Framework Well-Architected da AWS

Pilar Confiabilidade

Pilar Confiabilidade: Framework Well-Architected da AWS

Table of Contents

Pilar Confiabilidade: AWS Well-Architected Framework

Data de publicação: 6 de novembro 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, na entrega e na 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 custo
- Sustentabilidade

Este documento foca o pilar 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 destina-se 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 workload de executar a função pretendida correta e consistentemente quando esperado. Isso inclui a capacidade de operar e testar a workload 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 entender 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 é composta por hardware, software, redes e instalações que executam os serviços da Nuvem AWS. A AWS emprega esforços comercialmente razoáveis para disponibilizar esses serviços da Nuvem AWS, garantindo que a disponibilidade do serviço atenda ou exceda os acordos de nível de serviço (SLAs) da AWS.

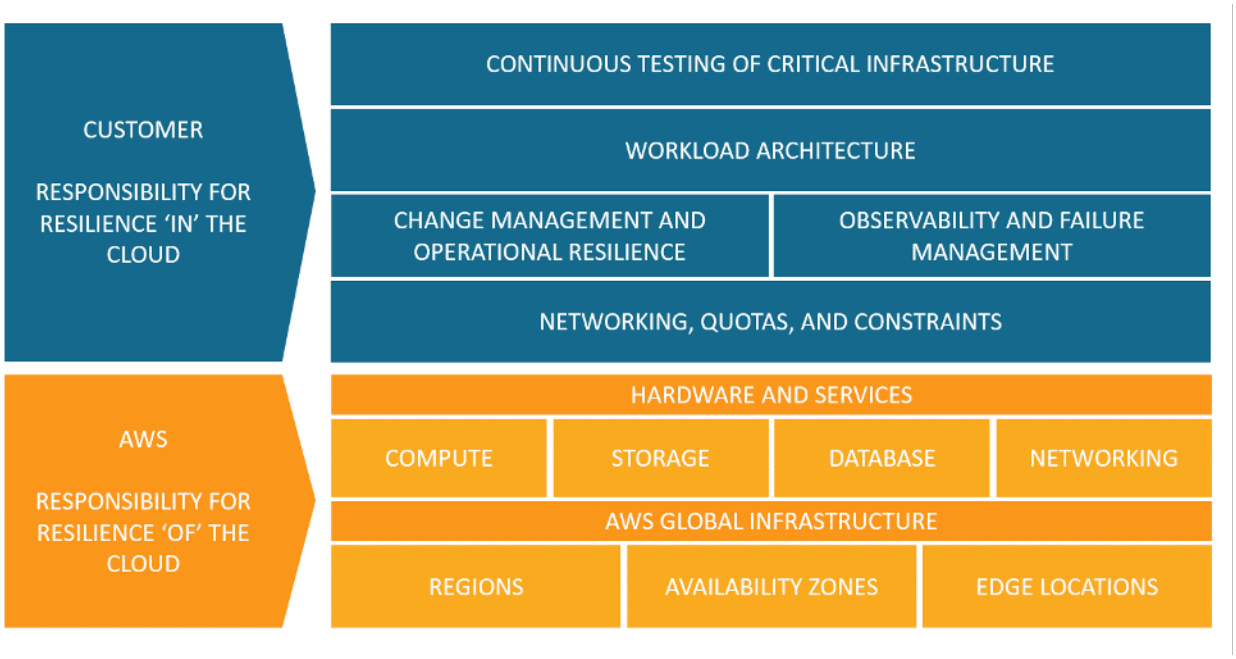
A infraestrutura de nuvem global da AWS foi projetada para permitir que os clientes criem arquiteturas de workload altamente resilientes. Cada Região da AWS é totalmente isolada e composta de várias zonas de disponibilidade, que são partições de infraestrutura fisicamente isoladas umas das outras. As 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 conexão de rede com alto throughput e baixa latência entre as zonas. Todo o tráfego entre as zonas é criptografado. A performance da rede é suficiente para realizar a replicação síncrona entre as zonas.

Quando uma aplicação é particionada entre AZs, as empresas permanecem mais bem isoladas e protegidas contra problemas como queda de energia, raios, tornados, furacões, entre outros.

Responsabilidade do cliente: resiliência na nuvem

Sua responsabilidade é determinada pelos serviços da Nuvem AWS que você escolhe. Isso determina o volume do trabalho de configuração que você deve executar como parte das 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 autorrecuperação usando serviços como o Auto Scaling e usar 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 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 são descritas em detalhes em Fundamentos.
- Planeje sua arquitetura com espaço adequado para escalar e entender as cotas de serviço e as restrições dos serviços que você incluiu com base nos aumentos esperados de solicitações de carga, quando aplicável.
- Projete sua topologia de rede para ser altamente disponível, redundante e escalável.

Gerenciamento de alterações e resiliência operacional

- O gerenciamento de alterações inclui como introduzir e gerenciar mudanças em seu ambiente. A implementação de alterações requer a criação e a manutenção de runbooks atualizados e estratégias de implantação para sua aplicação e infraestrutura.
- Uma estratégia resiliente para monitorar os recursos da workload considera todos os componentes, incluindo métricas técnicas e comerciais, notificações, automação e análise.

- As workloads na nuvem devem se adaptar às mudanças na escalação da demanda em reação a deficiências ou flutuações no uso.

Observabilidade e gerenciamento de falhas

- A observação de falhas por meio do monitoramento é necessária para automatizar a reparação para que suas workloads possam suportar falhas de componentes.
- O gerenciamento de falhas requer backup de dados, aplicação das práticas recomendadas para permitir que sua workload resista a falhas de componentes e planejamento para recuperação de desastres.

Arquitetura da workload

- Sua arquitetura de workload inclui como você projeta serviços em domínios de negócios, aplica SOA e design de sistema distribuído para evitar falhas e incorpora recursos como controle de utilização, novas tentativas, gerenciamento de filas, tempos limite e acionadores de emergência.
- Confie em soluções da AWS comprovadas, na Amazon Builders Library e em padrões sem servidor para se alinhar às práticas recomendadas e iniciar implementações.
- Use melhorias contínuas para decompor o sistema em serviços distribuídos para escalar e inovar mais rapidamente. Use a orientação de microsserviços da AWS e as 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 performance e caos, além de adotar análises de incidentes e práticas diárias para desenvolver experiência na resolução de problemas que não são bem compreendidos.
- 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 da forma esperada. Esses testes provarão a eficácia de sua resiliência geral e fornecerão um loop de feedback dos 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 práticas recomendadas:

- Recupere-se de falhas automaticamente: ao monitorar os indicadores-chave de performance (KPIs) de uma workloads, você pode executar a automação quando um limite é violado. Esses KPIs devem ser uma medida do valor comercial, e 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.
- Teste 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 workload 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 um cenário de falha real ocorra, reduzindo assim o risco.
- Escale 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.
- Pare de tentar 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 respectiva capacidade (esse muitas vezes é o objetivo dos ataques de negação de serviço). Na nuvem, você pode monitorar a demanda e a utilização da workload e automatizar a adição ou a remoção de recursos para manter o nível ideal e atender à demanda, sem provisionamento excessivo ou subprovisionamento. Ainda há limites, mas algumas cotas podem ser controladas e outras podem ser gerenciadas (consulte Gerenciar cotas e restrições de serviço).
- Gerencie alterações na automação: as alterações em sua infraestrutura devem ser feitas por meio de automação. Entre aquelas que precisam ser gerenciadas estão as alterações na automação, que podem então ser acompanhadas e analisadas.

Definições

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

- Fundamentos
- Arquitetura da workload
- Gerenciamento de alterações
- 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 workload. A arquitetura da workload do sistema distribuído deve ser projetada para evitar e mitigar falhas. A workload 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, o principal deles é a Resiliência:

- Resiliência é a capacidade de uma workload se recuperar de interrupções na infraestrutura ou nos serviços, 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 workload são:

- Excelência operacional, que inclui automação de alterações, uso de playbooks para responder a falhas e revisões de prontidão operacional (ORRs) para confirmar que as aplicações estão prontas 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 projetar para taxas máximas de solicitação e a minimização de latências para sua workload.
- Otimização de custos, que inclui compensações, como se você deseja gastar mais em instâncias do EC2 para alcançar ajuste de escala automático ou confiar no ajuste de escala automático quando mais capacidade for necessária.

A resiliência é o foco principal deste whitepaper.

Os outros quatro fatores 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

A Disponibilidade (também conhecida como disponibilidade de serviços) é uma métrica comumente usada para medir quantitativamente a resiliência, bem como um objetivo de resiliência alvo.

- A disponibilidade é a porcentagem de tempo durante a qual 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 a aplicação não está operando normalmente, incluindo interrupções agendadas e não agendadas. Definimos disponibilidade da seguinte forma:

$$Availability = \frac{Available\ for\ Use\ Time}{Total\ Time}$$

- 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 agendado do serviço (por exemplo, manutenção planejada) do tempo total na fórmula. No entanto, isso não é recomendado, 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 aplicação comuns e a máxima duração das interrupções que podem ocorrer em um ano sem que a meta deixe de ser atingida. A tabela contém exemplos dos tipos de aplicações comuns em cada nível de disponibilidade. Neste documento, vamos nos referir a esses valores.

Disponibilidade	Indisponibilidade máxima (por ano)	Categorias de aplicações
99%	3 dias e 15 horas	Tarefas de processamento em lote e de extração, transferência e carregamento de dados
99,9%	8 horas e 45 minutos	Ferramentas internas como gerenciamento de conhecimento e rastreamento de projetos
99,95%	4 horas e 22 minutos	Comércio online, ponto de vendas
99,99%	52 minutos	Workloads de entrega e broadcast de vídeo
99,999%	5 minutos	Workloads de transações em caixas eletrônicos, telecomunicações

Medição da disponibilidade com base nas solicitações. Para seu serviço, talvez seja 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:

$$Availability = \frac{Successful\ Responses}{Valid\ Requests}$$

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.

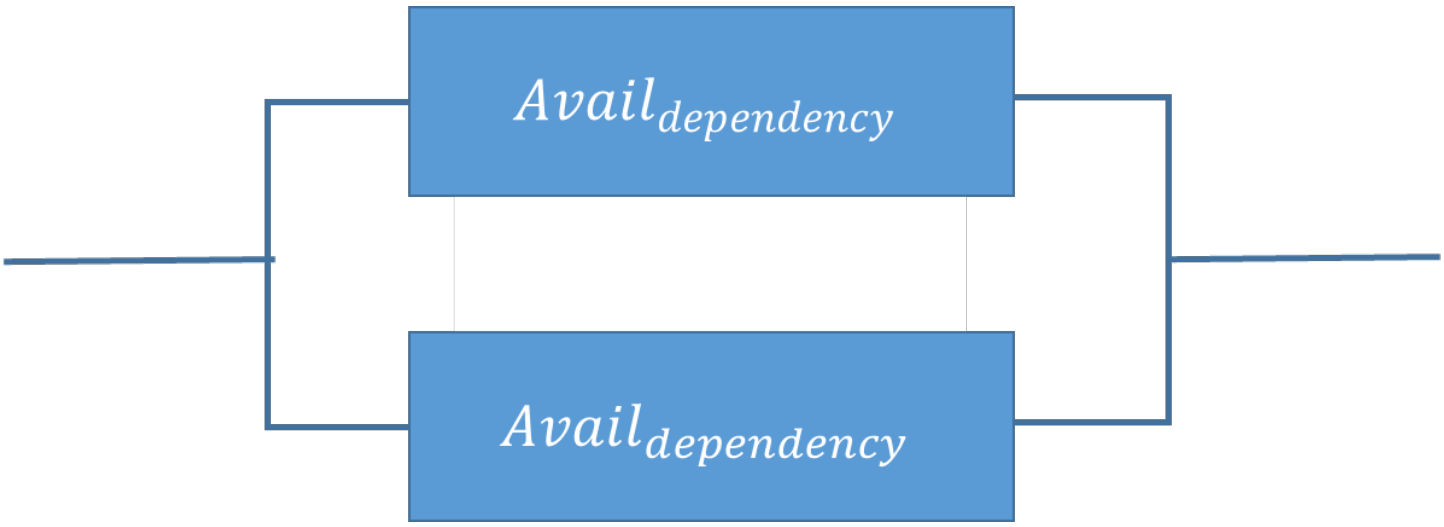
Cálculo de disponibilidade com dependências rígidas. Muitos sistemas têm dependências rígidas de outros sistemas, onde uma interrupção de um sistema dependente leva diretamente a uma interrupção do sistema que o invocou. Isso é o oposto de uma dependência flexível, em que uma falha do sistema dependente é compensada na aplicação. 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 workload teoricamente poderá atingir uma disponibilidade de 99,97%:

$Avail_{invok} \times Avail_{dep1} \times Avail_{dep2} = Avail_{workload}$

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%:



$Avail_{effective} = Avail_{MAX} - ((100\% - Avail_{dependency}) \times (100\% - Avail_{dependency}))$

99,9999% = 100% − (0,1%×0,1%)

Cálculo com atalho: se as disponibilidades de todos os componentes em seu cálculo consistirem apenas no dígito nove, 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 falhas (MTBF) e o tempo médio para recuperação (MTTR). É possível estabelecer a estimativa de disponibilidade da seguinte maneira:

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

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

Para obter detalhes adicionais, consulte Disponibilidade e além: como entender e melhorar a resiliência de sistemas distribuídos na AWS para obter ajuda para calcular sua disponibilidade.

Custos para a disponibilidade. Desenvolver aplicações para níveis mais altos de disponibilidade costuma resultar em mais custos. Assim, é importante identificar as verdadeiras necessidades de disponibilidade antes de iniciar o design da aplicação. 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. Contribuindo com 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 a meta 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 de recuperação de desastres (DR)

Além dos objetivos de disponibilidade, a estratégia de resiliência também deve incluir objetivos de recuperação de desastres (DR) baseados em estratégias para recuperar a workload em caso 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. O RTO é o atraso máximo aceitável entre a interrupção e a restauração do serviço. Ele determina o que é considerado uma janela de tempo aceitável quando o serviço não está disponível.

Objetivo de ponto de recuperação (RPO) definido pela organização. O RPO é o período máximo de tempo 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 objetivo de ponto de recuperação (RPO), objetivo do tempo de recuperação (RTO) 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 workload. No entanto, o 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.

Como entender as necessidades de disponibilidade

É comum pensar inicialmente na disponibilidade de uma aplicação como um único objetivo para a aplicação como um todo. Porém, com uma inspeção mais detalhada, é comum descobrirmos que determinados aspectos de uma aplicação 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 sobre 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 uma única aplicação nas partes que a 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 ambientes de gerenciamento são usados para configurar o ambiente. Por exemplo, instâncias do Amazon EC2, bancos de dados do Amazon RDS e operações de leitura/gravação de tabela do Amazon DynamoDB são operações no 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 no ambiente de gerenciamento. 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 ambientes de gerenciamento. 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 empregados 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. AWS Os arquitetos de soluções (SAs) 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 workload. 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 workload devem estar alinhadas à necessidade e à criticidade da empresa. Ao definir primeiro a estrutura de criticidade empresarial com RTO, RPO e disponibilidade específicos, é possível avaliar cada workload. Essa abordagem exige que as pessoas envolvidas na implementação da workload tenham conhecimento da estrutura e do impacto que sua workload tem sobre as necessidades empresariais.

Fundamentos

Os requisitos fundamentais são aqueles que têm um escopo que vai além de uma única workload 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 data center.

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

As seções a seguir explicam práticas recomendadas que focam essas considerações para fins de confiabilidade.

Tópicos

Gerenciar cotas e restrições de serviço

Para arquiteturas de workload baseadas em nuvem, existem cotas de serviço (que também são chamadas de limites de serviço). Essas cotas existem para evitar o provisionamento acidental de mais recursos do que você precisa e para limitar as taxas de solicitação nas operações de API, a fim de proteger os serviços contra uso abusivo. Também há restrições de recursos, por exemplo, a taxa em que você pode propagar bits por um cabo de fibra óptica ou a quantidade de armazenamento em um disco físico.

Se você estiver usando aplicações do AWS Marketplace, precisará entender as limitações dessas aplicações. Se você estiver usando software como serviço ou serviços Web de terceiros, também precisará estar ciente desses limites.

Práticas recomendadas

- REL01-BP03 Acomodar restrições e cotas de serviço fixo por meio de 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

REL01-BP01 Conhecer as cotas e restrições de serviços

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

Resultado desejado: os clientes podem evitar a degradação ou interrupção do serviço em suas Contas da AWS implementando diretrizes adequadas para monitorar as principais métricas, análises de infraestrutura e etapas de correção de automação para verificar se as cotas e restrições de serviços não foram atingidas, o que poderia causar degradação ou interrupção do serviço.

Práticas comuns que devem ser evitadas:

- 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 respectivo tamanho necessário.
- Provisionar capacidade em excesso 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 de implementar esta prática recomendada: o monitoramento e o gerenciamento automatizado de cotas de serviço e restrições de recursos podem reduzir proativamente as falhas. 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 risco exposto se esta prática recomendada não for estabelecida: Alto

Orientação para implementação

O Service Quotas é um serviço da AWS que ajuda a gerenciar em um único local suas cotas para mais de 250 serviços da AWS. Além de pesquisar os valores de cotas, você também pode solicitar e rastrear 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 com controle de utilização, 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 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 Suporte 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, uma cota de serviço limita o número de instâncias do Amazon EC2 DL usadas em uma região. Esse limite pode ser alcançado durante um evento de escalação 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. Elas podem ser a capacidade de armazenamento (por exemplo, gp2 tem um limite de tamanho de 1 GB a 16 TB) ou o throughput de disco. É essencial que a restrição de um tipo de recurso seja projetada e avaliada constantemente quanto a tipos de uso que podem 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 Suporte para ver se há mitigações. Para obter mais detalhes sobre o ajuste de cotas fixas, consulte REL01-BP03 Acomodar restrições e cotas de serviço fixo por meio de 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 utilizados 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 de implementação

Para o Service Quotas:

- Revise 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 pelo Service Quotas. 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.
- Aumente essa análise de cotas com o AWS Config para encontrar todos os recursos da AWS usados em suas Contas da AWS.

- Use dados do AWS CloudFormation para determinar seus recursos da AWS usados. Veja os recursos que foram criados no AWS Management Console ou com o comando `list-stack-resources` da AWS CLI. Também é possível 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 em várias contas e regiões e REL01-BP04 Monitorar e gerenciar cotas) para alertar e informar se as cotas de serviços estão próximas ou atingiram seu limite.
- Estabeleça um método automatizado e programático para verificar se uma cota de serviço foi alterada em uma região, mas não em outras regiões da mesma conta (consulte REL01-BP02 Gerenciar cotas de serviço em 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 procedimentos de engenharia para calcular a mudança necessária na cota (consulte REL01-BP05 Automatizar o gerenciamento de cotas) depois de identificar que cotas maiores são necessárias 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 procedimentos e processos existentes.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP02 Gerenciar cotas de serviço em várias contas e regiões
- REL01-BP03 Acomodar restrições e cotas de serviço fixo por meio de 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-BP04 Testar a resiliência com engenharia do caos

Documentos relacionados:

- Pilar Confiabilidade do AWS Well-Architected Framework: Disponibilidade
- AWS Service Quotas (antigamente conhecido como Limites de serviço)
- Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Limites de serviço)
- Monitor de limites da AWS em respostas da AWS
- Limites de serviço do Amazon EC2
- O que é o Service Quotas?
- Como solicitar um aumento da cota

- Endpoints e cotas de serviço
- Guia do usuário do Service Quotas
- Monitor de cotas para AWS
- Limites de isolamento de falhas da AWS
- Disponibilidade com redundância
- AWS para dados
- O que é integração contínua?
- O que é entrega contínua?
- Parceiro da APN: parceiros que podem ajudar no gerenciamento de configuração
- Gerenciar o ciclo de vida da conta em ambientes SaaS de conta por locatário na AWS
- Gerenciar e monitorar o controle de utilização de APIs em suas workloads
- Visualizar recomendações do AWS Trusted Advisor em grande escala com o AWS Organizations
- Automatizar aumentos de limites de serviço e suporte corporativo com o AWS Control Tower

Vídeos relacionados:

- AWS Live re:Inforce 2019: Service Quotas
- Visualizar e gerenciar cotas para serviços da AWS com o Service Quotas
- 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 em 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: serviços e aplicações não devem ser afetados pelo esgotamento da cota de serviço para configurações que abrangem contas ou regiões ou que tenham projetos de resiliência usando failover de zona, região ou conta.

Práticas comuns que devem ser evitadas:

- 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 reutilizar modelos de solicitação de cotas 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 de implementar esta prática recomendada: verificar se você pode lidar com sua carga atual em regiões ou contas 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 risco exposto se esta prática recomendada não for estabelecida: Alto

Orientação para 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 cada vez mais workloads abrangendo regiões devido à implementação de projetos usando as abordagens Ativo/Ativo, Ativo/Passivo – Quente, Ativo/Passivo – Frio e Ativo/Passivo – Luz piloto, é essencial entender todos os níveis de cota da região e da conta. 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 essas diferenças de forma contínua. 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.

Monitorar e avaliar o desvio da cota de serviço, a condição em que os limites da cota de serviço para uma cota específica são alterados em uma região e não em todas as regiões, é muito importante. 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 de implementação

- Analise os valores do Service Quotas que podem 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 caso qualquer desvio de cota de serviço tenha ocorrido 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 Conhecer as cotas e restrições de serviços
- REL01-BP03 Acomodar restrições e cotas de serviço fixo por meio de 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-BP04 Testar a resiliência com engenharia do caos

Documentos relacionados:

- Pilar Confiabilidade do AWS Well-Architected Framework: Disponibilidade
- AWS Service Quotas (antigamente conhecido como Limites de serviço)
- Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Limites de serviço)
- Monitor de limites da AWS em respostas da AWS
- Limites de serviço do Amazon EC2
- O que é o Service Quotas?
- Como solicitar um aumento da cota

- Endpoints e cotas de serviço
- Guia do usuário do Service Quotas
- Monitor de cotas para AWS
- Limites de isolamento de falhas da AWS
- Disponibilidade com redundância
- AWS para dados
- O que é integração contínua?
- O que é entrega contínua?
- Parceiro da APN: parceiros que podem ajudar no gerenciamento de configuração
- Gerenciar o ciclo de vida da conta em ambientes SaaS de conta por locatário na AWS
- Gerenciar e monitorar o controle de utilização de APIs em suas workloads
- Visualizar recomendações do AWS Trusted Advisor em grande escala com o AWS Organizations
- Automatizar aumentos de limites de serviço e suporte corporativo com o AWS Control Tower

Vídeos relacionados:

- AWS Live re:Inforce 2019: Service Quotas
- Visualizar e gerenciar cotas para serviços da AWS com o Service Quotas
- 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 restrições e cotas de serviço fixo por meio de arquitetura

Esteja ciente das cotas de serviço, das restrições de 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 controle de utilização para um gateway da API e conexões simultâneas de usuários com um banco de dados.

Resultado desejado: a aplicação ou o serviço funciona conforme o esperado em condições normais e de alto tráfego. Eles foram desenvolvidos para funcionar com as limitações referentes às restrições ficas ou cotas de serviço do recurso.

Práticas comuns que devem ser evitadas:

- 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 fora da realidade 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 observabilidade 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 de implementar esta prática recomendada: verificar se a aplicação será executada em todos os níveis de carga de serviços projetados sem interrupção ou degradação.

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

Orientação para 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 no design de uma aplicação.

Os limites rígidos são mostrados no console do Service Quotas. Se as colunas mostrarem `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 alternativos 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 severidade 1.

Depois que a aplicação for implantada em um ambiente de teste, estratégias devem ser usadas 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 de 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 baseados em 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 Conhecer as cotas e restrições de serviços
- REL01-BP02 Gerenciar cotas de serviço em 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-BP04 Testar a resiliência com engenharia do caos

Documentos relacionados:

- Pilar Confiabilidade do AWS Well-Architected Framework: Disponibilidade
- AWS Service Quotas (antigamente conhecido como Limites de serviço)
- Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Limites de serviço)
- Monitor de limites da AWS em respostas da AWS
- Limites de serviço do Amazon EC2
- O que é o Service Quotas?
- Como solicitar um aumento da cota
- Endpoints e cotas de serviço
- Guia do usuário do Service Quotas
- Monitor de cotas para AWS
- Limites de isolamento de falhas da AWS
- Disponibilidade com redundância
- AWS para dados
- O que é integração contínua?

- O que é entrega contínua?
- Parceiro da APN: parceiros que podem ajudar no gerenciamento de configuração
- Gerenciar o ciclo de vida da conta em ambientes SaaS de conta por locatário na AWS
- Gerenciar e monitorar o controle de utilização de APIs em suas workloads
- Visualizar recomendações do AWS Trusted Advisor em grande escala com o AWS Organizations
- Automatizar aumentos de limites de serviço e suporte corporativo com o AWS Control Tower
- Ações, recursos e chaves de condição para o Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019: Service Quotas
- Visualizar e gerenciar cotas para serviços da AWS com o Service Quotas
- Demonstração de cotas do AWS IAM
- AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e 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 para possibilitar o crescimento planejado do uso.

Resultado desejado: sistemas ativos e automatizados que gerenciam e monitoram foram implantados. Essas soluções de operações garantem que os limites de uso da cota estejam próximos de ser atingidos. Isso seria corrigido proativamente por mudanças solicitadas na cota.

Práticas comuns que devem ser evitadas:

- 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 apenas alarmes 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 gerenciamento de cotas em várias regiões.

Benefícios de implementar esta prática recomendada: o rastreamento automático das cotas de serviço da AWS e o monitoramento do seu uso em relação a essas cotas permitirão que você veja quando está perto de atingir uma cota. Também é possível usar esse monitoramento de dados para ajudar a limitar qualquer dano resultante da exaustão da cota.

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

Orientação para 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 invocados via AWS Config, funções do Lambda, Amazon CloudWatch ou AWS Trusted Advisor. Você também pode usar

filtros de métrica no CloudWatch Logs para pesquisar e extrair padrões nos logs para determinar se o uso está se aproximando dos limites de cota.

Etapas de implementação

Para monitoramento:

- Capture o consumo atual de recursos (por exemplo, buckets ou instâncias). Use as operações de API de serviço, como a API `DescribeInstances` do Amazon EC2, 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 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 a gerenciar em um único local suas cotas para mais de 250 serviços da AWS.
- 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 do Lambda agendada para verificar as cotas de serviço nas regiões e comparar as diferenças.
- Manual: verifique as cotas de serviço via AWS CLI, API ou Console da AWS para conferir as cotas de serviço nas regiões e comparar as diferenças. Informe as diferenças.
- Se diferenças nas cotas entre as regiões forem identificadas, solicite uma mudança na cota, se necessário.

- Revise o resultado de todas as solicitações.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP01 Conhecer as cotas e restrições de serviços
- REL01-BP02 Gerenciar cotas de serviço em várias contas e regiões
- REL01-BP03 Acomodar restrições e cotas de serviço fixo por meio de 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-BP04 Testar a resiliência com engenharia do caos

Documentos relacionados:

- Pilar Confiabilidade do AWS Well-Architected Framework: Disponibilidade
- AWS Service Quotas (antigamente conhecido como Limites de serviço)
- Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Limites de serviço)
- Monitor de limites da AWS em respostas da AWS
- Limites de serviço do Amazon EC2
- O que é o Service Quotas?
- Como solicitar um aumento da cota
- Endpoints e cotas de serviço
- Guia do usuário do Service Quotas
- Monitor de cotas para AWS
- Limites de isolamento de falhas da AWS

- Disponibilidade com redundância
- AWS para dados
- O que é integração contínua?
- O que é entrega contínua?
- Parceiro da APN: parceiros que podem ajudar no gerenciamento de configuração
- Gerenciar o ciclo de vida da conta em ambientes SaaS de conta por locatário na AWS
- Gerenciar e monitorar o controle de utilização de APIs em suas workloads
- Visualizar recomendações do AWS Trusted Advisor em grande escala com o AWS Organizations
- Automatizar aumentos de limites de serviço e suporte corporativo com o AWS Control Tower
- Ações, recursos e chaves de condição para o Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019: Service Quotas
- Visualizar e gerenciar cotas para serviços da AWS com o Service Quotas
- Demonstração de cotas do AWS IAM
- AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e 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

As cotas de serviço, também chamadas de limites nos serviços da AWS, são os valores máximos para os recursos na sua Conta da AWS. Cada serviço da AWS define um conjunto de cotas e respectivos valores padrão. Para que a workload tenha acesso a todos os recursos necessários, talvez seja necessário aumentar os valores de cota de serviço.

O crescimento no consumo de recursos da AWS pela workload pode ameaçar a estabilidade da workload e afetar a experiência do usuário se as cotas forem excedidas. Implemente ferramentas para alertar você quando a workload se aproximar dos limites e considere criar solicitações de aumento de cotas automaticamente.

Resultado desejado: as cotas são configuradas adequadamente para as workloads em execução em cada Conta da AWS e região.

Práticas comuns que devem ser evitadas:

- Deixar de considerar e ajustar as cotas de forma adequada para atender aos requisitos da workload.
- Monitorar as cotas e o uso usando métodos que podem ficar desatualizados, como planilhas.
- Atualizar os limites de serviço apenas em programações periódicas.
- A organização carece de processos operacionais para revisar as cotas existentes e solicitar aumentos na cota de serviço quando necessário.

Benefícios de implementar essa prática recomendada:

- Resiliência aprimorada da workload: você evita erros causados pela superação das cotas de recursos da AWS.
- Recuperação de desastres simplificada: você pode reutilizar mecanismos automatizados de gerenciamento de cotas incorporados na região primária durante a configuração de DR em outra Região da AWS.

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

Orientação para implementação

Visualize as cotas atuais e acompanhe o consumo contínuo de cotas por meio de mecanismos, como o console do AWS Service Quotas, a AWS Command Line Interface (AWS CLI) e AWS SDKs.

Você também pode integrar sistemas de bancos de dados de gerenciamento de configuração (CMDB) e de gerenciamento de serviços de TI (ITSM) com as APIs do AWS Service Quotas.

Gere alertas automatizados se o uso da cota atingir os limites definidos e defina um processo para enviar solicitações de aumento de cota ao receber alertas. Se a workload subjacente for fundamental para os negócios, você pode automatizar as solicitações de aumento de cotas, mas teste cuidadosamente a automação para evitar o risco de ações descontroladas, como um ciclo de feedback aumentado.

Aumentos menores de cota geralmente são aprovados automaticamente. Solicitações de cotas maiores podem precisar ser processadas manualmente pelo AWS Support e podem levar mais tempo para serem analisadas e processadas. Reserve mais tempo para processar várias solicitações ou solicitações de grandes aumentos.

Etapas de implementação

- Implemente o monitoramento automatizado das cotas de serviço e emita alertas se o crescimento da utilização de recursos pela workload se aproximar dos limites da cota. Por exemplo, o Quota Monitor for AWS pode fornecer monitoramento automatizado das cotas de serviço. Essa ferramenta se integra ao AWS Organizations e é implantada usando o CloudFormation StackSets para que novas contas sejam monitoradas automaticamente na criação.
- Use recursos, como os modelos de solicitação do Service Quotas ou o AWS Control Tower, para simplificar a configuração do Service Quotas para novas contas.
- Crie painéis do uso atual da cota de serviço em todas as regiões e Contas da AWS e consulte-os conforme necessário para evitar exceder as cotas. O painel Trusted Advisor Organizational (TAO), parte do Cloud Intelligence Dashboards, pode ajudar você a começar a usar esses painéis rapidamente.
- Acompanhe as solicitações de aumento do limite de serviço. O Consolidated Insights from Multiple Accounts (CIMA) pode fornecer uma visão ao nível organizacional de todas as solicitações.
- Teste a geração de alertas e as solicitações de aumento de cota definindo limites de cota mais baixos em contas que não sejam de produção. Não conduza esses testes em uma conta de produção.

Recursos

Práticas recomendadas relacionadas:

- OPS10-BP07 Automatizar respostas a eventos

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudar no gerenciamento da configuração
- AWS Marketplace: produtos CMDB que ajudam a rastrear os limites
- AWS Service Quotas (antigamente conhecido como Limites de serviço)
- Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Limites de serviço)
- Quota Monitor Solution on AWS - AWS Solution
- O que é o Service Quotas?
- O que são modelos de solicitação do Service Quotas?

Vídeos relacionados:

- AWS Live re:Inforce 2019: Service Quotas
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Ferramentas relacionadas:

- Monitor de cotas para AWS

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

Este artigo explica como manter uma distância entre a cota do recurso e seu uso e como isso pode beneficiar sua organização. Quando você termina de usar um recurso, a cota de uso pode continuar contabilizando esse recurso. Isso pode resultar em falha ou em um recurso inacessível. Previna a falha do recurso verificando se as cotas abrangem a sobreposição de recursos inacessíveis e suas substituições. Considere 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 nos recursos ou na acessibilidade dos recursos podem ser cobertas dentro dos limites atuais do serviço. As falhas de zona, falhas de rede ou até mesmo falhas regionais foram consideradas no planejamento de recursos.

Práticas comuns que devem ser evitadas:

- 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 de implementar esta prática recomendada: quando eventos de interrupção do serviço afetam a disponibilidade da aplicação, use a nuvem para implementar estratégias para se recuperar desses eventos. Um exemplo de estratégia é criar recursos adicionais para substituir recursos inacessíveis e acomodar condições de failover sem esgotar seu limite de serviço.

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

Orientação para implementação

Ao avaliar os limites de cota, considere casos de failover que podem ocorrer devido a algum dano. Considere os seguintes casos de falha:

- Uma VPC interrompida ou inacessível.
- Uma sub-rede inacessível.
- Uma zona de disponibilidade degradada que afeta a acessibilidade dos recursos.
- Rotas de rede ou pontos de ingresso e egresso são bloqueados ou alterados.
- Uma região degradada que afeta a acessibilidade dos recursos.
- Um subconjunto de recursos afetados por uma falha em uma região ou zona de disponibilidade.

A decisão de fazer o failover é única para cada situação, já que o impacto na empresa pode variar drasticamente. Aborde o planejamento da capacidade dos recursos no local de failover e as cotas dos recursos antes de decidir fazer o failover de uma aplicação ou serviço.

Considere picos de atividade acima do normal ao revisar as cotas de cada serviço. Esses picos podem estar relacionados a recursos que estão inacessíveis por questões de rede ou permissões, mas ainda estão ativos. Os recursos ativos não encerrados são contabilizados no limite de cota do serviço.

