastres.
- Depender das operações do ambiente de gerenciamento durante a recuperação.

Benefícios do estabelecimento desta prática recomendada:

- O uso de estratégias de recuperação definidas permite que você adote ferramentas comuns e procedimentos de teste.
- Usar estratégias de recuperação definidas melhora o compartilhamento de conhecimento entre as equipes e a implementação da DR nas workloads que possuem.

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto. Sem uma estratégia de DR planejada, implementada e testada, é improvável que você cumpra os objetivos de recuperação em caso de desastre.

Orientação de implementação

Uma estratégia de DR depende da capacidade de manter a workload em um site de recuperação se o local primário não puder executar a workload. Os objetivos de recuperação mais comuns são o RTO e o RPO, conforme discutido em REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados.

Uma estratégia de DR em várias zonas de disponibilidade (AZs) em uma única Região da AWS pode fornecer mitigação contra eventos de desastre, como incêndios, inundações e grandes interrupções de energia. Se for um requisito implementar proteção contra um evento improvável que impeça a execução da workload em determinada Região da AWS, você poderá optar por uma estratégia de DR que use várias regiões.

Ao arquitetar uma estratégia de DR em várias regiões, você deve escolher uma das estratégias a seguir. Elas estão listadas em ordem crescente de custo e complexidade e em ordem decrescente de RTO e RPO. Região de recuperação refere-se a uma Região da AWS diferente da primária usada para a workload.

Figura 17: Estratégias de recuperação de desastres (DR)

- Backup e restauração (RPO em horas, RTO em 24 horas ou menos): faça backup de seus dados e aplicações na região de recuperação. O uso de backups automatizados ou contínuos permitirá a recuperação a um ponto anterior no tempo, podendo reduzir o RPO para até cinco minutos em

alguns casos. Em caso de desastre, você implantará a infraestrutura (usando a infraestrutura como código para reduzir o RTO), implantará o código e restaurará os dados salvos para se recuperar de um desastre na região de recuperação.

- Luz piloto (RPO em minutos, RTO em dezenas de minutos): provisione uma cópia da infraestrutura de workload principal na região de recuperação. Replique seus dados na região de recuperação e crie backups deles lá. Os recursos necessários para permitir a replicação e o backup, como bancos de dados e armazenamento de objetos, estão sempre ativos. Outros elementos, como servidores de aplicações ou computação com tecnologia sem servidor, não são implantados. Porém, podem ser criados com a configuração e o código da aplicação necessários.
- Standby passivo (RPO em segundos, RTO em minutos): mantenha uma versão reduzida, mas totalmente funcional, da workload sempre em execução na região de recuperação. Os sistemas críticos para os negócios são totalmente duplicados e estão sempre ativados, mas com uma frota reduzida. Os dados são replicados e vivem na região de recuperação. Quando chega o momento da recuperação, o sistema é dimensionado rapidamente para processar a carga de produção. Quanto mais a escala do standby passivo for aumentada verticalmente, menor será a dependência do RTO e do ambiente de gerenciamento. Quando totalmente dimensionado, isso é conhecido como standby a quente.
- Ativo/ativo de várias regiões (multissite) (RPO próximo a zero, RTO potencialmente zero): a workload é implantada em várias Regiões da AWS e processa ativamente o tráfego delas. Essa estratégia exige que você sincronize os dados entre regiões. Deve-se evitar ou lidar com possíveis conflitos causados por gravações no mesmo registro em duas réplicas regionais diferentes, o que pode ser complexo. A replicação de dados é útil para a sincronização de dados e protegerá você contra alguns tipos de desastre, mas não contra corrupção ou destruição de dados, a menos que sua solução também inclua opções para recuperação a um ponto anterior no tempo.

> **Note**
>
> Às vezes, a diferença entre luz-piloto e standby passivo pode ser difícil de entender. Ambos incluem um ambiente na região de recuperação com cópias dos ativos da região primária. A diferença é que a luz-piloto não pode processar solicitações sem primeiro realizar uma ação adicional, enquanto o standby passivo pode processar o tráfego (em níveis de capacidade reduzidos) imediatamente. A luz piloto exigirá que você ative os servidores, possivelmente implante infraestrutura adicional (não essencial) e aumente a escala verticalmente. Já o standby passivo exige apenas que você aumente a escala verticalmente (tudo já está implantado e em execução). Escolha entre elas com base nas suas necessidades de RTO e RPO.

Quando o custo é uma preocupação e você deseja alcançar objetivos de RPO e RTO semelhantes, conforme definido na estratégia de standby passivo, é possível considerar soluções nativas da nuvem, como AWS Elastic Disaster Recovery, que adota a abordagem de luz piloto e oferece metas de RPO e RTO aprimoradas.

Etapas da implementação

1. Determine uma estratégia de DR que satisfaça os requisitos de recuperação para essa workload.

Escolher uma estratégia de DR é uma troca entre reduzir o tempo de inatividade e a perda de dados (RTO e RPO) e o custo e a complexidade da sua implementação. Você deve evitar implementar uma estratégia mais rigorosa do que necessário, pois isso resulta em custos desnecessários.

Por exemplo, no diagrama a seguir, a empresa determinou o RTO máximo permitido e o orçamento limite da estratégia de restauração de serviço. Considerando os objetivos empresariais, as estratégias de DR luz-piloto e standby passivo satisfarão tanto o RTO quanto os critérios de custo.

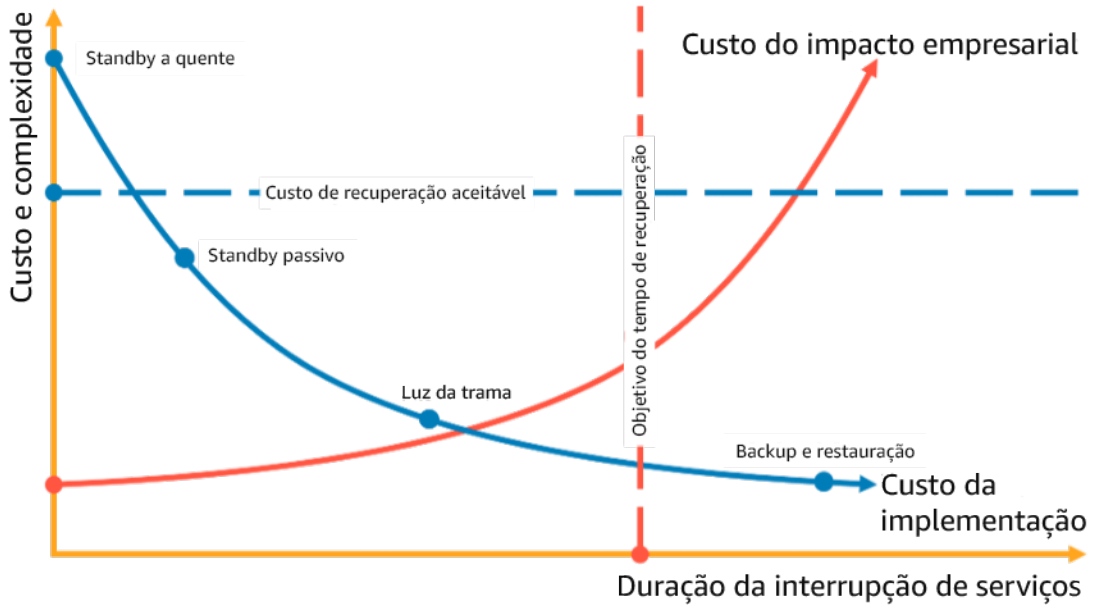


Figura 18: Escolha de uma estratégia de DR com base no RTO e no custo

Para saber mais, consulte Business Continuity Plan (BCP) (Plano de continuidade de negócios (BCP).

2. Revise os padrões de como a estratégia de DR selecionada pode ser implementada.

Essa etapa é para compreender como implementar a estratégia selecionada. As estratégias são explicadas usando as Regiões da AWS como locais primários e de recuperação. No entanto, também é possível optar por usar as zonas de disponibilidade em uma única região como sua estratégia de DR, que faz uso de elementos de várias dessas estratégias.

Nas etapas a seguir, é possível aplicar a estratégia para sua workload específica.

Backup e restauração

Backup e restauração é a estratégia menos complexa de implementar. Porém, exigirá mais tempo e esforço para restaurar a workload, levando a RTO e RPO mais altos. É uma boa prática sempre fazer backups dos dados e copiá-los para outro local (como outra Região da AWS).

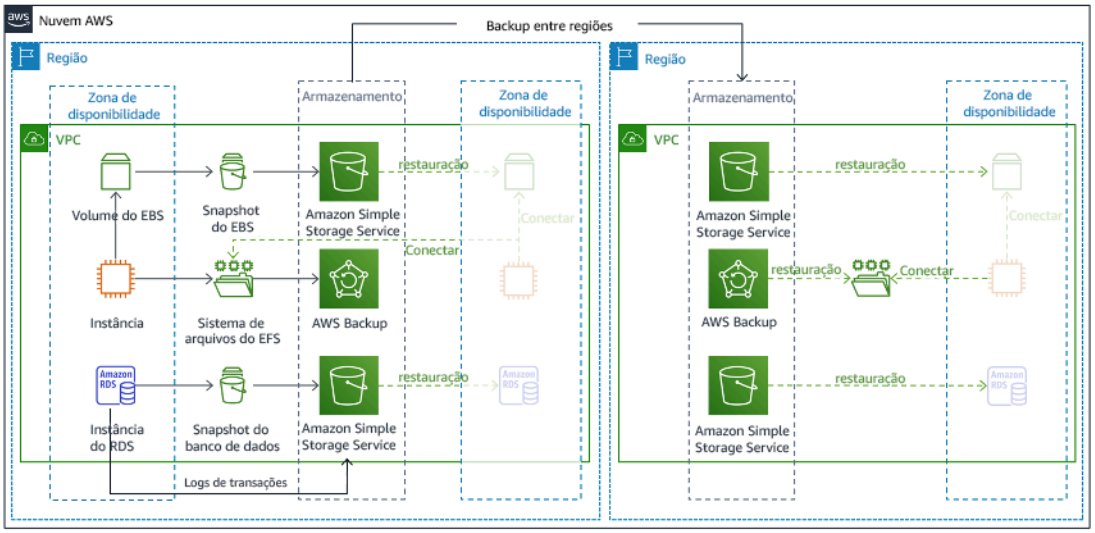


Figura 19: Arquitetura de backup e restauração

Para obter mais detalhes sobre essa estratégia, consulte Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery (Arquitetura de recuperação de desastres (DR) na AWS, parte II: backup e restauração com recuperação rápida).

Luz piloto

Com a abordagem de luz-piloto, você replica os dados da região primária para a região de recuperação. Os principais recursos usados para a infraestrutura da workload são implantados na região de recuperação. No entanto, recursos adicionais e as dependências ainda são necessários para tornar a pilha funcional. Por exemplo, na Figura 20, nenhuma instância de computação é implantada.