Etapas de implementação

- Mantenha distância suficiente entre a cota de serviço e o uso máximo para acomodar um failover ou uma perda de acessibilidade.
- Determine suas cotas de serviço. Considere os padrões típicos de implantação, os requisitos de disponibilidade e o crescimento do consumo.
- Solicite aumentos de cota, se necessário. Preveja um tempo de espera para a solicitação de aumento de cota.
- Determine os requisitos de confiabilidade (também conhecidos como "número de noves").
- Entenda possíveis cenários de falha, como perda de um componente, zona de disponibilidade ou região.
- Estabeleça a metodologia de implantação (por exemplo, canário, azul/verde, vermelho/preto ou gradual).
- Inclua uma reserva adequada do limite atual. Um exemplo de buffer pode ser de 15%.
- 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 usar reservas de capacidade sob demanda para programar a capacidade à frente de qualquer failover. Isso pode ser uma estratégia útil durante os cronogramas empresariais mais críticos a fim de reduzir possíveis riscos de obter a quantidade e o tipo certo de recursos durante o failover.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP01 Conhecer as cotas e restrições de serviços
- REL01-BP02 Gerenciar cotas de serviço em várias contas e regiões
- REL01-BP03 Acomodar restrições e cotas de serviço fixo por meio de 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-BP04 Testar a resiliência com engenharia do caos

Documentos relacionados:

- Pilar Confiabilidade do AWS Well-Architected Framework: Disponibilidade
- AWS Service Quotas (antigamente conhecido como Limites de serviço)
- Verificações de práticas recomendadas do AWS Trusted Advisor (consulte a seção Limites de serviço)
- Monitor de limites da AWS em respostas da AWS
- Limites de serviço do Amazon EC2
- O que é o Service Quotas?
- Como solicitar um aumento da cota
- Endpoints e cotas de serviço
- Guia do usuário do Service Quotas
- Monitor de cotas para AWS
- Limites de isolamento de falhas da AWS
- Disponibilidade com redundância
- AWS para dados
- O que é integração contínua?
- O que é entrega contínua?
- Parceiro da APN: parceiros que podem ajudar no gerenciamento de configuração
- Gerenciar o ciclo de vida da conta em ambientes SaaS de conta por locatário na AWS
- Gerenciar e monitorar o controle de utilização de APIs em suas workloads
- Visualizar recomendações do AWS Trusted Advisor em grande escala com o AWS Organizations
- Automatizar aumentos de limites de serviço e suporte corporativo com o AWS Control Tower
- Ações, recursos e chaves de condição para o Service Quotas

Vídeos relacionados:

- AWS Live re:Inforce 2019: Service Quotas
- Visualizar e gerenciar cotas para serviços da AWS com o Service Quotas
- Demonstração de cotas do AWS IAM
- AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e 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

Planeje sua topologia de rede

Geralmente, existem workloads em vários ambientes. Isso inclui vários ambientes de nuvem (acessíveis ao público e privados) e, possivelmente, a 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 privada e isolada da Nuvem AWS em que é possível executar recursos da AWS em uma rede virtual.

Práticas recomendadas

- REL02-BP01 Usar conectividade de rede altamente disponível em endpoints públicos de workloads
- REL02-BP02 Provisionar conectividade redundante entre redes privadas na nuvem e 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 de 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 em endpoints públicos de workloads

Construir 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), API gateways, balanceamento de carga ou proxies reversos.

Resultado desejado: é fundamental planejar, criar e operacionalizar a conectividade de rede altamente disponível para seus endpoints públicos. Se a workload se tornar 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.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, URLs de funções do AWS Lambda, APIs da AWS AppSync e Elastic Load Balancing (ELB): todos fornecem endpoints públicos altamente disponíveis. O Amazon Route 53 fornece um serviço de DNS altamente disponível para resolução de nomes de domínio para verificar se seus endereços de endpoints públicos podem ser resolvidos.

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

Práticas comuns que devem ser evitadas:

- 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 de DDoS representam um risco de obstruir o tráfego legítimo e reduzir a disponibilidade para os usuários.

Benefícios de implementar esta prática recomendada: projetar para uma conectividade de rede altamente disponível e resiliente garante que sua workload permaneça acessível e disponível para seus usuários.

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

Orientação para implementação

No centro da criação de 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) altamente disponível e dimensionável, como o Amazon Route 53 para gerenciar os registros de DNS do seu 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 sua workload está altamente disponível, use o Elastic Load Balancing (ELB). O Amazon Route 53 pode ser usado para direcionar o tráfego para o ELB, que distribui o tráfego para as instâncias computacionais 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 ativo/ativo de vários sites, a workload pode atender ao tráfego de várias regiões. Com um padrão ativo/passivo de vários sites, a workload atende ao tráfego da região ativa enquanto os dados são replicados para a região secundária e se tornam ativos no caso de uma falha na região primária. 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 sua workload está acessível e disponível para seus 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 à medida que a carga do conteúdo é transferida dos seus 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 a performance 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 a melhor performance 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 contra ataques de DDoS, a AWS oferece o AWS Shield Standard. O Shield Standard é ativado automaticamente e protege contra ataques comuns de infraestrutura (camadas 3 e 4), como inundações SYN/UDP e ataques de reflexão, para oferecer suporte à alta disponibilidade de 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 usar o AWS Shield Advanced. Para proteção contra ataques de camada de 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 de implementação

1. Configure o DNS altamente disponível: o Amazon Route 53 é um serviço Web de Sistema de Nomes de Domínio (DNS) altamente disponível e escalável. O Route 53 conecta solicitações de usuários a aplicações da Internet executadas na AWS ou on-premises. Para obter mais informações, consulte Como configurar o Amazon Route 53 como seu serviço de 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 Amazon Route 53 e configurando o failover de DNS. É importante levar em consideração os seguintes aspectos ao configurar verificações de integridade:
    a. Como o Amazon Route 53 determina a integridade de uma verificação de integridade
    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 para seu tráfego, crie um registro de alias usando o Amazon Route 53 que aponta para o endpoint regional do seu balanceador de carga. Durante a criação do registro de alias, defina a opção "Avaliar integridade do destino" como "Sim".
    b. Para workloads sem servidor ou APIs privadas quando o API Gateway é usado, use o Route 53 para direcionar o tráfego para o API Gateway.
4. Decida sobre uma rede de entrega de conteúdo.
    a. Para entregar conteúdo usando pontos de presença mais próximos do usuário, comece entendendo como o CloudFront entrega conteúdo.
    b. Aprenda os conceitos básicos de 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 o armazenamento em cache funciona 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 escudo de origem do Amazon CloudFront

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, analise como o AWS WAF funciona e, quando você estiver pronto para implementar proteções contra inundações de HTTP POST E GET na camada de aplicação, consulte Conceitos básicos do AWS WAF. Você também pode 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 executados no Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator e Amazon Route 53, você pode considerar o AWS Shield Advanced e revisar exemplos de arquiteturas resilientes a DDoS. Para proteger sua workload e seus endpoints públicos contra ataques de DDoS, consulte Conceitos básicos do AWS Shield Advanced.

Recursos

Práticas recomendadas relacionadas:

- REL10-BP01 Implantar a workload em 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 da APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede
- O que é o AWS Global Accelerator?
- O que é o Amazon CloudFront?
- O que é o Amazon Route 53?

- O que é Elastic Load Balancing?
- Recurso de conectividade de rede: estabelecer suas bases da nuvem
- O que é o Amazon API Gateway?
- O que são AWS WAF, AWS Shield e AWS Firewall Manager?
- What is Amazon Application Recovery Controller?
- Configurar verificações de integridade personalizadas para failover de DNS

Vídeos relacionados:

- AWS re:Invent 2022: Aprimorar a performance e a disponibilidade com o AWS Global Accelerator
- AWS re:Invent 2020: Gerenciamento de tráfego global com o Amazon Route 53
- AWS re:Invent 2022: Operar aplicações Multi-AZ altamente disponíveis
- AWS re:Invent 2022: Mergulho profundo na infraestrutura de rede da AWS
- AWS re:Invent 2022: Construir redes resilientes

Exemplos relacionados:

- Disaster Recovery with Amazon Application Recovery Controller (ARC)
- Workshop do AWS Global Accelerator

REL02-BP02 Provisionar conectividade redundante entre redes privadas na nuvem e 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.

Práticas comuns que devem ser evitadas:

- 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 implementar esta prática recomendada: ao implementar conectividade redundante entre seu ambiente de nuvem e o ambiente corporativo ou on-premises, você pode garantir que os serviços dependentes entre os dois ambientes possam se comunicar de forma confiável.

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

Orientação para implementação

Ao usar o AWS Direct Connect para conectar sua rede on-premises à AWS, é possível atingir a resiliência máxima 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 de 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.
- Use AWS Transit Gateway, quando possível, para encerrar sua conexão VPN.
- Avalie os appliances do AWS Marketplace para acabar com as VPNs ou estender 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:

- Recomendações de resiliência da AWS Direct Connect
- Usar conexões VPN site a site redundantes para realizar failover
- Políticas de roteamento e comunidades BGP
- Configurações Ativas/Ativas e Ativas/Passivas no AWS Direct Connect
- Parceiro da APN: parceiros que podem ajudar a planejar sua rede

- AWS Marketplace para infraestrutura de rede
- Whitepaper de opções de conectividade da Amazon Virtual Private Cloud
- Construção de uma infraestrutura de rede da AWS Multi-VPC escalável e segura
- Usar conexões VPN site a site redundantes para realizar failover
- Usar o Kit de ferramentas de resiliência do AWS Direct Connect para começar
- Endpoints da VPC e serviços de endpoint da VPC (AWS PrivateLink)
- O que é Amazon VPC?
- O que é um gateway de trânsito?
- O que é AWS Site-to-Site VPN?
- Trabalhar com gateways Direct Connect

Vídeos relacionados:

- AWS re:Invent 2018: Design de VPCs avançadas e novos recursos para Amazon VPC
- AWS re:Invent 2019: Arquiteturas de referência do AWS Transit Gateway para muitas VPCs

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

Os 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 balanceadores de carga, instâncias do EC2 e aplicações baseadas 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 expansão futura.
- 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 obter mais informações, consulte 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.

Práticas comuns que devem ser evitadas:

- Deixar de considerar o crescimento futuro, o que resulta em 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 implementar esta prática recomendada: isso garante que você possa acomodar o crescimento das suas workloads 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 regulatória e integração com outras pessoas. O crescimento pode ser subestimado, a conformidade regulatória 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 gateway de trânsito?
    - Conectividade com várias VPCs de região única
  - Determine se você precisa de rede segregada por questões regulatórias.
  - 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 usar o 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 do Well-Architected relacionadas:

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

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede
- Whitepaper de opções de conectividade da Amazon Virtual Private Cloud

- Conectividade de rede de alta disponibilidade de vários datacenters
- Conectividade com várias VPCs de região única
- O que é Amazon VPC?
- IPv6 na AWS
- IPv6 em arquiteturas de referência
- Amazon Elastic Kubernetes Service lança suporte a IPv6
- Recommendations for your VPC: Classic Load Balancers
- Sub-redes de zona de disponibilidade: Application Load Balancers
- Zonas de disponibilidade: Network Load Balancers

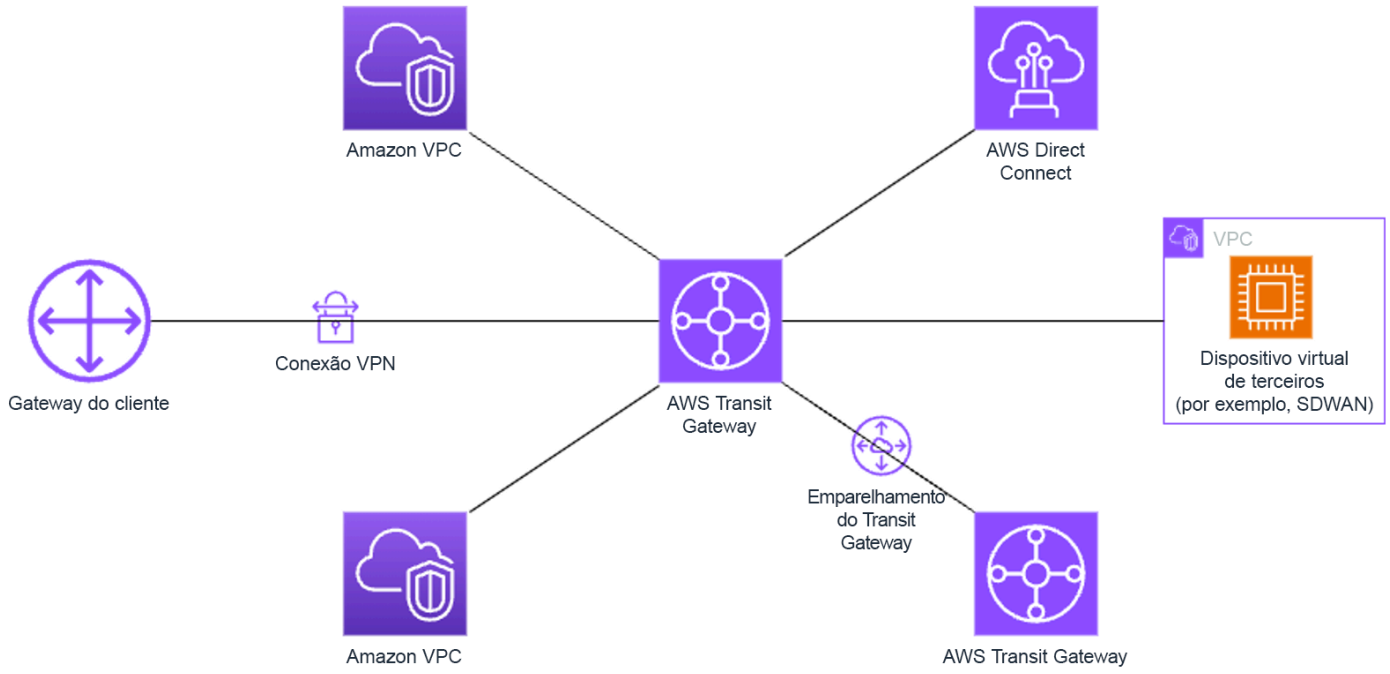
Vídeos relacionados:

- AWS re:Invent 2018: Design de VPCs avançadas e novos recursos para Amazon VPC (NET303)
- AWS re:Invent 2019: Arquiteturas de referência do AWS Transit Gateway para muitas VPCs (NET406-R1)
- AWS re:Invent 2023: AWS pronto para o que vem a seguir? Desenvolver redes para crescimento e flexibilidade (NET310)

REL02-BP04 Preferir topologias hub-and-spoke em vez de 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 na nuvem. A figura a seguir ilustra como você pode usar o AWS Transit Gateway para criar a topologia hub-and-spoke.



Resultado desejado: você conectou as nuvens privadas virtuais (VPCs) e redes on-premises por meio de um hub central. Você configura as conexões de emparelhamento por meio do hub, que atua como um roteador de nuvem altamente escalável. O roteamento é simplificado porque você não precisa trabalhar com relacionamentos de emparelhamento complexos. O tráfego entre redes é criptografado e você tem a capacidade de isolar redes.

Práticas comuns que devem ser evitadas:

- Criar regras complexas de emparelhamento de rede.
- Fornecer rotas entre redes que não devem se comunicar umas com as outras (por exemplo, workloads separadas que não têm interdependências).
- Há uma governança ineficaz da instância do hub.

Benefícios de implementar essa prática recomendada: à medida que o número de redes conectadas aumenta, o gerenciamento e a expansão da conectividade em malha se tornam cada vez mais desafiadores. Uma arquitetura de malha apresenta desafios adicionais, como componentes adicionais de infraestrutura, requisitos de configuração e considerações de implantação. A malha também introduz uma sobrecarga adicional para gerenciar e monitorar os componentes do plano de dados e do ambiente de gerenciamento. Você deve pensar em como fornecer alta disponibilidade da

arquitetura de malha, como monitorar a integridade e o desempenho da malha e como lidar com as atualizações dos componentes da malha.

Um modelo hub-and-spoke, por outro lado, estabelece o roteamento centralizado do tráfego em várias redes. Ele fornece uma abordagem mais simples para gerenciamento e monitoramento dos componentes do plano de dados e do ambiente de gerenciamento.

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

Orientação para implementação

Crie uma conta de serviços de rede se não houver uma. Coloque o hub na conta de Serviços de Rede da organização. Essa abordagem permite que o hub seja gerenciado centralmente por engenheiros de rede.

O hub do modelo hub-and-spoke atua como um roteador virtual para o tráfego que flui entre suas nuvens privadas virtuais (VPC) e redes on-premises. Essa abordagem reduz a complexidade da rede e facilita a solução de problemas de rede.

Considere o design da rede, incluindo as VPCs, o AWS Direct Connect e as conexões do Site-to-Site VPN que você deseja interconectar.

Considere usar uma sub-rede separada para cada anexo da VPC do gateway de trânsito. Para cada sub-rede, use um CIDR pequeno (por exemplo /28), para que você tenha mais espaço de endereço para recursos de computação. Adicionalmente, crie uma network ACL e associe-a a todas as sub-redes que estão associadas ao hub. Mantenha a ACL de rede aberta nas direções de entrada e saída.

Projete e implemente as tabelas de rotas de modo que as rotas sejam fornecidas somente entre redes que devem se comunicar. Omita rotas entre redes que não devem se comunicar umas com as outras (por exemplo, entre workloads separadas que não têm interdependências).

Etapas de implementação

1. Planeje sua rede. Determine quais redes você deseja conectar e verifique se elas não compartilham intervalos CIDR sobrepostos.
2. Crie um AWS Transit Gateway e conecte as VPCs.
3. Se necessário, crie conexões VPN ou gateways Direct Connect e associe-os ao gateway de trânsito.

4. Defina como o tráfego é direcionado entre as VPCs conectadas e outras conexões por meio da configuração das tabelas de rotas do gateway de trânsito.
5. Use o Amazon CloudWatch para monitorar e ajustar as configurações conforme necessário para otimizar a performance e os custos.

Recursos

Práticas recomendadas relacionadas:

- REL02-BP03 Garantir contas de alocação de sub-rede IP para expansão e disponibilidade
- REL02-BP05 Aplicar intervalos de endereços IP privados não sobrepostos a todos os espaços de endereços privados onde estão conectados

Documentos relacionados:

- O que é um gateway de trânsito?
- Melhores práticas de design do gateway de trânsito
- Construção de uma infraestrutura de rede da AWS Multi-VPC escalável e segura
- Criar uma rede global usando o emparelhamento entre regiões do AWS Transit Gateway
- Opções de conectividade da Amazon Virtual Private Cloud
- Parceiro da APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede

Vídeos relacionados:

- AWS re:Invent 2023: Fundamentos de rede na AWS
- AWS re:Invent 2023: Designs e novos recursos de VPCs avançadas

Workshops relacionados:

- Workshop do AWS Transit Gateway

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 o ajuste de escala da infraestrutura de rede para atender ao crescimento e às mudanças nos requisitos de rede.

Práticas comuns que devem ser evitadas:

- Usar o mesmo intervalo de IPs na VPC que você tem on-premises, na rede corporativa ou em outros provedores de nuvem
- Não rastrear 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 implementar esta prática recomendada: o planejamento ativo da rede garantirá que você não tenha várias ocorrências do mesmo endereço IP em redes interconectadas. Isso evita que problemas de roteamento ocorram em partes da workload que usam as diferentes aplicações.

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

Orientação para implementação

Use um IPAM, como o Gerenciador de endereços IP da Amazon VPC, para monitorar e gerenciar seu uso do 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 de 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 Gerenciador de endereços IP da Amazon VPC 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 Gerenciador de endereços IP da Amazon VPC para descobrir recursos.
- Registre o uso atual.
- Determine se você criou intervalos de IPs sobrepostos.
- 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 gateway NAT privado ou AWS PrivateLink se precisar conectar os intervalos sobrepostos.

Recursos

Práticas recomendadas relacionadas:

- Proteção de redes

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudar a planejar sua rede
- AWS Marketplace para infraestrutura de rede
- Whitepaper de opções de conectividade da Amazon Virtual Private Cloud
- Conectividade de rede de alta disponibilidade de vários datacenters
- Conectar redes com intervalos de IP sobrepostos
- O que é Amazon VPC?
- O que é IPAM?

Vídeos relacionados:

- AWS re:Invent 2023: Designs e novos recursos de VPCs avançadas

- AWS re:Invent 2019: Arquiteturas de referência do AWS Transit Gateway para muitas VPCs
- AWS re:Invent 2023: Pronto para o que vem a seguir? Desenvolver redes para crescimento e flexibilidade
- AWS re:Invent 2021: {Novo lancamento} Gerenciar seus endereços IP em grande escala na AWS

Arquitetura da workload

Uma workload confiável começa com as decisões iniciais de projeto que envolvem tanto o software quanto a infraestrutura. Suas decisões de arquitetura afetarão o comportamento da workload em todos os cinco pilares do Well-Architected. Para atingir a confiabilidade, há padrões específicos que devem ser seguidos.

As seções a seguir explicam as práticas recomendadas a serem usadas com esses padrões para fins de confiabilidade.

Tópicos

Projete a arquitetura de serviço da workload

Use uma arquitetura orientada a serviços (SOA) ou uma arquitetura de microsserviços para criar workloads altamente escaláveis e confiáveis. A arquitetura orientada a serviços (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 de arquitetura orientada a serviços (SOA) usam padrões de comunicação comuns para que possam ser rapidamente incorporadas a novas workloads. 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 voltados para 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 da 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 ser 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 ajuste de escala a partir das necessidades iniciais. Ao refatorar um monólito existente, será necessário considerar o quanto a aplicação poderá 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.

Práticas comuns que devem ser evitadas:

- 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 de estabelecer esta prática:

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

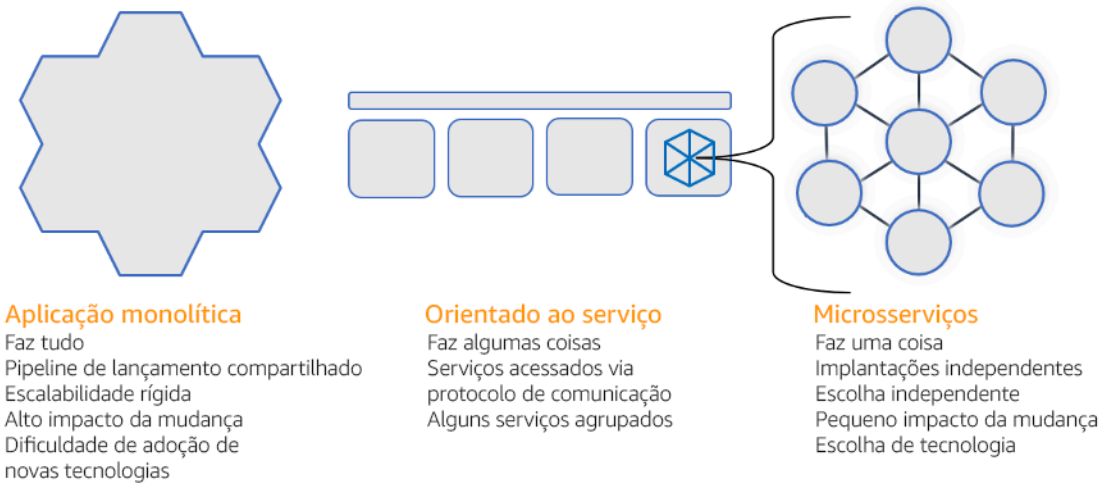
- Os componentes da aplicação podem ter requisitos de disponibilidade diferentes, aos quais uma segmentação mais atômica pode oferecer suporte.
- Responsabilidades bem definidas para as equipes que oferecem suporte à workload.

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

Orientação para 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, é necessário 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 de 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 workload aceitá-la e sua organização puder sustentá-la, use uma arquitetura de microsserviços para obter a melhor agilidade e confiabilidade. Para obter mais informações, consulte Implementar 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 novos serviços e aplicações. O AWS Migration Hub Refactor Spaces atua como ponto de partida para a 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. O AWS App Mesh pode ser usado com arquiteturas orientadas a serviços para fornecer descoberta e acesso confiáveis aos serviços. O AWS Cloud Map também pode ser usado para descoberta dinâmica de serviços baseada em DNS.
- Se você estiver migrando de um monólito para SOA, o Amazon MQ poderá ajudar a preencher a lacuna como um barramento de serviço ao redesenhar 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 Do SQL ao NoSQL.

Nível de esforço do plano de implementação: Alto

Recursos

Práticas recomendadas relacionadas:

- REL03-BP02 Criar serviços voltados para domínios e funcionalidades de negócios específicos

Documentos relacionados:

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

Exemplos relacionados:

- Workshop Modernização iterativa de aplicações

Vídeos relacionados:

- Como entregar excelência com microsserviços na AWS

REL03-BP02 Criar serviços voltados para domínios e funcionalidades de negócios específicos

A arquitetura orientada a serviços (SOA) define serviços com funções bem delineadas estabelecidas pelas 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 limitados, 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 por 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.

Práticas comuns que devem ser evitadas:

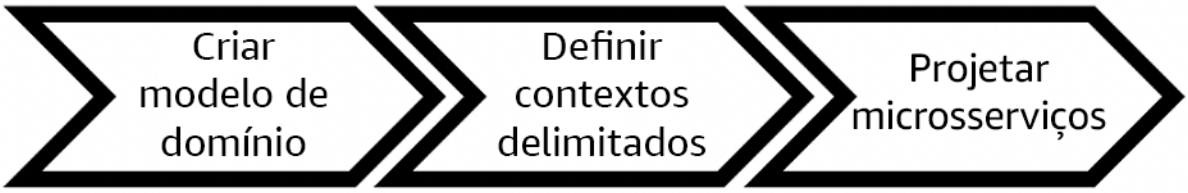
- As equipes são formadas em torno de domínios técnicos específicos, como UI e UX, middleware ou banco de dados, em vez de domínios empresariais específicos.
- As aplicações abrangem as responsabilidades do domínio. Serviços que abrangem contextos delimitados podem ser mais difíceis de manter, exigir 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 uma 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 implementar 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

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



Design orientado por domínio

Etapas de 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 funções de domínio forem formadas em um contexto de domínio, você poderá dividir seu domínio em serviços usando contexto limitado em que entidades que compartilham características 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, cronograma, 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.
- A decomposição de 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 limitado é estabelecido usando um mapeamento e coordenação de serviços, ou camada corrompimento, que protege o modelo de domínio recém-definido contra 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 simulem as regras de negócios do seu domínio. O desenvolvimento orientado por testes (TDD) e o desenvolvimento orientado por comportamento

(BDD) ajudam as equipes a manter os serviços voltados para a solução de problemas de negócios.

- Selecione os serviços da AWS que melhor atendem aos requisitos de domínio da sua empresa e à arquitetura de microsserviços:
  - A tecnologia sem servidor da AWS permite que sua equipe enfoque a lógica de domínio específica em vez de gerenciar servidores e infraestrutura.
  - Os contêineres na AWS simplificam o gerenciamento de sua infraestrutura para que você possa enfocar nos requisitos de domínio.
  - Os bancos de dados com propósito específico ajudam você a adequar seus requisitos de domínio ao tipo de banco de dados mais adequado.
- Criar 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:

- Microsserviços da AWS
- Implementar microsserviços na AWS
- Como dividir um monólito em microsserviços
- Conceitos básicos do DDD quando cercado por sistemas herdados
- 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 por comportamento

Exemplos relacionados:

- Como projetar microsserviços nativos da nuvem na AWS (do DDD/EventStormingWorkshop)

Ferramentas relacionadas:

- Bancos de dados na 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: as aplicações criadas com arquiteturas orientadas a serviços ou de microsserviços podem operar de forma independente e, ao mesmo tempo, ter uma dependência de tempo de execução 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.

Práticas comuns que devem ser evitadas:

- Criação de APIs de serviço sem esquemas altamente tipificados. Isso resulta em APIs que não podem ser usadas para gerar vinculações de API e payloads que não podem 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 implementar 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 o Amazon API Gateway, o AWS AppSync e o Amazon EventBridge, em sua arquitetura para usar contratos de serviços de API em sua aplicação. O Amazon API Gateway ajuda você a se integrar diretamente com serviços da AWS nativos e outros serviços Web. O API Gateway é compatível com a especificação da OpenAPI e versionamento. O AWS AppSync é um endpoint gerenciado do GraphQL que você configura definindo um esquema do GraphQL para definir uma interface de serviço para consultas, mutações e assinaturas. O Amazon

EventBridge usa esquemas de eventos para definir eventos e gerar vinculações de código para seus eventos.

Etapas de 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.
  - Importar uma definição da 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).
  - Exportar 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 as APIs do GraphQL com o AWS AppSync definindo um arquivo de esquema do GraphQL para gerar sua interface de contrato e simplificar a interação com modelos REST complexos, várias tabelas de banco de dados ou serviços legados.
- Os 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 solicitação e resposta 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 voltados para domínios e funcionalidades de negócios específicos
- REL04-BP02 Implementar dependências com acoplamento fraco
- REL05-BP03 Controlar e limitar chamadas de novas tentativas
- REL05-BP05 Definir tempos limite do cliente

Documentos relacionados:

- O que é uma API (interface de programação de aplicações)?
- Implementar 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 a OpenAPI
- Amazon API Gateway para a aplicação CRUD do Amazon DynamoDB usando a OpenAPI
- Padrões modernos de integração de aplicações em uma era sem servidor: integração do serviço do API Gateway

- Implementar o versionamento do API Gateway baseado em cabeçalho com o Amazon CloudFront
- AWS AppSync: como criar uma aplicação cliente

Vídeos relacionados:

- Usar OpenAPI na 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 de redes de comunicação para interconectar componentes, como servidores ou serviços. A workload deve operar de forma confiável, apesar da perda de dados ou da latência nessas redes. Os componentes do sistema distribuído devem operar de uma maneira que não afete negativamente outros componentes ou a workload. Essas práticas recomendadas evitam falhas e melhoram o tempo médio entre falhas (MTBF).

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 Garantir a idempotência das operações de mutação

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 uns com os outros fazendo chamadas síncronas de solicitação e resposta usando protocolos HTTP/S, REST ou de chamada de procedimento remoto (RPC). Os sistemas assíncronos se comunicam uns com os outros 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: crie uma workload que interaja efetivamente com dependências síncronas, assíncronas e em lote.

Práticas comuns que devem ser evitadas:

- 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 implementar esta prática recomendada: é comum que uma 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 risco exposto se esta prática recomendada não for estabelecida: Alto

Orientação para implementação

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

Orientações gerais

- 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 serviços de observabilidade da AWS para monitorar os tempos de resposta e as taxas de erro para garantir que sua dependência esteja fornecendo serviços nos níveis necessários para sua workload.
- Identifique os possíveis desafios que a workload pode enfrentar ao se comunicar com as dependências. Os sistemas distribuídos apresentam uma ampla variedade 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, ajuste de escala e atraso na replicação de dados.
- Implemente gerenciamento e registro de erros robustos para obter ajuda para solucionar problemas quando sua dependência apresentar 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 obter mais detalhes sobre o comportamento bimodal, consulte REL11-BP05 Usar estabilidade estática para evitar comportamento bimodal.
- Lembre-se 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 ver uma discussão útil sobre esse problema, consulte Ajustar as configurações de solicitação HTTP do AWS SDK Java para aplicações do Amazon DynamoDB com reconhecimento de latência.
- 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 sua workload se comunique com sua dependência enviando e recebendo mensagens por meio de um agente de mensagens. O streaming de eventos permite que sua workload e sua 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 é necessário 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 rastrear a última mensagem que leram.
  - Ordenação das mensagens: com o sistema de mensagens, não é garantido receber mensagens na ordem exata em que elas são enviadas, a menos que você use a abordagem FIFO (primeira a entrar, primeira a sair). O streaming de eventos sempre preserva a ordem na qual os dados foram produzidos.
  - Exclusão de mensagens: 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 sua workload invoca uma função do Lambda de forma assíncrona, o Lambda coloca o evento em uma fila e retorna uma resposta informando êxito, sem informações adicionais. Após a conclusão do

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

- Crie a workload para lidar com mensagens duplicadas utilizando 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 mensagens ou streaming reenviarão uma mensagem se ocorrer uma falha na rede ou se uma confirmação não for recebida.
- Se a workload não receber uma resposta da dependência, ela precisará reenviar a solicitação. Considere 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 de 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 serviços de mensagens e streaming de eventos. Você também pode instrumentar sua workload com o AWS X-Ray para obter insights rapidamente para solucionar problemas.

Dependência de 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 o Amazon Simple Storage Services (Amazon S3), o Amazon Elastic File System (Amazon EFS) e o Amazon FSx para Lustre, que permita que sua workload leia e grave arquivos em grande escala.
- Se sua workload precisar invocar vários trabalhos em lote, você poderá usar o AWS Step Functions para simplificar a orquestração de trabalhos em lote executados na AWS ou on-premises. Este projeto de exemplo demonstra a orquestração de trabalhos em lote usando Step Functions, o AWS Batch e 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 ambientes e trabalhos em AWS Batch. Nesse caso, a workload impediria o início do próximo trabalho e informaria a equipe relevante sobre a exceção.

Recursos

Documentos relacionados:

- Operações da Nuvem AWS: monitoramento e observabilidade
- Amazon Builders' Library: desafios com sistemas distribuídos
- REL11-BP05 Usar estabilidade estática para evitar comportamento bimodal
- Guia do desenvolvedor do AWS Lambda: tratamento de erros e novas tentativas automáticas no AWS Lambda
- Ajustar as configurações de solicitação HTTP do AWS SDK Java para aplicações do Amazon DynamoDB com reconhecimento de latência
- Sistema de mensagens da AWS
- O que é streaming de dados?
- Guia do desenvolvedor do AWS Lambda: invocação assíncrona
- Perguntas frequentes do Amazon Simple Queue Service: filas FIFO
- Guia do desenvolvedor do Amazon Kinesis Data Streams: tratar registros duplicados
- Guia do desenvolvedor do Amazon Simple Queue Service: métricas do CloudWatch disponíveis para Amazon SQS
- Guia do desenvolvedor do Amazon Kinesis Data Streams: monitorar o serviço Amazon Kinesis Data Streams com o Amazon CloudWatch
- Guia do desenvolvedor do AWS X-Ray: conceitos do AWS X-Ray
- 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: Observabilidade full-stack e monitoramento de aplicações com a AWS (COP310)

Ferramentas relacionadas:

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Service (Amazon S3)
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx para 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 balanceadores de carga, têm acoplamento fraco. O acoplamento fraco ajuda a isolar o comportamento de um componente de outros componentes que dependem dele, aumentando a resiliência e a agilidade.

Dependências de desacoplamento, como sistemas de filas, sistemas de streaming e fluxos de trabalho, ajudam a minimizar o impacto de alterações ou falhas em um sistema. Essa separação impede o comportamento de um componente de afetar outros que dependem dele, melhorando a resiliência e a agilidade.

Em sistemas fortemente acoplados, alterações em um componente podem exigir mudanças em outros componentes que dependem dele, o que resulta em performance degradada em todos eles. O acoplamento fraco interrompe essa dependência para que os componentes dependentes só precisem saber a interface versionada e publicada. A implementação de um acoplamento fraco 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 granular 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 acoplamento fraco, 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 Amazon Kinesis, ou o AWS Step Functions.

Figura 4: Dependências como sistemas de enfileiramento e balanceadores de carga têm acoplamento fraco

As filas do Amazon SQS e os AWS Step Functions são apenas duas maneiras de adicionar uma camada intermediária para acoplamento fraco. As arquiteturas orientadas a eventos também podem ser criadas na Nuvem AWS com 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 muitos para muitos. Usando tópicos do Amazon SNS, seus sistemas de publicadores 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 devem ser 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 de falha em um nível de componente, o que ajuda a diagnosticar e resolver problemas. Ela também simplifica os ciclos de desenvolvimento, permitindo que as equipes implementem mudanças em um nível modular sem impactar a performance 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.

Práticas comuns que devem ser evitadas:

- 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 com acoplamento forte, o que pode reintroduzir o acoplamento forte e impedir a escalabilidade.

- Ignorar a pressão contrária. 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 de implementar esta 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. Uma falha em um componente é isolada dos demais.

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

Orientação para 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.

- Crie arquiteturas orientadas a eventos: o Amazon EventBridge ajuda você a criar arquiteturas orientadas a eventos distribuídas e com acoplamento fraco.
- Implemente filas em sistemas distribuídos: é possível usar o Amazon Simple Queue Service (Amazon SQS) para integrar e desacoplar sistemas distribuídos.
- Containerize componentes na forma de microsserviços: os microsserviços permitem que as equipes criem aplicações formadas por 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 rápido.
- Gerencie fluxos de trabalho com Step Functions: o Step Functions ajuda você a coordenar vários serviços da AWS em fluxos de trabalho flexíveis.
- Utilize as arquiteturas de mensagens publicador-assinante (pub/sub): o Amazon Simple Notification Service (Amazon SNS) fornece entrega de mensagens de publicadores para assinantes (também conhecidos como produtores e consumidores).

Etapas de 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, um evento que aciona o próximo componente do sistema é gerado.

  - Criar aplicações orientadas por eventos com o Amazon EventBridge
  - AWS re:Invent 2022: Desenvolver integrações orientadas por eventos com o 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 fins de redundância.
  - Arquitetura básica do Amazon SQS
  - Envie mensagens entre aplicações distribuídas com o 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. Isso também permite o isolamento de comportamentos em um único componente em caso de alterações.
  - Implementar microsserviços na AWS
  - Vamos arquitetar! Arquitetar microsserviços com contêineres
- 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.
  - Criar um fluxo de trabalho sem servidor com o AWS Step Functions e o AWS Lambda
  - Conceitos básicos do AWS Step Functions

Recursos

Documentos relacionados:

- Amazon EC2: garantia da idempotência
- Amazon Builders' Library: desafios com sistemas distribuídos
- 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?

- Romper com seu monólito
- Organizar microsserviços baseados em filas com o AWS Step Functions e o Amazon SQS
- Arquitetura básica do Amazon SQS
- Arquitetura baseada em fila

Vídeos relacionados:

- AWS New York Summit 2019: Introdução a arquiteturas orientadas por eventos e ao Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e pequenos ARC337 (inclui acoplamento fraco, trabalho constante, estabilidade estática)
- AWS re:Invent 2019: Migrar para arquiteturas orientadas por eventos (SVS308)
- AWS re:Invent 2019: Aplicações sem servidor orientadas por eventos e escaláveis usando o Amazon SQS e o Lambda
- AWS re:Invent 2022: Desenvolver integrações orientadas por eventos com o Amazon EventBridge
- AWS re:Invent 2017: Mergulho profundo e práticas recomendadas do Elastic Load Balancing

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). Independentemente de nenhum servidor falhar ou todos eles, o sistema de verificação de integridade está realizando um trabalho constante sem alterações grandes e rápidas.

Por exemplo, se o sistema de verificação de integridade estiver monitorando 100 mil 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. 100.000 estados de integridade do servidor, cada um representado por um bit, seriam apenas uma carga útil de 12,5 KB. Independentemente de nenhum servidor ou falhar, ou se 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 verificações de integridade de endpoints (como endereços IP) para determinar como os usuários finais são roteados para eles.

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

Orientação para 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 balanceadores de carga, têm acoplamento fraco. O acoplamento fraco ajuda a isolar o comportamento de um componente de outros componentes que dependem dele, aumentando a resiliência e a agilidade.
  - Amazon Builders' Library: confiabilidade, trabalho constante e uma boa xícara de café
  - AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e pequenos ARC337 (inclui trabalho constante)
    - Para o exemplo de um sistema de verificação de integridade monitorando 100.000 servidores, crie workloads para que os tamanhos das cargas permaneçam constantes, independentemente do número de sucessos ou falhas.

Recursos

Documentos relacionados:

- Amazon EC2: garantia da idempotência
- Amazon Builders' Library: desafios com sistemas distribuídos
- Amazon Builders' Library: confiabilidade, trabalho constante e uma boa xícara de café

Vídeos relacionados:

- AWS New York Summit 2019: Introdução a arquiteturas orientadas por eventos e ao Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e pequenos ARC337 (inclui trabalho constante)
- AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e pequenos ARC337 (inclui acoplamento fraco, trabalho constante, estabilidade estática)

- AWS re:Invent 2019: Migrar para arquiteturas orientadas por eventos (SVS308)

REL04-BP04 Garantir a idempotência das operações de mutação

Um serviço idempotente garante que cada solicitação seja processada exatamente uma vez, de modo que fazer várias solicitações idênticas tenha o mesmo efeito que uma única solicitação. Isso 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, que 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, mesmo se o estado subjacente do sistema tiver mudado.

Em um sistema distribuído, é relativamente 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). Mas é difícil garantir que uma ação seja realizada exatamente uma vez, de modo que fazer várias solicitações idênticas tem o mesmo efeito que fazer uma única solicitação. Usando tokens de idempotência em APIs, os serviços podem receber uma solicitação mutante uma vez ou mais sem necessidade de criar registros duplicados nem efeitos colaterais.

Resultado desejado: você tem uma abordagem consistente, bem documentada e amplamente adotada para garantir a idempotência em todos os componentes e serviços.

Práticas comuns que devem ser evitadas:

- Aplicar a idempotência indiscriminadamente, mesmo quando não é necessária.
- Introduzir uma lógica excessivamente complexa para implementar a idempotência.
- Usar carimbos de data/hora como chaves para a idempotência. Isso pode causar imprecisões devido à distorção do relógio ou devido a vários clientes que usam os mesmos carimbos de data/hora para aplicar as alterações.
- Armazenar cargas úteis inteiras para fins de idempotência. Nessa abordagem, você salva cargas úteis de dados completas para cada solicitação e as substitui a cada nova solicitação. Isso pode degradar o desempenho e afetar a escalabilidade.
- Gerar chaves de maneira inconsistente nos serviços. Sem chaves consistentes, os serviços podem não reconhecer solicitações duplicadas, o que gera resultados indesejados.

Benefícios de implementar essa prática recomendada:

- Maior escalabilidade: o sistema pode lidar com novas tentativas e solicitações duplicadas sem precisar executar lógica adicional ou gerenciamento complexo de estados.
- Confiabilidade aprimorada: a idempotência ajuda os serviços a lidar com várias solicitações idênticas de maneira consistente, o que reduz o risco de efeitos colaterais indesejados ou registros duplicados. Isso é especialmente crucial em sistemas distribuídos, onde falhas e novas tentativas de rede são comuns.
- Consistência de dados aprimorada: como a mesma solicitação produz a mesma resposta, a idempotência ajuda a manter a consistência dos dados em sistemas distribuídos. Isso é essencial para manter a integridade das transações e operações.
- Tratamento de erros: os tokens de idempotência tornam o tratamento de erros mais simples. Se um cliente não receber uma resposta devido a um problema, ele poderá reenviar a solicitação com segurança com o mesmo token de idempotência.
- Transparência operacional: a idempotência permite um melhor monitoramento e registro. Os serviços podem registrar solicitações com seus tokens de idempotência, o que facilita o rastreamento e a depuração de problemas.
- Contrato de API simplificado: ele pode simplificar o contrato entre os sistemas do lado do cliente e do servidor e reduzir o medo de processamento incorreto de dados.

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

Orientação para implementação

Em um sistema distribuído, é relativamente fácil executar uma ação no máximo uma vez (o cliente faz apenas uma solicitação) ou pelo menos uma vez (o cliente continua solicitando até a confirmação do sucesso). No entanto, é difícil implementar um comportamento do tipo exatamente uma vez. Para conseguir isso, os clientes devem gerar e fornecer um token de idempotência para cada solicitação.

Ao usar tokens de idempotência, um serviço pode distinguir entre solicitações novas e repetidas. Quando um serviço recebe uma solicitação com um token de idempotência, ele verifica se o token já foi usado. Se o token tiver sido usado, o serviço recuperará e retornará a resposta armazenada. Se o token for novo, o serviço processará a solicitação, armazenará a resposta junto com o token e, em seguida, retornará a resposta. Esse mecanismo torna todas as respostas idempotentes, o que aumenta a confiabilidade e a consistência do sistema distribuído.

A idempotência também é um comportamento importante das arquiteturas orientadas por eventos. Essas arquiteturas geralmente são apoiadas por uma fila de mensagens, como Amazon SQS, Amazon MQ, Amazon Kinesis Streams ou Amazon Managed Streaming for Apache Kafka (MSK).

Em algumas circunstâncias, uma mensagem publicada somente uma vez pode ser entregue acidentalmente mais de uma vez. Quando um publicador gera e inclui tokens de idempotência nas mensagens, ele solicita que o processamento de qualquer mensagem duplicada recebida não resulte em uma ação repetida para a mesma mensagem. Os consumidores devem acompanhar cada token recebido e ignorar as mensagens que contêm tokens duplicados.

Os serviços e os consumidores também devem passar o token de idempotência recebido para qualquer serviço downstream que ele chame. Cada serviço downstream na cadeia de processamento é igualmente responsável por garantir que a idempotência seja implementada para evitar o efeito colateral de processar uma mensagem mais de uma vez.

Etapas de implementação

1. Identifique operações idempotentes

   Determine quais operações exigem idempotência. Elas geralmente incluem métodos HTTP POST, PUT e DELETE e operações de inserção, atualização ou exclusão do banco de dados. Operações que não alteram o estado, como consultas somente leitura, geralmente não exigem idempotência, a menos que tenham efeitos colaterais.
2. Usar identificadores exclusivos

   Inclua um token exclusivo em cada solicitação de operação de idempotência enviada pelo remetente, diretamente na solicitação ou como parte dos metadados (por exemplo, um cabeçalho HTTP). Isso permite que o destinatário reconheça e manipule solicitações ou operações duplicadas. Os identificadores comumente usados para tokens incluem Universally Unique Identifiers (UUIDs) e K-Sortable Unique Identifiers (KSUIDs).
3. Rastreie e gerencie o estado

   Mantenha o estado de cada operação ou solicitação na workload. Isso pode ser feito armazenando o token de idempotência e o estado correspondente (como pendente, concluído ou com falha) em um banco de dados, cache ou outro armazenamento persistente. Essas informações de estado permitem que a workload identifique e processe solicitações ou operações duplicadas.

   Mantenha a consistência e a atomicidade usando mecanismos de controle de concorrência apropriados, se necessário, como bloqueios, transações ou controles de simultaneidade otimistas. Isso inclui o processo de registrar o token de idempotência e executar todas as operações de mutação associadas ao atendimento da solicitação. Isso ajuda a evitar condições de corrida e verifica se as operações de idempotência são executadas corretamente.

Remova regularmente os tokens de idempotência antigos do datastore para gerenciar o armazenamento e o desempenho. Se o sistema de armazenamento oferecer suporte a isso, considere usar carimbos de data/hora de expiração para os dados (geralmente conhecidos como valores de vida útil, ou TTL). A probabilidade de reutilização do token de idempotência diminui com o tempo.

As opções de armazenamento da AWS normalmente usadas para armazenar tokens de idempotência e estados relacionados incluem:

- Amazon DynamoDB: O DynamoDB é um serviço de banco de dados NoSQL que fornece desempenho de baixa latência e alta disponibilidade, o que o torna adequado para o armazenamento de dados relacionados à idempotência. O modelo de dados de documentos e valores-chave do DynamoDB permite o armazenamento e a recuperação eficientes dos tokens de idempotência e das informações de estado associadas. O DynamoDB também pode expirar automaticamente os tokens de idempotência se a aplicação definir um valor de TTL ao inseri-los.
- Amazon ElastiCache: o ElastiCache pode armazenar tokens de idempotência com alto throughput, baixa latência e baixo custo. Tanto o ElastiCache (Redis) quanto o ElastiCache (Memcached) também podem expirar automaticamente os tokens de idempotência se a aplicação definir um valor de TTL ao inseri-los.
- Amazon Relational Database Service (RDS): você pode usar o Amazon RDS para armazenar tokens de idempotência e informações de estado relacionadas, especialmente se a aplicação já usa um banco de dados relacional para outros fins.
- Amazon Simple Storage Service (S3): o Amazon S3 é um serviço de armazenamento de objetos altamente escalável e durável que pode ser usado para armazenar tokens de idempotência e metadados relacionados. Os recursos de versionamento do S3 podem ser particularmente úteis para a manutenção do estado das operações de idempotência. A escolha do serviço de armazenamento geralmente depende de fatores como o volume de dados relacionados à idempotência, as características de desempenho necessárias, a necessidade de durabilidade e disponibilidade e como o mecanismo de idempotência se integra à arquitetura geral da workload.

4. Implemente operações de idempotência

Projete os componentes de API e workload para serem idempotentes. Incorpore verificações de idempotência nos componentes de workload. Antes de processar uma solicitação ou realizar uma operação, verifique se o identificador exclusivo já foi processado. Se já tiver sido processado,

retorne o resultado anterior em vez de executar a operação novamente. Por exemplo, se um cliente enviar uma solicitação para criar um usuário, verifique se já existe um usuário com o mesmo identificador exclusivo. Se o usuário existir, deverão ser retornadas as informações do usuário existente em vez de criar outro. Da mesma forma, se um consumidor da fila receber uma mensagem com um token de idempotência duplicado, ele deverá ignorá-la.

Crie conjuntos de testes abrangentes que validem a idempotência das solicitações. Eles devem abranger uma ampla variedade de cenários, como solicitações bem-sucedidas, solicitações malsucedidas e solicitações duplicadas.

Se a workload utilizar funções do AWS Lambda, considere o Powertools para AWS Lambda. O Powertools for AWS Lambda é um kit de ferramentas para desenvolvedores para implementar as práticas recomendadas da tecnologia sem servidor e aumentar a velocidade do desenvolvedor ao trabalhar com funções do AWS Lambda. Em particular, ele fornece um utilitário para converter funções do Lambda em operações de idempotência que podem ser repetidas com segurança.

5. Comunique a idempotência com clareza

   Documente a API e os componentes da workload para comunicar claramente a natureza de idempotência das operações. Isso ajuda os clientes a entender o comportamento esperado e a interagir com a workload de forma confiável.

6. Monitore e audite

   Implemente mecanismos de monitoramento e auditoria para detectar quaisquer problemas relacionados à idempotência das respostas, como variações inesperadas de respostas ou tratamento excessivo de solicitações duplicadas. Isso pode ajudar você a detectar e investigar quaisquer problemas ou comportamentos inesperados na workload.

Recursos

Práticas recomendadas relacionadas:

- REL05-BP03 Controlar e limitar chamadas de novas tentativas
- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP03 Enviar notificações (processamento e emissão de alarmes em tempo real)
- REL08-BP02 Integrar testes funcionais como parte da sua implantação

Documentos relacionados:

- Amazon Builders' Library: Como tornar as novas tentativas seguras com APIs idempotentes
- Amazon Builders' Library: desafios com sistemas distribuídos
- Amazon Builders' Library: confiabilidade, trabalho constante e uma boa xícara de café
- Amazon Elastic Container Service: Ensuring idempotency
- Como faço para tornar minha função do Lambda idempotente?
- Ensuring idempotency in Amazon EC2 API requests

Vídeos relacionados:

- Building Distributed Applications with Event-driven Architecture - AWS Online Tech Talks
- AWS re:Invent 2023 - Building next-generation applications with event-driven architecture
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2018 - Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019 - Moving to event-driven architectures (SVS308)

Ferramentas relacionadas:

- Idempotência com AWS Lambda Powertools (Java)
- Idempotência com AWS Lambda Powertools (Python)
- AWS LambdaPágina do Powertools no GitHub

Projete 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 workload deve operar de forma confiável, apesar da perda de dados ou da latência nessas redes. Os componentes do sistema distribuído devem operar de uma maneira que não afete negativamente outros componentes ou a workload. Essas práticas recomendadas permitem que as workloads resistam a estresses ou falhas, recuperem-se mais rapidamente e reduzam o impacto de possíveis prejuízos. Como resultado, o tempo médio para recuperação (MTTR) é melhorado.

Essas práticas recomendadas evitam falhas e melhoram o tempo médio entre falhas (MTBF).

Práticas recomendadas

- REL05-BP01 Implementar uma degradação normal para transformar dependências rígidas aplicáveis em dependências flexíveis
- REL05-BP02 Controlar a utilização de solicitações
- REL05-BP03 Controlar e limitar chamadas de novas tentativas
- REL05-BP04 Antecipar-se à falha e limitar filas
- 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 normal 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.

Práticas comuns que devem ser evitadas:

- 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 implementar 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 afeta 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 falham.

Trata-se tanto de uma discussão empresarial quanto técnica. Todos os requisitos comerciais 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 novas tentativas. 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 novas tentativas 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 de 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. Em geral, estratégias de fallback devem ser evitadas.

Etapas de 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 constantemente, não fará sentido continuar tentando novamente. Isso pode criar carga adicional em um sistema já sobrecarregado e dificultar

a recuperação. O padrão de disjuntor pode ser utilizado aqui, o qual 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. Uma loja 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 estar indisponível: o Amazon Relational Database Service, como quase todos os bancos de dados relacionais, só pode ter uma instância de gravador principal. 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 do Amazon Aurora Sem Servidor 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, é possível usar réplicas de leitura que fornecem redundância e a capacidade de aumentar a escala horizontalmente, e não apenas 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 principal esteja temporariamente indisponível.

Recursos

Documentos relacionados:

- Amazon API Gateway: controlar as solicitações de API para um melhor throughput
- CircuitBreaker (resume "Disjuntor" do livro "Release It!")
- Novas tentativas em caso de erro e recuo exponencial na AWS
- Michael Nygard "Release It! Design and Deploy Production-Ready Software"
- Amazon Builders' Library: evitar fallback em sistemas distribuídos

- Amazon Builders' Library: evitar backlogs de fila insuperáveis
- Amazon Builders' Library: desafios e estratégias de armazenamento em cache
- Amazon Builders' Library: tempos limite, novas tentativas e recuo com jitter

Vídeos relacionados:

- Novas tentativas, recuo e jitter: AWS re:Invent 2019: Introdução à Amazon Builders' Library (DOP328)

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.

Práticas comuns que devem ser evitadas:

- 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 para os 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.

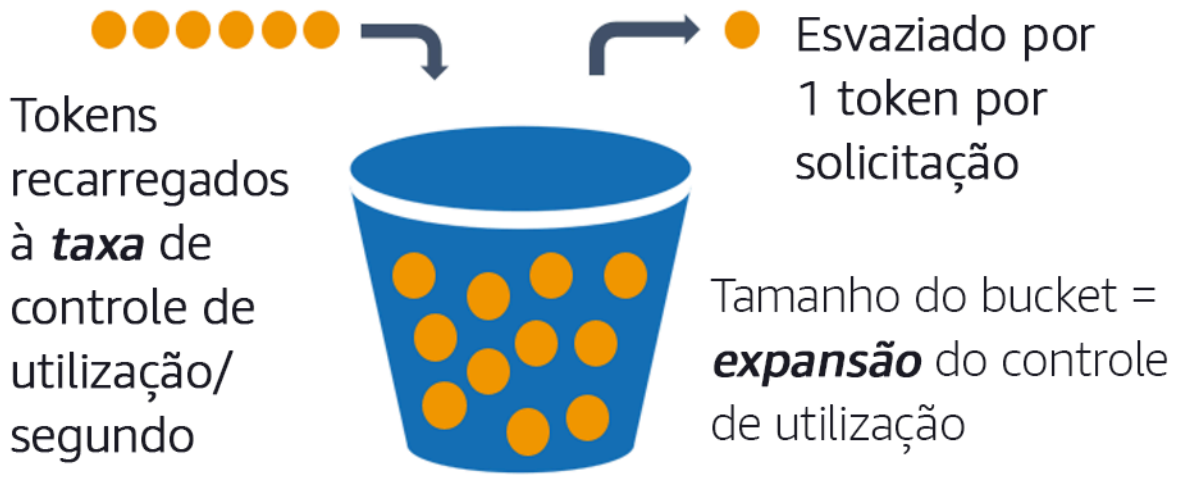
Benefícios de implementar esta prática recomendada: as workloads que definem limites de controle de utilização podem operar normalmente e processar a carga de solicitações aceitas com êxito em picos de volume inesperados. Os picos repentinos ou contínuos de solicitações para APIs e filas têm controle 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 limitação mais altas para solicitações que podem ser atendidas. Por fim, é possível implementar a limitação de taxa com o AWS WAF para limitar consumidores de API específicos que geram uma carga excepcionalmente alta.

Etapas de implementação

É possível configurar o API Gateway com limites de limitação para suas APIs e retornar erros `429 Too Many Requests` quando os limites são excedidos. É possível usar o AWS WAF com seus endpoints do AWS AppSync e do APÌ 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, você pode configurar a simultaneidade máxima para evitar que altas taxas de eventos consumam a cota de execução simultânea disponível da conta 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, é possível utilizar as implementações de código aberto específicas da linguagem (veja exemplos relacionados em Recursos) do bucket de token para seus serviços.

- Entenda e configure os limites de controle de utilização do API Gateway no nível da conta por região, API por estágio e chave de API por nível do plano de uso.
- Aplique regras de limitação de taxa do AWS WAF ao API Gateway e aos endpoints do AWS AppSync para se proteger contra inundações e bloquear IPs maliciosos. 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 as filas do Amazon SQS são configuradas como acionadores para consumidores de filas do Lambda, defina a simultaneidade máxima para 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 ao 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 foi estabelecido 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 chamadas de novas tentativas

Documentos relacionados:

- Amazon API Gateway: controlar as solicitações de API para um melhor throughput
- AWS WAF: declaração de regra baseada em intervalos
- Introduzir simultaneidade máxima 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 .NET System

Vídeos relacionados:

- Implementar as práticas recomendadas de segurança da API GraphQL com o AWS AppSync

Ferramentas relacionadas:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF
- Sala de Espera Virtual na AWS

REL05-BP03 Controlar e limitar chamadas de novas tentativas

Use o recuo exponencial para tentar as solicitações novamente em intervalos progressivamente maiores entre cada nova tentativa. Introduza jitter entre as novas tentativas para tornar os intervalos de repetição aleatórios. Limite o número máximo de novas tentativas.

Resultado desejado: os componentes típicos em um sistema de software distribuído incluem servidores, balanceadores de carga, 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 nova tentativa. Como resultado, os serviços e os processos de backend recebem alívio da carga e do tempo de recuperação automática, ocasionando uma recuperação mais rápida e atendimento bem-sucedido das solicitações.

Práticas comuns que devem ser evitadas:

- Implementar novas tentativas sem adicionar recuo exponencial, jitter e valores máximos de novas tentativas. O recuo e o jitter ajudam a evitar picos artificiais de tráfego devido a novas tentativas coordenadas involuntariamente em intervalos comuns.

- Implementar novas tentativas sem testar seus efeitos ou presumir que as novas tentativas já estejam incorporadas a um SDK sem testar cenários de repetição.
- Não entender os códigos de erro publicados das dependências, ocasionando novas tentativas 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 novas tentativas personalizados quando os recursos de novas tentativas integrados ou de terceiros são suficientes.
- Tentar novamente em várias camadas da pilha de aplicações de uma forma que agrava as novas tentativas, consumindo ainda mais recursos em uma tempestade de repetições. Entenda como esses erros afetam sua aplicação, as dependências nas quais você confia e implemente novas tentativas em apenas um nível.
- Tentar novamente chamadas de serviço que não são idempotentes, causando efeitos colaterais inesperados, como resultados duplicados.

Benefícios de implementar esta prática recomendada: as novas tentativas 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 novas tentativas funcionam bem. Quando as falhas são causadas pela sobrecarga de recursos, as novas tentativas podem piorar as coisas. Adicionar um recuo exponencial com jitter às novas tentativas 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 a escalação de carga causado pela adição de repetições à carga normal da solicitação. Por fim, é importante configurar um número máximo de novas tentativas 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 novas tentativas. Use o recuo exponencial para tentar novamente após intervalos progressivamente mais longos. Introduza jitter para tornar esses intervalos de novas tentativas aleatórios e limite o número máximo de repetições.

Alguns AWS SDKs implementam novas tentativas 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 simule cenários de teste para esses casos de uso de novas tentativas.

Etapas de implementação

- Determine a camada ideal em sua pilha de aplicações para implementar novas tentativas para os serviços dos quais sua aplicação depende.
- Conheça os SDKs existentes que implementam estratégias comprovadas de novas tentativas 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 novas tentativas. Depois que as novas tentativas forem implementadas, elas deverão ser testadas e simuladas regularmente na produção.
- Ao chamar as APIs de serviço da AWS, use os AWS SDKs e a AWS CLI e entenda as opções de configuração de nova tentativa. Determine se os padrões funcionam para seu caso de uso, teste e ajuste conforme necessário.

Recursos

Práticas recomendadas relacionadas:

- REL04-BP04 Garantir a idempotência das operações de mutação
- REL05-BP02 Controlar a utilização de solicitações
- REL05-BP04 Antecipar-se à falha e limitar filas
- REL05-BP05 Definir tempos limite do cliente
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas

Documentos relacionados:

- Novas tentativas em caso de erro e recuo exponencial na AWS
- Amazon Builders' Library: tempos limite, novas tentativas e recuo com jitter
- Recuo exponencial e jitter
- Tornar as tentativas seguras com APIs idempotentes

Exemplos relacionados:

- Spring Retry
- Resilience4j Retry

Vídeos relacionados:

- Novas tentativas, recuo e jitter: AWS re:Invent 2019: Introdução à Amazon Builders' Library (DOP328)

Ferramentas relacionadas:

- AWS SDKs e ferramentas: comportamento de novas tentativas
- AWS Command Line Interface: Novas tentativas via AWS CLI

REL05-BP04 Antecipar-se à falha e limitar filas

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 backend. Ao armazenar solicitações em filas, estratégias de gerenciamento de filas são implementadas para evitar backlogs intransponíveis.

Práticas comuns que devem ser evitadas:

- 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 implementar esta prática recomendada: sistemas que se antecipam às falhas 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 de 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, em seguida, 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 backend 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 chamadas de novas tentativas
- REL06-BP02 Definir e calcular métricas (agregação)
- REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema

Documentos relacionados:

- Evitar backlogs de fila intransponíveis
- Antecipar-se à falha
- Como posso evitar um aumento no atraso das mensagens na minha fila Amazon SQS?
- Elastic Load Balancing: mudança de zona
- Amazon Application Recovery Controller: controle de roteamento para failover de tráfego

Exemplos relacionados:

- Padrões de integração empresarial: canal de mensagens não entregues

Vídeos relacionados:

- AWS re:Invent 2022: Operar aplicações Multi-AZ altamente disponíveis

Ferramentas relacionadas:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- 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.

Práticas comuns que devem ser evitadas:

- 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 implementar 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 sejam 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 backend 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 Os SDKs e a AWS CLI permitem a configuração de tempos limite de conexão e solicitação e novas tentativas com recuo exponencial e jitter. As funções do AWS Lambda são compatíveis com a configuração de tempos limite. E, com o AWS Step Functions, você pode

criar disjuntores com pouco código que aproveitam as integrações pré-construídas com os serviços e SDKs da AWS. AWS App Mesh O Envoy oferece recursos de tempo limite e disjuntor.

Etapas de 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 para `connectTimeoutInMillis` e `tlsNegotiationTimeoutInMillis`.
- Aplique opções de linha de comando `cli-connect-timeout` e `cli-read-timeout` para controlar comandos da AWS CLI únicos 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 nas funções do Lambda.
- Os clientes do API Gateway devem implementar suas próprias repetições ao lidar com os tempos limite. O API Gateway oferece suporte a um tempo limite de integração de 50 milissegundos a 29 segundos para integrações downstream e não tenta novamente quando a integração solicita 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, revise os recursos do App Mesh Envoy para aproveitar 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 SDKs nativos da AWS e integrações do Step Functions compatíveis para simplificar sua workload.

Recursos

Práticas recomendadas relacionadas:

- REL05-BP03 Controlar e limitar chamadas de novas tentativas
- REL05-BP04 Antecipar-se à falha e limitar filas
- REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema

Documentos relacionados:

- AWS SDK: novas tentativas e tempos limite
- Amazon Builders' Library: tempos limite, novas tentativas e recuo com jitter
- Cotas do Amazon API Gateway e notas importantes
- Opções de linha de comando do AWS Command Line Interface
- 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 para .NET: novas tentativas e tempos limite
- AWS Lambda: configurar as opções da 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
- 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 uma aplicação, eles geralmente executam uma série de interações que formam uma sessão. Uma sessão são dados exclusivos para usuários que persistem entre solicitações enquanto usam a aplicação. Uma aplicação sem estado é uma aplicação 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 das aplicações sem estado é que elas 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 implementar esta prática recomendada: os sistemas projetados para serem sem estado são mais adaptáveis ao dimensionamento 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 risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

Crie aplicações sem estado. As aplicações sem estado permitem o ajuste de escala 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 de 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 pode dissociar os dados do usuário do código da aplicação usando recursos, como bancos de identidades, grupos de usuários e o Amazon Cognito Sync.
  - É possível usar o AWS Secrets Manager para desacoplar dados do usuário 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.
  - Considere 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.
  - O Amazon ElastiCache, o Amazon DynamoDB, o Amazon Elastic File System (Amazon EFS) e o Amazon MemoryDB são exemplos de serviços da AWS que você pode usar para descarregar dados da sessão.
- 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:

- Amazon Builders' Library: evitar fallback em sistemas distribuídos
- Amazon Builders' Library: evitar backlogs de fila insuperáveis
- Amazon Builders' Library: desafios e estratégias de armazenamento em cache

- Práticas recomendadas para níveis na 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, você pode estabelecer processos em boas condições para manter a disponibilidade de componentes essenciais em sua workload. A workload deve se degradar normalmente e continuar desempenhando suas funções essenciais aos negócios durante a ativação de uma medida emergencial. Para obter detalhes sobre a degradação normal, consulte REL05-BP01 Implementar uma degradação normal para transformar dependências rígidas aplicáveis em dependências flexíveis.

Práticas comuns que devem ser evitadas:

- 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 de implementar esta prática recomendada: a implementação de medidas emergenciais pode melhorar a disponibilidade dos componentes críticos em sua workload, fornecendo aos seus resolvedores processos estabelecidos para responder a picos inesperados na demanda ou falhas de dependências não críticas.

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

Orientação para 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 de implementação

- Identifique 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 exemplos de funcionalidades críticas e não críticas na Amazon, consulte Qualquer dia pode ser o Prime Day: Como a pesquisa da Amazon.com usa a engenharia do caos para lidar com mais de 84 mil solicitações por segundo.
  - 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 comportamento bimodal. Para obter mais detalhes, consulte REL11-BP05 Usar estabilidade estática para evitar 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 redução de carga, controle de utilização de solicitações ou implementação de degradação normal.

Recursos

Práticas recomendadas relacionadas:

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

Documentos relacionados:

- Automatizar implantações seguras e sem intervenção
- Qualquer dia pode ser o Prime Day: como a pesquisa da Amazon.com usa a engenharia do caos para lidar com mais de 84 mil solicitações por segundo

Vídeos relacionados:

- AWS re:Invent 2020: Confiabilidade, consistência e confiança por meio da imutabilidade

Gerenciamento de alterações

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

As seções a seguir explicam as práticas recomendadas para o gerenciamento de alterações.

Tópicos

Monitore os recursos da workload

Logs e métricas são ferramentas avançadas para obter informações sobre a integridade da workload. Você pode configurar a workload para monitorar logs e métricas e enviar notificações quando os limites forem ultrapassados ou ocorrerem eventos significativos. O monitoramento permite que sua workload reconheça quando os limites de baixa performance são ultrapassados ou quando há 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. Alertar quando problemas ocorrem 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 eles ocorram. Não consideramos apenas a latência média. Focamos ainda mais em exceções de latência, como 99,9 e 99,99 percentil. Isso ocorre porque, se uma solicitação em 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, isso acabará se tornando um problema à medida que seu tráfego aumentar.

O monitoramento na AWS consiste em quatro fases distintas:

1. Geração: monitorar todos os componentes da workload
2. Agregação: definir e calcular métricas
3. Processamento e alarmes em tempo real: enviar notificações e automatizar respostas
4. Armazenamento e análise

Práticas recomendadas

- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP02 Definir e calcular métricas (agregação)
- REL06-BP03 Enviar notificações (processamento e alarmes em tempo real)
- REL06-BP04 Automatizar respostas (processamento e alarmes em tempo real)
- REL06-BP05 Analisar logs
- REL06-BP06 Revisar regularmente o escopo e as métricas de monitoramento
- 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 workload com o Amazon CloudWatch ou ferramentas de terceiros. Monitore os serviços da AWS com o AWS Health Dashboard.