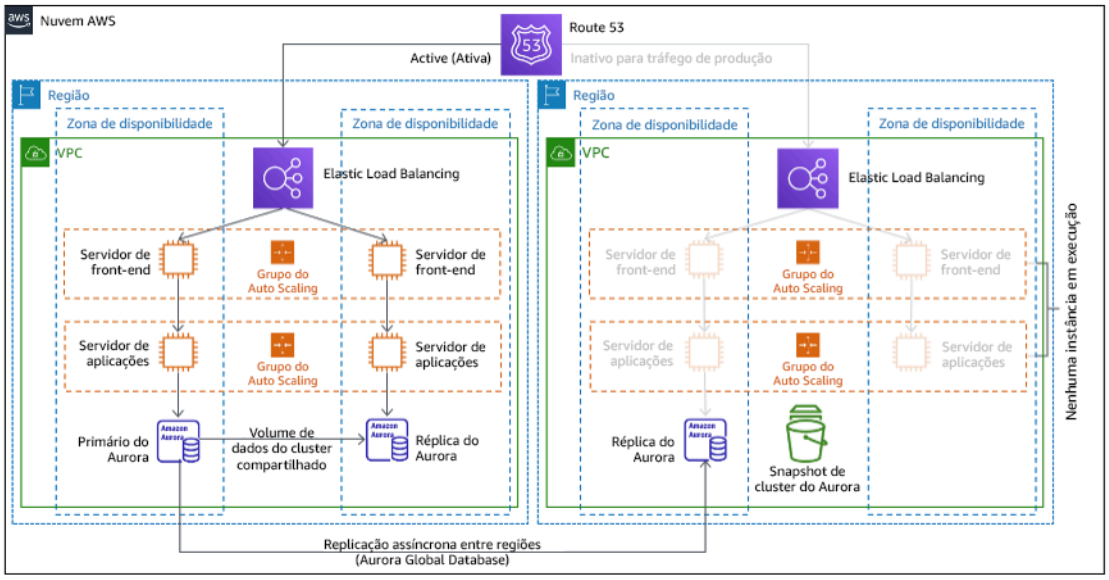


Figura 20: Arquitetura de luz-piloto

Para obter mais detalhes sobre essa estratégia, consulte Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby (Arquitetura de recuperação de desastres (DR) na AWS, parte III: luz-piloto e standby passivo).

Modo de espera passivo

A abordagem de standby passivo envolve garantir que haja uma cópia com escala reduzida verticalmente, mas totalmente funcional, do seu ambiente de produção em outra região. Essa abordagem estende o conceito de luz-piloto e diminui o tempo de recuperação, já que a workload está sempre ativa em outra região. Se a região de recuperação estiver implantada com sua capacidade total, isso é conhecido como standby a quente.

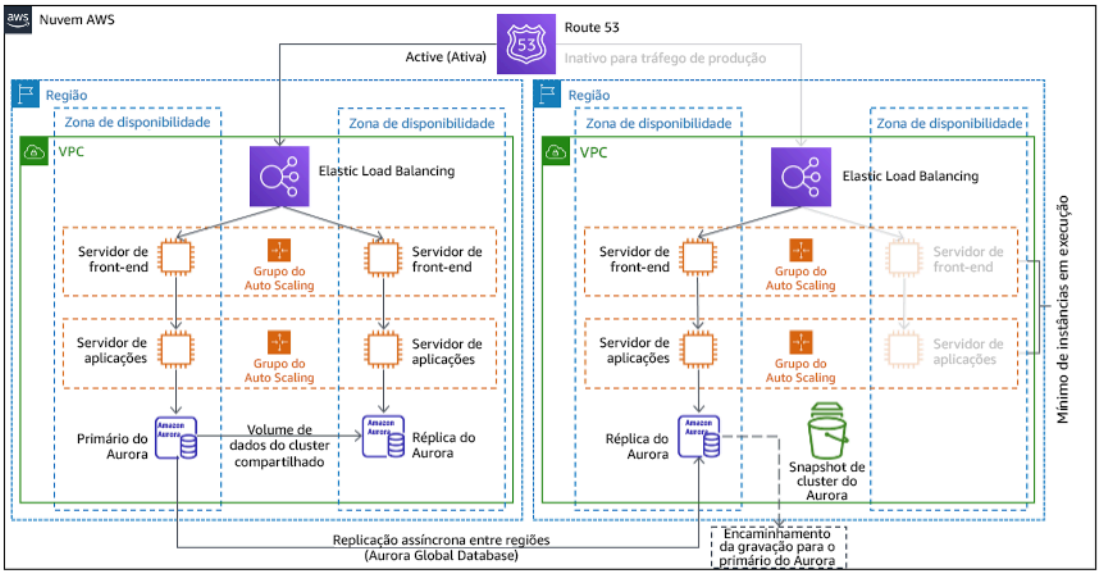


Figura 21: Arquitetura de standby passivo

O uso de standby passivo ou luz-piloto requer que a escala dos recursos seja aumentada verticalmente na região de recuperação. Para verificar se a capacidade está disponível quando necessário, considere o uso de reservas de capacidade para instâncias do EC2. Se você usar o AWS Lambda, a simultaneidade provisionada poderá fornecer ambientes de execução para que eles sejam preparados para responder imediatamente às invocações da função.

Para obter mais detalhes sobre essa estratégia, consulte Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby (Arquitetura de recuperação de desastres (DR) na AWS, parte III: luz-piloto e standby passivo).

Multissite ativo/ativo

É possível executar a workload simultaneamente em várias regiões como parte de uma estratégia de multissite ativo/ativo. O multissite ativo-ativo atende ao tráfego de todas as regiões onde está implantado. Os clientes podem selecionar essa estratégia por outros motivos, além da DR. Ela pode ser usada para aumentar a disponibilidade ou ao implantar uma workload para um público global (a fim de aproximar o endpoint dos usuários e/ou implantar pilhas localizadas para o público nessa região). Como uma estratégia de DR, se a workload não for compatível com uma das Regiões da AWS onde está implantada, essa região será evacuada e as regiões restantes serão usadas para manter a disponibilidade. O multissite ativo-ativo é a estratégia de DR mais complexa operacionalmente e deve ser selecionada apenas quando os requisitos empresariais exigirem.

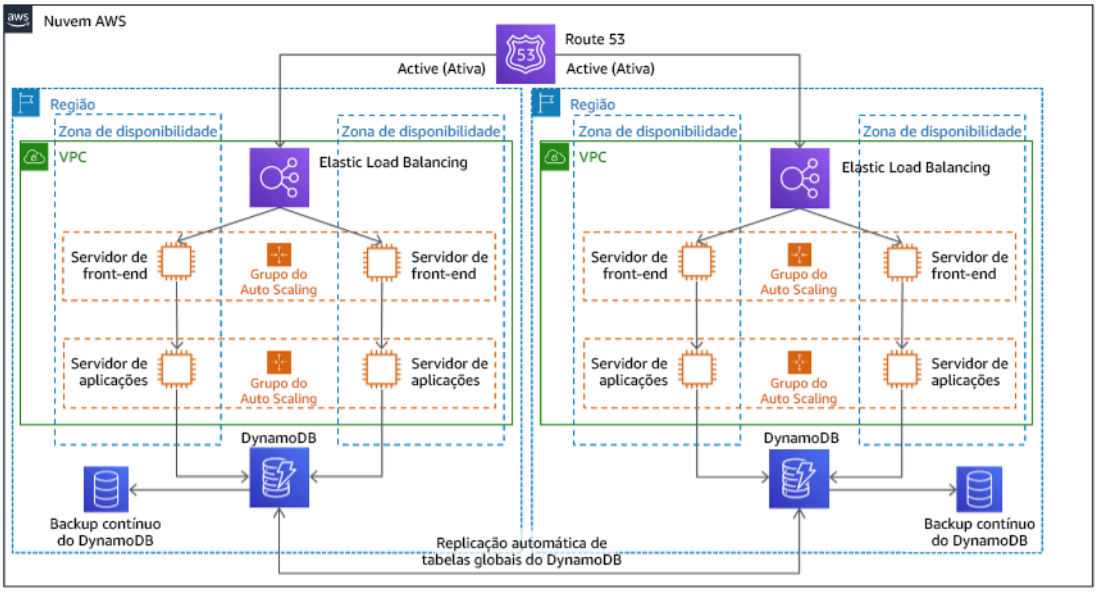


Figura 22: Arquitetura de multissite ativo-ativo

Para obter mais detalhes sobre essa estratégia, consulte Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active (Arquitetura de recuperação de desastres (DR) na AWS, parte IV: multissite ativo/ativo).

AWS Elastic Disaster Recovery

Se você estiver considerando a estratégia de luz-piloto ou de standby passivo para a recuperação de desastres, o AWS Elastic Disaster Recovery poderia fornecer uma abordagem alternativa com benefícios melhorados. O Elastic Disaster Recovery pode oferecer uma meta de RPO e RTO semelhante à do standby passivo, mas mantém a abordagem de baixo custo da luz-piloto. O Elastic Disaster Recovery replica os dados da região primária para a região de recuperação, usando a proteção contínua de dados para alcançar um RPO medido em segundos e um RTO que pode ser medido em minutos. Somente os recursos necessários para replicar os dados são implantados na região de recuperação, o que mantém os custos baixos, semelhante à estratégia de luz-piloto. Ao usar o Elastic Disaster Recovery, o serviço coordena e orquestra a recuperação de recursos de computação quando iniciado como parte de um failover ou de uma simulação.