Todos os componentes da workload devem ser monitorados, incluindo frontend, lógica de negócios e níveis de armazenamento. Defina as principais métricas e como extraí-las dos logs (se necessário) e defina limites para invocação de eventos de alarme correspondentes. Garanta que as métricas sejam relevantes para os indicadores-chave de performance (KPIs) da sua workload e use métricas e logs para identificar sinais precoces de degradação do serviço. Por exemplo, uma métrica relacionada aos resultados comerciais, como o número de pedidos processados com sucesso por minuto, pode indicar problemas de workload mais rapidamente do que uma métrica técnica, como a utilização da CPU. Use o AWS Health Dashboard para obter 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 hooks personalizáveis e pode fornecer informações para ajudar você a monitorar

várias camadas da sua workload. Serviços da AWS como o Amazon CloudWatch aplicam algoritmos estatísticos e de machine learning para analisar continuamente métricas de sistemas e aplicações, determinar linhas de base normais e apontar 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 enorme quantidade de informações de monitoramento e log para consumo que podem ser usadas para definir métricas específicas da workload e 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. Esse monitoramento ativo pode ser feito com transações sintéticas (às vezes chamadas de canários do usuário, mas que não devem ser confundidas com implantações canários) que executam periodicamente diversas tarefas comuns correspondentes a ações executadas pelos consumidores da workload. Mantenha essas tarefas com curta duração e certifique-se de não sobrecarregar sua workload durante o teste. O Amazon CloudWatch Synthetics permite criar canários sintéticos para monitorar 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:

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

Práticas comuns que devem ser evitadas:

- Monitorar apenas as interfaces externas com sua workload.
- Não gerar nenhuma métrica específica da workload nem depender apenas das métricas fornecidas pelos serviços da AWS usados por sua workload.
- Usar apenas métricas técnicas em sua workload e não monitorar nenhuma métrica relacionada a KPIs não técnicos para os quais a workload contribui.
- Contar com o tráfego de produção e com verificações de saúde simples para monitorar e avaliar o estado da workload.

Benefícios de implementar esta prática recomendada: o monitoramento em todos os níveis de sua workload permite que você antecipe e resolva problemas mais rapidamente nos componentes que fazem parte da workload.

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

Orientação para implementação

1. Ative o registro quando disponível. Os dados de monitoramento devem ser obtidos de todos os componentes das workloads. Ative o log adicional, como os logs de acesso do S3, e permita que sua workload registre dados específicos da workload. Colete métricas para médias de CPU, E/S de rede e E/S de disco de serviços como Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling e Amazon EMR. Consulte Serviços da AWS que publicam métricas do CloudWatch para obter uma lista dos serviços da AWS que publicam métricas no CloudWatch.
2. Analise 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 dos componentes afetam a workload. Você também pode publicar suas próprias métricas no CloudWatch usando a AWS CLI ou uma API.
3. Avalie todas as métricas para decidir quais delas alertar para cada serviço da AWS em sua workload. Você pode optar por selecionar um subconjunto de métricas que tenham um grande impacto na confiabilidade da workload. Concentrar-se em métricas e limites críticos permite refinar o número de alertas e pode ajudar a minimizar os falsos positivos.
4. Defina alertas e o processo de recuperação para sua workload após a chamada do alerta. A definição de alertas permite que você notifique, escale e siga rapidamente as etapas necessárias para se recuperar de um incidente e atingir seu objetivo de tempo de recuperação (RTO) prescrito. Você pode usar os 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 executa as mesmas ações que um cliente, o que possibilita verificar continuamente a experiência do cliente, mesmo quando você não tem nenhum tráfego de cliente em suas workloads. Ao usar transações sintéticas, é possível 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 AWS Health Dashboard: integridade da conta
- Serviços da AWS que publicam métricas do CloudWatch
- Logs de acesso do Network Load Balancer
- Logs de acesso para seu Application Load Balancer
- Acessar o Amazon CloudWatch Logs para AWS Lambda
- Registro em log de acesso ao servidor do Amazon S3
- Habilitar logs de acesso para seu Classic Load Balancer
- Exportar dados de log para o Amazon S3
- Instalar o agente do CloudWatch em uma instância do Amazon EC2
- Publicar métricas personalizadas
- Usar painéis do Amazon CloudWatch
- Usar métricas do Amazon CloudWatch
- Usar canários (Amazon CloudWatch Synthetics)
- O que são Amazon CloudWatch Logs?

Guias do usuário:

- Criar uma trilha
- Monitorar métricas de memória e disco para instâncias Linux do Amazon EC2
- Usar o CloudWatch Logs com Instâncias de contêiner
- VPC Flow Logs
- O que é o Amazon DevOps Guru?
- O que é AWS X-Ray?

Blogs relacionados:

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

Exemplos relacionados:

- Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- Workshop Observability

REL06-BP02 Definir e calcular métricas (agregação)

Colete métricas e logs dos componentes da workload e calcule métricas agregadas relevantes com base neles. Essas métricas fornecem uma observabilidade ampla e profunda da workload e podem melhorar significativamente sua postura de resiliência.

A observabilidade é mais do que apenas coletar métricas dos componentes da workload e poder visualizá-las e criar alertas com base nelas. Trata-se de ter uma compreensão holística sobre o comportamento da workload. Essas informações comportamentais vêm de todos os componentes das workloads, incluindo os serviços de nuvem dos quais elas dependem, logs bem elaborados e métricas. Esses dados fornecem uma visão geral do comportamento da workload, bem como uma compreensão da interação de cada componente com cada unidade de trabalho em um nível preciso de detalhes.

Resultado desejado:

- Coletar logs dos componentes da workload e das dependências de serviços da AWS e os publicar em um local central onde eles podem ser facilmente acessados e processados.
- Os logs contêm carimbos de data/hora precisos e de alta fidelidade.
- Os logs contêm informações relevantes sobre o contexto de processamento, como identificador de rastreamento, identificador de usuário ou de conta e endereço IP remoto.
- Criar métricas agregadas com base nos logs que representam o comportamento da workload de uma perspectiva de alto nível.
- Poder consultar os logs agregados para obter insights profundos e relevantes sobre a workload e identificar problemas reais e potenciais.

Práticas comuns que devem ser evitadas:

- Não coletar logs ou métricas relevantes das instâncias de computação em que as workloads são executadas ou dos serviços de nuvem que elas usam.
- Ignora a coleção de logs e métricas relacionados aos indicadores-chave de desempenho (KPIs) da empresa.
- Analisar a telemetria relacionada à workload de maneira isolada, sem agregação e correlação.

- Permitir que métricas e logs expirem muito rapidamente, o que dificulta a análise de tendências e a identificação de problemas recorrentes.

Benefícios de implementar essas práticas recomendadas: você pode detectar mais anomalias e correlacionar eventos e métricas entre diferentes componentes da workload. Você pode criar insights a partir dos componentes da workload com base nas informações contidas nos logs que frequentemente não estão disponíveis apenas nas métricas. Você pode determinar as causas de falha mais rapidamente consultando os logs em grande escala.

Nível de risco exposto se estas práticas recomendadas não forem estabelecidas: Alto

Orientação para implementação

Identifique as fontes de dados de telemetria que são relevantes para as workloads e para os componentes delas. Esses dados não vêm apenas de componentes que publicam métricas, como o sistema operacional (SO) e runtimes de aplicações como Java, mas também de logs de aplicações e serviços de nuvem. Por exemplo, os servidores web normalmente registram cada solicitação em log com informações detalhadas, como carimbo de data/hora, latência de processamento, ID do usuário, endereço IP remoto, caminho e string de consulta. O nível de detalhe nesses logs ajuda você a realizar consultas detalhadas e gerar métricas que talvez não estivessem disponíveis de outra forma.

Colete as métricas e os logs usando ferramentas e processos apropriados. Os logs gerados por aplicações executadas na instância do Amazon EC2 podem ser coletados por um agente, como o agente do Amazon CloudWatch, e publicados em um serviço de armazenamento central, como o Amazon CloudWatch Logs. Os serviços de computação gerenciados pela AWS, como o AWS Lambda e o Amazon Elastic Container Service, publicam automaticamente logs no CloudWatch Logs para você. Habilite a coleta de logs para serviços de armazenamento e processamento da AWS usados pelas workloads, como Amazon CloudFront, Amazon S3, Elastic Load Balancing e Amazon API Gateway.

Enriqueça os dados de telemetria com dimensões que podem ajudar você a ver padrões de comportamento com mais clareza e isolar problemas correlacionados em grupos de componentes relacionados. Depois de adicionar, você pode observar o comportamento dos componentes em um nível mais detalhado, detectar falhas correlacionadas e tomar as medidas corretivas apropriadas. Exemplos de dimensões úteis incluem zona de disponibilidade, ID da instância EC2 e tarefa do contêiner ou ID do pod.

Depois de coletar as métricas e os logs, você pode escrever consultas e gerar métricas agregadas com base neles que forneçam informações úteis sobre comportamentos normais e anormais. Por exemplo, você pode usar o Amazon CloudWatch Logs Insights para derivar métricas personalizadas dos logs de aplicações, o Amazon CloudWatch Metrics Insights para consultar as métricas em grande escala, o Amazon CloudWatch Container Insights para coletar, agregar e resumir métricas e logs de aplicações e microsserviços em contêineres ou o Lambda Insights do Amazon CloudWatch, se você estiver usando funções do AWS Lambda. Para criar uma métrica agregada de taxa de erros, você pode incrementar um contador sempre que uma resposta ou mensagem de erro for encontrada nos logs de componentes ou calcular o valor agregado de uma métrica de taxa de erros existente. Você pode usar esses dados para gerar histogramas que mostrem o comportamento final, como as solicitações ou processos com pior desempenho. Você também pode verificar esses dados em tempo real em busca de padrões anormais usando soluções como a detecção de anomalias do CloudWatch Logs. Esses insights podem ser colocados em painéis para mantê-los organizados de acordo com suas necessidades e preferências.

A consulta de logs pode ajudar você a entender como solicitações específicas foram tratadas pelos componentes da workload e revelar padrões de solicitação ou outro contexto que tenha impacto na resiliência da workload. Pode ser útil pesquisar e preparar consultas com antecedência, com base no seu conhecimento de como as aplicações e outros componentes se comportam, para que você possa executá-las com mais facilidade, conforme necessário. Com o CloudWatch Logs Insights, você pode pesquisar e analisar de maneira interativa dados de log armazenados no CloudWatch Logs. Você também pode usar o Amazon Athena para consultar logs de várias fontes, incluindo vários serviços da AWS, em escala de petabytes.

Ao definir uma política de retenção de logs, considere o valor dos logs históricos. Os logs históricos podem ajudar a identificar padrões comportamentais e de uso a longo prazo, regressões e melhorias no desempenho da workload. Os logs excluídos permanentemente não poderão ser analisados posteriormente. No entanto, o valor dos logs históricos tende a diminuir em longos períodos. Escolha uma política que equilibre suas necessidades conforme apropriado e esteja em conformidade com quaisquer requisitos legais ou contratuais aos quais você possa estar sujeito.

Etapas de implementação

1. Escolha mecanismos de coleta, armazenamento, análise e exibição dos dados de observabilidade.
2. Instale e configure coletores de métricas e logs nos componentes apropriados da workload (por exemplo, em instâncias do Amazon EC2 e em contêineres auxiliares). Configure esses coletores para que sejam reiniciados automaticamente em caso de interrupção inesperada. Habilite o buffer

de disco ou memória para os coletores de modo que falhas temporárias de publicação não afetem as aplicações nem resultem em perda de dados.

3. Habilite o registro em log nos serviços da AWS que você usa como parte das workloads e encaminhe esses logs para o serviço de armazenamento selecionado, se necessário. Consulte os guias do usuário ou do desenvolvedor dos respectivos serviços para obter instruções detalhadas.
4. Defina as métricas operacionais relevantes para as workloads com base nos dados de telemetria. Elas podem ser baseadas em métricas diretas emitidas pelos componentes da workload, que podem incluir métricas relacionadas ao KPI comercial ou resultados de cálculos agregados, como somas, taxas, percentis ou histogramas. Calcule essas métricas usando o analisador de logs e coloque-as em painéis conforme apropriado.
5. Prepare consultas de log apropriadas para analisar os componentes da workload, as solicitações ou o comportamento da transação, conforme necessário.
6. Defina e habilite uma política de retenção de logs para os logs dos componentes. Exclua periodicamente os logs quando eles ficarem mais antigos do que o permitido pela política.

Recursos

Práticas recomendadas relacionadas:

- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP03 Enviar notificações (processamento e emissão de alarmes em tempo real)
- REL06-BP04 Automatizar respostas (processamento e alarmes em tempo real)
- REL06-BP05 Analisar logs
- REL06-BP06 Revisar regularmente o escopo e as métricas de monitoramento
- REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema

Documentação relacionada:

- Como o Amazon CloudWatch funciona
- Prometheus gerenciado pela Amazon
- Amazon Managed Grafana
- Analisar logs de dados com o CloudWatch Logs Insights
- Lambda Insights do Amazon CloudWatch

- Amazon CloudWatch Container Insights
- Consultar métricas com o CloudWatch Metrics Insights
- AWS Distro para OpenTelemetry
- Consultas de exemplo do Amazon CloudWatch Logs Insights
- Depurar com o Amazon CloudWatch Synthetics e o AWS X-Ray
- Pesquisar e filtrar dados de log
- Enviar logs diretamente para o Amazon S3
- Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional

Workshops relacionados:

- Workshop One Observability

Ferramentas relacionadas:

- AWS Distro for OpenTelemetry (GitHub)

REL06-BP03 Enviar notificações (processamento e 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 em métricas de serviços e aplicações. Quando os limites do alarme são violados, a equipe e os sistemas apropriados são notificados para que possam resolver os problemas subjacentes.

Práticas comuns que devem ser evitadas:

- Configurar 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 implementar 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 capacidade de detecção 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.

O Amazon CloudWatch permite criar alarmes métricos 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 alarmes que é possível 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 com os 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 executar uma ou mais ações, como enviar uma notificação a um tópico do Amazon SNS, executar uma ação do Amazon EC2 ou uma ação do Amazon EC2 Auto Scaling ou criar 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 as notificações do Amazon SNS, consulte Configurar o Amazon SNS.

O CloudWatch envia eventos do EventBridge sempre que um alarme do CloudWatch é criado, atualizado, excluído ou muda de estado. 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 a automação do Systems Manager.

Mantenha-se a par do AWS Health. O AWS Health é a fonte de informações autorizada sobre a integridade dos seus recursos da Nuvem AWS. Use o AWS Health para receber notificações sobre quaisquer eventos de serviço confirmados para que possa tomar medidas rapidamente e mitigar qualquer impacto. Crie notificações de eventos do AWS Health ajustadas à finalidade para canais de e-mail e chat por meio do Notificações de Usuários da AWS e integre-as programaticamente às suas ferramentas de monitoramento e alerta por meio do Amazon EventBridge. Se você usar o AWS Organizations, agregue eventos do AWS Health em todas as contas.

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 das 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 terceirizados. 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 para um destino. No momento, o EventBridge oferece suporte a mais de 20 serviços da AWS como destino, incluindo o AWS Lambda, Amazon SQS, Amazon SNS, Amazon Kinesis Data Streams e Amazon Data Firehose.

O Amazon SNS é recomendado para aplicações que precisam de alta distribuição (milhares ou milhões de endpoints). Um padrão comum que observamos é que os clientes usam o Amazon SNS como destino para a regra para 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 oferece suporte ao encaminhamento de mensagens para seis tipos diferentes de destinos, incluindo Lambda, Amazon SQS, endpoints do HTTP/S, SMS, push móvel e e-mail. No Amazon SNS, a latência típica é inferior a 30 milissegundos. Uma ampla variedade de serviços da AWS enviam mensagens do Amazon SNS configurando o serviço para fazer isso (mais de 30, incluindo o Amazon EC2, o Amazon S3 e o Amazon RDS).

Etapas de implementação

1. Crie um alarme usando os 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 ser enviar uma notificação a um tópico do Amazon SNS, executar uma ação do Amazon EC2 ou uma ação do Amazon EC2 Auto Scaling ou criar 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 entrará 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. Os alarmes compostos podem enviar notificações do Amazon SNS quando mudam de estado e podem criar OpsItems ou incidentes do Systems Manager quando entram no estado de alarme, mas não podem executar ações do EC2 ou ações do Auto Scaling.

2. Configure 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 correspondam aos alarmes especificados do CloudWatch. 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 os alvos do EventBridge, consulte Destinos do EventBridge.

Recursos

Práticas recomendadas do Well-Architected relacionadas:

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

Documentos relacionados:

- Amazon CloudWatch
- Insights do CloudWatch Logs
- Usar alarmes do Amazon CloudWatch
- Usar painéis do Amazon CloudWatch

- Usar métricas do Amazon CloudWatch
- Configurar notificações do Amazon SNS
- Detecção de anomalias do CloudWatch
- Proteção de dados do CloudWatch Logs
- Amazon EventBridge
- Amazon Simple Notification Service

Vídeos relacionados:

- Vídeos sobre observabilidade do re:Invent 2022
- AWS re:Invent 2022: Práticas recomendadas de observabilidade na Amazon

Exemplos relacionados:

- Workshop One Observability
- Amazon EventBridge para AWS Lambda com controle de feedback por alarmes do Amazon CloudWatch

REL06-BP04 Automatizar respostas (processamento e 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 tomadas para manter os objetivos de nível de serviço e os contratos de nível de serviço (SLAs). A automação pode variar de atividades de autorrecuperação de componentes individuais a failover de todo o site.

Práticas comuns que devem ser evitadas:

- 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 de implementar esta prática recomendada: automatizar o 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 risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para 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 playbooks 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 de implementação

1. Crie um inventário de alarmes: para obter uma lista de todos os alarmes, é possível usar a AWS CLI com o comando `describe-alarms` do Amazon CloudWatch. Dependendo de quantos alarmes você configurou, talvez seja necessário usar a paginação para recuperar um subconjunto

de alarmes para cada chamada ou, alternativamente, você pode usar o AWS SDK para obter os alarmes usando uma chamada de API.

2. Documente todas as ações de alarme: atualize um runbook com todos os alarmes e suas ações, independentemente de serem manuais ou automatizados. O AWS Systems Manager fornece runbooks predefinidos. Para obter informações sobre como usar runbooks, consulte Trabalhado com runbooks. Para obter detalhes sobre como visualizar o conteúdo do runbook, consulte Visualizar o conteúdo do runbook.
3. Configure e gerencie ações de alarme: 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. É possível criar regras para indicar os eventos de seu interesse e quais ações deverão ser executadas quando um evento corresponder a uma regra. As ações que podem ser iniciadas automaticamente incluem invocar uma função do AWS Lambda, invocar o `Run Command` do Amazon EC2, retransmitir o evento para o Amazon Kinesis Data Streams e consultar Automatizar o Amazon EC2 usando o EventBridge.
4. Procedimentos operacionais padrão (SOPs): com base nos componentes da aplicação, o AWS Resilience Hubrecomenda vários modelos de SOP. É possível usar esses SOPs para documentar todos os processos que um operador deve seguir caso um alerta seja emitido. Você também pode criar um SOP com base nas recomendações do Resilience Hub, onde você precisa de uma aplicação do Hub de Resiliência com uma política de resiliência associada, bem como uma avaliação histórica da resiliência em relação a essa aplicação. As recomendações para o SOP são produzidas pela avaliação de resiliência.

   O Hub de Resiliência trabalha com o Systems Manager para automatizar as etapas de seus SOPs, fornecendo vários documentos do SSM que você pode usar como base para esses SOPs. Por exemplo, o Hub de Resiliência pode recomendar um SOP para adicionar espaço em disco com base em um documento de automação do SSM existente.
5. Execute ações automatizadas com o Amazon DevOps Guru: é possível usar o Amazon DevOps Guru para monitorar automaticamente recursos de aplicações em busca de comportamento anômalo e entregar recomendações direcionadas para acelerar os tempos de identificação e correção do problema. Com o DevOps Guru, você pode monitorar fluxos de dados operacionais quase em tempo real de várias fontes, incluindo métricas do Amazon CloudWatch, AWS Config,

AWS CloudFormation e AWS X-Ray. Você também pode usar o DevOps Guru para criar automaticamente OpsItems no OpsCenter e enviar eventos para o EventBridge para automação adicional.

Recursos

Práticas recomendadas relacionadas:

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

Documentos relacionados:

- Automação do AWS Systems Manager
- Criar uma regra do EventBridge que seja acionada por um evento de um recurso da AWS
- Workshop One Observability
- Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- O que é o 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: automatizar qualquer coisa com o AWS Systems Manager
- Introdução ao AWS Resilience Hub
- Criar sistemas de tickets personalizados para notificações do Amazon DevOps Guru
- Habilitar a agregação de insights de várias contas com o Amazon DevOps Guru

Exemplos relacionados:

- Workshop do Amazon CloudWatch e Systems Manager

REL06-BP05 Analisar logs

Colete arquivos de log e históricos de métricas e analise-os para obter tendências mais abrangentes e informações sobre a workload.

O Amazon CloudWatch Logs Insights oferece suporte a uma linguagem de consulta simples, porém 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 para o Amazon S3, onde você pode usar o ou o Amazon Athena para consultar os dados. Ele também oferece suporte a consultas em uma grande variedade de formatos. Para obter mais informações, consulte SerDes e formatos de dados compatíveis no Guia do usuário do Amazon Athena. 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 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 da aplicação 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 é compatível com 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 para o 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 risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

- O CloudWatch Logs Insights permite pesquisar e analisar dados de log de modo interativo no Amazon CloudWatch Logs.
  - Analisar logs de dados com o CloudWatch Logs Insights
  - Consultas de exemplo do Amazon CloudWatch Logs Insights
- Use o Amazon CloudWatch Logs para enviar logs ao Amazon S3, no qual você pode usar o Amazon Athena para consultar os 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. Fazer isso reduz a quantidade de dados que o Athena analisa para cada consulta.
    - Como faço para criar uma política de ciclo de vida para um bucket do S3?

Recursos

Documentos relacionados:

- Consultas de exemplo do Amazon CloudWatch Logs Insights
- Analisar logs de dados com o CloudWatch Logs Insights
- Depurar com o Amazon CloudWatch Synthetics e o AWS X-Ray
- Como faço para criar uma política de ciclo de vida para um bucket do S3?
- Como analiso meus logs de acesso ao servidor do Amazon S3 usando o Athena?
- Workshop One Observability
- Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional

REL06-BP06 Revisar regularmente o escopo e as métricas de monitoramento

Revise frequentemente a maneira como o monitoramento da workload é implementado e atualize-o à medida que a workload e a arquitetura evoluem. Auditorias regulares do monitoramento ajudam a reduzir o risco de indicadores de problemas perdidos ou negligenciados e colaboram ainda mais para que a workload atinja as metas de disponibilidade.

O monitoramento eficaz está ancorado nas principais métricas de negócios, que evoluem à medida que suas prioridades comerciais mudam. Seu processo de análise de monitoramento deve enfatizar os indicadores de nível de serviço (SLIs) e incorporar insights da infraestrutura, das aplicações, dos clientes e dos usuários.

Resultado desejado: Você tem uma estratégia de monitoramento eficaz que é regularmente revisada e atualizada periodicamente, bem como após quaisquer eventos ou mudanças significativas. Você verifica se os principais indicadores de integridade da aplicação ainda são relevantes à medida que a workload e os requisitos de negócios evoluem.

Práticas comuns que devem ser evitadas:

- Coletar apenas as métricas padrão.
- Definir uma estratégia de monitoramento, mas nunca revisá-la.
- Não discutir o monitoramento quando alterações importantes são implantadas.
- Confiar em métricas desatualizadas para determinar a integridade da workload.
- Equipes de operações sobrecarregadas com alertas falso-positivos devido a métricas e limites desatualizados.
- Faltar observabilidade dos componentes da aplicação que não estão sendo monitorados.
- Concentrar-se apenas em métricas técnicas de baixo nível e excluir métricas de negócios no monitoramento.

Benefícios de implementar essa prática recomendada: ao revisar regularmente seu monitoramento, você pode antecipar possíveis problemas e verificar se é capaz de detectá-los. Ela também permite que você descubra pontos cegos que pode ter perdido durante as avaliações anteriores, o que melhora ainda mais sua capacidade de detectar problemas.

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

Orientação para implementação

Analise as métricas e o escopo do monitoramento durante seu processo de revisão de prontidão operacional (ORR). Realize revisões periódicas de prontidão operacional em um cronograma consistente para avaliar se há alguma lacuna entre sua workload atual e o monitoramento que você configurou. Estabeleça um ritmo regular para revisões de desempenho operacional e compartilhamento de conhecimento a fim de aprimorar sua capacidade de obter um desempenho superior das equipes operacionais. Valide se os limites de alerta existentes ainda são adequados e verifique as situações em que as equipes operacionais estão recebendo alertas falso-positivos ou não estão monitorando aspectos da aplicação que devem ser monitorados.

O Resilience Analysis Framework fornece orientações úteis que podem ajudar você a navegar pelo processo. O foco do framework é identificar possíveis modos de falha e os controles preventivos e corretivos que você pode usar para mitigar o impacto deles. Esse conhecimento pode ajudar você a identificar as métricas e os eventos certos para monitorar e alertar.

Etapas de implementação

1. Programe e realize revisões regulares dos painéis da workload. É possível ter cadências diferentes para a profundidade de inspeção.
2. 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 ser investigado. Os exemplos incluem o aumento da latência, diminuição da função primária de negócios e aumento das respostas a falhas.
3. Verifique se há discrepâncias e anomalias em suas métricas, que podem ser mascaradas por médias ou medianas. Examine os valores mais altos e mais baixos durante o período e investigue as causas das observações que estão muito fora dos limites normais. À medida que você continua a remover essas causas, pode estreitar os limites das métricas esperadas em resposta à consistência aprimorada do desempenho da workload.
4. 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 que talvez você precise para adicionar e rastrear outras métricas.
5. Analise se a estratégia de monitoramento atual permanece relevante para a aplicação. Com base em uma análise de incidentes anteriores (ou no Resilience Analysis Framework), avalie se há aspectos adicionais da aplicação que devem ser incorporados ao escopo de monitoramento.
6. Analise suas métricas de monitoramento de usuários reais (RUM) para determinar se há alguma lacuna na cobertura da funcionalidade da aplicação.
7. Revise seu processo de gerenciamento de alterações. Atualize seus procedimentos, se necessário, para incluir uma etapa de análise de monitoramento que deve ser executada antes de você aprovar uma alteração.
8. Implemente a revisão de monitoramento como parte da revisão de prontidão operacional e dos processos de correção de erro.

Recursos

Práticas recomendadas relacionadas:

- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP02 Definir e calcular métricas (agregação)
- REL06-BP07 Monitorar o rastreamento completo das solicitações por meio de seu sistema
- REL12-BP02 Realizar análises pós-incidentes

- REL12-BP06 Conduzir dias de jogo regularmente

Documentos relacionados:

- Por que você deve desenvolver uma correção de erro (COE)
- Usar painéis do Amazon CloudWatch
- Criação de painéis para visibilidade operacional
- Advanced Multi-AZ Resilience Patterns - Gray failures
- Consultas de exemplo do Amazon CloudWatch Logs Insights
- Depurar com o Amazon CloudWatch Synthetics e o AWS X-Ray
- Workshop One Observability
- Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- Usar painéis do Amazon CloudWatch
- Práticas recomendadas de observabilidade da AWS
- Resilience Analysis Framework
- Resilience Analysis Framework - Observability
- Operational Readiness Review - ORR

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: workloads com rastreamento abrangente em todos os componentes são fáceis de depurar, melhorando o tempo médio de resolução (MTTR) de erros e a latência ao simplificar 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.

Práticas comuns que devem ser evitadas:

- 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.

- Os canários sintéticos ou o monitoramento de usuários reais (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 implementar 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 investigando-as. 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 os serviços de instrumentação do AWS X-Ray e do Monitoramento de aplicações do Amazon CloudWatch para fornecer uma visão completa das solicitações à medida que elas percorrem sua aplicação. O X-Ray coleta telemetria da aplicação e permite que você a visualize

e filtre em cargas, funções, rastreamentos, serviços, APIs. Além disso, ele pode ser ativado para componentes do sistema sem código ou com pouco código. O monitoramento de aplicações do CloudWatch inclui o ServiceLens para integrar seus rastreamentos com métricas, logs e alarmes. O monitoramento de aplicações do CloudWatch também inclui sintéticos para monitorar seus endpoints e APIs, bem como monitoramento de usuários reais para instrumentar seus clientes de aplicações Web.

Etapas de 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 infraestrutura como código, AWS SDKs ou o AWS Management Console.
- Instrumente aplicações AWS Distro para Open Telemetry e X-Ray ou agentes de coleção de terceiros.
- Consulte o Guia do desenvolvedor da AWS X-Ray para obter informações sobre implementações específicas de linguagens 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 do Amazon CloudWatch Synthetic e Amazon CloudWatch RUM para analisar o caminho da solicitação do cliente do usuário final ao longo de sua infraestrutura da AWS downstream.
- 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.
- Ative a integração do X-Ray para ferramentas de rastreamento de terceiros, como Datadog, New Relic ou 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 AWS X-Ray?
- Amazon CloudWatch: monitoramento de aplicações
- Depurar com o Amazon CloudWatch Synthetics e o AWS X-Ray
- Amazon Builders' Library: instrumentação de sistemas distribuídos para visibilidade operacional
- Integrar o AWS X-Ray a outros serviços da AWS
- AWS Distro para OpenTelemetry e AWS X-Ray
- Amazon CloudWatch: usar monitoramento sintético
- Amazon CloudWatch: usar o CloudWatch RUM
- Configurar o canário do Amazon CloudWatch Synthetics e o alarme do Amazon CloudWatch
- Disponibilidade e além: como entender e melhorar a resiliência de sistemas distribuídos na AWS

Exemplos relacionados:

- Workshop One Observability

Vídeos relacionados:

- AWS re:Invent 2022: Como monitorar aplicações em várias contas
- Como monitorar suas aplicações da AWS

Ferramentas relacionadas:

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

Projetar a workload para se adaptar às mudanças na demanda

Uma workload escalável fornece elasticidade para adicionar ou remover recursos automaticamente, de modo que eles correspondam perfeitamente à demanda atual em determinado momento.

Práticas recomendadas

- REL07-BP01: Usar automação ao obter ou escalar recursos

- REL07-BP02 Obter recursos após a detecção de danos em uma workload
- REL07-BP03 Obter recursos após determinar que mais recursos são necessários para uma workload
- REL07-BP04 Fazer o teste de carga da workload

REL07-BP01: Usar automação ao obter ou escalar recursos

A base da confiabilidade na nuvem é a definição programática, o provisionamento e o gerenciamento de sua infraestrutura e recursos. A automação ajuda você a otimizar o provisionamento de recursos, facilitar implantações consistentes e seguras e escalar recursos em toda a sua infraestrutura.

Resultado desejado: você gerencia sua infraestrutura como código (IaC). Você define e mantém seu código de infraestrutura em sistemas de controle de versão (VCS). Você delega o provisionamento de recursos da AWS a mecanismos automatizados e aproveita serviços gerenciados, como Application Load Balancer (ALB), Network Load Balancer (NLB) e grupos do Auto Scaling. Você provisiona seus recursos usando pipelines de integração contínua/entrega contínua (CI/CD) para que as alterações do código iniciem automaticamente as atualizações dos recursos, incluindo as atualizações das configurações do Auto Scaling.

Práticas comuns que devem ser evitadas:

- Implantar recursos manualmente usando a linha de comandos ou pelo AWS Management Console (também conhecido como click-ops).
- Acoplar fortemente os componentes ou recursos da aplicação e, como resultado, criar arquiteturas inflexíveis.
- Implementar políticas de escalabilidade inflexíveis que não se adaptam às mudanças nos requisitos de negócios, aos padrões de tráfego ou aos novos tipos de recursos.
- Estimar manualmente a capacidade para atender à demanda prevista.

Benefícios de implementar essa prática recomendada: a infraestrutura como código (IaC) permite que a infraestrutura seja definida de maneira programática. Isso ajuda você a gerenciar as mudanças na infraestrutura por meio do mesmo ciclo de vida de desenvolvimento de software das alterações nas aplicações, o que promove consistência e repetibilidade e reduz o risco de tarefas manuais propensas a erros. Você pode simplificar ainda mais o processo de provisionamento e atualização de recursos por meio da implementação de IaC com pipelines de entrega automatizados. Você pode implantar atualizações de infraestrutura de forma confiável e eficiente sem a necessidade de

intervenção manual. Essa agilidade é particularmente importante ao escalar recursos para atender às demandas flutuantes.

Você pode alcançar uma escalabilidade dinâmica e automatizada de recursos em conjunto com IaC e pipelines de entrega. Ao monitorar as principais métricas e aplicar políticas de escalabilidade predefinidas, o Auto Scaling pode provisionar ou desprovisionar recursos automaticamente conforme necessário, o que melhora o desempenho e a economia. Isso reduz o potencial de erros manuais ou atrasos em resposta às mudanças nos requisitos da aplicação ou da workload.

A combinação de IaC, pipelines de entrega automatizados e Auto Scaling ajuda as organizações a provisionar, atualizar e escalar seus ambientes com confiança. Essa automação é essencial para manter uma infraestrutura de nuvem responsiva, resiliente e gerenciada com eficiência.

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

Orientação para implementação

Para configurar a automação com pipelines de CI/CD e infraestrutura como código (IaC) para sua arquitetura da AWS, escolha um sistema de controle de versão, como o Git, para armazenar modelos e configurações de IaC. Esses modelos podem ser escritos usando ferramentas como AWS CloudFormation. Para começar, defina os componentes da infraestrutura (como VPCs da AWS, grupos do Amazon EC2 Auto Scaling e bancos de dados Amazon RDS) dentro desses modelos.

Depois, integre esses modelos de IaC a um pipeline de CI/CD para automatizar o processo de implantação. O AWS CodePipeline fornece uma solução integrada e nativa da AWS, mas você também pode usar outras soluções de CI/CD de terceiros. Crie um pipeline que seja ativado quando ocorrerem alterações no seu repositório de controle de versão. Configure o pipeline para incluir estágios que vinculam e validam os modelos de IaC, implantam a infraestrutura em um ambiente de preparação, executam testes automatizados e, por fim, implantam na produção. Incorpore etapas de aprovação quando necessário para manter o controle sobre as mudanças. Esse pipeline automatizado não apenas acelera a implantação, mas também facilita a consistência e a confiabilidade em todos os ambientes.

Configure o ajuste de escala automático dos recursos, como instâncias do Amazon EC2, tarefas do Amazon ECS e réplicas de banco de dados, na IaC para aumentar e reduzir horizontalmente a escala de forma automática, conforme necessário. Essa abordagem aprimora a disponibilidade e o desempenho das aplicações e otimiza os custos ajustando dinamicamente os recursos com base na demanda. Para conferir uma lista dos recursos compatíveis, consulte Amazon EC2 Auto Scaling e AWS Auto Scaling.

Etapas de implementação

1. Crie e use um repositório de código-fonte para armazenar o código que controla sua configuração de infraestrutura. Confirme as alterações nesse repositório para refletir as alterações em andamento que você queira fazer.
2. Selecione uma solução de infraestrutura como código, como o AWS CloudFormation, para manter a infraestrutura atualizada e detectar inconsistências (desvios) em relação ao estado pretendido.
3. Integre sua plataforma de IaC ao seu pipeline de CI/CD para automatizar as implantações.
4. Determine e colete as métricas apropriadas para o ajuste de escala automático de recursos.
5. Configure o ajuste de escala automático dos recursos usando políticas de aumento e redução horizontal da escala para os componentes da workload. Considere usar a escalabilidade programada para padrões de uso previsíveis.
6. Monitore as implantações para detectar falhas e regressões. Implemente mecanismos de reversão em sua plataforma de CI/CD para reverter as alterações, se necessário.