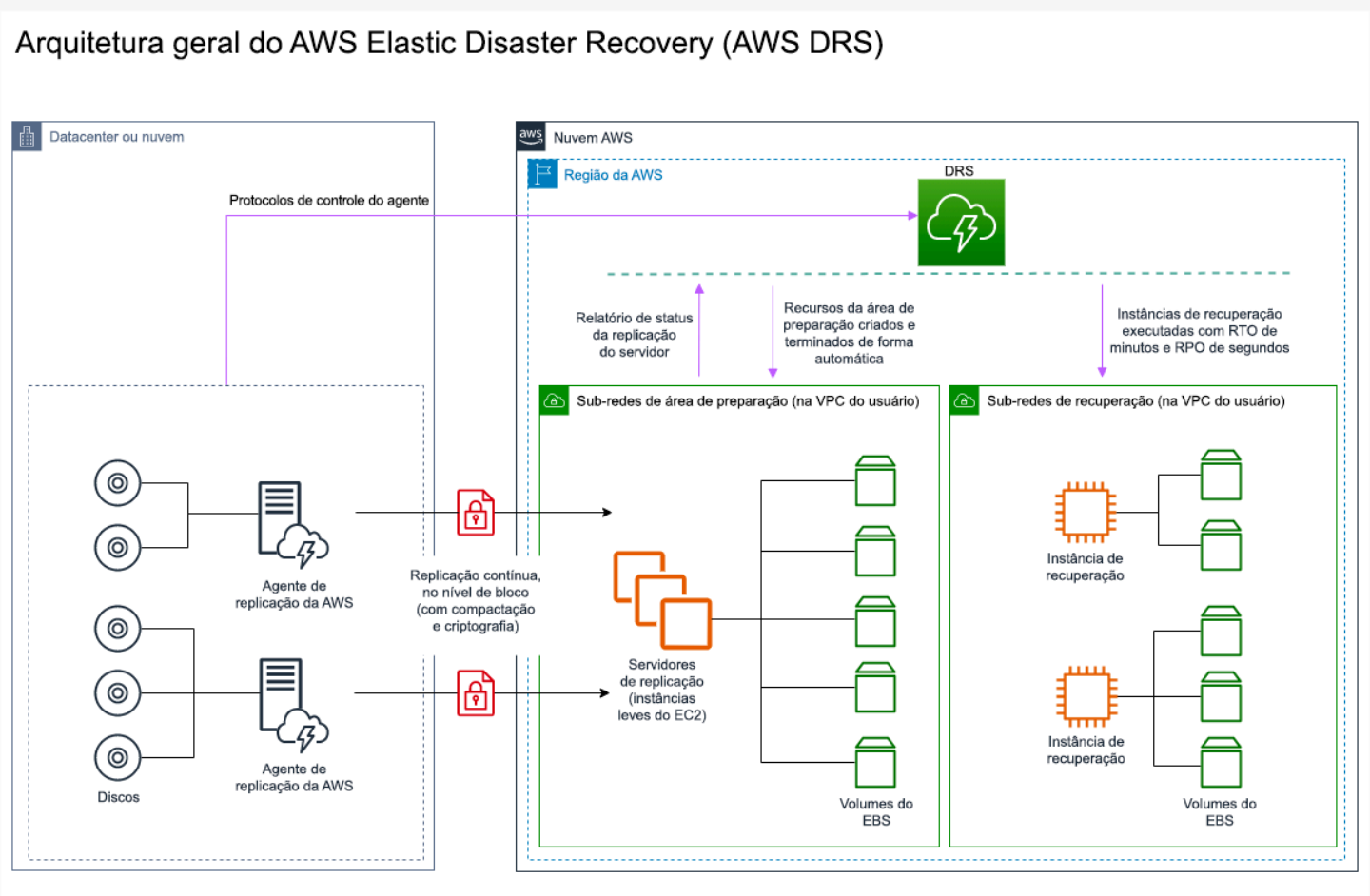


Figura 23: Arquitetura do AWS Elastic Disaster Recovery

Práticas adicionais para proteção de dados

Com todas as estratégias, você também deve atenuar um desastre de dados. A replicação contínua de dados protege você contra alguns tipos de desastre, mas não contra corrupção ou destruição de dados, a menos que sua solução também inclua o versionamento de dados armazenados ou opções para recuperação a um ponto anterior no tempo. Você também deve fazer backup dos dados replicados no local de recuperação para criar backups pontuais além das réplicas.

O uso de várias zonas de disponibilidade (AZs) em uma única Região da AWS

Ao utilizar várias AZs em uma única região, a implementação de DR usa vários elementos das estratégias acima. Primeiro, você deve criar uma arquitetura de alta disponibilidade (HA), usando várias AZs, conforme mostrado na Figura 23. Essa arquitetura usa uma abordagem multissite ativo/ativo, já que as instâncias do Amazon EC2 e o Elastic Load Balancer tem recursos implantados em várias AZs, processando solicitações ativamente. A arquitetura também demonstra o standby a

quente, no qual, caso a instância primária do Amazon RDS falhe (ou a própria AZ falhe), a instância de standby será promovida a primária.

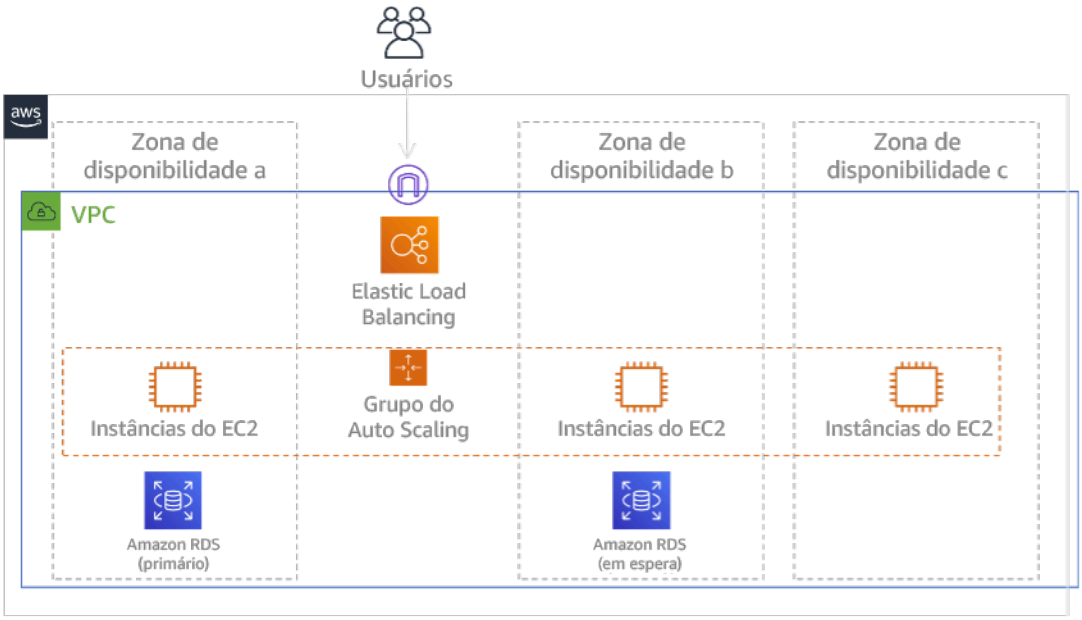


Figura 24: Arquitetura Multi-AZ

Além da arquitetura de alta disponibilidade, você precisa adicionar backups de todos os dados necessários para executar a workload. Isso é especialmente importante para dados restritos a uma única zona, como volumes do Amazon EBS ou clusters do Amazon Redshift. Se uma AZ falhar, você precisará restaurar esses dados para outra AZ. Sempre que possível, você também deve copiar backups de dados para outra Região da AWS, como uma camada adicional de proteção.

Uma abordagem alternativa menos comum para uma única região, a DR Multi-AZ está ilustrada nesta publicação de blog, Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack (Criar aplicações altamente resilientes usando o Amazon Route 53 Application Recovery Controller, parte 1: pilha de região única). Aqui, a estratégia é manter o máximo de isolamento possível entre as AZs, assim como as regiões operam. Ao usar esta estratégia alternativa, você pode escolher uma abordagem ativa/ativa ou ativa/passiva.

> **Note**
>
> Algumas workloads têm requisitos regulamentares de residência de dados. Se isso se aplicar à sua workload em uma localidade que atualmente tem apenas uma Região da AWS, a multirregião não atenderá às suas necessidades empresariais. As estratégias Multi-AZ fornecem boa proteção contra a maioria dos desastres.

3. Avalie os recursos da workload e qual será sua configuração na região de recuperação antes do failover (durante a operação normal).

Para recursos de infraestrutura e da AWS, use a infraestrutura como código, como o AWS CloudFormation, ou ferramentas de terceiros, como o Hashicorp Terraform. Para implantar em várias contas e regiões com uma única operação, você pode usar o AWS CloudFormation StackSets. Para estratégias multissite ativo-ativo e standby a quente, a infraestrutura implantada na região de recuperação tem os mesmos recursos que a região primária. Para as estratégias de luz-piloto e standby passivo, a infraestrutura implantada exigirá ações adicionais para ficar pronta para produção. Usando parâmetros e lógica condicional do CloudFormation, é possível controlar se uma pilha implantada está ativa ou em espera com um único modelo. Ao usar o Elastic Disaster Recovery, o serviço vai replicar e orquestrar a restauração de configurações da aplicação e os recursos de computação.

Todas as estratégias de DR exigem que seja feito backup das fontes de dados na Região da AWS e, então, esses backups são copiados para a região de recuperação. O AWS Backup fornece uma visão centralizada em que é possível configurar, programar e monitorar backups para esses recursos. Para luz-piloto, standby passivo e multissite ativo-ativo, você também deve replicar dados da região primária para recursos de dados na região de recuperação, como instâncias de banco de dados do Amazon Relational Database Service (Amazon RDS) ou tabelas do Amazon DynamoDB. Esses recursos de dados estão ativos e prontos para atender a solicitações na região de recuperação.

Para saber mais sobre como os serviços da AWS operam entre regiões, consulte essa série de blog em Creating a Multi-Region Application with AWS Services (Criar uma aplicação de várias regiões com os serviços da AWS).

4. Determine e implemente como deixar sua região de recuperação pronta para failover quando necessário (durante um evento de desastre).

Para multissite ativo-ativo, failover significa evacuar uma região e confiar nas regiões ativas restantes. No geral, essas regiões estão prontas para aceitar tráfego. Para as estratégias de luz-piloto e standby passivo, as ações de recuperação precisarão implantar os recursos ausentes, como as instâncias do EC2 na Figura 20, além de quaisquer outros recursos ausentes.

Para todas as estratégias acima, pode ser necessário promover instâncias somente leitura de bancos de dados para se tornar a instância primária de leitura/gravação.

Para backup e restauração, a restauração de dados do backup cria recursos para esses dados, como volumes do EBS, instâncias de banco de dados do RDS e tabelas do DynamoDB. Você também precisa restaurar a infraestrutura e implantar o código. É possível usar o AWS Backup para restaurar dados na região de recuperação. Perceber REL09-BP01 Identificar e fazer backup de todos os dados que precisam de backup ou reproduzir os dados das fontes para obter mais detalhes. A reconstrução da infraestrutura inclui a criação de recursos como instâncias do EC2, além da Amazon Virtual Private Cloud (Amazon VPC), de sub-redes e dos grupos de segurança necessários. É possível automatizar grande parte do processo de restauração. Para saber como, consulte esta publicação do blog.

5. Determine e implemente como redirecionar o tráfego para failover quando necessário (durante um evento de desastre).

Essa operação de failover pode ser iniciada de forma manual ou automática. O failover iniciado automaticamente com base em verificações de integridade ou alarmes deve ser usado com cautela, pois um failover desnecessário (alarme falso) resulta em custos como indisponibilidade e perda de dados. Portanto, geralmente é usado o failover iniciado manualmente. Nesse caso, você ainda deve automatizar as etapas para failover, para que a inicialização manual seja como apertar um botão.

Há várias opções de gerenciamento de tráfego a serem consideradas ao usar os serviços da Regiões da AWS. Uma opção é usar o Amazon Route 53. Ao usar o Amazon Route 53, você pode associar vários endpoints de IP em uma ou mais Regiões da AWS a um nome de domínio do Route 53. Para implementar um failover iniciado manualmente, é possível usar o Amazon Route 53 Application Recovery Controller, o que fornece uma API de plano de dados altamente disponível para rotear novamente o tráfego para a região de recuperação. Ao implementar o failover, use as operações do plano de dados e evite as do ambiente de gerenciamento, conforme descrito em REL11-BP04 Confiar no plano de dados e não no ambiente de gerenciamento durante a recuperação.