Recursos

Documentos relacionados:

- AWS Auto Scaling: como os planos de ajuste de escala funcionam
- AWS Marketplace: produtos que podem ser usados com o Auto Scaling
- Gerenciar a capacidade de throughput automaticamente com o ajuste de escala automático do DynamoDB
- Usar um balanceador de carga com um grupo do Auto Scaling
- O que é o AWS Global Accelerator?
- O que é o Amazon EC2 Auto Scaling?
- O que é AWS Auto Scaling?
- O que é o Amazon CloudFront?
- O que é o Amazon Route 53?
- O que é Elastic Load Balancing?
- O que é um Network Load Balancer?
- O que é um Application Load Balancer?
- Integrating Jenkins with AWS CodeBuild and AWS CodeDeploy
- Creating a four stage pipeline with AWS CodePipeline

Vídeos relacionados:

- De volta ao básico: implante seu código no Amazon EC2
- AWS Supports You | Starting Your Infrastructure as Code Solution Using AWS CloudFormation Templates
- Streamline Your Software Release Process Using AWS CodePipeline
- Monitor AWS Resources Using Amazon CloudWatch Dashboards
- Create Cross Account & Cross Region CloudWatch Dashboards | Amazon Web Services

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 workload.

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 escalação (automática ou manual) são iniciadas para restaurar a disponibilidade após a detecção de uma falha ou degradação da experiência do cliente.

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

Orientação para 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 de implementação

- Defina 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 usa o AWS CloudFormation ou adiciona tags a recursos da AWS, é possível configurar planos de ajuste de escala para diferentes conjuntos de recursos por aplicação. O Auto Scaling faz recomendações de estratégias de ajuste de escala personalizadas para cada recurso. Depois que você criar seu plano de ajuste de escala, o Auto Scaling combinará o ajuste de escala dinâmico com os métodos de escalabilidade preditiva para oferecer suporte à sua estratégia de ajuste de escala. Para obter mais detalhes, consulte Como os planos de ajuste de escala funcionam.
  - O Amazon EC2 Auto Scaling verifica se você tem o número correto de instâncias do Amazon EC2 disponíveis para processar a carga da sua aplicação. Você cria coleções de instâncias EC2, chamadas de grupos de 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 Auto Scaling do Amazon DynamoDB usa o serviço Application Auto Scaling para ajustar dinamicamente a capacidade de throughput provisionado em seu nome em resposta aos padrões de tráfego reais. Isso permite que uma tabela ou um índice secundário global aumente a capacidade provisionada de leitura e gravação para processar aumentos repentinos no tráfego, sem limitações. Para obter mais detalhes, consulte Gerenciar a capacidade de throughput automaticamente com o ajuste de escala automático do DynamoDB.

Recursos

Práticas recomendadas relacionadas:

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

Documentos relacionados:

- AWS Auto Scaling: como os planos de ajuste de escala funcionam

- Gerenciar a capacidade de throughput automaticamente com o ajuste de escala automático do DynamoDB
- O que é o Amazon EC2 Auto Scaling?

REL07-BP03 Obter recursos após determinar que mais recursos são necessários para uma workload

Um dos atributos mais valiosos da computação em nuvem é a capacidade de provisionar recursos de maneira dinâmica.

Em ambientes computacionais on-premises tradicionais, você deve identificar e provisionar capacidade suficiente com antecedência para atender aos picos de demanda. Isso é um problema porque é caro e porque representa riscos à disponibilidade se você subestimar as necessidades de capacidade máxima da workload.

Na nuvem, isso não é necessário. Em vez disso, você pode provisionar a capacidade de computação, banco de dados e outros recursos conforme necessário para atender à demanda atual e prevista. Soluções automatizadas, como o Amazon EC2 Auto Scaling e o Application Auto Scaling, podem disponibilizar recursos on-line para você com base em métricas especificadas. Isso pode tornar o processo de escalabilidade mais fácil e previsível, além de tornar a workload significativamente mais confiável, garantindo que você tenha recursos suficientes disponíveis o tempo todo.

Resultado desejado: você configura o ajuste de escala automático da computação e de outros recursos para atender à demanda. Você fornece espaço suficiente nas políticas de escalabilidade para permitir que piscos de tráfego sejam atendidos enquanto recursos adicionais são colocados on-line.

Práticas comuns que devem ser evitadas:

- Provisionar um número fixo de recursos escaláveis.
- Escolher uma métrica de escalabilidade que não se correlaciona com a demanda real.
- Não fornecer espaço suficiente nos planos de escalabilidade para acomodar picos de demanda.
- As políticas de escalabilidade aumentam a capacidade tarde demais, o que leva à exaustão da capacidade e à degradação do serviço, enquanto recursos adicionais são colocados on-line.
- Não configurar corretamente as contagens mínima e máxima de recursos, o que leva a falhas de escalabilidade.

Benefícios de implementar essa prática recomendada: ter recursos suficientes para atender à demanda atual é fundamental para fornecer alta disponibilidade de workload e cumprir os objetivos definidos de nível de serviço (SLOs). O ajuste de escala automático permite que você forneça a quantidade certa de recursos de computação, banco de dados e outros que a workload precisa para atender à demanda atual e prevista. Você não precisa determinar as necessidades de pico de capacidade e alocar recursos estaticamente para atendê-las. Em vez disso, à medida que a demanda aumenta, você pode alocar mais recursos para acomodá-la e, depois que a demanda cair, pode desativar recursos para reduzir custos.

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

Orientação para implementação

Primeiro, determine se o componente da workload é adequado para o ajuste de escala automático. Esses componentes são chamados de escaláveis horizontalmente porque fornecem os mesmos recursos e se comportam de forma idêntica. Exemplos de componentes escaláveis horizontalmente incluem instâncias do EC2 que são configuradas da mesma forma, tarefas do Amazon Elastic Container Service (ECS) e pods executados no Amazon Elastic Kubernetes Service (EKS). Esses recursos computacionais geralmente estão localizados atrás de um balanceador de carga e são chamados de réplicas.

Outros recursos replicados podem incluir réplicas de leitura de banco de dados, tabelas do Amazon DynamoDB e clusters do Amazon ElastiCache (Redis OSS). Para obter uma lista completa dos recursos compatíveis, consulte Serviços da AWS que você pode usar com o Application Auto Scaling.

Para arquiteturas baseadas em contêineres, talvez seja necessário escalar de duas maneiras diferentes. Primeiro, talvez seja necessário escalar os contêineres que fornecem serviços escaláveis horizontalmente. Depois, talvez seja necessário escalar os recursos de computação a fim de criar espaço para novos contêineres. Existem diferentes mecanismos de ajuste de escala automático para cada camada. Para escalar tarefas do ECS, você pode usar o Application Auto Scaling. Para escalar os pods do Kubernetes, você pode usar o Horizontal Pod Autoscaler (HPA) ou o Kubernetes Event-driven Autoscaling (KEDA). Para escalar os recursos de computação, você pode usar provedores de capacidade para o ECS ou, para Kubernetes, você pode usar o Karpenter ou o Cluster Autoscaler.

Em seguida, selecione como você executará o ajuste de escala automático. Há três opções principais: escalabilidade baseada em métrica, escalabilidade programada e escalabilidade preditiva.

Escalabilidade baseada em métrica

A escalabilidade baseada em métrica provisiona recursos com base no valor de uma ou mais métricas de escalabilidade. Uma métrica de escalabilidade corresponde à demanda da workload. Uma boa maneira de determinar métricas de escalabilidade adequadas é executar testes de carga em um ambiente que não seja de produção. Durante os testes de carga, mantenha fixo o número de recursos escaláveis e aumente aos poucos a demanda (por exemplo, throughput, simultaneidade ou usuários simulados). Em seguida, procure métricas que aumentem (ou diminuam) à medida que a demanda cresce e, inversamente, diminuam (ou aumentem) à medida que a demanda cai. Métricas de escalabilidade típicas incluem utilização da CPU, profundidade da fila de trabalhos (como uma fila do Amazon SQS), número de usuários ativos e throughput da rede.

> **Note**
>
> A AWS observou que, na maioria das aplicações, a utilização de memória aumenta à medida que a aplicação se aquece, depois atinge um valor estável. Quando a demanda diminui, a utilização da memória normalmente permanece elevada em vez de diminuir paralelamente. Como a utilização da memória não corresponde à demanda em ambas as direções, ou seja, cresce e diminui com a demanda, reflita antes de selecionar essa métrica para ajuste de escala automático.

A escalabilidade baseada em métrica é uma operação latente. Pode levar vários minutos para que as métricas de utilização se propaguem para os mecanismos de ajuste de escala automático e esses mecanismos normalmente esperam por um sinal claro de aumento da demanda antes de reagir. Então, à medida que o mecanismo de ajuste de escala automático cria recursos, pode levar mais tempo para que eles fiquem totalmente operacionais. Por isso, é importante não definir suas metas métricas de escalabilidade muito próximas da utilização total (por exemplo, 90% de utilização da CPU). Fazer isso apresenta o risco de esgotar a capacidade de recursos existente antes que a capacidade adicional possa ser disponibilizada. As metas típicas de utilização de recursos podem variar entre 50 e 70% para uma disponibilidade ideal, dependendo dos padrões de demanda e do tempo necessário para provisionar recursos adicionais.

Escalabilidade programada

A escalabilidade programada provisiona ou remove recursos com base no calendário ou na hora do dia. É frequentemente usada para workloads com demanda previsível, como pico de utilização durante o horário comercial em dias úteis ou eventos de promoções. Tanto o Amazon EC2 Auto Scaling quanto o Application Auto Scaling oferecem suporte à escalabilidade programada. O cron scaler da KEDA oferece suporte à escalabilidade programada de pods do Kubernetes.

Escalabilidade preditiva

A escalabilidade preditiva usa machine learning para escalar automaticamente os recursos com base na demanda prevista. A escalabilidade preditiva analisa o valor histórico de uma métrica de utilização fornecida por você e prevê continuamente o valor futuro dela. O valor previsto é então usado para aumentar ou diminuir a escala do recurso. O Amazon EC2 Auto Scaling pode executar escalabilidade preditiva.

Etapas de implementação

1. Determine se o componente da workload é adequado para o ajuste de escala automático.
2. Determine que tipo de mecanismo de escalabilidade é mais apropriado para a workload: baseada em métrica, programada ou preditiva.
3. Selecione o mecanismo de ajuste de escala automático apropriado para o componente. Para instâncias do Amazon EC2, use o Amazon EC2 Auto Scaling. Para outros serviços da AWS, use o Application Auto Scaling. Para pods do Kubernetes (como aqueles executados em um cluster do Amazon EKS), considere o Horizontal Pod Autoscaler (HPA) ou o Kubernetes Event-driven Autoscaling (KEDA). Para nós do Kubernetes ou EKS, considere o Karpenter e o Cluster Auto Scaler (CAS).
4. Para escalabilidade baseada em métrica ou programada, realize testes de carga para determinar as métricas de escalabilidade e os valores-alvo apropriados para a workload. Para a escalabilidade programada, determine o número de recursos necessários nas datas e horários selecionados. Determine o número máximo de recursos necessários para atender ao pico de tráfego esperado.
5. Configure o escalador automático com base nas informações coletadas acima. Consulte a documentação do serviço de ajuste de escala automático para obter mais detalhes. Verifique se os limites máximo e mínimo de escalabilidade estão configurados corretamente.
6. Verifique se a configuração de escalabilidade está funcionando conforme esperado. Execute testes de carga em um ambiente que não seja de produção, observe como o sistema reage e ajuste conforme necessário. Ao ativar o ajuste de escala automático na produção, configure os alarmes apropriados para notificá-lo sobre qualquer comportamento inesperado.

Recursos

Documentos relacionados:

- O que é o Amazon EC2 Auto Scaling?

- AWS Prescriptive Guidance: Load testing applications
- AWS Marketplace: produtos que podem ser usados com o Auto Scaling
- Gerenciar a capacidade de throughput automaticamente com o ajuste de escala automático do DynamoDB
- Ajuste de escala preditivo para o EC2 com Machine Learning
- Ajuste de escala agendado para o Amazon EC2 Auto Scaling
- Histórias sobre a Lei de Little

REL07-BP04 Fazer o teste de carga da workload

Adote uma metodologia de teste de carga para avaliar se a ação de ajuste de escala atende aos requisitos da workload.

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, é possível criar um ambiente de teste em escala de produção sob demanda, concluir seus testes e desativar 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 game days 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.

Práticas comuns que devem ser evitadas:

- Executar testes de carga em implantações que não têm a mesma configuração da sua produção.
- Executar testes de carga apenas em componentes individuais da workload, 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 implementar 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. Diversas combinações de solicitações são possíveis. Portanto, você deve examinar vários períodos ao identificar a combinação de tráfego.
    - Implemente um direcionador de carga. É possível usar código personalizado, código aberto ou um software comercial para implementar um direcionador de carga.
    - Faça o teste de carga inicialmente com uma pequena capacidade. Você percebe 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, recomenda-se testar o mais próximo possível de um ambiente de produto.

Recursos

Documentos relacionados:

- Teste de carga distribuída na AWS: simular milhares de usuários conectados
- Aplicações de teste de carga

Vídeos relacionados:

- AWS Summit ANZ 2023: Acelerar com confiança com o teste de carga distribuída da AWS

Implementar alterações

Alterações controladas são necessárias para implantar novas funcionalidades e garantir que as workloads e o ambiente operacional estejam executando software conhecido e com os patches mais

atuais. Se essas alterações não forem controladas, será difícil prever o efeito dessas alterações ou resolver os problemas que surgem por causa delas.

Padrões de implantação adicionais para minimizar riscos

Sinalizadores de recursos (também conhecidos como alternâncias de recursos) são opções de configuração em uma aplicação. É possível implantar o software com um recurso desativado para que seus clientes não o vejam. Você então pode ativar o recurso, como faria para uma implantação canário, ou pode definir o ritmo da alteração como 100% para ver o efeito. Se a implantação apresentar problemas, você poderá simplesmente desativar o recurso outra vez sem reverter.

Implantação por zonas isoladas de falhas: uma das regras mais importantes que a AWS estabeleceu para as próprias implantações é evitar trabalhar em várias zonas de disponibilidade em uma região ao mesmo tempo. Isso é fundamental para garantir a independência das zonas de disponibilidade para os fins de nossos cálculos de disponibilidade. Recomendamos empregar considerações similares em suas implantações.

Revisões de prontidão operacional (ORRs)

A AWS recomenda realizar revisões de prontidão operacional para avaliar se o teste é completo, a habilidade de monitorar e, mais importante, a capacidade de auditar a performance das aplicações conforme seus SLAs e fornecer dados no caso de uma interrupção ou outra anomalia de operação. Uma ORR formal é realizada antes da produção inicial. A AWS repetirá ORRs periodicamente (uma vez por ano ou antes de cada período de performance crítico) para garantir que não tenha ocorrido um desvio das expectativas operacionais. Para obter mais informações sobre a prontidão operacional, consulte o pilar Excelência operacional do AWS Well-Architected Framework.

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 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 manual ou automaticamente. Os exemplos incluem a

implantação de workloads, aplicação de patches a workloads ou a realização de modificações de DNS.

Por exemplo, coloque processos em vigor para garantir a segurança da 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 a execução automatizada quando adequado.

Mesmo para workloads sofisticadas altamente automatizadas, os runbooks ainda são úteis para executar game days 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.

Práticas comuns que devem ser evitadas:

- 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 possibilidade de reversão.

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

Orientação para implementação

- Habilite respostas consistentes e rápidas para eventos bem conhecidos, documentando procedimentos nos runbooks.
  - 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 sua infraestrutura, você pode usar software de controle de versão para controlar as versões e rastrear as alterações.
  - Use o AWS CloudFormation (ou um provedor terceirizado confiável) para definir sua infraestrutura.

    - O que é 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.
        - Como controlar o acesso com o AWS Identity and Access Management
    - Use um sistema hospedado de gerenciamento de código-fonte baseado em uma tecnologia popular, como o Git, para armazenar seu código-fonte e configuração de infraestrutura como código (IaC).

Recursos

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudar 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 AWS CloudFormation?

Exemplos relacionados:

- Automatizar operações com playbooks e runbooks

REL08-BP02 Integrar testes funcionais como parte da sua implantação

Use técnicas como testes de unidade e testes de integração que validem a funcionalidade necessária.

O teste de unidade é o processo em que você testa a menor unidade funcional de código para validar o comportamento dela. O teste de integração busca validar que cada recurso da aplicação funciona de acordo com os requisitos do software. Enquanto os testes de unidade se concentram em testar parte de uma aplicação isoladamente, os testes de integração consideram os efeitos colaterais (por exemplo, o efeito da alteração de dados por meio de uma operação de mutação). Em ambos os casos, os testes devem ser integrados em um pipeline de implantação e, se os critérios de sucesso 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.

Os melhores resultados são obtidos quando esses testes são executados automaticamente como parte das ações de compilação e implantação. Por exemplo, com o AWS CodePipeline, os desenvolvedores confirmam alterações em um repositório de origem onde o CodePipeline detecta automaticamente as alterações. A aplicação é compilada e os testes de unidade são executados. Após os testes de unidade serem aprovados, o código criado será implantado nos servidores de preparação para a realização de testes. No servidor de preparação, o CodePipeline executa mais testes, como testes de integração ou de carregamento. Após a conclusão bem-sucedida desses testes, o CodePipeline implanta o código testado e aprovado nas instâncias de produção.

Resultado desejado: você usa a automação para realizar testes de unidade e integração a fim de validar que o código se comporta conforme o esperado. Esses testes são integrados ao processo de implantação, e uma falha no teste aborta a implantação.

Práticas comuns que devem ser evitadas:

- Ignorar ou pular falhas e planos de teste durante o processo de implantação para acelerar o cronograma de implantação.
- Você realiza testes manualmente fora do pipeline de implantação.
- Ignorar as etapas de teste na automação por meio de fluxos de trabalho manuais de emergência.
- Executar testes automatizados em um ambiente que não se parece muito com o ambiente de produção.
- Criar um conjunto de testes que não é suficientemente flexível e que é difícil de manter, atualizar ou escalar à medida que a aplicação evolui.

Benefícios de implementar essa prática recomendada: testes automatizados durante o processo de implantação detectam problemas de maneira antecipada, o que reduz o risco de uma liberação para produção com bugs ou comportamento inesperado. Os testes de unidade validam que o código se comporta conforme desejado e que os contratos de API são respeitados. Os testes de integração validam a operação do sistema de acordo com os requisitos especificados. Esses tipos de testes verificam a ordem de funcionamento pretendida dos componentes, como interfaces de usuário, APIs, bancos de dados e código-fonte.

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

Orientação para implementação

Adote a abordagem de desenvolvimento orientado por testes (TDD) para gravar software, em que você desenvolve casos de teste e valida o código. Para começar, crie casos de teste para cada

função. Se o teste falhar, escreva um novo código para passar no teste. Essa abordagem ajuda você a validar o resultado esperado de cada função. Execute testes de unidade e valide se eles foram aprovados antes de confirmar o código em um repositório de código-fonte.

Implemente testes de unidade e de integração como parte dos estágios de compilação, teste e implantação do pipeline de CI/CD. Automatize os testes e inicie-os automaticamente sempre que uma nova versão da aplicação estiver pronta para ser implantada. Se os critérios de êxito não forem atendidos, o pipeline será interrompido ou revertido.

Se a aplicação for um aplicativo web ou móvel, realize testes de integração automatizados em vários navegadores de desktop ou dispositivos reais. Essa abordagem é particularmente útil para validar a compatibilidade e a funcionalidade dos aplicativos móveis em uma ampla variedade de dispositivos.

Etapas de implementação

1. Escreva testes de unidade antes de escrever código funcional (desenvolvimento orientado a testes, ou TDD). Estabeleça diretrizes de código para que escrever e executar testes de unidade sejam um requisito de codificação não funcional.
2. Crie um conjunto de testes de integração automatizados que cubram as funcionalidades testáveis identificadas. Esses testes devem simular as interações do usuário e validar os resultados esperados.
3. Crie o ambiente de teste necessário para executar os testes de integração. Isso pode incluir ambientes de preparação ou pré-produção que imitam o ambiente de produção.
4. Configure os estágios de código-fonte, compilação, teste e implantação usando o console do AWS CodePipeline ou a AWS Command Line Interface (CLI).
5. Implante a aplicação depois que o código for criado e testado. O AWS CodeDeploy pode implantá-lo nos ambientes de preparação (teste) e produção. Esses ambientes podem incluir instâncias do Amazon EC2, funções do AWS Lambda ou servidores on-premises. O mesmo mecanismo de implantação deve ser usado para implantar a aplicação em todos os ambientes.
6. Monitore o progresso do pipeline e o status de cada estágio. Utilize verificações de qualidade para bloquear o pipeline de acordo com o status dos testes. Você também pode receber notificações sobre qualquer falha ou conclusão do estágio do pipeline.
7. Monitore continuamente os resultados dos testes e procure padrões, regressões ou áreas que exijam mais atenção. Use essas informações para melhorar o conjunto de testes, identificar áreas da aplicação que precisam de testes mais robustos e otimizar o processo de implantação.

Recursos

Práticas recomendadas relacionadas:

- REL07-BP04 Fazer o teste de carga da workload
- REL08-BP03 Integrar testes de resiliência como parte da implantação
- REL12-BP04 Testar a resiliência com engenharia do caos

Documentos relacionados:

- AWS Prescriptive Guidance: Test automation
- Integração e entrega contínuas
- Indicadores para testes funcionais
- Monitorar pipelines
- Usar o AWS CodePipeline com o AWS CodeBuild para testar código e executar compilações
- AWS Device Farm

REL08-BP03 Integrar testes de resiliência como parte da 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 focam a 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 game days.

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

Práticas comuns que devem ser evitadas:

- 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 as práticas recomendadas: os testes de resiliência integrados em suas implantações ajudam a identificar problemas desconhecidos no sistema que, de outra forma, passariam despercebidos e poderiam levar à inatividade da 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 risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para 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 AWS Resilience Hub podem ser integrados ao seu pipeline de CI/CD para estabelecer avaliações contínuas de resiliência que são avaliadas 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, standby passivo, standby a quente e ativo-ativo, e todas elas diferem em custo e complexidade. Antes do teste de recuperação de desastres, recomendamos definir 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 a AWS Elastic Disaster Recoveryque ajudam 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 AWS Fault Injection Service para ganhar confiança antes da implantação na produção.

Recursos

Documentos relacionados:

- Experimentar com falhas usando testes de resiliência para aumentar a preparação para a recuperação
- Avaliar continuamente a resiliência da aplicação com o AWS Resilience Hub e o AWS CodePipeline
- Arquitetura de recuperação de desastres (DR) na AWS, Parte I: estratégias de recuperação na nuvem
- Verificar a resiliência de suas workloads via engenharia do caos
- Princípios da engenharia do caos
- Workshop de engenharia do caos

Vídeos relacionados:

- AWS re:Invent 2020: Testar a resiliência via engenharia do caos
- Melhorar a resiliência de aplicações com o AWS Fault Injection Service
- Prepare e proteja suas aplicações contra interrupções com o AWS Resilience Hub

REL08-BP04 Implantar usando infraestrutura imutável

A infraestrutura imutável é um modelo que não requer atualizações, patches de segurança ou alterações na configuração 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 uma 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.

Práticas comuns que devem ser evitadas:

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

Benefícios de implementar esta prática recomendada:

- Maior consistência em todos os ambientes: como não há diferenças nos recursos de infraestrutura entre os ambientes, a consistência aumenta e os testes são simplificados.
- Redução nos desvios de configuração: ao substituir recursos de infraestrutura por uma configuração conhecida e com controle de versão, a infraestrutura é redefinida para um estado conhecido, testado e confiável, evitando assim desvios de configuração.
- Implantações atômicas confiáveis: as implantações são concluídas com sucesso ou 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 foi alterada. Em caso de erros, é possível reverter para ela.
- Postura de segurança aprimorada: ao não permitir alterações na infraestrutura, os mecanismos de acesso remoto (como o SSH) podem ser desabilitados. Isso reduz o vetor de ataque, melhorando a postura de segurança da organização.

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

Orientação para 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 o potencial de erro humano. Para obter mais detalhes, consulte REL08-BP05 Implantar alterações com automação e Automatizar implantações seguras e sem intervenção manual.

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 código-fonte. 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: a prática de direcionar um pequeno número de seus 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. O tráfego poderá ser removido da implantação canário se problemas críticos forem encontrados, e os usuários poderão ser enviados de volta para a versão anterior. Se a implantação for bem-sucedida, você poderá continuar implantando na velocidade desejada enquanto monitora as alterações em busca de erros até a implantação ser concluída. O AWS CodeDeploy pode ser configurado com uma configuração de implantação que permita uma implantação canário.

Implantação azul/verde: 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 chaveado de uma só vez. No entanto, você também pode 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.

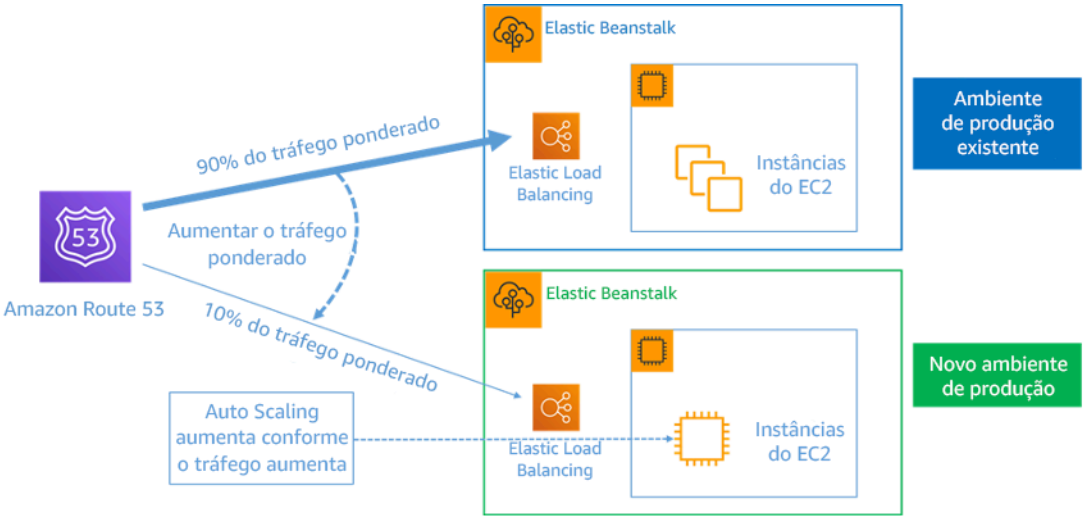


Figura 8: Implantação azul/verde com o AWS Elastic Beanstalk e o Amazon Route 53

Detecção de desvios

Um desvio é definido como qualquer alteração que faz com que um recurso de infraestrutura tenha um estado ou 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 de implementação

- Proíba 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 centralmente permissões refinadas e analisar o acesso para refinar permissões na AWS.
- Automatize a implantação dos recursos da infraestrutura para aumentar a reprodutibilidade e minimizar a possibilidade de erro humano.
  - Conforme descrito em Introdução a DevOps no whitepaper da AWS, a automação é a base dos serviços da AWS e possui suporte interno em todos os serviços, recursos e ofertas.
  - Pré-preparar a imagem de máquina da Amazon (AMI) pode acelerar o tempo de lançamento. O EC2 Image Builder é um serviço da AWS totalmente gerenciado que ajuda a automatizar a criação, a manutenção, a validação, o compartilhamento e a implantação de AMIs personalizadas do Linux ou do Windows.
  - Alguns dos serviços compatíveis com automação são:

    - O AWS Elastic Beanstalk é um serviço para implantação e escalação rápidas de aplicações Web desenvolvidas com Java, .NET, PHP, Node.js, Python, Ruby, Go e Docker em servidores familiares, como Apache, NGINX, Passenger e IIS.
    - O AWS Proton ajuda as equipes de plataforma a conectar e coordenar todas as diferentes ferramentas que suas equipes de desenvolvimento precisam para provisionamento de infraestrutura, implantação de código, monitoramento e atualizações. O AWS Proton torna possível a infraestrutura automatizada na forma de provisionamento de código e implantação de aplicações de tecnologia sem servidor baseadas em contêiner.
- 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, insere-os em um sistema de controle de versão, e o AWS 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 se integra 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 de aplicações 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 API Cloud Control apresenta um conjunto comum de APIs de criação, leitura, atualização, exclusão e lista (CRUDL) para ajudar os desenvolvedores a gerenciar sua infraestrutura de nuvem de forma fácil e consistente. As APIs comuns do Cloud Control 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 canário do API Gateway
    - Criar um pipeline com implantações canário para o Amazon ECS usando o AWS App Mesh
  - Implantações azuis/verdes: Implantações azuis/verdes no whitepaper da AWS descreve exemplos de técnicas para implementar estratégias de implantação azul/verde.
- Detecte variações de configuração ou estado. Para obter mais informações, consulte Como detectar alterações de configuração não gerenciadas para pilhas e recursos.

Recursos

Práticas recomendadas relacionadas:

- REL08-BP05 Implantar alterações com automação

Documentos relacionados:

- Automatizar implantações seguras e sem intervenção
- Como utilizar o AWS CloudFormation para criar uma infraestrutura imutável no Nubank
- Infraestrutura como código
- Implementar um alarme para detectar automaticamente desvios nas pilhas do AWS CloudFormation

Vídeos relacionados:

- AWS re:Invent 2020: Confiabilidade, consistência e confiança por meio da imutabilidade

REL08-BP05 Implantar alterações com automação

As implantações e a aplicação de patches são automatizadas para eliminar impactos negativos.

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 negócios abordados pelo software. 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 no processo de lançamento com testes extensivos de pré-produção, reversões automáticas e implantações de produção em etapas. 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.

Práticas comuns que devem ser evitadas:

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

Benefícios de implementar esta prática recomendada: ao usar a automação para implantar todas as alterações, você remove o potencial de introdução de erros humanos 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 risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para 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 contínuas (CI/CD) em que uma confirmação ou alteração de código passa por vários estágios automatizados, desde os estágios de construção e teste até a implantação em ambientes de produção.

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

Etapas de implementação

É possível automatizar as implantações para remover as operações manuais seguindo estas etapas:

- Configure um repositório de código para armazenar o código com segurança: use um sistema hospedado de gerenciamento de código-fonte baseado em uma tecnologia popular, como o Git, para armazenar o código-fonte e a configuração de infraestrutura como código (IaC).
- Configure um serviço de integração contínua para compilar seu código-fonte, executar testes e criar artefatos de implantação: para configurar um projeto de compilação para essa finalidade, consulte Introdução ao AWS CodeBuild usando o 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 toda uma variedade de serviços computacionais, como Amazon EC2, AWS Fargate, AWS Lambda e seus servidores on-premises. Para configurar essas etapas, consulte Introdução ao CodeDeploy.
- Configure um serviço de entrega contínua que automatize os pipelines de lançamento para atualizações rápidas e confiáveis de aplicações e infraestrutura: considere usar o AWS CodePipeline para obter ajuda para automatizar os pipelines de lançamento. Para obter mais detalhes, consulte os tutoriais do CodePipeline.

Recursos

Práticas recomendadas relacionadas:

- OPS05-BP04 Usar sistemas de gerenciamento de compilação e 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:

- Entrega contínua de pilhas do AWS CloudFormation aninhadas usando o AWS CodePipeline
- Parceiro da APN: parceiros que podem ajudar a criar soluções de implantação automatizada
- AWS Marketplace: produtos que podem ser usados para automatizar suas implantações
- Automatizar mensagens de chat com webhooks
- Amazon Builders' Library: como garantir a segurança da reversão durante implantações
- Amazon Builders' Library: como aumentar a velocidade com a entrega contínua
- O que é o AWS CodePipeline?

- O que é CodeDeploy?
- Gerenciador de patches do AWS Systems Manager
- O que é o Amazon SES?
- O que é o Amazon Simple Notification Service?

Vídeos relacionados:

- AWS Summit 2019: CI/CD na AWS

Gerenciamento de falhas

> Falhas são inevitáveis e, com o tempo, tudo acabará falhando: de roteadores a discos rígidos, de sistemas operacionais a unidades de memória corrompendo pacotes TCP, de erros transitórios a falhas permanentes. Isso é um fato, não importa se você usa 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 data center on-premises. Na nuvem, no entanto, você deve estar protegido contra a maioria desses tipos de falhas. Por exemplo, os 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 disponibilidade de 99,999%. Os objetos do Amazon S3 são armazenados no mínimo em três zonas de disponibilidade, fornecendo 99,999999999% de durabilidade de objetos durante um determinado ano. Independentemente do seu provedor de nuvem, há o potencial de falhas para afetar sua workload. Portanto, você deve tomar medidas para implementar a resiliência se precisar que sua workload.

Um pré-requisito para aplicar as práticas recomendadas discutidas aqui é que você deve garantir que as pessoas que projetam, implementam e operam suas workloads 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 práticas recomendadas para gerenciar falhas para evitar impacto na workload.

Tópicos

Fazer backup dos dados

Faça backup de dados, aplicações 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 backups 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: fundamental

Práticas comuns que devem ser evitadas:

- 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 de implementar esta prática recomendada: identificar os locais onde os backups são necessários e implementar um mecanismo para criar backups, ou ser capaz de reproduzir os dados de uma fonte externa, melhora a capacidade de restaurar e recuperar dados durante uma interrupção.

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

Orientação para implementação

Todos os datastores da AWS oferecem recursos de backup. Serviços como o Amazon RDS e o Amazon DynamoDB oferecem suporte adicional ao backup automatizado que possibilita a recuperação para um ponto no tempo (PITR), permitindo restaurar um backup a qualquer momento até cinco minutos ou menos antes da hora atual. Muitos serviços da AWS oferecem a capacidade de copiar backups para outra Região da AWS. O AWS Backup é uma ferramenta que permite centralizar e automatizar a proteção de dados em todos os serviços da AWS. O AWS Elastic Disaster Recovery permite copiar workloads completas do servidor e manter a proteção contínua dos dados no local, entre AZ 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. Serviços da AWS, como o Amazon EBS, o Amazon RDS e o Amazon DynamoDB, oferecem recursos integrados de criação de backups. É possível também usar um software de backup de terceiros.

Os dados on-premises podem ser copiados para a Nuvem AWS via AWS Storage Gateway ou 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 o custo do armazenamento de dados.