Para saber mais sobre essa e outras opções, consulte esta seção do whitepaper de recuperação de desastres.

6. Projete um plano de como será feito failback da workload.

Failback é quando você retorna a operação de workload para a região primária, após o término de um evento de desastre. O provisionamento de infraestrutura e código para a região primária geralmente segue as mesmas etapas que foram usadas inicialmente, contando com a infraestrutura

como código e pipelines de implantação de código. O desafio com o failback é restaurar os armazenamentos de dados e garantir sua consistência com a região de recuperação em operação.

No estado de failover, os bancos de dados na região de recuperação estão ativos e têm dados atualizados. O objetivo é ressincronizar da região de recuperação para a região primária, garantindo que ela esteja atualizada.

Alguns serviços da AWS farão isso automaticamente. Se você usar tabelas globais do Amazon DynamoDB, mesmo que a tabela na região primária tenha ficado indisponível, quando ela voltar a ficar online, o DynamoDB retomará a propagação das gravações pendentes. Se você usar o banco de dados global do Amazon Aurora e o failover planejado e gerenciado, a topologia da replicação existente do banco de dados global do Aurora será mantida. Portanto, a antiga instância de leitura/gravação na região primária se tornará uma réplica e receberá atualizações da região de recuperação.

Em casos nos quais isso não seja automático, você precisará restabelecer o banco de dados na região primária como uma réplica do banco de dados na região de recuperação. Em muitos casos, isso envolverá a exclusão do banco de dados primário antigo e a criação de outras réplicas. Por exemplo, para obter instruções de como fazer isso com o banco de dados global do Amazon Aurora presumindo um failover não planejado, consulte este laboratório: Fail Back a Global Database (Executar failback em um banco de dados global).

Após um failover, se você puder continuar a execução na região de recuperação, considere torná-la a nova região primária. Você ainda seguiria todas as etapas acima para transformar a antiga região primária em uma região de recuperação. Algumas organizações fazem uma alternância programada, trocando as regiões primárias e de recuperação periodicamente (por exemplo, a cada três meses).

Todas as etapas necessárias para failover e failback devem ser mantidas em um manual disponível para todos os membros da equipe e que seja revisado periodicamente.

Ao usar o Elastic Disaster Recovery, o serviço auxiliará na orquestração e automatização do processo de failback. Para obter mais detalhes, consulte Performing a failback (Como executar failback).

Nível de esforço do plano de implementação: alto

Recursos

Práticas recomendadas relacionadas:

- the section called “REL09-BP01 Identificar e fazer backup de todos os dados que precisam de backup ou reproduzir os dados das fontes”
- the section called “REL11-BP04 Confiar no plano de dados e não no ambiente de gerenciamento durante a recuperação”
- the section called “REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados”

Documentos relacionados:

- Blog de arquitetura da AWS: série de recuperação de desastres
- Recuperação de desastres de workloads na AWS: recuperação na nuvem (whitepaper da AWS)
- Opções de recuperação de desastres na nuvem
- Crie uma solução de backend ativo-ativo multirregional sem servidor em uma hora
- Backend multirregional sem servidor: recarregado
- RDS: replicação de uma réplica de leitura entre regiões
- Route 53: configuração do failover de DNS
- S3: replicação entre regiões
- O que é o AWS Backup?
- O que é o Route 53 Application Recovery Controller?
- AWS Elastic Disaster Recovery
- (HashiCorp Terraform: conceitos básicos – AWS
- Parceiro do APN: parceiros que podem ajudar com a recuperação de desastres
- AWS Marketplace: produtos que podem ser usados para recuperação de desastres

Vídeos relacionados:

- Recuperação de desastres de workloads na AWS
- AWS re:Invent 2018: padrões de arquitetura para aplicações ativas/ativas de várias regiões (ARC209-R2)
- Conceitos básicos do AWS Elastic Disaster Recovery | Amazon Web Services

Exemplos relacionados:

- Well-Architected Lab - Disaster Recovery - Series of workshops illustrating DR strategies (Laboratório do Well-Architected – Recuperação de desastres: série de workshops ilustrando estratégias de DR)

REL13-BP03 Testar a implementação da recuperação de desastres para validá-la

Teste regularmente o failover no site de recuperação para verificar se a operação está correta e que o RTO e o RPO sejam cumpridos.

Antipadrões comuns:

- Nunca execute failovers na produção.

Benefícios do estabelecimento dessa prática recomendada: testar regularmente seu plano de recuperação de desastres garante que ele funcione quando necessário e que sua equipe saiba como executar a estratégia.

Nível de exposição a riscos quando esta prática recomendada não é estabelecida: alto

Orientações para a implementação

Um padrão que deve ser evitado é o desenvolvimento de caminhos de recuperação que raramente são executados. Por exemplo, você pode ter um repositório de dados secundário utilizado para consultas somente leitura. Quando você grava em um repositório de dados e o repositório de dados primário falha, pode ser necessário fazer o failover para o repositório de dados secundário. Se você não testar esse failover com frequência, poderá descobrir que suas suposições sobre as capacidades do armazenamento de dados secundário são incorretas. A capacidade do secundário, que talvez tenha sido suficiente quando testado pela última vez, pode não conseguir mais tolerar a carga nesse cenário. Nossa experiência mostrou que a única recuperação de erro que funciona é o caminho que você testa com frequência. É por isso que é melhor ter um pequeno número de caminhos de recuperação. Você pode estabelecer padrões de recuperação e testá-los regularmente. Se você tiver um caminho de recuperação complexo ou crítico, ainda precisará executar regularmente essa falha na produção para garantir o funcionamento desse caminho. No exemplo que acabamos de discutir, você deve realizar o failover para o standby regularmente, não importa a necessidade.

Etapas da implementação

1. Projete suas cargas de trabalho para recuperação. Teste regularmente seus caminhos de recuperação. A computação orientada para a recuperação identifica as características em sistemas que aprimoram a recuperação: isolamento e redundância, capacidade de reverter alterações em todo o sistema, capacidade de monitorar e determinar a integridade, capacidade de realizar diagnósticos, recuperação automatizada, design modular e capacidade de reinicialização. Pratique o caminho da recuperação para verificar se é possível realizá-la no tempo especificado para o estado determinado. Use seus runbooks durante essa recuperação para documentar problemas e encontrar soluções para eles antes do próximo teste.
2. Para workloads com base no Amazon EC2, use o AWS Elastic Disaster Recovery para implementar e iniciar instâncias de simulação para a estratégia de DR. O AWS Elastic Disaster Recovery permite executar simulações com eficiência, o que ajuda você a se preparar para um evento de failover. Também é possível iniciar frequentemente as instâncias usando o Elastic Disaster Recovery para fins de teste e simulação sem redirecionar o tráfego.

Recursos

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar com a recuperação de desastres
- Blog de arquitetura da AWS: série de recuperação de desastres
- AWS Marketplace: produtos que podem ser usados para recuperação de desastres
- AWS Elastic Disaster Recovery
- Recuperação de desastres de workloads na AWS: recuperação na nuvem (whitepaper da AWS)
- AWS Elastic Disaster Recovery Preparing for Failover (AWS Elastic Disaster Recovery: preparação para failover)
- O projeto de computação orientado por recuperação de Berkeley/Stanford
- What is AWS Fault Injection Simulator? (O que e o AWS Fault Injection Simulator?)

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (AWS re:Invent 2018: padrões de arquitetura para aplicações ativas/ativas de várias regiões)
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS (AWS re:Invent 2019: soluções de backup e restauração e de recuperação de desastres com a AWS)

Exemplos relacionados:

- Well-Architected Lab - Testing for Resiliency (Laboratório do Well-Architected: teste de resiliência)

REL13-BP04 Gerenciar o desvio de configuração para o local ou a região de DR

Certifique-se de que a infraestrutura, os dados e a configuração estejam conforme necessário no local ou na região de DR. Por exemplo, verifique se as AMIs e as cotas de serviço estão atualizadas.

O AWS Config monitora e registra continuamente as configurações dos recursos da AWS. Ele pode detectar desvios e acionar o AWS Systems Manager Automation para corrigi-lo e gerar alarmes. O AWS CloudFormation também pode detectar desvios nas pilhas que você implantou.

Antipadrões comuns:

- Falhar ao atualizar os locais de recuperação, ao fazer alterações de configuração ou infraestrutura nos locais primários.
- Não considerar possíveis limitações (como diferenças de serviço) nos locais primários e de recuperação.

Benefícios do estabelecimento desta prática recomendada: Garantir que o ambiente de DR seja consistente com seu ambiente existente para assegurar a recuperação completa.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- Garanta que seus pipelines de entrega enviem para seus locais primário e de backup. Os pipelines de entrega para implantação de aplicativos em produção devem ser distribuídos para todos os locais de estratégia de recuperação de desastres especificados, incluindo os ambientes de desenvolvimento e de teste.
- Habilite o AWS Config para acompanhar possíveis locais de desvio. Use as regras do AWS Config para criar sistemas que aplicam suas estratégias de recuperação de desastres e geram alertas ao detectar desvios.
  - Correção de recursos não compatíveis do Regras do AWS Config pela AWS
  - AWS Systems Manager Automation

- Use o AWS CloudFormation para implantar a infraestrutura. O AWS CloudFormation pode detectar desvios entre as especificações dos modelos do CloudFormation e o que é realmente implantado.
  - AWS CloudFormation: detectar desvios em uma pilha inteira do CloudFormation

Recursos

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar com a recuperação de desastres
- Blog de arquitetura da AWS: série de recuperação de desastres
- AWS CloudFormation: detectar desvios em uma pilha inteira do CloudFormation
- AWS Marketplace: produtos que podem ser usados para recuperação de desastres
- AWS Systems Manager Automation
- Recuperação de desastres de workloads na AWS: recuperação na nuvem (whitepaper da AWS)
- Como faço para implementar uma solução de gerenciamento de configuração de infraestrutura na AWS?
- Correção de recursos não compatíveis do Regras do AWS Config pela AWS

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

REL13-BP05 Automatizar a recuperação

Use ferramentas da AWS ou de terceiros para automatizar a recuperação do sistema e rotear o tráfego para o local ou a região de DR.

Com base em verificações de integridade configuradas, os serviços da AWS, como o Elastic Load Balancing e o AWS Auto Scaling, podem distribuir a carga para zonas de disponibilidade íntegras, enquanto outros serviços, como o Amazon Route 53 e o AWS Global Accelerator, podem rotear a carga para Regiões da AWS íntegras. O Amazon Route 53 Application Recovery Controller ajuda a gerenciar e coordenar o failover usando verificações de prontidão e recursos de controle de roteamento. Esses recursos monitoram continuamente a capacidade da aplicação de se recuperar

de falhas, permitindo que você controle a recuperação da aplicação em várias Regiões da AWS, zonas de disponibilidade e ambientes on-premises.