É possível 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 poderão ser usados para reproduzir dados se os dados primários forem perdidos. Nos casos em que fontes como essa podem ser usadas para atender ao objetivo de ponto de recuperação (RPO) e ao objetivo de tempo de recuperação (RTO), talvez um backup não seja necessário. Outro exemplo, se estiver trabalhando com o Amazon EMR, é que talvez não seja necessário fazer backup do seu datastore HDFS, desde que você consiga reproduzir os dados no Amazon EMR a partir do 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 respeitar o RTO para a workload.

Etapas de 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 em log e armazenamento de objetos. Consulte a seção Recursos para encontrar Documentos relacionados sobre os diferentes serviços da AWS em que os dados são armazenados e sobre a capacidade de backup oferecida por esses serviços.
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 serviços da AWS serviços ou 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 cuida da replicação automatizada de dados em menos de um segundo para 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. Se você tiver carregado dados do Amazon S3 para um data warehouse (como o Amazon Redshift) ou cluster MapReduce (como o Amazon EMR) para fazer análises nesses dados, isso pode ser 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 cadência para fazer backup dos 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 AWS Backup?
- O que é o AWS DataSync?
- O que é Gateway de Volumes?
- Parceiro da APN: parceiros que podem ajudar com o backup
- AWS Marketplace: produtos que podem ser usados para backup
- Snapshots do Amazon EBS
- Fazer backup do Amazon EFS
- Fazer backup do Amazon FSx para Windows File Server
- Backup e restauração do ElastiCache para Redis
- Criar um snapshot de cluster de banco de dados no Neptune
- Criar um snapshot de banco de dados
- Criar uma regra do EventBridge que é acionada de acordo com uma programação
- Replicação entre regiões com o Amazon S3
- EFS para AWS Backup de EFS
- Exportar dados de log para o Amazon S3
- Gerenciamento do ciclo de vida de objetos
- Backup e restauração sob demanda para o DynamoDB
- Recuperação para um ponto no tempo para o DynamoDB
- Como trabalhar com snapshots de índices no Amazon OpenSearch Service
- O que é o AWS Elastic Disaster Recovery?

Vídeos relacionados:

- AWS re:Invent 2021: Backup, recuperação de desastres e proteção contra ransomware com a AWS
- Demonstração do AWS Backup: backup entre contas e regiões
- AWS re:Invent 2019: Mergulho profundo no AWS Backup com destaque para o Rackspace (STG341)

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.

Práticas comuns que devem ser evitadas:

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

Benefícios de implementar esta prática recomendada: proteger os backups impede a violação dos dados, e a criptografia dos dados impede o acesso a eles caso sejam expostos por engano.

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

Orientação para 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 em repouso. Ao usar a criptografia do lado do servidor, o Amazon S3 aceita os objetos como dados não criptografados e, depois, criptografa-os à medida que são armazenados. 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. Quando o AWS Elastic Disaster Recovery é usado, 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 de implementação

1. Use criptografia em cada um dos seus datastores. 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 nos 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 pertencente à 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. É possível especificar as chaves de destino ao configurar a replicação.
   - Escolha se deseja usar a criptografia padrão ou usar a criptografia do Amazon EBS para Elastic Disaster Recovery. 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: produtos que podem ser usados para backup
- Criptografia do Amazon EBS

- Amazon S3: proteger dados usando criptografia
- Configuração adicional de CRR: replicar objetos criados com a criptografia do lado do servidor (SSE) usando as chaves de criptografia armazenadas no AWS KMS
- Criptografia do DynamoDB em repouso
- Como criptografar recursos do Amazon RDS
- Criptografar dados e metadados no Amazon EFS
- Criptografia para backups no AWS
- Como gerenciar tabelas criptografadas
- Pilar Segurança: AWS Well-Architected Framework
- O que é o AWS Elastic Disaster Recovery?

REL09-BP03 Realizar backups 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 das fontes de dados em uma cadência estabelecida.

Práticas comuns que devem ser evitadas:

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

Benefícios de implementar esta prática recomendada: automatizar os backups verifica se eles são feitos regularmente com base no seu RPO e alerta você caso não sejam feitos.

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

Orientação para implementação

O AWS Backup pode ser usado para criar backups de dados automatizados de várias fontes de dados da AWS. É possível fazer backup de instâncias do Amazon RDS quase continuamente a

cada cinco minutos, e objetos do Amazon S3 podem ser feitos backup quase continuamente a cada quinze minutos, proporcionando recuperação para um ponto no tempo (PITR) dentro do 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 para um ponto no tempo incluem Amazon DynamoDB, Amazon RDS e Amazon Keyspaces (para Apache Cassandra): esses podem ser restaurados em um momento específico no histórico de 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 para um ponto no tempo. O versionamento do Amazon S3, uma vez habilitado, é automático. O Amazon Data Lifecycle Manager pode ser usado para automatizar a criação, a retenção e a exclusão de AMIs compatíveis com o 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 controle de versões, 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 região diferente.

Etapas de implementação

1. Identifique as fontes de dados que estão sendo copiadas para backup manualmente no momento. 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 para a 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 automatizada ou um serviço de backup gerenciado. O AWS Backup é um serviço totalmente gerenciado que ajuda você a centralizar e automatizar a proteção de dados nos serviços da AWS, na nuvem e on-premises. Usando planos de backup no AWS Backup, crie 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 orientação prática sobre como criar backups automatizados usando o AWS Backup, consulte Testar o backup e a restauração de dados. A maioria dos serviços da AWS que armazenam dados oferecem recursos de backup nativos. Por exemplo, o RDS pode ser utilizado para fazer backups automatizados com recuperação para um ponto no tempo (PITR).
4. Para fontes de dados sem suporte de uma solução de backup automatizada ou serviço gerenciado, como fontes de dados on-premises ou filas de mensagens, considere usar uma solução terceirizada confiável para criar backups automatizados. Como alternativa, você pode criar automação para fazer isso usando a AWS CLI ou os SDKs. É possível usar o AWS Lambda Functions ou o AWS Step Functions para definir a lógica envolvida na criação de um backup de dados e usar o Amazon EventBridge para executá-la em uma frequência baseada no RPO.

Nível de esforço do plano de implementação: Baixo.

Recursos

Documentos relacionados:

- Parceiro da APN: parceiros que podem ajudar com o backup
- AWS Marketplace: produtos que podem ser usados para backup
- Criar uma regra do EventBridge que é acionada de acordo com uma programação
- O que é o AWS Backup?
- O que é o AWS Step Functions?
- O que é o AWS Elastic Disaster Recovery?

Vídeos relacionados:

- AWS re:Invent 2019: Mergulho profundo no AWS Backup com destaque para o Rackspace (STG341)

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 dos backups são recuperados periodicamente usando mecanismos bem definidos para verificar se a recuperação é 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).

Práticas comuns que devem ser evitadas:

- Restaurar um backup, mas não consultar ou recuperar os dados para garantir que a restauração é utilizável.
- Presumir a existência de um backup.
- Presumir que o backup de um sistema esteja totalmente operacional e que os dados possam ser recuperados.
- 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 não seguir um procedimento automatizado estabelecido.

Benefícios de estabelecer essa prática recomendada: testar a recuperação dos backups verifica se os dados podem ser restaurados quando necessário, sem a preocupação de que os dados possam estar ausentes ou corrompidos, que a restauração e a recuperação sejam possíveis dentro do RTO da workload e que qualquer perda de dados esteja dentro do RPO da workload.

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

Orientação para implementação

Testar o recurso de backup e restauração aumenta a confiança na capacidade 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, se eles não estão corrompidos e se eles estão acessíveis, bem como 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 para um ponto no tempo (PITR). Ao usar o backup contínuo, você pode restaurar o conjunto de dados para o estado em que ele se encontrava em uma data e hora especificadas.

Se todos os dados estão 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 para um ponto no tempo de volumes do Amazon EBS. À medida que os servidores de origem são replicados, os estados pontuais são registrados ao longo do tempo com base na política configurada. O Elastic Disaster Recovery ajuda você a verificar a integridade desses snapshots lançando instâncias para fins de teste e detalhamento sem redirecionar o tráfego.

Etapas de implementação

1. Identifique as fontes de dados que estão sendo copiadas 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 validação de dados para 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 dos dados. 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 Hub de Resiliência da AWS, você pode executar uma avaliação da sua 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 usando 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 poderá ser tão simples quanto restaurar o backup em um novo recurso. Se você usou o AWS Elastic Disaster Recovery, pode iniciar um exercício de recuperação.
6. Valide a recuperação de dados do recurso restaurado com base nos critérios que você estabeleceu anteriormente para validação de 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. Meça o tempo necessário para restauração e recuperação e compare-o com seu 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 demora. Todas as chamadas da API da AWS contêm informações de data e 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 estiver usando um processo automatizado, 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 a 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 de seus metadados que podem ser usadas para medir o tempo necessário para restauração e recuperação.
8. Notifique 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 push, como e-mail ou SMS, às partes interessadas. Essas mensagens também podem ser publicadas em aplicações de

mensagens, como Amazon Chime, Slack ou 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 invocar 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, esse laboratório do Well-Architected fornece uma experiência prática sobre uma forma de automatizar várias das etapas descritas aqui.

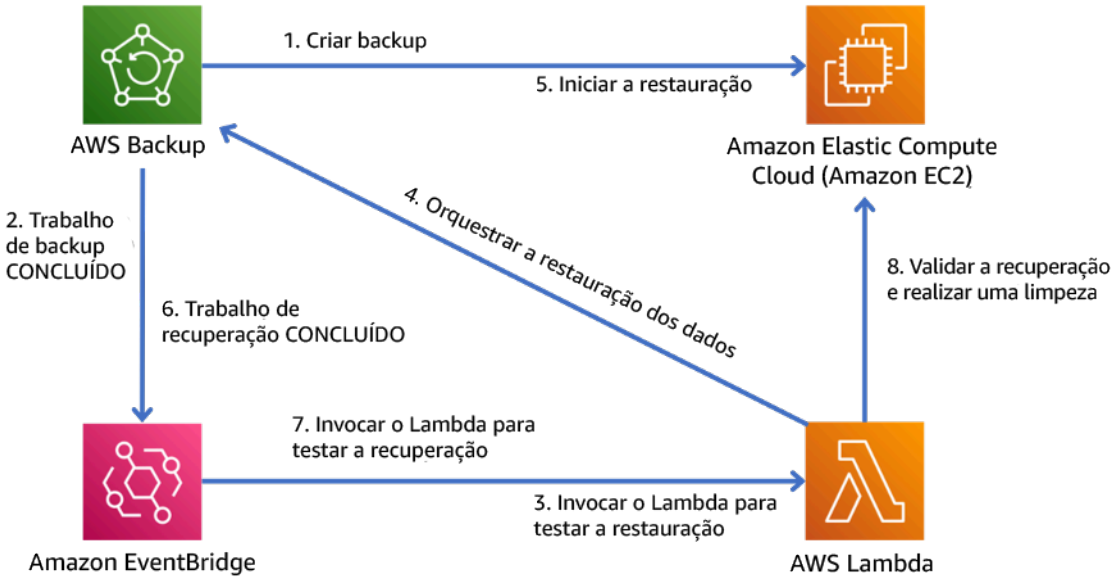


Figura 9. Um processo de backup e restauração automatizado

Nível de esforço para o plano de implementação: Moderado a alto, dependendo da complexidade dos critérios de validação.

Recursos

Documentos relacionados:

- Automatizar a validação da recuperação de dados com o AWS Backup.
- Parceiro da APN: parceiros que podem ajudar com o backup
- AWS Marketplace: produtos que podem ser usados para backup
- Criar uma regra do EventBridge que é acionada de acordo com uma programação

- Backup e restauração sob demanda para o DynamoDB
- O que é o AWS Backup?
- O que é o AWS Step Functions?
- O que é o AWS Elastic Disaster Recovery
- AWS Elastic Disaster Recovery

Use o isolamento de falhas para proteger a workload

O isolamento de falhas limita o impacto de uma falha de componente ou do sistema a um limite definido. Com o isolamento adequado, os componentes fora do limite não são afetados pela falha. Executar a workload em vários limites de isolamento de falhas pode torná-la mais resistente a falhas.

Práticas recomendadas

- REL10-BP01 Implantar a workload em vários locais
- REL10-BP02 Automatizar a recuperação de componentes restritos a um único local
- REL10-BP03 Usar arquiteturas de anteparo para limitar o escopo de impactos

REL10-BP01 Implantar a workload em vários locais

Distribua os dados e os recursos da workload por várias zonas de disponibilidade ou, quando necessário, por Regiões da AWS.

Um princípio fundamental do design de serviço na AWS é evitar pontos únicos de falha, incluindo a infraestrutura física subjacente. A AWS fornece recursos e serviços de computação em nuvem globalmente em várias localizações geográficas chamadas regiões. Cada região é física e logicamente independente e consiste em três ou mais zonas de disponibilidade (AZs). As zonas de disponibilidade ficam geograficamente próximas umas das outras, mas estão fisicamente separadas e isoladas. Ao distribuir as workloads entre zonas de disponibilidade e regiões, você reduz o risco de ameaças, como incêndios, inundações, desastres relacionados ao clima, terremotos e erro humano.

Crie uma estratégia de localização para fornecer alta disponibilidade adequada às workloads.

Resultado desejado: as workloads de produção são distribuídas entre várias zonas de disponibilidade (AZs) ou regiões para oferecer tolerância a falhas e alta disponibilidade.

Práticas comuns que devem ser evitadas:

- A workload de produção existe somente em uma única zona de disponibilidade.
- Implantar uma arquitetura multirregional quando uma arquitetura multi-AZ é suficiente para satisfazer os requisitos de negócios.
- As implantações ou os dados ficam dessincronizados, o que resulta em desvio na configuração ou em dados sub-replicados.
- Não contabilizar as dependências entre os componentes da aplicação se os requisitos de resiliência e de vários locais são diferentes entre esses componentes.

Benefícios de implementar essa prática recomendada:

- A workload é mais resistente a incidentes, como falhas de energia ou de controle ambiental, desastres naturais, falhas no serviço upstream ou problemas de rede que afetam uma AZ ou uma região inteira.
- Você pode acessar um inventário mais amplo de instâncias do Amazon EC2 e reduzir a probabilidade de InsufficientCapacityExceptions (ICE) ao iniciar tipos específicos de instância do EC2.

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

Orientação para implementação

Implemente e opere todas as workloads de produção em pelo menos duas zonas de disponibilidade (AZs) em uma região.

Usar várias zonas de disponibilidade

As zonas de disponibilidade são locais de hospedagem de recursos fisicamente separados uns dos outros para evitar falhas correlacionadas devido a riscos como incêndios, inundações e tornados. Cada zona de disponibilidade tem uma infraestrutura física independente, incluindo conexões de energia elétrica, fontes de energia de backup, serviços mecânicos e conectividade de rede. Esse arranjo limita as falhas em qualquer um desses componentes apenas à zona de disponibilidade afetada. Por exemplo, se um incidente em toda a AZ tornar as instâncias do EC2 indisponíveis na zona de disponibilidade afetada, suas instâncias em outra zona de disponibilidade permanecerão disponíveis.

Apesar de serem separadas fisicamente, as zonas de disponibilidade na mesma Região da AWS são próximas o suficiente para fornecer redes de alto throughput e baixa latência (milissegundo de um

dígito). Você pode replicar dados de forma síncrona entre as zonas de disponibilidade para a maioria das workloads sem afetar significativamente a experiência do usuário. Assim, você pode usar as zonas de disponibilidade de uma região em uma configuração ativa/ativa ou ativa/em espera.

Toda a computação associada à workload deve ser distribuída entre várias zonas de disponibilidade. Isso inclui instâncias do Amazon EC2, tarefas do AWS Fargate e funções do AWS Lambda anexadas à VPC. Os serviços de computação da AWS, incluindo EC2 Auto Scaling, Amazon Elastic Container Service (ECS) e Amazon Elastic Kubernetes Service (EKS), fornecem maneiras de você iniciar e gerenciar a computação em todas as zonas de disponibilidade. Configure-os para substituir automaticamente a computação conforme necessário em uma zona de disponibilidade diferente para manter a disponibilidade. Para direcionar o tráfego para as zonas de disponibilidade disponíveis, coloque um balanceador de carga na frente da computação, como um Application Load Balancer ou Network Load Balancer. Os balanceadores de carga da AWS podem redirecionar o tráfego para instâncias disponíveis em caso de falha na zona de disponibilidade.

Você também deve replicar dados para a workload e disponibilizá-los em várias zonas de disponibilidade. Alguns serviços de dados gerenciados pela AWS, como o Amazon S3, o Amazon Elastic File Service (EFS), o Amazon Aurora, o Amazon DynamoDB, o Amazon Simple Queue Service (SQS) e o Amazon Kinesis Data Streams, replicam dados em várias zonas de disponibilidade por padrão e são robustos contra o comprometimento da zona de disponibilidade. Com outros serviços de dados gerenciados pela AWS, como Amazon Relational Database Service (RDS), Amazon Redshift e Amazon ElastiCache, você deve habilitar a replicação multi-AZ. Depois de habilitados, esses serviços detectam automaticamente uma deficiência na zona de disponibilidade, redirecionam as solicitações para uma zona de disponibilidade disponível e replicam novamente os dados conforme necessário após a recuperação, sem a intervenção do cliente. Familiarize-se com o guia do usuário de cada serviço de dados gerenciado pela AWS que você usa para entender os recursos, os comportamentos e as operações multi-AZ deles.

Se você estiver usando armazenamento autogerenciado, como volumes do Amazon Elastic Block Store (EBS) ou armazenamento de instâncias do Amazon EC2, deverá gerenciar a replicação multi-AZ por conta própria.

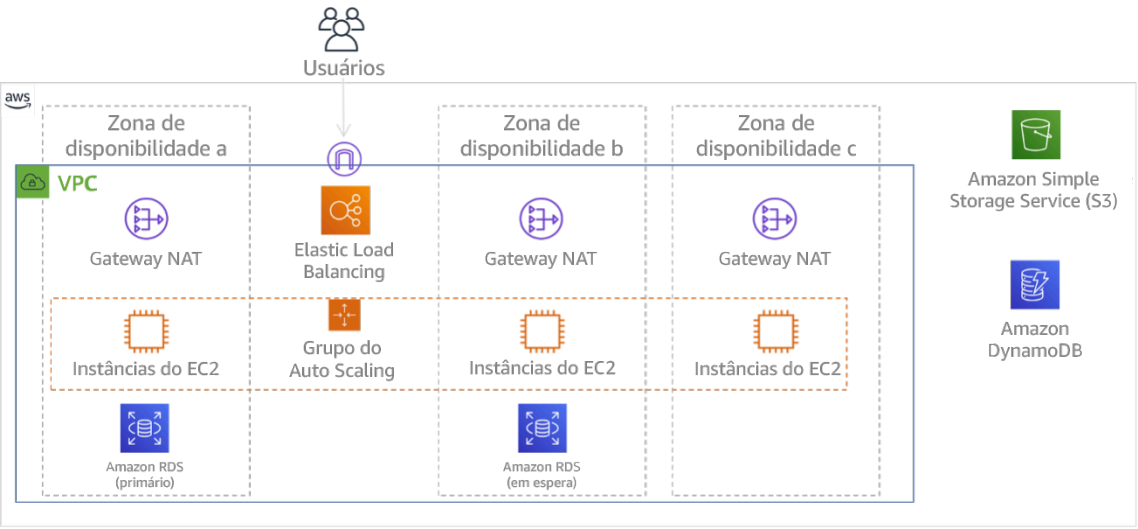


Figura 9: arquitetura multicamadas implantada em três Zonas de disponibilidade. Observe que o Amazon S3 e o Amazon DynamoDB são sempre Multi-AZ automaticamente. O ELB também é implantado em todas as três zonas.

Usar várias Regiões da AWS

Se você tem workloads que exigem extrema resiliência (como infraestrutura crítica, aplicações relacionadas à saúde ou serviços com rigorosos requisitos de clientes ou de disponibilidade obrigatória), pode precisar de disponibilidade adicional além do que uma única Região da AWS pode oferecer. Nesse caso, você deve implantar e operar a workload em pelo menos duas Regiões da AWS (supondo que os requisitos de residência de dados permitam isso).

As Regiões da AWS estão localizadas em diferentes regiões geográficas ao redor do mundo e em vários continentes. As Regiões da AWS têm separação física e isolamento ainda maiores do que as zonas de disponibilidade sozinhas. Os serviços da AWS, com poucas exceções, aproveitam esse design para operar de forma totalmente independente entre diferentes regiões (também conhecidos como serviços regionais). Uma falha em um serviço que opera por Região da AWS não deve afetar o serviço em uma região diferente.

Ao operar a workload em várias regiões, você deve considerar requisitos adicionais. Como os recursos em diferentes regiões são separados e independentes uns dos outros, você deve duplicar os componentes da workload em cada região. Isso inclui infraestrutura básica, como VPCs, além de serviços de computação e dados.

OBSERVAÇÃO: ao considerar um design multirregional, verifique se a workload é capaz de ser executada em uma única região. Se você criar dependências entre regiões, em que um componente

em uma região depende de serviços ou componentes de outra região, poderá aumentar o risco de falha e enfraquecer significativamente sua postura de confiabilidade.

Para facilitar as implantações multirregionais e manter a consistência, o AWS CloudFormation StackSets pode replicar toda a sua infraestrutura da AWS em várias regiões. O AWS CloudFormation também pode detectar desvios na configuração e informar você quando os recursos da AWS em uma região estiverem fora de sincronia. Muitos serviços da AWS oferecem replicação multirregional para ativos importantes de workload. Por exemplo, o EC2 Image Builder pode publicar as imagens de máquina (AMIs) do EC2 após cada compilação em cada região que você usa. O Amazon Elastic Container Registry (ECR) pode replicar as imagens de contêiner nas regiões selecionadas.

Você também deve replicar os dados em cada uma das regiões escolhidas. Muitos serviços de dados gerenciados pela AWS oferecem capacidade de replicação entre regiões, incluindo Amazon S3, Amazon DynamoDB, Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon ElastiCache e Amazon EFS. As tabelas globais do Amazon DynamoDB aceitam gravações em qualquer região compatível e replicarão dados entre todas as outras regiões configuradas. Com outros serviços, você deve designar uma região primária para gravações, pois outras regiões contêm réplicas somente leitura. Para cada serviço de dados gerenciado pela AWS que a workload usa, consulte o guia do usuário e o guia do desenvolvedor para entender as capacidades e limitações multirregionais de cada um. Preste atenção especial para onde as gravações devem ser direcionadas, aos recursos e limitações transacionais, à forma como a replicação é executada e à forma de monitorar a sincronização entre regiões.

A AWS também fornece a capacidade de rotear o tráfego de solicitações para as implantações regionais com grande flexibilidade. Por exemplo, você pode configurar os registros DNS usando o Amazon Route 53 a fim de direcionar o tráfego para a região disponível mais próxima do usuário. Como alternativa, você pode configurar os registros DNS em uma configuração ativa/em espera, em que designa uma região como primária e retorna a uma réplica regional somente se a região primária não estiver íntegra. Você pode configurar as verificações de integridade do Route 53 para detectar endpoints não íntegros e realizar failover automático, além de usar o Controlador de Recuperação de Aplicações (ARC) para fornecer um controle de roteamento altamente disponível a fim de redirecionar manualmente o tráfego conforme necessário.

Mesmo que você opte por não operar em várias regiões para obter alta disponibilidade, considere várias regiões como parte da sua estratégia de recuperação de desastres (DR). Se possível, replique os componentes e os dados da infraestrutura da workload em uma configuração de espera passiva ou de luz piloto em uma região secundária. Nesse design, você replica a infraestrutura

de linha de base da região primária, como VPCs, grupos do Auto Scaling, orquestradores de contêineres e outros componentes, mas configura os componentes de tamanho variável na região de espera (como o número de instâncias do EC2 e réplicas de banco de dados) para ter um tamanho minimamente operável. Você também organiza a replicação contínua de dados da região primária para a região de standby. Se ocorrer um incidente, você poderá aumentar a escala horizontalmente ou aumentar os recursos na região de standby e promovê-la para se tornar a região primária.

Etapas de implementação

1. Trabalhe com partes interessadas empresariais e especialistas em residência de dados para determinar quais Regiões da AWS podem ser usadas para hospedar os recursos e os dados.
2. Trabalhe com partes interessadas de áreas comerciais e técnicas para avaliar a workload e determinar se as necessidades de resiliência podem ser atendidas por uma abordagem multi-AZ (Região da AWS única) ou se elas exigem uma abordagem multirregional (se várias regiões forem permitidas). O uso de várias regiões pode alcançar maior disponibilidade, mas pode envolver complexidade e custo adicionais. Considere os seguintes fatores em sua avaliação:
    a. Objetivos de negócios e requisitos do cliente: quanto tempo de inatividade é permitido caso ocorra um incidente que afete a workload em uma zona de disponibilidade ou região? Avalie os objetivos de ponto de recuperação conforme discutido em REL13-BP01 Definir objetivos de recuperação para tempo de inatividade e perda de dados.
    b. Requisitos de recuperação de desastres (DR): contra qual tipo de desastre potencial você quer se proteger? Considere a possibilidade de perda de dados ou indisponibilidade de longo prazo em diferentes escopos de impacto, de uma única zona de disponibilidade a uma região inteira. Se você replicar dados e recursos em todas as zonas de disponibilidade e uma única zona de disponibilidade apresentar uma falha contínua, você poderá recuperar o serviço em outra zona de disponibilidade. Se você replicar dados e recursos entre regiões, poderá recuperar o serviço em outra região.
3. Implemente seus recursos computacionais em várias zonas de disponibilidade.
    a. Na VPC, crie várias sub-redes em diferentes zonas de disponibilidade. Configure cada uma para ser grande o suficiente para acomodar os recursos necessários e atender à workload, mesmo durante um incidente. Para conferir mais detalhes, consulte REL02-BP03 Garantir contas de alocação de sub-rede IP para expansão e disponibilidade.
    b. Se você estiver usando instâncias do Amazon EC2, use o EC2 Auto Scaling para gerenciar suas instâncias. Especifique as sub-redes que você escolheu na etapa anterior ao criar os grupos do Auto Scaling.

c. Se você estiver usando a computação do AWS Fargate para o Amazon ECS ou o Amazon EKS, selecione as sub-redes escolhidas na primeira etapa ao criar um serviço do ECS, inicializar uma tarefa do ECS ou criar um perfil do Fargate para o EKS.

d. Se você estiver usando funções do AWS Lambda que precisam ser executadas na VPC, selecione as sub-redes escolhidas na primeira etapa ao criar a função do Lambda. Para qualquer função que não tenha uma configuração de VPC, o AWS Lambda gerencia a disponibilidade para você automaticamente.

e. Coloque diretores de tráfego, como balanceadores de carga, na frente dos recursos de computação. Se o balanceamento de carga entre zonas estiver habilitado, os AWS Application Load Balancers e Network Load Balancers detectarão quando destinos, como instâncias e contêineres do EC2, estiverem inacessíveis devido ao comprometimento da zona de disponibilidade e redirecionarão o tráfego para destinos em zonas de disponibilidade íntegras. Se você desabilitar o balanceamento de carga entre zonas, use o Controlador de Recuperação de Aplicações (ARC) para fornecer a capacidade de mudança de zona. Se você estiver usando um balanceador de carga de terceiros ou tiver implementado seus próprios balanceadores de carga, configure-os com vários front-ends em diferentes zonas de disponibilidade.

4. Replique os dados da workload em várias zonas de disponibilidade.

   a. Se você usa um serviço de dados gerenciado pela AWS, como Amazon RDS, Amazon ElastiCache ou Amazon FSx, estude os guias do usuário para entender os recursos de replicação e resiliência de dados deles. Habilite o failover e a replicação entre AZs, se necessário.

   b. Se você usa serviços de armazenamento gerenciados pela AWS, como Amazon S3, Amazon EFS e Amazon FSx, evite usar configurações single-AZ ou One Zone para dados que exijam alta durabilidade. Use uma configuração multi-AZ para esses serviços. Consulte o guia do usuário do respectivo serviço para determinar se a replicação multi-AZ está habilitada por padrão ou se você deve habilitá-la.

   c. Se você executar um banco de dados, uma fila ou outro serviço de armazenamento autogerenciado, providencie a replicação multi-AZ de acordo com as instruções ou práticas recomendadas da aplicação. Familiarize-se com os procedimentos de failover da aplicação.

5. Configure o serviço DNS para detectar deficiências na AZ e redirecionar o tráfego para uma zona de disponibilidade íntegra. O Amazon Route 53, quando usado em combinação com Elastic Load Balancers, pode fazer isso automaticamente. O Route 53 também pode ser configurado com registros de failover que usam verificações de integridade para responder a consultas somente com endereços IP íntegros. Para registros DNS usados para failover, especifique um valor curto de vida útil (TTL) (por exemplo, 60 segundos ou menos) para ajudar a evitar que o

armazenamento em cache de registros impeça a recuperação (os registros de alias do Route 53 fornecem TTLs apropriados para você).

Etapas adicionais ao usar várias Regiões da AWS

1. Replique todo o sistema operacional (SO) e código de aplicação usados pela workload nas regiões selecionadas. Replique as imagens de máquina da Amazon (AMIs) usadas pelas instâncias do EC2, se necessário, usando soluções como o Amazon EC2 Image Builder. Replique imagens de contêineres armazenadas em registros usando soluções como a replicação entre regiões do Amazon ECR. Habilite a replicação regional para qualquer bucket do Amazon S3 usado para armazenar recursos de aplicações.
2. Implante os recursos de computação e os metadados de configuração (como parâmetros armazenados no AWS Systems Manager Parameter Store) em várias regiões. Use os mesmos procedimentos descritos nas etapas anteriores, mas replique a configuração para cada região em que você está usando a workload. Use soluções de infraestrutura como código, como o AWS CloudFormation, para reproduzir uniformemente as configurações entre as regiões. Se você estiver usando uma região secundária em uma configuração de luz piloto para recuperação de desastres, poderá reduzir o número de recursos de computação a um valor mínimo para reduzir os custos, com um aumento correspondente no tempo de recuperação.
3. Replique os dados da região primária para as regiões secundárias.
    a. As tabelas globais do Amazon DynamoDB fornecem réplicas globais dos dados que podem receber gravações de qualquer região compatível. Com outros serviços de dados gerenciados pela AWS, como Amazon RDS, Amazon Aurora e Amazon ElastiCache, designe uma região primária (leitura/gravação) e regiões de réplica (somente leitura). Consulte os guias do usuário e do desenvolvedor dos respectivos serviços para obter detalhes sobre a replicação regional.
    b. Se você estiver executando um banco de dados autogerenciado, providencie a replicação multirregional de acordo com as instruções ou as práticas recomendadas da aplicação. Familiarize-se com os procedimentos de failover da aplicação.
    c. Se a workload usa o AWS EventBridge, talvez seja necessário encaminhar eventos selecionados da região primária para as regiões secundárias. Para fazer isso, especifique os barramentos de eventos nas regiões secundárias como metas para eventos correspondentes na região primária.
4. Considere se e em que medida você deseja usar chaves de criptografia idênticas em todas as regiões. Uma abordagem típica que equilibra segurança e facilidade de uso é usar chaves com escopo regional para dados e autenticação locais em uma região e usar chaves com escopo

global para criptografia de dados que são replicadas entre diferentes regiões. O AWS Key Management Service (KMS) oferece suporte a chaves multirregionais para distribuição segura e proteção das chaves compartilhadas entre regiões.

5. Considere o AWS Global Accelerator para melhorar a disponibilidade da aplicação direcionando o tráfego para regiões que contêm endpoints íntegros.

Recursos

Práticas recomendadas relacionadas:

- REL02-BP03 Garantir contas de alocação de sub-rede IP para expansão e disponibilidade
- REL11-BP05 Usar estabilidade estática para evitar comportamento bimodal
- REL13-BP01 Definir objetivos de recuperação para tempo de inatividade e perda de dados

Documentos relacionados:

- Infraestrutura global da AWS
- White paper: AWS Fault Isolation Boundaries
- Resiliência no Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling: exemplo: distribuir instâncias entre zonas de disponibilidade
- Como o EC2 Image Builder funciona
- Como o Amazon ECS posiciona tarefas em instâncias de contêineres (inclui o Fargate)
- Resiliência no AWS Lambda
- Amazon S3: Visão geral sobre a replicação de objetos
- Replicação de imagem privada no Amazon ECR
- Tabelas globais: replicação em várias regiões com o DynamoDB
- Amazon ElastiCache for Redis OSS: Replication across Regiões da AWS using global datastores
- Resiliência no Amazon RDS
- Usar bancos de dados globais do Amazon Aurora
- Guia do desenvolvedor do AWS Global Accelerator
- Chaves de multirregiões no AWS KMS
- Amazon Route 53: configurar o failover de DNS

- Guia do desenvolvedor do Controlador de Recuperação de Aplicações (ARC) da Amazon
- Como enviar e receber eventos do Amazon EventBridge entre regiões da Regiões da AWS
- Série de blogs Criar aplicações multirregiões com serviços da AWS
- Arquitetura de recuperação de desastres (DR) na AWS, Parte I: estratégias de recuperação na nuvem
- Arquitetura de recuperação de desastres (DR) na AWS, parte III: luz piloto e standby passivo

Vídeos relacionados:

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

REL10-BP02 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 data center on-premises, será necessário 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 for estabelecida: Médio

Orientação para implementação

Se a prática recomendada para implantar a workload 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 aplicações 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 porque 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. É possível provisionar vários nós. Usando o EMR File System (EMRFS),Regiões da AWS 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 .

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 data center 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 sua workload em uma zona ou região de disponibilidade alternativa. O Elastic Disaster Recovery usa a replicação contínua em nível de bloco para uma área temporária leve para fornecer recuperação rápida e confiável de aplicações on-premises e baseadas na nuvem.

Etapas de implementação

1. Implemente a autorrecuperação. Quando possível, use o ajuste de escala automático para implantar instâncias ou contêineres. Quando não for possível, use a recuperação automática de instâncias do EC2 ou implemente a automação de autorrecuperação 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 instâncias e workloads de contêiner que não têm requisitos de endereço IP de instância única, endereço IP privado, endereço IP elástico e metadados de 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 exigem um endereço do ID de instância única, endereço IP privado, endereço IP elástico e metadados de instância.
     - A recuperação automática enviará alertas de status de recuperação para um tópico do SNS quando a falha na instância for detectada.
   - Use eventos de ciclo de vida da instância do Amazon EC2 ou eventos do Amazon ECS para automatizar a autorrecuperação quando ajuste de escala automático ou a recuperação do EC2 não puderem ser usadas.
     - Use os eventos para invocar a automação que recuperará seu componente de acordo com a lógica do processo necessária.
   - Proteja workloads monitoradas que estão 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
- Recuperar a instância
- Ajuste de escala automático do serviço
- O que é o Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP03 Usar arquiteturas de anteparo para limitar o escopo de impactos

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 conhecida como 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 uma falha ocorrer, 90% das solicitações gerais não serão afetadas pela falha.