Para workloads em datacenters físicos ou virtuais existentes ou nuvens privadas, o AWS Elastic Disaster Recovery, disponível por meio do AWS Marketplace, permite que as organizações configurem uma estratégia automatizada de recuperação de desastres para a AWS. O CloudEndure também oferece suporte à recuperação de desastres entre regiões e entre AZs na AWS.

Antipadrões comuns:

- A implementação de failover e failback automatizados idênticos pode causar oscilação quando uma falha ocorre.

Benefícios do estabelecimento dessa prática recomendada: A recuperação automatizada reduz o tempo de recuperação ao eliminar a oportunidade de erros manuais.

Nível de exposição a riscos quando esta prática recomendada não for estabelecida: Médio

Orientações para a implementação

- Automatize caminhos de recuperação. No caso de tempos de recuperação curtos, não é possível adotar critério e ação humanos em cenários de alta disponibilidade. O sistema deve recuperar-se automaticamente sob qualquer situação.
  - Use o CloudEndure Disaster Recovery para automatizar failover e failback. Ele replica continuamente suas máquinas (incluindo sistema operacional, configuração de estado do sistema, bancos de dados, aplicações e arquivos) em uma área de preparação de baixo custo na Conta da AWS de destino e na região de preferência. Em caso de desastre, você pode instruir o CloudEndure Disaster Recovery a executar automaticamente milhares de máquinas em seu estado totalmente provisionado em minutos.
    - Realizar um failover e failback de recuperação de desastres
    - CloudEndure Disaster Recovery

Recursos

Documentos relacionados:

- Parceiro do APN: parceiros que podem ajudar com a recuperação de desastres
- Blog de arquitetura da AWS: série de recuperação de desastres

- AWS Marketplace: produtos que podem ser usados para recuperação de desastres
- AWS Systems Manager Automation
- CloudEndure Disaster Recovery para AWS
- Recuperação de desastres de workloads na AWS: recuperação na nuvem (whitepaper da AWS)

Vídeos relacionados:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

Exemplos de implementações de metas de disponibilidade

Nesta seção, vamos analisar designs da carga de trabalho usando a implantação de um aplicativo web típico que consiste em um proxy reverso, conteúdo estático no Amazon S3, um servidor de aplicativo e um banco de dados SQL para o armazenamento persistente de dados. Para cada objetivo de disponibilidade, mostramos um exemplo de implementação. Essa carga de trabalho poderia usar contêineres ou o AWS Lambda para computação e NoSQL (como o Amazon DynamoDB) para o banco de dados, mas as abordagens são semelhantes. Em cada cenário, demonstramos como atender às metas de disponibilidade por meio do design da carga de trabalho para estes tópicos:

Tópico	Para obter mais informações, consulte esta seção
Monitorar recursos	Monitorar os recursos da workload
Adaptar-se às mudanças de demanda	Projetar a workload de modo que ela se adapte às alterações na demanda
Implementar alterações	Implementar alterações
Fazer o backup de dados	Fazer o backup de dados
Arquitetar para resiliência	Use o isolamento de falhas para proteger a carga de trabalho Projete a workload para resistir às falhas de componentes
Testar a resiliência	Testar a confiabilidade
Planejar para a recuperação de desastres (DR)	Planejar para a recuperação de desastres (DR)

Seleção de dependência

Escolhemos usar o Amazon EC2 para nossos aplicativos. Vamos mostrar como usar o Amazon RDS e várias zonas de disponibilidade melhora a disponibilidade de nossos aplicativos. Usaremos o

Amazon Route 53 para DNS. Quando usarmos várias Zonas de disponibilidade, usaremos o Elastic Load Balancing. O Amazon S3 é usado para backups e conteúdo estático. Uma vez que projetamos para maior confiabilidade, devemos usar serviços que tenham uma disponibilidade maior.

Cenários de região única

Tópicos

Cenário de dois noves (99%)

Essas cargas de trabalho são úteis para a empresa, mas isso só é um inconveniente se elas estiverem indisponíveis. Esse tipo de carga de trabalho pode ser ferramentas internas, gerenciamento de conhecimento interno ou rastreamento de projetos. Ou podem ser cargas de trabalho reais voltadas ao cliente, mas servidas de um serviço experimental, com um recurso de alternância que pode ocultar o serviço, se necessário.

Essas cargas de trabalho podem ser implantadas em uma Região e em uma zona de disponibilidade.

Monitorar recursos

Teremos monitoramento simples, indicando se a página inicial do serviço está retornando um status HTTP 200 OK. Quando ocorrem problemas, nosso playbook indica que o registro da instância pode ser usado para estabelecer a causa raiz.

Adaptar-se às mudanças de demanda

Teremos manuais para falhas de hardware comuns, atualizações de software urgentes e outras alterações que causam interrupção.

Implementar alterações

Usaremos o AWS CloudFormation para definir nossa infraestrutura como código e especificamente para acelerar a reconstrução em caso de falha.

As atualizações de software são realizadas manualmente usando um runbook, sendo necessário tempo de inatividade para a instalação e o reinício do serviço. No caso de um problema durante a implantação, o runbook descreve como reverter para a versão anterior.

Todas as correções do erro são feitas usando a análise de logs pelas equipes de operações e desenvolvimento, e a correção é implantada depois que a correção é priorizada e concluída.

Fazer o backup de dados

Usaremos uma solução de backup especialmente criada para enviar dados de backup criptografados para o Amazon S3 usando um runbook. Testaremos se os backups funcionam restaurando os dados e garantindo a disponibilidade de usá-los regularmente empregando um runbook. Configuramos o controle de versões de nossos objetos do Amazon S3 e removemos permissões de exclusão de backups. Usamos uma política de ciclo de vida de bucket do Amazon S3 para arquivar ou excluir em caráter permanente conforme nossos requisitos.

Arquitetar para resiliência

As cargas de trabalho são implantadas com uma região e uma zona de disponibilidade. Implantamos o aplicativo, incluindo o banco de dados, em uma única instância.

Testar a resiliência

O pipeline de implantação do novo software é programado, com algum teste de unidade, mas principalmente teste de caixa branca/caixa preta da carga de trabalho montada.

Planejar para a recuperação de desastres (DR)

Durante falhas, aguardamos a falha terminar, opcionalmente roteando as solicitações a um site estático usando modificação de DNS por meio de um runbook. O tempo de recuperação para isso será determinado pela velocidade em que a infraestrutura pode ser implantada e o banco de dados restaurado para o backup mais recente. Essa implantação pode ser na mesma zona de disponibilidade ou em uma zona de disponibilidade diferente no caso de uma falha da zona de disponibilidade usando um runbook.

Meta de design de disponibilidade

Levamos 30 minutos para entender e decidir executar a recuperação, implantamos toda a pilha no AWS CloudFormation em 10 minutos. Vamos supor que implantamos em uma nova zona de disponibilidade e o banco de dados pode ser restaurado em 30 minutos. Isso implica cerca de

70 minutos para a recuperação de uma falha. Supondo uma falha por trimestre, nosso tempo de impacto estimado para o ano é de 280 minutos, ou quatro horas e 40 minutos.

Isso significa que o limite superior de disponibilidade é de 99,9%. A disponibilidade geral depende também da taxa real de falha, da duração da falha e da rapidez com que cada falha se recupera realmente. Para essa arquitetura, precisamos que o aplicativo fique offline para atualizações (estimando 24 horas por ano: quatro horas por alteração, seis vezes por ano) mais eventos reais. Assim, consultando a tabela sobre disponibilidade de aplicativo anterior no whitepaper, vemos que nossa meta de design de disponibilidade é 99%.

Resumo

Tópico	Implementação
Monitorar recursos	Somente verificação de integridade do site; sem alertas.
Adaptar-se às mudanças de demanda	Escalabilidade vertical por meio de nova implantação.
Implementar alterações	Runbook para implantar e reverter.
Fazer o backup de dados	Runbook para backup e restauração.
Arquitetar para resiliência	Recompilação completa, restaurar de um backup.
Testar a resiliência	Recompilação completa, restaurar de um backup.
Planejar para a recuperação de desastres (DR)	Backups criptografados são restaurados para uma zona de disponibilidade diferente, se necessário.

Cenário de três noves (99,9%)

A próxima meta de disponibilidade refere-se a aplicativos para os quais a alta disponibilidade é importante, mas que podem tolerar curtos períodos de indisponibilidade. Esse tipo de carga de trabalho costuma ser usado para operações internas que, quando estão inativas, afetam os

funcionários. Esse tipo de carga de trabalho também pode ser voltada para o cliente, mas não é de alta receita para os negócios e pode tolerar um tempo de recuperação ou ponto de recuperação maior. Essas cargas de trabalho incluem aplicativos administrativos para gerenciamento de conta ou informações.

Podemos melhorar a disponibilidade de cargas de trabalho usando duas zonas de disponibilidade para nossa implantação e separando os aplicativos em níveis distintos.

Monitorar recursos

O monitoramento será expandido para alertar sobre a disponibilidade do site geral verificando um status HTTP 200 OK na página inicial. Além disso, haverá alertas em cada substituição de um servidor da web e quando o banco de dados realizar o failover. Também monitoraremos o conteúdo estático no Amazon S3 para disponibilidade e alerta se ele ficar indisponível. O registro será agregado para facilitar o gerenciamento e ajudar na análise de causa raiz.

Adaptar-se às mudanças de demanda

A escalabilidade automática é configurada para monitorar a utilização da CPU em instâncias do EC2 e adicionar ou remover instâncias para manter o destino da CPU em 70%, mas com pelo menos uma instância do EC2 por zona de disponibilidade. Se os padrões de carga em nossa instância do RDS indicarem que a escala é necessária, alteraremos o tipo de instância durante uma janela de manutenção.

Implementar alterações

As tecnologias de implantação de infraestrutura permanecem as mesmas do cenário anterior.

A entrega de novo software ocorre conforme um cronograma fixo a cada duas a quatro semanas. As atualizações de software serão automatizadas, não usando padrões de implantação canário ou azul/verde, mas usando a substituição no local. A decisão de reverter será tomada usando o runbook.

Teremos playbooks para estabelecer a causa raiz dos problemas. Depois da identificação da causa raiz, a correção do erro será identificada por uma combinação das equipes de operações e desenvolvimento. A correção será implantada depois que for desenvolvida.

Fazer o backup de dados

O backup e a restauração podem ser feitos usando o Amazon RDS. A operação será executada regularmente usando um runbook para garantir que os requisitos de recuperação possam ser cumpridos.

Arquitetar para resiliência

Podemos melhorar a disponibilidade de aplicativos usando duas zonas de disponibilidade para nossa implantação e separando os aplicativos em níveis distintos. Usaremos serviços que funcionam em várias Zonas de disponibilidade, como o Elastic Load Balancing, oAuto Scaling e o Amazon RDS Multi-AZ com armazenamento criptografado por meio do AWS Key Management Service. Isso garantirá a tolerância a falhas no nível do recurso e no nível da zona de disponibilidade.

O load balancer apenas roteará o tráfego para instâncias de aplicativo íntegras. A verificação de integridade precisa estar na camada de aplicativo/plano de dados indicando a capacidade do aplicativo na instância. Essa verificação não deve ser contra o plano de controle. Uma URL de verificação de integridade para o aplicativo web estará presente e configurada para uso pelo load balancer e pelo Auto Scaling, para que as instâncias que falharem sejam removidas e substituídas. O Amazon RDS gerenciará o mecanismo de banco de dados ativo para estar disponível na segunda Zona de disponibilidade se a instância falhar na Zona de disponibilidade primária e, em seguida, fará reparos para restaurar na mesma resiliência.

Depois de termos separado os níveis, podemos usar padrões de resiliência de sistema distribuído para aumentar a confiabilidade do aplicativo de modo que ele ainda possa estar disponível mesmo quando o banco de dados estiver temporariamente indisponível durante um failover da Zona de disponibilidade.

Testar a resiliência

Fazemos testes funcionais, da mesma forma que no cenário anterior. Não testamos os recursos de recuperação automática do ELB, da escalabilidade automática ou do failover do RDS.

Teremos playbooks para problemas de banco de dados comuns, incidentes relacionados à segurança e implantações com falha.

Planejar para a recuperação de desastres (DR)

Existem runbooks para recuperação total da carga de trabalho e emissão de relatórios comuns. A recuperação usa backups armazenados na mesma região que a carga de trabalho.

Meta de design de disponibilidade

Presumimos que pelo menos algumas falhas exigirão uma decisão manual para executar a recuperação. Porém, com maior automação nesse cenário, presumimos que apenas dois eventos

por ano exigirão essa decisão. Levamos 30 minutos para decidir executar a recuperação e presumimos que a recuperação estará concluída dentro de 30 minutos. Isso implica 60 minutos para a recuperação da falha. Presumindo dois incidentes por ano, nosso tempo de impacto estimado para o ano é de 120 minutos.

Isso significa que o limite superior de disponibilidade é de 99,95%. A disponibilidade geral depende também da taxa real de falha, da duração da falha e da rapidez com que cada falha se recupera realmente. Para essa arquitetura, precisamos que o aplicativo fique brevemente online para atualizações, mas essas atualizações são automatizadas. Estimamos 150 minutos por ano para isso: 15 minutos por alteração, 10 vezes por ano. Isso adiciona até 270 minutos por ano quando o serviço não está disponível, assim, nossa meta de design de disponibilidade é 99,9%.

Resumo

Tópico	Implementação
Monitorar recursos	Somente verificação de integridade do site; alertas enviados quando inativo.
Adaptar-se às mudanças de demanda	ELB para a camada de aplicativo web e escalabilidade automática, redimensionamento de RDS Multi-AZ.
Implementar alterações	Implantação automatizada em vigor e runbook para reversão.
Fazer o backup de dados	Backups automatizados por meio de RDS para cumprir o RPO e runbook para restauração.
Arquitetar para resiliência	Escalabilidade automática para fornecer um nível de aplicativo e web com autorrecu peração; RDS é Multi-AZ.
Testar a resiliência	ELB e aplicativo têm autorrecuperação; RDS é Multi-AZ; não há teste explícito.
Planejar para a recuperação de desastres (DR)	Backups criptografados por meio do RDS para a mesma região da AWS.

Cenário de quatro noves (99,99%)

Essa meta de disponibilidade exige que o aplicativo esteja altamente disponível e seja tolerante a falhas de componentes. O aplicativo deve poder absorver falhas sem a necessidade de obter mais recursos. Essa meta de disponibilidade é para aplicativos de missão crítica que são essenciais ou importantes para a geração de receitas para uma corporação, como um site de comércio eletrônico, um web service business to business ou um site de mídia/conteúdo de alto volume de tráfego.

Podemos melhorar ainda mais a disponibilidade usando uma arquitetura que será estaticamente estável dentro da Região. Essa meta de disponibilidade não exige uma alteração no plano de controle no comportamento de nossa carga de trabalho para tolerar a falha. Por exemplo, deve haver capacidade suficiente para suportar a perda de uma zona de disponibilidade. Não devemos exigir atualizações no DNS do Amazon Route 53. Não deve ser preciso criar nenhuma infraestrutura nova, seja criando ou modificando um bucket do S3, criando novas políticas do IAM (ou modificações de políticas) ou modificando configurações de tarefa do Amazon ECS.

Monitorar recursos

O monitoramento incluirá métricas de sucesso, bem como alertas quando ocorrem problemas. Além disso, haverá alertas em cada substituição de um servidor da web com falha, quando o banco de dados realizar o failover ou quando uma AZ falhar.

Adaptar-se às mudanças de demanda

Usaremos o Amazon Aurora como nosso RDS, o que permite a escalabilidade automática de réplicas de leitura. Para esses aplicativos, a engenharia de disponibilidade de leitura em vez da disponibilidade de gravação de conteúdo principal também é uma decisão de arquitetura importante. O Aurora também pode aumentar automaticamente o armazenamento conforme necessário, em incrementos de 10 GB de até 64 TB.

Implementar alterações

Vamos implantar atualizações usando uma implantação canário ou azul/verde em cada de zona de isolamento separadamente. As implantações são totalmente automatizadas, incluindo uma reversão se os KPIs indicarem um problema.

Os Runbooks existirão para requisitos de relatório rigorosos e acompanhamento de performance. Se operações bem-sucedidas estiverem seguindo uma tendência de falha em atingir a performance ou as metas de disponibilidade, um manual será usado para determinar o que está causando essa

tendência. Há manuais para modos de falha e incidentes de segurança não descobertos. Também existem playbooks para estabelecer a causa raiz das falhas. Também atuaremos com o AWS Support para a oferta de Gerenciamento de eventos de infraestrutura.

A equipe que cria e opera o site identificará a correção de erro de qualquer falha inesperada e priorizará a correção a ser implantada depois de implementada.

Fazer o backup de dados

O backup e a restauração podem ser feitos usando o Amazon RDS. A operação será executada regularmente usando um runbook para garantir que os requisitos de recuperação possam ser cumpridos.

Arquitetar para resiliência

Recomendamos três zonas de disponibilidade para essa abordagem. Usando uma implantação de três zonas de disponibilidade, cada AZ tem uma capacidade estática de 50% do pico. Duas zonas de disponibilidade podem ser usadas, mas o custo da capacidade estaticamente estável seria maior, pois ambas as zonas precisariam ter 100% da capacidade de pico. Adicionaremos o Amazon CloudFront para fornecer armazenamento em cache geográfico, bem como solicitar a redução do plano de dados de nosso aplicativo.

Usaremos o Amazon Aurora como RDS e implantaremos réplicas de leitura nas três zonas.

Os aplicativos serão criados usando os padrões de resiliência de software/aplicativo em todas as camadas.

Testar a resiliência

O pipeline de implantação terá um pacote de teste completo, incluindo testes de performance, carga e injeção de falha.

Praticaremos nossos procedimentos de recuperação de falha constantemente em dias de jogo usando runbooks para garantir que possamos realizar as tarefas e não nos desviemos dos procedimentos. A equipe que cria o site também o opera.

Planejar para a recuperação de desastres (DR)

Existem runbooks para recuperação total da carga de trabalho e emissão de relatórios comuns. A recuperação usa backups armazenados na mesma região que a carga de trabalho. Os procedimentos de restauração são exercidos regularmente como parte dos dias de jogos.

Meta de design de disponibilidade

Presumimos que pelo menos algumas falhas exigirão uma decisão manual de executar a recuperação, porém, com maior automação neste cenário, presumimos que apenas dois eventos por ano exigirão essa decisão e as ações de recuperação serão rápidas. Levamos 10 minutos para decidir executar a recuperação, e presumimos que a recuperação estará concluída dentro de cinco minutos. Isso implica 15 minutos para a recuperação da falha. Presumindo duas falhas por ano, nosso tempo de impacto estimado para o ano é de 30 minutos.

Isso significa que o limite superior de disponibilidade é 99,99%. A disponibilidade geral dependerá também da taxa real de falha, da duração da falha e da rapidez com que cada falha se recupera realmente. Para essa arquitetura, presumimos que o aplicativo está online continuamente por meio de atualizações. Com base nisso, nossa meta de design de disponibilidade é 99,99%.

Resumo

Tópico	Implementação
Monitorar recursos	Verificações de integridade em todas as camadas e em KPIs; alertas enviados quando os alarmes configurados são desarmados; alerta de todas as falhas. As reuniões operacionais são rigorosas para detectar tendências e gerenciar conforme as metas de design.
Adaptar-se às mudanças de demanda	ELB para nível de aplicativo de escalabilidade automática e web; armazenamento de escalabilidade automática e réplicas de leitura em várias zonas para o Aurora.
Implementar alterações	Implantação automatizada por meio de canário ou azul/verde e reversão automatizada quando KPIs ou alertas indicam problemas não detectados no aplicativo. As implantações são feitas por zona de isolamento.

Tópico	Implementação
Fazer o backup de dados	Backups automatizados por meio de RDS para cumprir o RPO e restauração automatizada praticada regularmente em um dia de jogo.
Arquitetar para resiliência	Implementadas zonas de isolamento de falhas para o aplicativo; auto scaling para fornecer um nível de aplicativo e web com autorrecu peração; RDS é Multi-AZ.
Testar a resiliência	O teste de falha de zona de isolamento e componente está no pipeline e é praticado com a equipe operacional regularmente em um dia de jogo; existe um manual para diagnosticar problemas desconhecidos e um processo de Análise de causa raiz.
Planejar para a recuperação de desastres (DR)	Backups criptografados por meio do RDS para a mesma região da AWS que é praticada em um dia de jogo.

Cenários de várias regiões

A implementação da aplicação em várias regiões da AWS aumentará o custo da operação, em parte porque isolamos regiões para manter a autonomia delas. Seguir esse caminho deve ser uma decisão muito ponderada. Dito isso, as regiões fornecem um limite de isolamento forte e fazemos grandes esforços para evitar correlacionar falhas entre regiões. O uso de várias regiões fornece mais controle sobre o tempo de recuperação no caso de uma falha de dependência rígida em um serviço regional da AWS. Nesta seção, vamos discutir vários padrões de implementação e sua disponibilidade típica.

Tópicos

3½ noves (99,95%) com um tempo de recuperação entre 5 e 30 minutos

Essa meta de disponibilidade para aplicativos exige um tempo de inatividade extremamente curto e perda de dados muito pequena durante tempos específicos. Aplicativos com essa meta de disponibilidade incluem aqueles nas áreas de: bancos, investimento, serviços de emergência e captura de dados. Esses aplicativos têm pontos de recuperação e tempos de recuperação muito curtos.

Podemos melhorar ainda mais o tempo de recuperação usando uma abordagem de Standby passivo abordagem de duas regiões da AWS. Implantaremos toda a carga de trabalho em ambas as regiões, com o nosso site passivo reduzido e todos os dados mantidos eventualmente consistentes. Ambas as implantações terão um comportamento estaticamente estável em suas respectivas regiões. Os aplicativos devem ser criados usando os padrões de resiliência do sistema distribuído. Precisaremos criar um componente leve de roteamento que monitore a integridade da carga de trabalho e que possa ser configurado para rotear o tráfego para a região passiva, se necessário.