Práticas comuns que devem ser evitadas:

- 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 de implementar esta prática recomendada: com arquiteturas baseadas em células, muitos tipos comuns de falha são contidos na própria célula, o que permite o isolamento adicional das

falhas. Esses limites de falha podem fornecer resiliência contra tipos de falha que, de outra forma, seriam difíceis de conter, como implantações de código malsucedidas ou solicitações corrompidas ou que invocam um modo de falha específico (também conhecido como solicitações de pílulas venenosas).

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

Orientação para implementação

Em uma embarcação, os anteparos 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. Os componentes fora do limite não são afetados pela falha. Ao usar vários limites isolados de falhas, é possível limitar o impacto na workload. 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. 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. 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.

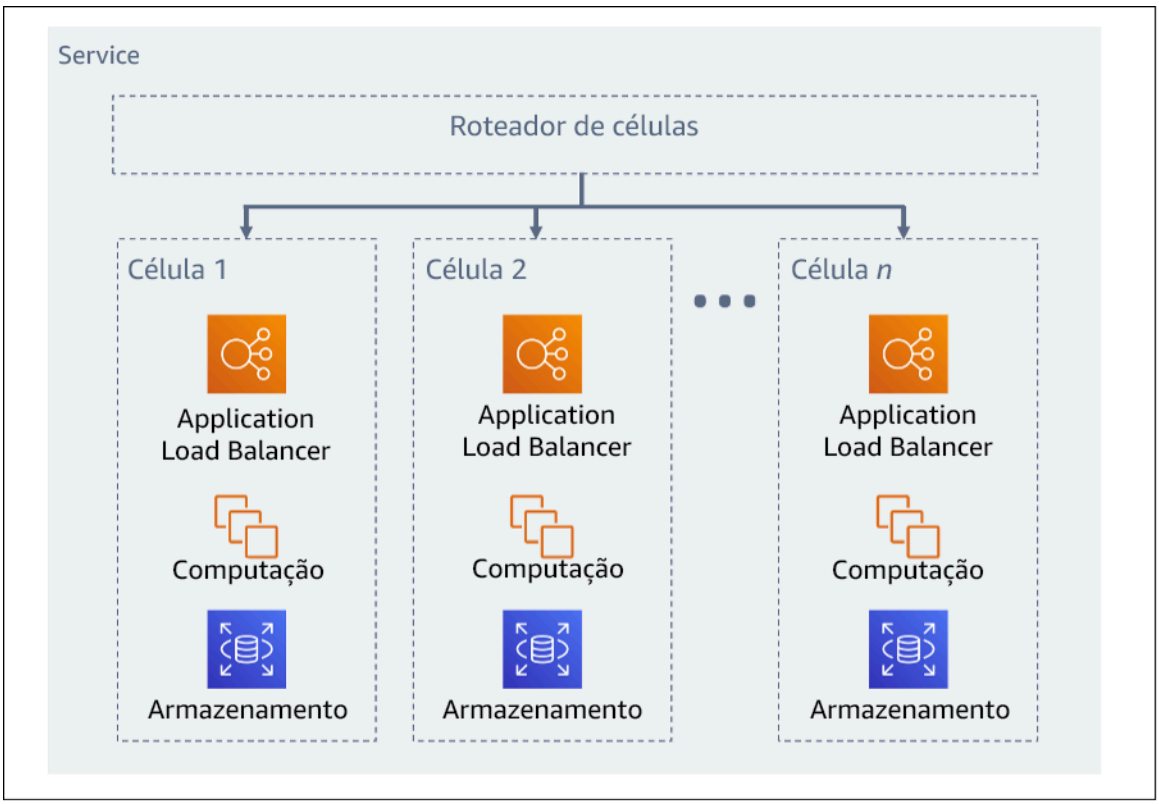


Figura 11: arquitetura baseada em células

Etapas de implementação

Ao projetar uma arquitetura baseada em células, há várias considerações de design a levar em conta:

1. Chave de partição: consideração especial deve ser feita em relação à chave de partição.
    - Ela precisa se alinhar à granularidade do serviço, ou à maneira natural que a workload de um serviço pode ser subdividida em interações mínimas entre células. Os exemplos são `customer ID` ou `resource ID`.
    - 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 celular persistente: os serviços upstream devem interagir somente com uma única célula durante o ciclo de vida de seus 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 se torna 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 as células e, portanto, não pode seguir a mesma estratégia de compartimentação 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 traz o benefício adicional de facilitar a compreensão de seu comportamento esperado em todos os momentos, permitindo a realização de testes rigorosos. Conforme explicado por Colm MacCárthaigh em 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 se estender além dele.
   - O tamanho máximo deve ser identificado com a realização de testes completos até que os pontos de ruptura sejam atingidos e margens operacionais seguras sejam estabelecidas. Para obter mais detalhes sobre como implementar práticas de testes, consulte REL07-BP04 Fazer o teste de carga da workload
   - A workload geral deve crescer com a adição de mais células, permitindo que a workload seja escalada com aumentos na demanda.
5. Estratégias multi-AZ ou multirregiões: utilize várias camadas de resiliência para oferecer 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 multirregiões 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 multirregiões pode ser complexa e, geralmente, não é necessária para a maioria das workloads. Para obter mais detalhes, consulte REL10-BP01 Implantar a workload em vários locais.

6. Implantação de código: uma estratégia de implantação de código em etapas deve ser preferida à implantação de alterações de código em todas as células ao mesmo tempo.
   - Isso ajuda a reduzir a possibilidade de falhas em várias células devido a uma implantação incorreta ou a erro humano. Para obter mais detalhes, consulte Automatizar implantações seguras e sem intervenção manual.

Recursos

Práticas recomendadas relacionadas:

- REL07-BP04 Fazer o teste de carga da workload
- REL10-BP01 Implantar a workload em vários locais

Documentos relacionados:

- Confiabilidade, trabalho constante e uma boa xícara de café
- AWS e a compartimentalização
- Isolamento de workloads via fragmentação aleatória
- Automatizar implantações seguras e sem intervenção manual

Vídeos relacionados:

- AWS re:Invent 2018: Fechar loops e abrir mentes: como assumir o controle de sistemas grandes e pequenos
- AWS re:Invent 2018: Como a AWS minimiza o raio de ação das falhas (ARC338)
- Fragmentação aleatória: AWS re:Invent 2019: Introdução à Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021: Tudo falha, o tempo todo: projetando para resiliência

Projete a workload para resistir a falhas de componentes

As workloads 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 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 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 recuperação e correção 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.

Práticas comuns que devem ser evitadas:

- 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.
- Monitores em excesso são criados.

Benefícios de implementar 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, será necessário 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 Padrões de resiliência Multi-AZ avançados.

Etapas de 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 das instâncias do EC2 e do Auto Scaling é necessário. 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 avançado 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, API Gateway, Amazon EKS, Amazon ECS e todos os tipos de balanceadores de carga.
  - Determine os requisitos de monitoramento dos componentes de armazenamento para Amazon S3, Amazon FSx, Amazon EFS e Amazon EBS.
- Crie métricas personalizadas para medir os indicadores-chave de performance (KPIs) de negócios. As workloads implementam as principais funções empresariais, as quais devem ser usadas como KPIs para ajudar a identificar quando um problema indireto ocorre.
- Utilize os canários de usuário para monitorar a experiência do usuário e verificar se há falhas. O teste de transações sintéticas (também conhecido como teste canário, que não deve ser confundidos com as implantações canário), capaz de 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 acompanhem 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 o ajuste de escala automático dos recursos deve ser feito. Os alarmes podem ser exibidos visualmente em painéis, enviar alertas via Amazon SNS ou e-mail e trabalhar com o Auto Scaling para aumentar ou reduzir a escala dos recursos da workload.
- Crie painéis para visualizar as 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.
- Mantenha-se a par das degradações de serviço por meio do AWS Health. Crie notificações de eventos do AWS Health ajustadas à finalidade para canais de e-mail e chat por meio do Notificações de Usuários da AWS e integre-as programaticamente às suas ferramentas de monitoramento e alerta por meio do Amazon EventBridge.

Recursos

Práticas recomendadas relacionadas:

- Definição de disponibilidade
- REL11-BP06 Enviar notificações quando os eventos afetarem a disponibilidade

Documentos relacionados:

- O Amazon CloudWatch Synthetics permite criar canários de usuário
- Habilitar ou desabilitar o monitoramento detalhado da instância
- Monitoramento avançado
- Monitorar grupos do Auto Scaling e instâncias usando o Amazon CloudWatch
- Publicar métricas personalizadas
- Usar alarmes do Amazon CloudWatch
- Usar painéis do CloudWatch
- Usar painéis do CloudWatch entre regiões e contas
- Usar o rastreamento do X-Ray entre regiões e contas
- Noções básicas da disponibilidade

Vídeos relacionados:

- Como mitigar falhas cinzentas

Exemplos relacionados:

- Workshop One Observability: explorar o X-Ray

Ferramentas relacionadas:

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 Failover para recursos íntegros

Se uma falha ocorrer 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 que não apresentam 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 por meio da transferência do 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 extensivamente 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, rastrear 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.

Práticas comuns que devem ser evitadas:

- 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 multirregiões 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 autocorreção sem notificar que a reparação era necessária.
- Falta de um período de amortecimento a fim de evitar o failback cedo demais.

Benefícios de implementar esta prática recomendada: é possível 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 por meio do desvio do tráfego para os recursos íntegros restantes.

Para workloads multirregiões, os designs são mais complicados. Por exemplo, as 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 Amazon Application Recovery Controller (ARC), o Amazon CloudFront e o AWS Global Accelerator podem ajudar a direcionar o tráfego nas Regiões da AWS.

Os 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.

O Multi-AZ é uma opção de configuração para o Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS ou Amazon ECS. A AWS pode 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, escolha em quais zonas de disponibilidade deseja fazer a implantação. Em alguns designs, o Elastic Load Balancing fornece a solução para detectar as instâncias nas zonas com problemas de integridade e rotear o tráfego para as instâncias íntegras. O Elastic Load Balancing também pode rotear tráfego para componentes no seu data center on-premises.

No caso de failover de tráfego em várias regiões, o redirecionamento pode utilizar o Amazon Route 53, o Amazon Application Recovery Controller, o AWS Global Accelerator, o Route 53 Private DNS para VPCs ou o CloudFront para oferecer uma maneira de definir domínios da internet e atribuir políticas de roteamento, incluindo verificações de integridade, e rotear o tráfego para regiões íntegras. O AWS Global Accelerator fornece endereços IP estáticos que atuam como um ponto de entrada fixo para a 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 a performance e a confiabilidade.

Etapas de 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 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 playbook de failover e inclua todas as etapas para realizar o failover de cada recurso, aplicação e serviço.
- Crie um playbook 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 playbook. Use simulações e testes de caos para testar a automação e as etapas do playbook.
- 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 que não

apresentam problemas. Confira a cota, os níveis de ajuste de escala automático e os recursos em execução antes do teste de failover.

Recursos

Práticas recomendadas do Well-Architected relacionadas:

- REL13: planejar para DR
- REL10: usar o isolamento de falhas para proteger a workload

Documentos relacionados:

- Definir metas de RTO e RPO
- Failover usando o roteamento ponderado do Route 53
- Disaster Recovery with Amazon Route 53 Application Recovery Controller (ARC)
- EC2 com ajuste de escala automático
- Implantações do EC2: Multi-AZ
- Implantações do ECS: Multi-AZ
- Switch traffic using Amazon Application Recovery Controller
- Lambda com um Application Load Balancer e failover
- Replicação e failover do ACM
- Replicação e failover do repositório de parâmetros
- Replicação entre regiões e failover do ECR
- Configuração da replicação entre regiões do Secrets Manager
- Habilitar a replicação entre regiões para EFS e failover
- Replicação entre regiões e failover do EFS
- Failover de rede
- Failover de endpoint do S3 usando MRAP
- Criar replicação entre regiões para o S3
- Guidance for Cross Region Failover and Graceful Failback on AWS
- Failover com o acelerador global multirregiões
- Failover com DRS

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 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 workloads. 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 exponencial e jitter. 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.

Práticas comuns que devem ser evitadas:

- 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 da recuperação automática.
- Sem automação para failover dos bancos de dados.
- Não há métodos automatizados para redirecionar o tráfego para novos endpoints.
- Sem replicação de armazenamento.

Benefícios de implementar 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 do Auto Scaling. O Elastic Load Balancing (ELB) distribui automaticamente o tráfego de entrada das aplicações entre vários destinos e appliances 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. Exemplos de serviços são 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 automatizado 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 invocar o AWS Lambda, o Systems Manager Automation (ou outros destinos) para

executar a lógica de correção personalizada na workload. O Amazon EC2 Auto Scaling pode ser configurado para verificar a integridade da instância do 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 a instância como não íntegra e iniciará 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 de implementação

- Use grupos do Auto Scaling para implantar camadas em uma workload. O Auto Scaling pode executar a autocorreção em aplicações sem estado e adicionar e remover capacidade.
- Para instâncias computacionais mencionadas anteriormente, use o balanceamento de carga e escolha o tipo apropriado de balanceador de carga.
- Considere a possibilidade de reparação do Amazon RDS. Com instâncias em espera, configure o failover automático para a instância em espera. Para a réplica de leitura do Amazon RDS, é necessário um fluxo de trabalho automatizado para transformar uma réplica de leitura em primária.
- Implemente a recuperação automática em instâncias do EC2 que tenham aplicações implantadas que não possam ser implantadas em vários locais e possam 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 e os endereços IP associados da instância são mantidos, assim como os volumes do EBS e os pontos de montagem para Amazon Elastic File Systems ou File Systems para Lustre e Windows. Com o AWS OpsWorks, é possível configurar a autocorreção das instâncias do EC2 no nível da camada.
- Implemente a recuperação automatizada por meio do AWS Step Functions e do 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.
- 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 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 o AWS Auto Scaling funciona
- Recuperação automática do Amazon EC2
- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- O que é o Amazon FSx para Lustre?
- O que é o Amazon FSx para Windows File Server?
- AWS OpsWorks: Como usar a reparação automática para substituir instâncias com falha
- O que é AWS Step Functions?
- O que é AWS Lambda?
- O que é o Amazon EventBridge?
- Usar alarmes do Amazon CloudWatch
- Failover do Amazon RDS
- SSM: Systems Manager Automation
- Práticas recomendadas de arquitetura resiliente

Vídeos relacionados:

- Provisionar e escalar automaticamente o serviço OpenSearch
- Failover automático do Amazon RDS

Exemplos relacionados:

- Workshop 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.

Práticas comuns que devem ser evitadas:

- Dependência da alteração dos registros DNS para redirecionar o tráfego.
- Dependência das operações de escalação 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 implementar 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 essa 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.

Utilize o Amazon 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 às consultas ao DNS, além de realizarem e avaliarem verificações de integridade. Eles são distribuídos globalmente e projetados para um acordo de serviço (SLA) com 100% de disponibilidade.

As APIs e os 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.

Projete sua infraestrutura de computação de modo que ela seja estaticamente estável para evitar o uso do ambiente de gerenciamento durante um incidente. Por exemplo, se você estiver usando instâncias do Amazon EC2, evite provisionar novas instâncias manualmente ou instruir grupos do Auto Scaling a adicionar instâncias em resposta. 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, a utilização desses serviços faz uso do plano de dados. No entanto, criar novas funções, balanceadores de carga, API gateways ou tabelas do 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 às metas de alta disponibilidade, consulte o whitepaper Estabilidade estática usando zonas de disponibilidade.

Entenda quais operações estão no plano de dados e quais estão no ambiente de gerenciamento.

Etapas de 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) para recursos do Amazon EC2 pré-escalados (plano de dados)
- Ajuste de escala de instâncias do Amazon EC2 (ambiente de gerenciamento) para ajuste de escala do AWS Lambda (plano de dados)
- Avalie qualquer design 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. Usar nós superprovisionados 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 do Route 53 (ambiente de gerenciamento) ou Amazon Application Recovery Controller (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 roteamento de tráfego para 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 tolerância a falhas
- AWS Marketplace: produtos que podem ser usados para tolerância a falhas
- Amazon Builders' Library: evitar 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 de AWS Elemental MediaStore
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack
- Creating Disaster Recovery Mechanisms Using Amazon Route 53
- What is Amazon Application Recovery Controller
- Ambiente de gerenciamento e plano de dados do Kubernetes

Vídeos relacionados:

- De volta ao básico: uso da estabilidade estática
- Como criar workloads resilientes em vários sites usando serviços globais da AWS

Exemplos relacionados:

- Introducing Amazon Application Recovery Controller
- Amazon Builders' Library: evitar a sobrecarga em sistemas distribuídos colocando o menor serviço no controle
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack
- Estabilidade estática com zonas de disponibilidade

Ferramentas relacionadas:

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 Usar estabilidade estática para evitar 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.

Práticas comuns que devem ser evitadas:

- 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.

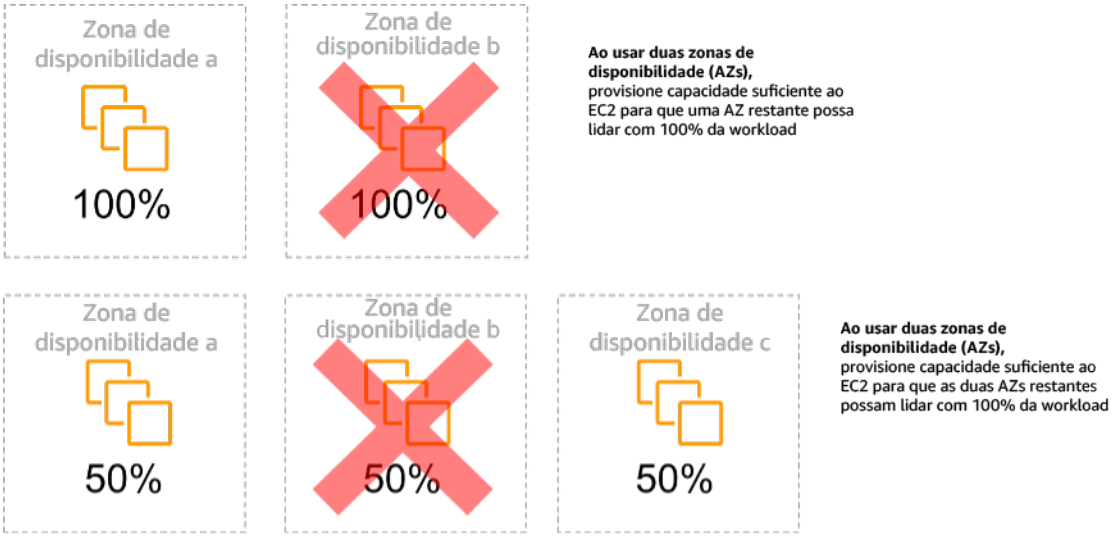
- Pensar em projetos estáticos estáveis somente para recursos computacionais.

Benefícios de implementar 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 designs estáveis do Amazon EC2 provisionam instâncias suficientes em cada zona de disponibilidade para lidar com a workload se uma AZ for removidas. O Elastic Load Balancing ou o Amazon Route 53 verificariam a integridade para afastar a carga das instâncias danificadas. 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, designs de réplicas de leitura de banco de dados, designs de cluster Kubernetes (EKS) e arquiteturas de failover multirregiões.

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: sob determinados designs

Etapas de 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. Diversos serviços podem ser usados para roteamento a recursos íntegros, como:
  - Roteamento de DNS entre regiões

  - Roteamento multirregiões MRAP do Amazon S3
  - AWS Global Accelerator
  - Amazon Application Recovery Controller
- Configure réplicas de leitura de 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 críticos no armazenamento do S3, projetado para ser estaticamente estável para dados armazenados em caso de falha na zona de disponibilidade. Se a classe de armazenamento One Zone-IA do Amazon S3 for usada, ela não deverá ser considerada estaticamente estável, pois a perda dessa zona minimiza o acesso a esses dados armazenados.
- Os balanceadores de carga algumas vezes são configurados incorreta 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 do Well-Architected relacionadas:

- 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
- Amazon Builders' Library: estabilidade estática usando zonas de disponibilidade
- Limites de isolamento de falhas
- Estabilidade estática usando zonas de disponibilidade
- RDS Multi-AZ
- Minimizar dependências em um plano de recuperação de desastres

- Roteamento de DNS entre regiões
- Roteamento multirregiões MRAP do Amazon S3
- AWS Global Accelerator
- Amazon Application Recovery Controller
- Amazon S3 de zona única
- Balanceamento de carga entre zonas

Vídeos relacionados:

- Estabilidade estática na AWS: AWS re:Invent 2019: introdução à Amazon Builders' Library (DOP328)

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 ocultar problemas subjacentes que precisam ser resolvidos. Implemente eventos e monitoramento apropriados para que você possa detectar padrões de problemas, incluindo aqueles abordados pela autocorreção, e consiga 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.

Práticas comuns que devem ser evitadas:

- Enviar muitos alarmes.
- Enviar alarmes não acionáveis.

- 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 implementar esta prática recomendada: as notificações de recuperação alertam as equipes operacionais e empresariais sobre as degradações do serviço para que elas 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.

Orientação para 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 `IN ALARM` e `OK`) e detalhes sobre o que o acionou. Em muitos casos, um evento de alarme deve ser detectado e uma notificação por e-mail deve ser enviada. 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 limites mais complexo, considere usar 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 de 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.
- Os alarmes de infraestrutura indicam quando escalar os recursos. Os alarmes podem ser exibidos visualmente em painéis, enviar alertas via Amazon SNS ou e-mail e trabalhar com o Auto Scaling para aumentar ou reduzir a escala dos recursos da workload.
- Alarmes estáticos de exemplo 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 remediação ou comunicação.
- Mantenha-se a par das degradações de serviço por meio do AWS Health. Crie notificações de eventos do AWS Health ajustadas à finalidade para canais de e-mail e chat por meio do Notificações de Usuários da AWS e integre-as programaticamente às suas ferramentas de monitoramento e alerta por meio do Amazon EventBridge.

Recursos

Práticas recomendadas do Well-Architected relacionadas:

- 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
- Usar alarmes do Amazon CloudWatch
- Configurar alarmes do CloudWatch Composite
- Novidades no AWS Observability na re:Invent 2022

Ferramentas relacionadas:

- CloudWatch
- CloudWatch X-Ray

REL11-BP07 Arquitetar o produto para cumprir as metas de disponibilidade e os acordos de serviço (SLAs) de tempo de atividade

Arquitete o produto para cumprir as metas de disponibilidade e os acordos 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 de disponibilidade e um SLA para métricas de performance, as quais podem ser monitoradas e mantidas para atingir os resultados comerciais.

Práticas comuns que devem ser evitadas:

- Planejar e implantar workloads sem definir SLAs.
- As métricas de SLA são definidas muito altas sem justificativas ou requisitos comerciais.
- Definir SLAs sem considerar as dependências e o SLA subjacente.
- Os designs das aplicações são criados sem considerar o modelo de responsabilidade compartilhada para resiliência.

Benefícios de implementar esta prática recomendada: desenvolver aplicações com base nas principais metas de resiliência ajuda a atingir os objetivos de negócios e as 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 risco exposto se esta prática recomendada não for estabelecida: Médio

Orientação para implementação

Os designs da aplicação precisam levar em conta um conjunto de requisitos diversos que são derivados de 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).
- Falhas menos frequentes (MTBF mais longo), tempos de detecção de falhas mais curtos (MTTD mais curto) e tempos de reparo mais curtos (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, a performance, o ajuste de escala e os custos.

Etapas de implementação

- Analise e documente o design da workload considerando as seguintes questões:
  - Onde os ambientes de gerenciamento são usados na workload?
  - Como a workload implementa tolerância a falhas?
  - Quais são os padrões de design para componentes de ajuste de escala, ajuste de escala automático, 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 o design que cumprirá o SLA for definido, 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-BP04 Testar a resiliência com engenharia do caos
- REL13-BP01 Definir os objetivos de recuperação para tempo de inatividade e perda de dados
- Como a integridade da workload funciona

Documentos relacionados:

- Disponibilidade com redundância
- Pilar Confiabilidade: disponibilidade
- Como medir a disponibilidade
- Limites de isolamento de falhas da AWS
- Modelo de responsabilidade compartilhada para resiliência
- Estabilidade estática com zonas de disponibilidade
- Acordos de serviço (SLAs) da AWS
- Orientação para arquitetura baseada em células na AWS
- Infraestrutura da AWS

- Whitepaper Padrões avançados de resiliência multi-AZ

Serviços relacionados:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

Testar a confiabilidade

Depois de projetar sua workload 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 workload atende aos requisitos funcionais e não funcionais, pois erros ou gargalos de performance podem afetar a confiabilidade de sua workload. Teste a resiliência de sua workload para obter ajuda para encontrar erros latentes que apenas aparecem na produção. Simule esses testes regularmente.

Práticas recomendadas

- REL12-BP01 Usar playbooks para investigar falhas
- REL12-BP02 Realizar análises pós-incidentes
- REL12-BP03 Testar os requisitos de escalabilidade e desempenho
- REL12-BP04 Testar a resiliência com engenharia do caos
- REL12-BP05 Conduzir dias de jogo regularmente

REL12-BP01 Usar playbooks para investigar falhas

Documente o processo de investigação em playbooks para permitir respostas consistentes e rápidas em cenários de falha. Os playbooks consistem em 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 escalado.

O playbook é um planejamento proativo que deve ser feito para poder executar ações reativas com eficácia. Quando cenários de falha não cobertos pelo playbook forem encontrados na produção,

aborde o problema primeiro ("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 os playbooks são usados em resposta a incidentes específicos, enquanto runbooks são usados para alcançar resultados específicos. Muitas vezes, os runbooks são usados para atividades de rotina e os playbooks são usados para responder a eventos que não são rotineiros.

Práticas comuns que devem ser evitadas:

- Planejar a implantação de uma workload 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 por tempo suficiente para recuperar os dados.

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

Orientação para implementação

- Use playbooks para identificar problemas. Os playbooks são processos documentados para investigar problemas. Documente os 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 (ou seja, faça 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 seus playbooks operacionais com o AWS Systems Manager
    - AWS Systems Manager Run Command

- AWS Systems Manager Automation
- O que é 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 seus playbooks operacionais com o AWS Systems Manager
- Usar alarmes do Amazon CloudWatch
- Usar canários (Amazon CloudWatch Synthetics)
- O que é o Amazon EventBridge?
- O que é AWS Lambda?

Exemplos relacionados:

- Automatizar operações com playbooks e runbooks

REL12-BP02 Realizar análises pós-incidentes

Analise os eventos que afetam o cliente e identifique os fatores contribuintes e os itens de ação preventiva. Use essas informações para desenvolver mitigações e 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 consensual 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 que cria mecanismos e barreiras de proteção para limitar a probabilidade do mesmo incidente ocorrer novamente.

Práticas comuns que devem ser evitadas:

- 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 de implementar esta 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 permitir que elas implementem a mitigação ou a recuperação automatizada antes que ocorra um incidente.

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

Orientação para 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-raiz essenciais e garantir que elas sejam analisadas e abordadas. Atribua uma propriedade clara ao processo de análise pós-incidente. Atribua uma equipe ou uma pessoa responsável para supervisionar as investigações e o rastreamento 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-raiz 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 de 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 comerciais podem determinar 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.
  - Que lições você aprendeu?
  - Que ações corretivas você está adotando?
    - 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. Por exemplo, 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:

- Por que você deve desenvolver uma correção de erro (COE)

Vídeos relacionados:

- A abordagem da Amazon para falhar com sucesso
- AWS re:Invent 2021: Amazon Builders' Library: excelência operacional da Amazon

REL12-BP03 Testar os requisitos de escalabilidade e desempenho

Use técnicas como teste de carga para validar se a workload atende aos requisitos de escalabilidade e desempenho.

É possível criar na nuvem um ambiente de teste em escala de produção sob demanda para a workload. Em vez de depender de um ambiente de teste com escala vertical reduzida, o que pode levar a previsões imprecisas dos comportamentos de produção, você pode usar a nuvem para provisionar um ambiente de teste que espelhe o ambiente de produção esperado. Esse ambiente ajuda você a testar em uma simulação mais precisa das condições reais que a aplicação enfrenta.

Além das ações de teste de desempenho, é essencial validar que os recursos básicos, as configurações escalabilidade, as cotas de serviço e o design de resiliência operem conforme o esperado sob carga. Essa abordagem holística verifica se a aplicação pode ser escalada e executada de forma confiável conforme necessário, mesmo sob as condições mais exigentes.

Resultado desejado: a workload mantém o comportamento esperado mesmo quando está sujeita a picos de carga. Você aborda proativamente quaisquer problemas relacionados ao desempenho que possam surgir à medida que a aplicação cresce e evolui.

Práticas comuns que devem ser evitadas:

- Usar ambientes de teste que não são muito parecidos com o ambiente de produção.
- Tratar o teste de carga como uma atividade separada e única, em vez de uma parte integrada do pipeline de integração contínua (CI) da implantação.
- Não definir requisitos de desempenho claros e mensuráveis, como metas de tempo de resposta, throughput e escalabilidade.
- Executar testes com cenários de carga irrealistas ou insuficientes e não conseguir testar cargas de pico, picos repentinos e alta carga sustentada.
- Não testar o estresse da workload excedendo os limites de carga esperados.
- Usar ferramentas de teste de carga e perfil de desempenho inadequadas ou indevidas.
- Não ter sistemas abrangentes de monitoramento e alerta para rastrear métricas de desempenho e detectar anomalias.

Benefícios de implementar essa prática recomendada:

- O teste de carga ajuda você a identificar possíveis gargalos de desempenho no sistema antes da entrada em produção. Ao simular tráfego e workloads em nível de produção, você pode identificar áreas em que o sistema pode ter dificuldades para lidar com a carga, como tempos de resposta lentos, restrições de recursos ou falhas do sistema.
- Ao testar o sistema sob várias condições de carga, você pode entender melhor os requisitos de recursos necessários para oferecer suporte à workload. Essas informações podem ajudar você a tomar decisões informadas sobre a alocação de recursos e evitar o provisionamento excessivo ou insuficiente de recursos.
- Para identificar possíveis pontos de falha, você pode observar o desempenho da workload em condições de alta carga. Essas informações ajudam você a melhorar a confiabilidade e a resiliência da workload, implementando mecanismos de tolerância a falhas, estratégias de failover e medidas de redundância, conforme apropriado.
- Você identifica e aborda problemas de desempenho com antecedência, o que ajuda a evitar as consequências dispendiosas de interrupções do sistema, tempos de resposta lentos e usuários insatisfeitos.

- Dados detalhados de desempenho e informações de perfil coletados durante o teste podem ajudar você a solucionar problemas relacionados ao desempenho que possam surgir na produção. Isso pode proporcionar mais agilidade na resposta e resolução de incidentes, o que reduz o impacto nos usuários e nas operações da organização.
- Em certos setores, os testes proativos de desempenho podem ajudar a workload a atender aos padrões de conformidade, o que reduz o risco de penalidades ou problemas legais.

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

Orientação para implementação

A primeira etapa é definir uma estratégia de teste abrangente que cubra todos os aspectos dos requisitos de escalabilidade e desempenho. Para começar, defina claramente os objetivos de nível de serviço (SLOs) da workload com base nas necessidades de empresa, como throughput, histograma de latência e taxa de erros. Depois, crie um conjunto de testes que possa simular vários cenários de carga, que variam de uso médio a picos repentinos e picos contínuos de carga, e verifique se o comportamento da workload atende aos SLOs. Esses testes devem ser automatizados e integrados ao pipeline contínuo de integração e implantação para detectar regressões de desempenho no início do processo de desenvolvimento.

Para testar com eficácia a escalabilidade e o desempenho, invista nas ferramentas e na infraestrutura certas. Isso inclui ferramentas de teste de carga que podem gerar tráfego realista de usuários, ferramentas de perfil de desempenho para identificar gargalos e soluções de monitoramento para rastrear as principais métricas. É importante ressaltar que você deve verificar se os ambientes de teste correspondem perfeitamente ao ambiente de produção em termos de infraestrutura e condições do ambiente para tornar os resultados dos testes o mais precisos possível. Para facilitar a replicação e a escalabilidade confiáveis de configurações semelhantes às de produção, use a infraestrutura como código e aplicações baseadas em contêineres.

Os testes de escalabilidade e desempenho são um processo contínuo, não uma atividade que deve ser realizada uma única vez. Implemente monitoramento e alertas abrangentes para monitorar o desempenho da aplicação em produção e use esses dados para refinar continuamente as estratégias de teste e os esforços de otimização. Analise regularmente os dados de desempenho para identificar problemas emergentes, testar novas estratégias de escalabilidade e implementar otimizações para melhorar a eficiência e a confiabilidade da aplicação. Ao adotar uma abordagem iterativa e aprender constantemente com os dados de produção, você pode verificar se a aplicação pode se adaptar às demandas variáveis do usuário e manter a resiliência e o desempenho ideal ao longo do tempo.

Etapas de implementação

1. Estabeleça requisitos de desempenho claros e mensuráveis, como metas de tempo de resposta, throughput e escalabilidade. Esses requisitos devem se basear nos padrões de uso da workload, nas expectativas dos usuários e nas necessidades comerciais.
2. Selecione e configure uma ferramenta de teste de carga que possa imitar com precisão os padrões de carga e o comportamento do usuário no ambiente de produção.
3. Configure um ambiente de teste que corresponda perfeitamente ao ambiente de produção, incluindo condições ambientais e de infraestrutura, para melhorar a precisão dos resultados dos testes.
4. Crie um conjunto de testes que cubra uma ampla variedade de cenários, desde padrões de uso médio até picos de carga, picos rápidos e altas cargas contínuas. Integre os testes aos pipelines contínuos de integração e implantação para capturar regressões de desempenho no início do processo de desenvolvimento.
5. Realize testes de carga para simular o tráfego real de usuários e entender como a aplicação se comporta sob diferentes condições de carga. Para testar o estresse da aplicação, exceda a carga esperada e observe o comportamento dela, como degradação do tempo de resposta, esgotamento de recursos ou falhas do sistema, o que ajuda a identificar o ponto de ruptura da aplicação e informar as estratégias de escalabilidade. Avalie a escalabilidade da workload aumentando incrementalmente a carga e meça o impacto no desempenho para identificar limites de escalabilidade e planejar futuras necessidades de capacidade.
6. Implemente monitoramento e alertas abrangentes para rastrear métricas de desempenho, detectar anomalias e iniciar ações ou notificações de escalabilidade quando os limites forem excedidos.
7. Monitore e analise continuamente os dados de desempenho para identificar áreas de melhoria. Itere as estratégias de teste e os esforços de otimização.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP04 Monitorar e gerenciar cotas
- REL06-BP01 Monitorar todos os componentes da workload (geração)
- REL06-BP03 Enviar notificações (processamento e emissão de alarmes em tempo real)

Documentos relacionados:

- Aplicações de teste de carga
- Teste de carga distribuída na AWS
- Monitoramento do desempenho de aplicações
- Amazon EC2 Testing Policy

Exemplos relacionados:

- Distributed Load Testing on AWS (GitHub)

Ferramentas relacionadas:

- Amazon CodeGuru Profiler
- Amazon CloudWatch RUM
- Apache JMeter
- K6
- Vegeta
- Hey
- ab
- trabalho
- Teste de carga distribuída na AWS

REL12-BP04 Testar a resiliência com engenharia do caos

Execute experimentos de caos regularmente em ambientes que estão em produção ou muito próximos de entrar em produção para entender como seu sistema responde a condições adversas.