Monitorar recursos

Haverá alertas em cada substituição de um servidor da web, quando o banco de dados realizar o failover ou quando a Região realizar o failover. Também monitoraremos o conteúdo estático no Amazon S3 para disponibilidade e alerta se ele ficar indisponível. O registro será agregado para facilitar o gerenciamento e ajudar na análise de causa raiz em cada Região.

O componente de roteamento monitora a integridade do aplicativo e todas as dependências rígidas regionais que temos.

Adaptar-se às mudanças de demanda

Igual ao cenário de quatro noves.

Implementar alterações

A entrega de novo software ocorre conforme um cronograma fixo a cada duas a quatro semanas. As atualizações de software serão automatizadas usando padrões de implantação canário ou azul/verde.

Existem runbooks para quando ocorre o failover da Região, para problemas comuns do cliente que ocorrem durante esses eventos e para relatórios comuns.

Teremos manuais para problemas de banco de dados comuns, incidentes relacionados à segurança, implantações com falha, problemas inesperados do cliente no failover da Região e estabelecimento

da causa raiz dos problemas. Depois da identificação da causa raiz, a correção do erro será identificada por uma combinação das equipes de operações e desenvolvimento e implantada quando a correção for desenvolvida.

Também atuaremos com o AWS Support para Gerenciamento de eventos de infraestrutura.

Fazer o backup de dados

Como no cenário de quatro noves, usamos os backups automáticos do RDS e o versionamento do S3. Os dados são replicados de forma automática e assíncrona do cluster do Aurora RDS na região ativa para réplicas de leitura entre regiões na região passiva. A replicação entre regiões do S3 é usada para mover dados de forma automática e assíncrona da região ativa para a passiva.

Arquitetar para resiliência

Igual ao cenário de quatro noves, e também o failover regional é possível. Isso é gerenciado manualmente. Durante o failover, rotearemos solicitações a um site estático usando failover de DNS até a recuperação na segunda Região.

Testar a resiliência

Igual ao cenário de quatro noves, e ainda validaremos a arquitetura por meio de dias de jogos usando runbooks. A correção de RCA também tem prioridade sobre versões de recursos para implementação e implantação imediatas.

Planejar para a recuperação de desastres (DR)

O failover regional é gerenciado manualmente. Todos os dados são replicados de forma assíncrona. A infraestrutura no standby passivo é expandida. Isso pode ser automatizado usando um fluxo de trabalho executado no AWS Step Functions. O AWS Systems Manager (SSM) também pode ajudar com essa automação, pois você pode criar documentos do SSM que atualizam grupos de Auto Scaling e redimensionam instâncias.

Meta de design de disponibilidade

Presumimos que pelo menos algumas falhas exigirão uma decisão manual de executar a recuperação, porém, com uma boa automação neste cenário, presumimos que apenas dois eventos por ano exigirão essa decisão. Levamos 20 minutos para decidir executar a recuperação e presumimos que a recuperação estará concluída dentro de 10 minutos. Isso implica cerca de 30

minutos para a recuperação de uma falha. Presumindo duas falhas por ano, nosso tempo de impacto estimado para o ano é de 60 minutos.

Isso significa que o limite superior de disponibilidade é de 99,95%. A disponibilidade geral dependerá também da taxa real de falha, da duração da falha e da rapidez com que cada falha se recupera realmente. Para essa arquitetura, presumimos que o aplicativo está online continuamente por meio de atualizações. Com base nisso, nossa meta de design de disponibilidade é 99,95%.

Resumo

Tópico	Implementação
Monitorar recursos	Verificações de integridade em todas as camadas, incluindo integridade de DNS no nível de região da AWS, e em KPIs; em alertas enviados quando os alarmes configurados são acionados; enviando alertas sobre todas as falhas. As reuniões operacionais são rigorosas para detectar tendências e gerenciar conforme as metas de design.
Adaptar-se às mudanças de demanda	ELB para nível de aplicativo de escalabilidade automática e web; armazenamento de escalabilidade automática e réplicas de leitura em várias zonas nas regiões ativas e passivas do Aurora. Dados e infraestrutura sincronizados entre regiões da AWS para estabilidade estática.
Implementar alterações	Implantação automatizada por meio de canário ou azul/verde e reversão automatizada quando os KPIs ou os alertas indicam problemas não detectados na aplicação, as implantações são feitas em uma zona de isolamento e em uma região da AWS de cada vez.
Fazer o backup de dados	Backups automatizados em cada região da AWS por meio do RDS para atender ao RPO e restauração automatizada praticada

Tópico	Implementação
	regularmente em um dia de jogo. Os dados do Aurora RDS e do S3 são replicados de forma automática e assíncrona da região ativa para a passiva.
Arquitetar para resiliência	Escalabilidade automática para fornecer nível de aplicativo e web com autorrecuperação; RDS é Multi-AZ; failover regional é gerenciado manualmente com local estático apresentado durante o failover.
Testar a resiliência	O teste de falha de zona de isolamento e componente está no pipeline e é praticado com a equipe operacional regularmente em um dia de jogo; existe um manual para diagnosticar problemas desconhecidos e existe um processo de Análise de causa raiz com caminhos de comunicação para qual era o problema e como ele foi corrigido ou prevenido . A correção de RCA tem prioridade sobre versões de recursos para implementação e implantação imediatas.
Planejar para a recuperação de desastres (DR)	Standby passivo implantado em outra região. A infraestrutura é expandida usando fluxos de trabalho executados com o uso de documento s do AWS Step Functions ou do AWSSystem s Manager. Backups criptografados via RDS. Réplicas de leitura entre regiões entre duas regiões da AWS. Replicação entre regiões de ativos estáticos no Amazon S3. A restauraç ão é para a região da AWS ativa no momento, praticada em um dia de jogo e coordenada com a AWS.

Cenário de cinco noves (99,999%) ou mais alto com tempo de recuperação inferior a um minuto

Essa meta de disponibilidade para aplicativos não exige quase nenhum tempo de inatividade ou perda de dados para tempos específicos. Aplicativos que poderiam ter essa meta de disponibilidade incluem, por exemplo, determinados aplicativos bancários, de investimento, financeiros, do governo e críticos para o negócio central de empresas que geram receitas extremamente grandes. O objetivo é ter os repositórios de dados fortemente consistentes e redundância total em todas as camadas. Selecionamos um repositório de dados baseado em SQL. Porém, em alguns cenários, será difícil alcançar um RPO muito pequeno. Se você puder particionar seus dados, poderá não ter perda de dados. Isso pode exigir que você adicione lógica de aplicativo e latência para garantir que haja dados consistentes entre locais geográficos, além da capacidade de transferir ou copiar dados entre partições. Pode ser mais fácil realizar esse particionamento usando o banco de dados NoSQL.

Podemos melhorar ainda mais a disponibilidade usando uma abordagem Ativo-ativo em várias regiões da AWS. A carga de trabalho será implantada em todas as Regiões desejadas com um comportamento estaticamente estável entre regiões (para que as regiões restantes possam lidar com a carga após a perda de uma região). A roteamento direciona o tráfego para localizações geográficas íntegras e altera automaticamente o destino quando um local não está íntegro, bem como interrompe temporariamente as camadas de replicação de dados. O Amazon Route 53 oferece verificações de integridade a intervalos de 10 segundos e também oferece TTL de apenas um segundo em seus conjuntos de registro.

Monitorar recursos

Igual ao cenário de 3½ noves, além de alertas quando uma região é detectada como não íntegra e o tráfego é roteado para fora dela.

Adaptar-se às mudanças de demanda

Igual ao cenário de 3½ noves.

Implementar alterações

O pipeline de implantação terá um pacote de teste completo, incluindo testes de performance, carga e injeção de falha. Implantaremos as atualizações usando implantações canário ou azul/verde a uma zona de isolamento por vez, em uma Região antes de começarmos na outra. Durante a implantação, as versões antigas ainda serão mantidas em execução nas instâncias para facilitar uma reversão

mais rápida. Elas são totalmente automatizadas, incluindo uma reversão se os KPIs indicarem um problema. O monitoramento incluirá métricas de sucesso, bem como alertas quando ocorrem problemas.

Os Runbooks existirão para requisitos de relatório rigorosos e acompanhamento de performance. Se operações bem-sucedidas estiverem seguindo uma tendência de falha em atingir a performance ou as metas de disponibilidade, um manual será usado para determinar o que está causando essa tendência. Há manuais para modos de falha e incidentes de segurança não descobertos. Também existem playbooks para estabelecer a causa raiz das falhas.

A equipe que cria o site também o opera. Essa equipe identificará a correção de erro de qualquer falha inesperada e priorizará a correção a ser implantada depois de implementada. Também atuaremos com o AWS Support para Gerenciamento de eventos de infraestrutura.

Fazer o backup de dados

Igual ao cenário de 3½ noves.

Arquitetar para resiliência

Os aplicativos devem ser criados usando os padrões de resiliência de software/aplicativo. Talvez muitas outras camadas de roteamento sejam necessárias para implementar a disponibilidade exigida. Não subestime a complexidade dessa implementação adicional. O aplicativo será implementado em zonas de isolamento de falha de implantação e particionado e implantado de modo que até mesmo um evento em toda a Região não afete todos os clientes.

Testar a resiliência

Validaremos a arquitetura em dias de jogo usando runbooks para garantir que possamos realizar as tarefas e não nos desviemos dos procedimentos.

Planejar para a recuperação de desastres (DR)

Ativo-ativo em várias regiões com infraestrutura completa de carga de trabalho e dados em várias regiões. Usando um local de leitura, uma estratégia global de gravação, uma região é o banco de dados primário para todas as gravações, e os dados são replicados para leituras para outras regiões. Se a região do banco de dados primário falhar, será necessário promover um novo banco de dados. Leitura local, gravação global tem usuários atribuídos a uma região de origem onde as gravações de banco de dados são processadas. Isso permite que os usuários leiam ou gravem de qualquer

região, mas requer lógica complexa para gerenciar possíveis conflitos de dados entre gravações em diferentes regiões.

Quando uma região é detectada como não íntegra, a camada de roteamento roteia automaticamente o tráfego para as regiões íntegras restantes. Nenhuma intervenção manual é necessária.

Os repositórios de dados devem ser replicados entre as Regiões de maneira que possam resolver possíveis conflitos. Será preciso criar ferramentas e processos automatizados ou mover dados entre as partições por motivos de latência e para equilibrar as solicitações ou as quantidades de dados em cada partição. A remedição da resolução de conflito de dados também exigirá mais runbooks operacionais.

Meta de design de disponibilidade

Presumimos que são feitos investimentos pesados para automatizar toda a recuperação e que a recuperação pode ser concluída dentro de um minuto. Não presumimos recuperações acionadas manualmente, mas até uma ação de recuperação automatizada por trimestre. Isso implica quatro minutos por ano para recuperação. Presumimos que o aplicativo está online continuamente por meio de atualizações. Com base nisso, nossa meta de design de disponibilidade é 99,999%.

Resumo

Tópico	Implementação
Monitorar recursos	Verificações de integridade em todas as camadas, incluindo integridade de DNS no nível de região da AWS, e em KPIs; em alertas enviados quando os alarmes configurados são acionados; enviando alertas sobre todas as falhas. As reuniões operacionais são rigorosas para detectar tendências e gerenciar conforme as metas de design.
Adaptar-se às mudanças de demanda	ELB para nível de aplicativo de escalabilidade automática e web; armazenamento de escalabil idade automática e réplicas de leitura em várias zonas nas regiões ativas e passivas do Aurora RDS. Dados e infraestrutura sincroniz

Tópico	Implementação
	ados entre regiões da AWS para estabilidade estática.
Implementar alterações	Implantação automatizada por meio de canário ou azul/verde e reversão automatizada quando os KPIs ou os alertas indicam problemas não detectados na aplicação, as implantações são feitas em uma zona de isolamento e em uma região da AWS de cada vez.
Fazer o backup de dados	Backups automatizados em cada região da AWS por meio do RDS para atender ao RPO e restauração automatizada praticada regularmente em um dia de jogo. Os dados do Aurora RDS e do S3 são replicados de forma automática e assíncrona da região ativa para a passiva.
Arquitetar para resiliência	Implementadas zonas de isolamento de falhas para o aplicativo; auto scaling para fornecer um nível de aplicativo e web com autorrecu peração; RDS é Multi-AZ; failover regional automatizado.
Testar a resiliência	O teste de falha de zona de isolamento e componente está no pipeline e é praticado com a equipe operacional regularmente em um dia de jogo; existe um manual para diagnosticar problemas desconhecidos e existe um processo de Análise de causa raiz com caminhos de comunicação para qual era o problema e como ele foi corrigido ou prevenido . A correção de RCA tem prioridade sobre versões de recursos para implementação e implantação imediatas.

Tópico	Implementação
Planejar para a recuperação de desastres (DR)	Ativo-ativo implantado em pelo menos duas regiões. A infraestrutura é totalment e dimensionada e estaticamente estável entre regiões. Os dados são particionados e sincronizados entre regiões. Backups criptografados via RDS. A falha na região é praticada em um dia de jogo e é coordenada com a AWS. Durante a restauração, pode ser necessário promover um novo banco de dados primário.

Recursos

Documentação

- Amazon Builders' Library - Como a Amazon cria e opera softwares
- Centro de Arquitetura da AWS

Laboratórios

- Laboratórios de confiabilidade do AWS Well-Architected

Links externos

- Padrão de enfileiramento adaptável: falha em grande escala
- Disponibilidade e além: compreensão e melhoria da resiliência de sistemas distribuídos na AWS

Livros

- Robert S. Hammer "Patterns for fault tolerant software"
- Andrew Tanenbaum e Marten van Steen "Distributed systems: principles and paradigms"

Conclusão

Seja você novo nos tópicos de disponibilidade e confiabilidade ou um veterano experiente buscando informações para maximizar sua disponibilidade de carga de trabalho de missão crítica, esperamos que este whitepaper o tenha feito refletir, oferecido novas perspectivas ou apresentado uma nova linha de questionamento. Esperamos que isso leve a uma compreensão mais profunda do nível certo de disponibilidade conforme as necessidades de sua empresa, e como projetar a confiabilidade para consegui-la. Incentivamos você a aproveitar as recomendações orientadas à recuperação, operacionais e de design oferecidas aqui, bem como o conhecimento e a experiência dos arquitetos de soluções da AWS. Queremos ouvir seus comentários, principalmente suas histórias de sucesso sobre como foram atingidos altos níveis de disponibilidade com a AWS. Entre em contato com a equipe da sua conta ou use a opção Entre em contato conosco em nosso site.

Colaboradores

Os colaboradores desse documento incluem:

- Seth Eliot, ativista-chefe de desenvolvedores, Amazon Web Services
- Mahanth Jayadeva, arquiteto de soluções, Well-Architected, Amazon Web Services
- Amulya Sharma, arquiteta-chefe de soluções, Amazon Web Services
- Jason DiDomenico, arquiteto sênior de soluções, Cloud Foundations (Fundamentos de nuvem), Amazon Web Services
- Marcin Bednarz, arquiteto-chefe de soluções, Amazon Web Services
- Tyler Applebaum, arquiteto sênior de soluções, Amazon Web Services
- Rodney Lester, arquiteto-chefe de soluções, App Modernization (Modernização de aplicações), Amazon Web Services
- Joe Chapman, arquiteto sênior de soluções, Amazon Web Services
- Adrian Hornsby, engenheiro-chefe de desenvolvimento de sistemas, Amazon Web Services
- Kevin Miller, vice-presidente, S3, Amazon Web Services
- Shannon Richards, gerente-chefe de programas técnicos, Amazon Web Services
- Laurent Domb, tecnólogo chefe da Fed Fin, Amazon Web Services
- Kevin Schwarz, arquiteto sênior de soluções, Amazon Web Services
- Rob Martell, arquiteto-chefe de resiliência na nuvem, Amazon Web Services
- Priyam Reddy, gerente sênior de arquitetura de soluções, DR, da Amazon Web Services
- Jeff Ferris, tecnólogo-chefe da Amazon Web Services
- Matias Battaglia, arquiteto sênior de soluções da Amazon Web Services

Leitura adicional

Para obter informações adicionais, consulte:

- AWS Well-Architected Framework
- Centro de Arquitetura da AWS

Revisões do documento

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

Alteração	Descrição	Data
Whitepaper atualizado	Práticas recomendadas atualizadas com novas orientações para implement ação.	June 27, 2024
Orientação sobre práticas recomendadas atualizada	As práticas recomendadas foram atualizadas com novas orientações nas seguintes áreas: Projetar interações em um sistema distribuído para evitar falhas, Projetar interações em um sistema distribuído para mitigar ou resistir a falhas, Monitorar os recursos da workload, Projetar a workload para se adaptar às mudanças na demanda, Implementar mudanças e Testar a confiabilidade.	December 6, 2023
Orientação sobre práticas recomendadas atualizada	As práticas recomendadas foram atualizadas com novas orientações nas seguintes áreas: Monitorar os recursos da workload e Projetar a workload para resistir a falhas de componentes.	October 3, 2023
Orientação sobre práticas recomendadas atualizada	As práticas recomendadas foram atualizadas com novas orientações nas seguintes	July 13, 2023

	áreas: Projetar a arquitetura de serviço da workload, Projetar interações em um sistema distribuído para mitigar ou resistir a falhas e Monitorar os recursos da workload.	
Atualização secundária	Remoção de linguagem não inclusiva.	April 13, 2023
Atualizações para o novo Framework	Práticas recomendadas atualizadas com orientações prescritivas e novas práticas recomendadas adicionadas.	April 10, 2023
Whitepaper atualizado	Práticas recomendadas atualizadas com novas orientações para implementação.	December 15, 2022
Atualizações secundárias	Correção de números das figuras e alterações secundárias em geral.	November 17, 2022
Whitepaper atualizado	Práticas recomendadas ampliadas e planos de melhoria adicionados.	October 20, 2022
Whitepaper atualizado	Adição de duas práticas recomendadas ao Pilar Confiabilidade nas seções Isolamento de falhas para proteger a workload e Projete a workload para resistir a falhas de componentes.	May 5, 2022

Atualização secundária	Pilar Sustentabilidade adicionado à introdução.	December 2, 2021
Whitepaper atualizado	Atualização das orientaçõ es sobre recuperação de desastres para incluir o Route 53 Application Recovery Controller. Adição de referênci as ao DevOps Guru. Atualizaç ão de vários links de recursos e outras alterações editoriais secundárias.	October 26, 2021
Atualização secundária	Adição de informações sobre o AWS Fault Injection Service (AWS FIS).	March 15, 2021
Atualização secundária	Atualização de texto secundári a.	January 4, 2021

Whitepaper atualizado	Atualização do Apêndice A para: atualizar as seções Meta de design de disponibilidade do Amazon SQS e do Amazon MQ; reordenar as linhas em tabelas para maior facilidad e de pesquisa; melhorar a explicação das diferença s entre disponibilidade e recuperação de desastres e como elas contribuem para a resiliência; expandir a cobertura de arquiteturas de várias regiões (para disponibi lidade) e as estratégias de varias regiões (para recuperaç ão de desastres); atualizar o manual de referência para a versão mais recente; expandir os cálculos de disponibilidade para incluir cálculos baseados em solicitações e cálculos de atalhos; melhorar a descrição de dias de jogo.	December 7, 2020
Atualização secundária	Atualização do Apêndice A para atualizar a meta de design de disponibilidade do AWS Lambda.	October 27, 2020
Atualização secundária	Atualização do Apêndice A para adicionar a meta de design de disponibilidade do AWS Global Accelerator.	July 24, 2020

Atualizações para o novo Framework	Atualizações substanciais e conteúdo novo/revisado, incluindo: adição da seção de práticas recomendadas de "Arquitetura de workload" , reorganização das práticas recomendadas nas seções Gerenciamento de alteraçõe s e Gerenciamento de falhas, atualização de recursos, atualização para incluir os recursos e serviços mais recentes da AWS, como o AWS Global Accelerator, o AWS Service Quotas e o AWS Transit Gateway, adição/at ualização de definições de Confiabilidade, Disponibi lidade, Resiliência, melhor alinhamento do whitepape r ao AWS Well-Architected Tool (perguntas e práticas recomendadas) usado para Revisões do Well-Architected, reordenação dos princípio s de projeto, mudança de posição de Recuperação de falhas automática para antes de Teste de procedimentos de recuperação, atualizaç ão de diagramas e formatos de equações, remoção das seções de principais serviços e implantação de referênci as aos serviços principais da	July 8, 2020

	AWS nas práticas recomendadas.	
Atualização secundária	Link quebrado corrigido	October 1, 2019
Whitepaper atualizado	Atualização do Apêndice A	April 1, 2019
Whitepaper atualizado	Adição de recomendações de rede do AWS Direct Connect específicas e mais metas de design de serviço	September 1, 2018
Whitepaper atualizado	Adicionadas as seções de Princípios de design e Gerenciamento de limite. Atualizados links, removida a ambiguidade da terminologia upstream/downstream e adicionadas referências explícitas aos tópicos de Pilar de confiabilidade restantes nos cenários de disponibilidade.	June 1, 2018
Whitepaper atualizado	Alteração da solução entre regiões do DynamoDB para tabelas globais do DynamoDB. Adição de metas de design de serviço	March 1, 2018
Atualizações secundárias	Pequena correção ao cálculo de disponibilidade para incluir a disponibilidade da aplicação	December 1, 2017

Whitepaper atualizado	Atualização para fornecer orientação sobre designs de alta disponibilidade, incluindo conceitos, melhores práticas e implementações de exemplo.	November 1, 2017
Publicação inicial	Publicação do Pilar Confiabilidade: AWS Well Architected Framework.	November 1, 2016