Resultado desejado:

A resiliência da workload é verificada regularmente por meio da aplicação de engenharia do caos na forma de experimentos 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 do 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.

Práticas comuns que devem ser evitadas:

- Projetar para resiliência, mas não verificar como a workload funciona como um todo quando falhas ocorrem.
- Nunca realizar experimentos sob condições reais e de carga esperada.
- Não tratar seus experimentos como código nem mantê-los ao longo do ciclo de desenvolvimento.
- Não realizar experimentos 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 experimentos.

Benefícios de implementar esta 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 para implementação

A engenharia do 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. Ela 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 do 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 experimentos de caos devem ser mantidos como testes de regressão automatizados. Dessa forma, os experimentos 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 possa sobreviver à falha de componentes, injete eventos reais como parte dos experimentos. Por exemplo, realize experimentos 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.

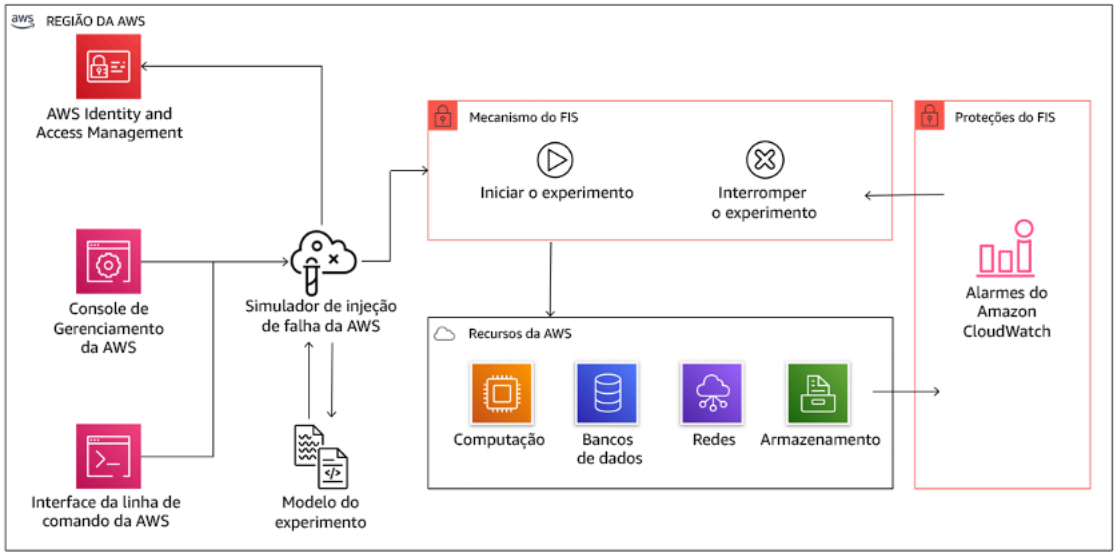
Para falhas no nível da aplicação (como travamentos), você pode começar com fatores de estresse, como exaustão de memória e CPU.

Para validar os 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). Experimentos 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 do 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 game days de engenharia de caos. Ele 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, é possível definir condições de parada como barreiras de proteção para reverter um experimento se ele 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. Isso inclui ferramentas de código aberto, como Chaos Toolkit, Chaos Mesh e Litmus Chaos, além de 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 ao Litmus Chaos, permitindo 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 de implementação

1. Determine quais falhas usar em seus experimentos.

   Avalie o design da workload quanto à resiliência. Esses designs (criados usando as práticas recomendadas do Well-Architected Framework) consideram os riscos com base em dependências críticas, eventos passados, problemas conhecidos e requisitos de conformidade. Liste cada elemento do design destinado a manter a resiliência e as falhas para o qual foi projetado para mitigar. Para obter mais informações sobre a criação dessas listas, consulte o whitepaper Revisões de prontidão operacional, o qual orienta você sobre como criar um processo para evitar a recorrência de incidentes anteriores. 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 é descrito com mais detalhes por Adrian Cockcroft em Modos de falha e resiliência contínua.

2. Atribua uma prioridade a cada falha.

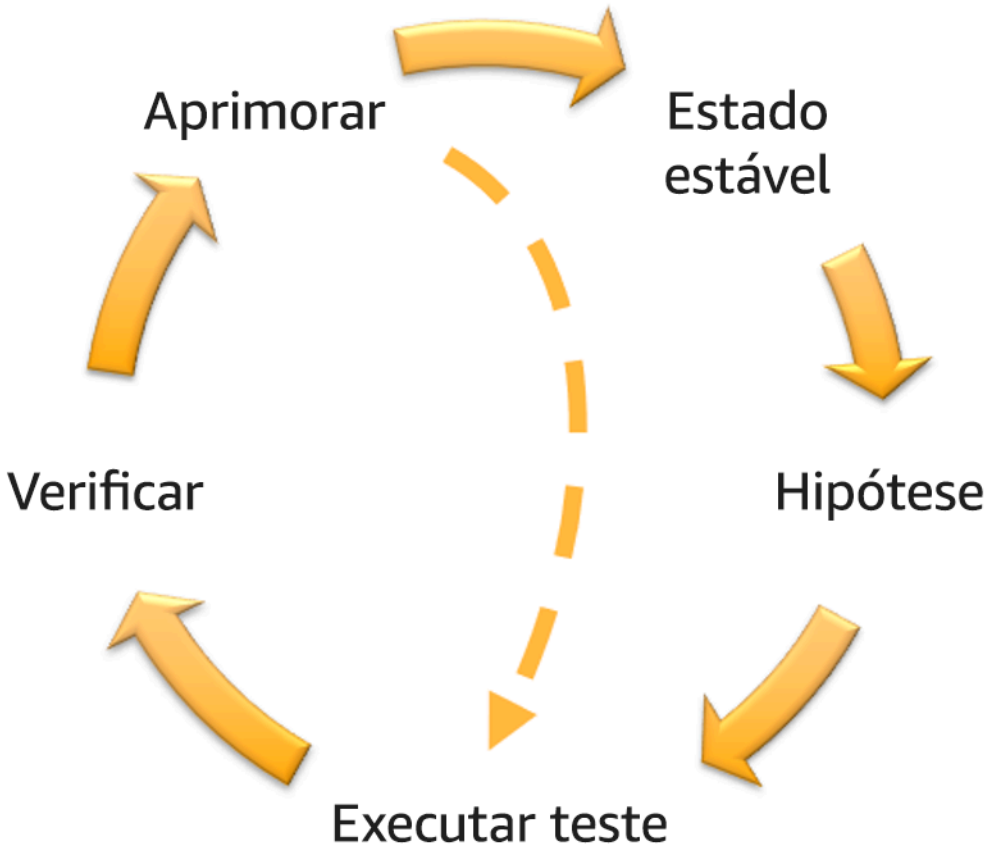
   Comece com uma categorização bruta, como alta, média e baixa. Para avaliar a prioridade, considere a frequência da falha e o impacto da falha na workload total.

   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 experimentos 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 experimentar primeiro e a sequência para desenvolver novos experimentos de injeção de falhas.
3. Para cada experimento realizado, siga o flywheel de engenharia do caos e resiliência contínua na figura a seguir.



Flywheel de engenharia do caos e resiliência contínua usando o método científico, por Adrian Hornsby.

a. 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 poderá observar durante o experimento, o que irá 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.

b. Formule uma hipótese sobre como a workload irá reagir à falha.

Uma boa hipótese baseia-se 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 permanecerá 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 uma redação diferente também é aceitável):

> **Note**
>
> Se uma *falha específica* ocorrer, o *nome da workload descreverá os controles de mitigação* para manter o *impacto da métrica técnica ou comercial*.

Por exemplo:

- Se 20% dos nós no grupo de nós do Amazon EKS forem desativados, a API Transaction Create continuará atendendo ao 99º percentil das solicitações em menos de 100 ms (estado estável). Os nós do Amazon EKS se recuperarão 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 uma única falha de instância do Amazon EC2 ocorrer, a verificação de integridade do Elastic Load Balancing do sistema de ordem fará 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).

c. Execute o experimento injetando a falha.

Um experimento deve, por padrão, ser seguro contra falhas e tolerado pela workload. Se você sabe que a workload irá falhar, não execute o experimento. A engenharia do caos deve ser usada para encontrar incertezas conhecidas ou desconhecidas. Incertezas conhecidas

são coisas que você conhece, mas não entende completamente, enquanto incertezas desconhecidas são coisas das quais você não está ciente nem compreende totalmente. Realizar experimentos em uma workload que você sabe que não funciona não oferecerá novos insights. Seu experimento 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 experimento, prossiga com o teste. Há diversas opções para injetar as falhas. Para workloads na AWS, o AWS FIS fornece muitas simulações de falhas predefinidas chamadas ações. Você também pode definir ações personalizadas que são executadas no AWS FIS usando documentos do AWS Systems Manager.

Recomendamos não usar scripts personalizados para experimentos de caos, a menos que os scripts tenham a capacidade de 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 do caos deve monitorar o estado atual de um experimento, emitir logs e fornecer mecanismos de rollback para acomodar à execução controlada de um experimento. Comece com um serviço estabelecido, como o AWS FIS, que permita que você realize experimentos com um escopo claramente definido e mecanismos de segurança que reverterão o experimento se ele introduzir turbulência inesperada. Para saber mais sobre uma variedade maior de experimentos usando o AWS FIS, consulte também o laboratório Aplicações resilientes e bem arquitetadas com engenharia do caos. Além disso, o AWS Resilience Hub analisará sua workload e criará experimentos que podem ser escolhidos para implementação e execução no AWS FIS.

> **Note**
>
> Para cada experimento, entenda claramente o escopo e seu impacto. Recomendamos que as falhas sejam simuladas primeiro em um ambiente de não produção, antes de serem executadas em produção.

Os experimentos devem ser executados em produção sob carga real usando implantações canário que ativam a implantação de um sistema de controle e experimental, sempre que possível. A realização de experimentos durante horários fora de pico é uma boa prática para mitigar o impacto potencial durante o primeiro experimento na produção. Além disso, se o uso de tráfego real de clientes for algo muito arriscado, você poderá executar experimentos usando

tráfego sintético na infraestrutura de produção nas implantações de controle e experimentais. Quando não for possível usar a produção, realize os experimentos em ambientes de pré-produção que sejam o mais parecido possível com a produção.

Estabeleça e monitore barreiras de proteção para garantir que o experimento não afete o tráfego de produção ou outros sistemas além dos limites aceitáveis. Estabeleça condições de parada para interromper um experimento 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. Um monitor sintético (também conhecido como canário de usuário) é uma métrica que geralmente deve ser incluída como proxy de usuário. As condições de parada para AWS FIS são aceitas como parte do modelo de experimento, permitindo até cinco condições de parada por modelo.

Um dos princípios de caos é minimizar o escopo do experimento 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 experimentos sejam minimizadas e contidas.

Um método para verificar o escopo e o impacto potencial é realizar o experimento primeiro em um ambiente de não produção, verificando se os limites para as condições de parada são ativados conforme o esperado durante o experimento e se há observabilidade em vigor para identificar uma exceção, em vez de testar diretamente em produção.

Ao executar experimentos 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 experimentos serão realizados e o que esperar. Ofereça a essas equipes ferramentas de comunicação para que informem os responsáveis pela execução do experimento 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 irá se restaurar automaticamente. No entanto, alguns designs de falhas ou experimentos malsucedidos podem deixar a workload em um estado de falha inesperado. Ao final do experimento, você deverá estar ciente disso e restaurar a workload e os sistemas. Com o AWS FIS, é possível definir uma configuração de reversão (também chamada de ação posterior) nos parâmetros de ação. Uma ação posterior retorna o destino ao 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.

d. Verifique a hipótese.

   Os 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. O 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 experimentos, a engenharia de caos verifica se o sistema de fato funciona em vez de tentar validar como ele funciona.

   Nos dois exemplos anteriores, incluímos 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 vivenciará 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.

e. 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. Orientações e recursos adicionais podem ser encontrados na AWS Builder's Library, a que qual contém artigos sobre como melhorar suas verificações de interidade ou empregar novas tentativas com atraso no código da aplicação, entre outros.

   Depois de implementar essas mudanças, execute o experimento 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.

4. Execute experimentos regularmente.

Um experimento de caos é um ciclo, e os experimentos devem ser realizados regularmente como parte da engenharia de caos. Depois que uma workload cumprir a hipótese do teste, o experimento 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 realizar experimentos do AWS FIS usando o AWS CodePipeline. Este laboratório sobre experimentos do AWS FIS recorrentes em um pipeline de CI/CD permite que você trabalhe de forma prática.

Os experimentos de injeção de falhas também fazem parte dos game days (consulte REL12-BP05 Conduzir dias de jogo regularmente). Os game days simulam uma falha ou um evento para verificar sistemas, processos e respostas das equipes. O objetivo é executar de fato as ações que a equipe executaria como se um evento excepcional acontecesse.

5. Capture e armazene os resultados do experimento.

   Os resultados dos experimentos de 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 experimento 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 experimento. O registro em log de experimentos com o AWS FIS pode fazer parte dessa captura de dados.

Recursos

Práticas recomendadas relacionadas:

- REL08-BP03 Integrar testes de resiliência como parte da implantação
- REL13-BP03 Testar a implementação da recuperação de desastres para validá-la

Documentos relacionados:

- O que é AWS Fault Injection Service?
- O que é AWS Resilience Hub?
- Princípios da engenharia do caos
- Engenharia de caos: como planejar seu primeiro experimento
- Engenharia de resiliência: como aprender a aceitar falhas

- Histórias sobre engenharia do caos
- Como evitar fallback em sistemas distribuídos
- Implantação canário para experimentos de caos

Vídeos relacionados:

- AWS re:Invent 2020: Testar a resiliência via engenharia do caos (ARC316)
- AWS re:Invent 2019: Melhorar a resiliência com engenharia do caos (DOP309-R1)
- AWS re:Invent 2019: Aplicar a engenharia do caos em um universo de tecnologia sem servidor (CMY301)

Ferramentas relacionadas:

- AWS Fault Injection Service
- AWS Marketplace: Plataforma de engenharia de caos Gremlin
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP05 Conduzir dias de jogo regularmente

Conduza dias de jogo para exercitar regularmente os procedimentos de resposta a eventos e deficiências que afetam a workload. Envolva as mesmas equipes que seriam responsáveis por lidar com os cenários de produção. Esses exercícios ajudam a aplicar medidas para evitar o impacto do usuário causado por eventos de produção. Ao praticar os procedimentos de resposta em condições realistas, você pode identificar e resolver quaisquer lacunas ou fragilidades antes que um evento real ocorra.

Os dias de jogo simulam eventos em ambientes parecidos com os de produção para testar sistemas, processos e respostas das equipes. O objetivo é executar as mesmas ações que a equipe executaria se um evento excepcional acontecesse. Esses exercícios ajudam você a compreender onde é possível aplicar melhorias e podem ajudar a desenvolver experiência organizacional para lidar com eventos e deficiências. Eles devem ser realizados regularmente para que a equipe desenvolva hábitos enraizados sobre como responder.

Os dias de jogo preparam as equipes para lidar com eventos de produção com maior confiança. Equipes bem treinadas são mais capazes de detectar e responder rapidamente a vários cenários. Isso resulta em uma postura de prontidão e resiliência significativamente melhor.

Resultado desejado: você realiza dias de jogos de resiliência de forma consistente e programada. Esses dias de jogo são vistos como uma parte normal e esperada dos negócios. A organização desenvolveu uma cultura de preparação, e quando ocorrem problemas de produção, as equipes estão bem preparadas para responder com eficácia, resolver os problemas de maneira eficiente e mitigar o impacto nos clientes.

Práticas comuns que devem ser evitadas:

- Documentar seus procedimentos, mas nunca colocá-los em prática.
- Excluir os responsáveis pelas decisões de negócios das simulações de teste.
- Organizar um dia de jogo, mas não informar todas as partes interessadas relevantes.
- Concentrar-se apenas nas falhas técnicas, mas não envolver as partes interessadas empresariais.
- Não incorporar as lições aprendidas nos dias de jogo nos processos de recuperação.
- Culpar as equipes por falhas ou bugs.

Benefícios de implementar essa prática recomendada:

- Melhore as habilidades de resposta: nos dias de jogo, as equipes praticam tarefas e testam mecanismos de comunicação durante eventos simulados, o que cria uma resposta mais coordenada e eficiente em situações de produção.
- Identifique e resolva dependências: ambientes complexos geralmente envolvem dependências complexas entre vários sistemas, serviços e componentes. Os dias de jogo podem ajudar você a identificar e resolver essas dependências e a verificar se os sistemas e serviços essenciais estão devidamente cobertos pelos procedimentos do runbook e podem passar por aumento vertical da escala ou ser recuperados em tempo hábil.
- Promova uma cultura de resiliência: os dias de jogos podem ajudar a cultivar uma mentalidade de resiliência dentro de uma organização. Quando você envolve equipes multifuncionais e partes interessadas, esses exercícios promovem a conscientização, a colaboração e uma compreensão compartilhada da importância da resiliência em toda a organização.
- Melhoria e adaptação contínuas: dias de jogo periódicos ajudam você a avaliar e adaptar continuamente as estratégias de resiliência, o que as mantém relevantes e eficazes diante das mudanças nas circunstâncias.

- Aumente a confiança no sistema: os dias de jogo bem-sucedidos podem ajudar você a aumentar a confiança na capacidade do sistema de resistir e se recuperar das interrupções.

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

Orientação para implementação

Depois de projetar e implementar as medidas de resiliência necessárias, realize um dia de jogo para validar se tudo funciona conforme planejado na produção. Um dia de jogo, especialmente o primeiro, deve envolver todos os membros da equipe. Todas as partes interessadas e participantes devem ser informados com antecedência sobre a data, a hora e os cenários simulados.

Durante o dia do jogo, as equipes envolvidas simulam vários eventos e cenários potenciais de acordo com os procedimentos prescritos. Os participantes monitoram e avaliam de perto o impacto desses eventos simulados. Se o sistema operar conforme projetado, os mecanismos automatizados de detecção, escalabilidade e autorrecuperação deverão ser ativados e resultar em pouco ou nenhum impacto nos usuários. Se a equipe observar algum impacto negativo, ela reverte o teste e soluciona os problemas identificados, seja por meios automatizados ou por meio de intervenção manual documentada nos runbooks aplicáveis.

Para melhorar continuamente a resiliência, é fundamental documentar e incorporar as lições aprendidas. Esse processo é um ciclo de feedback que captura sistematicamente os insights dos dias de jogo e os usa para aprimorar sistemas, processos e capacidades da equipe.

Para ajudar você a reproduzir cenários do mundo real em que serviços ou componentes do sistema podem falhar inesperadamente, injete falhas simuladas como um exercício do dia de jogo. As equipes podem testar a resiliência e a tolerância a falhas dos sistemas e simular os processos de resposta e recuperação de incidentes em um ambiente controlado.

Na AWS, os dias de jogo podem ser realizados com réplicas do ambiente de produção usando infraestrutura como código. Por meio desse processo, você pode realizar testes em um ambiente seguro muito semelhante ao seu ambiente de produção. Considere usar o AWS Fault Injection Service para criar diferentes cenários de falha. Use serviços, como o Amazon CloudWatch e o AWS X-Ray, para monitorar o comportamento do sistema durante os dias de jogo. Use o AWS Systems Manager para gerenciar e executar playbooks e use o AWS Step Functionspara orquestrar fluxos de trabalho recorrentes no dia de jogo.

Etapas de implementação

- Estabeleça um programa de dias de jogo: desenvolva um programa estruturado que defina a frequência, o escopo e os objetivos dos dias de jogo. Envolva as principais partes interessadas e especialistas no assunto no planejamento e execução desses exercícios.
- Prepare o dia do jogo:
  1. Identifique os principais serviços essenciais para os negócios que serão o foco do dia do jogo. Catalogue e mapeie as pessoas, os processos e as tecnologias que oferecem suporte a esses serviços.
  2. Defina a agenda para o dia de jogo e prepare as equipes envolvidas para participar do evento. Prepare os serviços de automação para simular os cenários planejados e executar os processos de recuperação apropriados. Os serviços da AWS, como AWS Fault Injection Service, AWS Step Functions e AWS Systems Manager, podem ajudar você a automatizar vários aspectos dos dias de jogo, como injeção de falhas e início das ações de recuperação.
- Execute a simulação: no dia de jogo, execute o cenário planejado. Observe e documente como as pessoas, os processos e as tecnologias reagem ao evento simulado.
- Faça análises após o exercício: depois do dia de jogo, faça uma sessão retrospectiva para revisar as lições aprendidas. Identifique áreas de melhoria e quaisquer ações necessárias para melhorar a resiliência operacional. Documente as descobertas e acompanhe todas as mudanças necessárias para aprimorar as estratégias de resiliência e preparação para a conclusão.

Recursos

Práticas recomendadas relacionadas:

- REL12-BP01 Usar playbooks para investigar falhas
- REL12-BP04 Testar a resiliência com engenharia do caos
- OPS04-BP01 Identificar indicadores-chave de performance
- OPS07-BP03 Usar runbooks para realizar procedimentos
- OPS10-BP01 Usar um processo para gerenciamento de eventos, incidentes e problemas

Documentos relacionados:

- O que é o AWS GameDay?
- AWS Well-Architected Concepts - Game Day

Vídeos relacionados:

- AWS re:Invent 2023 - Practice like you play: How Amazon scales resilience to new heights

Exemplos relacionados:

- AWS Workshop - Navigate the storm: Unleashing controlled chaos for resilient systems
- Build Your Own Game Day to Support Operational Resilience

Planejar para a recuperação de desastres (DR)

Implementar backups e componentes redundantes de workload é o ponto de partida da sua estratégia de DR. O RTO e o RPO são os objetivos para restaurar a 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 workload. 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.

Tanto a disponibilidade quanto a recuperação de desastres dependem das mesmas práticas recomendadas, como monitoramento de falhas, implantação em vários locais e failover automático. No entanto, a disponibilidade se concentra nos componentes da workload, enquanto a recuperação de desastres foca cópias discretas de toda a workload. A recuperação de desastres tem objetivos diferentes da disponibilidade, com foco 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 no local ou na região de recuperação de desastres
- REL13-BP05 Automatizar a recuperação

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

As falhas podem afetar os negócios de várias maneiras. Primeiro, elas podem causar interrupção do serviço (tempo de inatividade). Em segundo lugar, as falhas podem fazer com que os dados se tornem perdidos, inconsistentes ou obsoletos. Para orientar a maneira como você responde e se recupera de falhas, defina um objetivo de tempo de recuperação (RTO) e um objetivo de ponto de recuperação (RPO) para cada workload. O objetivo de tempo de recuperação (RTO) é o atraso aceitável entre a interrupção e a restauração do serviço. O objetivo de ponto de recuperação (RPO) é o tempo máximo aceitável após o último ponto de recuperação de dados.

Resultado desejado: cada workload tem um RTO e um RPO designados com base em considerações técnicas e no impacto comercial.

Práticas comuns que devem ser evitadas:

- Não designar objetivos de recuperação.
- Selecionar objetivos de recuperação arbitrários.
- Selecionar objetivos de recuperação que são muito permissivos e que não atendem aos objetivos de negócios.
- Não avaliar o impacto do tempo de inatividade e da perda de dados.
- Selecionar objetivos de recuperação irreais, como tempo de recuperação zerado ou nenhuma perda de dados, o que pode não ser possível para a configuração da workload.
- Selecionar objetivos de recuperação mais rigorosos do que os objetivos de negócios reais. Isso força implementações de recuperação mais caras e complicadas do que as necessidades da workload.
- Selecionar objetivos de recuperação incompatíveis com os de uma workload dependente.
- Deixar de considerar os requisitos regulatórios e de conformidade.

Benefícios de implementar essa prática recomendada: ao definir RTOs e RPOs para as workloads, você estabelece metas claras e mensuráveis de recuperação com base nas necessidades de negócios. Depois de definir essas metas, você pode criar planos de recuperação de desastres (DR) personalizados para atendê-las.

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

Orientação para implementação

Elabore uma matriz ou planilha para ajudar a orientar o planejamento da recuperação de desastres. Na matriz, crie diferentes categorias ou níveis de workload com base no impacto delas nos negócios (como crítico, alto, médio e baixo) e nas metas de RTO e RPO associadas a cada um. A seguinte matriz fornece um exemplo (observe que os valores de RTO e RPO podem ser diferentes) que você pode seguir:

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

Exemplo de matriz de recuperação de desastres

Para cada workload, investigue e entenda o impacto do tempo de inatividade e da perda de dados para os negócios. O impacto geralmente aumenta com tempo de inatividade e perda de dados, mas a forma do impacto pode ser diferente com base no tipo de workload. Por exemplo, o tempo de inatividade por até uma hora pode ter baixo impacto, mas depois disso o impacto pode se intensificar rapidamente. O impacto pode assumir várias formas, incluindo impacto financeiro (como perda de receita), impacto sobre a reputação (inclusive perda de confiança do cliente), impacto operacional (como folha de pagamento perdida ou diminuição da produtividade) e risco regulatório. Depois de concluído, atribua a workload ao nível apropriado.

Considere as seguintes perguntas ao analisar o impacto da falha:

1. Qual é o tempo máximo em que a workload pode permanecer inativa antes que um impacto inaceitável ocorra nos negócios?
2. Quanto impacto e de que tipo será causado pela empresa devido a uma interrupção na workload? Considere todos os tipos de impacto, incluindo financeiro, reputacional, operacional e regulatório.
3. Qual é a quantidade máxima de dados que podem ser perdidos ou não recuperados antes que um impacto inaceitável ocorra nos negócios?

4. Os dados perdidos podem ser recriados a partir de outras fontes (também conhecidas como dados derivados)? Nesse caso, considere também os RPOs de todos os dados de origem usados para recriar os dados da workload.
5. Quais são os objetivos de recuperação e as expectativas de disponibilidade das workloads das quais esta depende (downstream)? Os objetivos da workload devem ser alcançáveis com base nos recursos de recuperação das respectivas dependências downstream. Considere possíveis soluções alternativas ou mitigações das dependências downstream que possam melhorar a capacidade de recuperação dessa workload.
6. Quais são os objetivos de recuperação e as expectativas de disponibilidade das workloads que dependem desta (upstream)? Os objetivos da workload upstream podem exigir que ela tenha recursos de recuperação mais rigorosos do que parece à primeira vista.
7. Há objetivos de recuperação diferentes com base no tipo de incidente? Por exemplo, você pode ter RTOs e RPOs diferentes, dependendo se o incidente afeta uma zona de disponibilidade ou uma região inteira.
8. Os objetivos de recuperação mudam durante determinados eventos ou épocas do ano? Por exemplo, você pode ter RTOs e RPOs diferentes em temporadas de compras natalinas, eventos esportivos, vendas especiais e lançamentos de novos produtos.
9. Como os objetivos de recuperação se alinham às estratégias de recuperação de desastres organizacional e de linha de negócios que você possa ter?
10. Há ramificações legais ou contratuais a serem consideradas? Por exemplo, você está contratualmente obrigado a fornecer um serviço com um determinado RTO ou RPO? Quais penalidades você pode receber por não cumpri-las?
11. Você precisa manter a integridade dos dados para atender aos requisitos normativos ou de conformidade?

A planilha a seguir pode ajudar na avaliação de cada workload. É possível modificá-la para atender às suas necessidades específicas, como incluir perguntas adicionais.

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

Etapas de implementação

1. Identifique as partes interessadas empresariais e as equipes técnicas responsáveis por cada workload e interaja com elas.
2. Crie categorias ou níveis de criticidade para o impacto da workload na organização. As categorias de exemplo incluem: crítica, alta, média e baixa. Para cada categoria, escolha um RTO e um RPO que reflitam seus objetivos e requisitos de negócios.
3. Atribua uma das categorias de impacto que você criou na etapa anterior a cada workload. Para decidir como é o mapeamento entre uma workload e uma categoria, considere a importância dela para os negócios e o impacto da interrupção ou perda de dados e use as perguntas acima como guia. Isso resulta em um RTO e um RPO para cada workload.
4. Considere o RTO e o RPO para cada workload determinada na etapa anterior. Envolva as equipes técnicas e comerciais da workload para determinar se os objetivos devem ser ajustados. Por exemplo, as partes interessadas da empresa podem determinar que metas mais rigorosas são necessárias. Como alternativa, as equipes técnicas poderiam determinar que as metas

deveriam ser modificadas para torná-las alcançáveis com os recursos disponíveis e as restrições tecnológicas.

Recursos

Práticas recomendadas relacionadas:

- REL09-BP04 Realizar a recuperação periódica dos dados para verificar a integridade e os processos de backup
- REL12-BP01 Usar playbooks para investigar falhas
- 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

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)
- Gerenciar políticas de resiliência com o AWS Resilience Hub
- Parceiro da 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
- 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 (ativa/passiva) ou ativa/ativa.

Resultado desejado: para cada workload, há uma estratégia de DR definida e implementada que permite que a workload alcance os objetivos de DR. As estratégias de DR entre workloads fazem uso de padrões reutilizáveis (como as estratégias descritas anteriormente).

Práticas comuns que devem ser evitadas:

- Implementar procedimentos de recuperação inconsistentes para workloads com objetivos de DR semelhantes.
- Deixar que a estratégia de DR seja implementada ad hoc quando um desastre ocorrer.
- Não ter um plano para a recuperação de desastres.
- Depender das operações do ambiente de gerenciamento durante a recuperação.

Benefícios de implementar esta 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 pertencentes a elas.

Nível de risco exposto se esta prática recomendada não for 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 para 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 implementar proteção contra um evento improvável que impeça a execução da workload em determinada Região da AWS for um requisito, você poderá optar por uma estratégia de DR que use várias regiões.

Ao arquitetar uma estratégia de DR em várias regiões, é necessário escolher uma das estratégias a seguir. Elas são listadas em ordem crescente de custo e complexidade e em ordem decrescente de

RTO e RPO. Região de recuperação se refere a uma Região da AWS diferente da principal usada para sua 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 para um ponto no tempo (PITR), o que pode 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 sua infraestrutura principal de workload na região de recuperação. Replique seus dados na região de recuperação e crie backups deles nessa região. 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. No entanto, eles podem ser criados com a configuração e o código da aplicação necessários.
- Standby passivo (RPO em segundos, RTO em minutos): mantenha uma versão em escala vertical 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 residem na região de recuperação. No momento da recuperação, a escala do sistema é aumentada vertical e 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 em várias regiões (vários sites) (RPO próximo de zero, RTO potencialmente zero): sua workload é implantada e atende ativamente ao tráfego de várias Regiões da AWS. Essa estratégia exige que você sincronize os dados entre regiões. É necessário 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 abordagem de 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 ambas as opções 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 de implementação

1. Determine uma estratégia de recuperação de desastres que satisfaça os requisitos de recuperação dessa workload.

   Escolher uma estratégia de recuperação de desastres é uma troca entre reduzir o tempo de inatividade e a perda de dados (RTO e RPO) e o custo e a complexidade da implementação da estratégia. 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 recuperação de desastres de luz-piloto e standby passivo atenderão tanto ao RTO quanto aos critérios de custo.

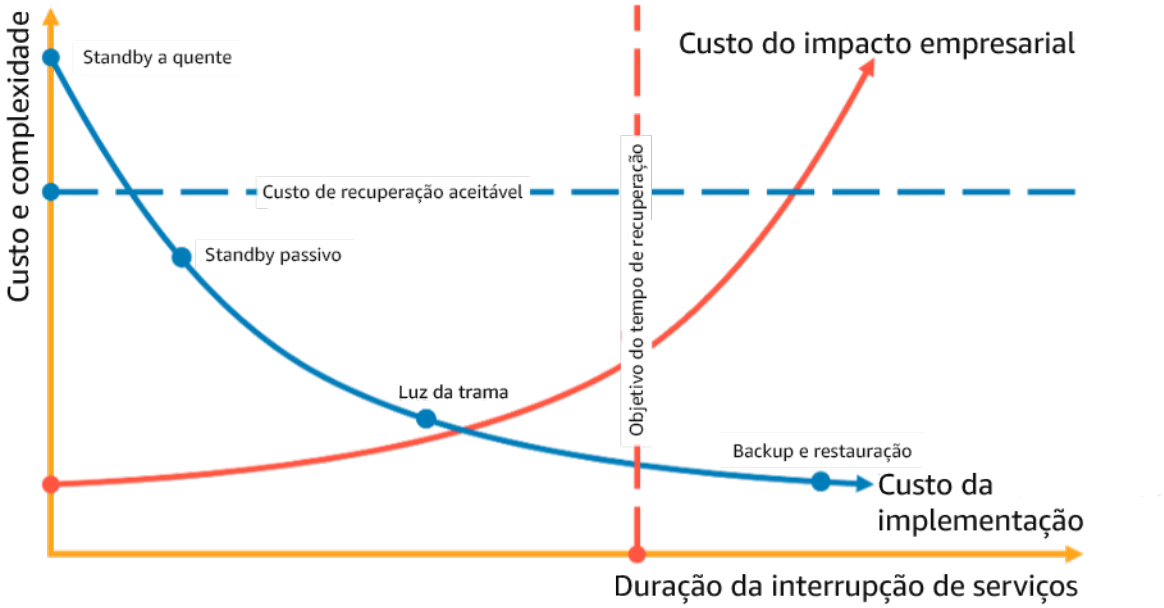


Figura 18: Escolher uma estratégia de recuperação de desastres com base no RTO e no custo

Para saber mais, consulte Plano de continuidade de negócios (BCP).

2. Analise os padrões de como a estratégia de recuperação de desastres selecionada pode ser implementada.

   O objetivo dessa etapa é entender 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 recuperação de desastres, a qual 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, mas exigirá mais tempo e esforços para restaurar a workload, resultando em maior RTO e RPO. É uma boa prática sempre fazer backups dos dados e copiá-los para outro local (como outra Região da AWS).

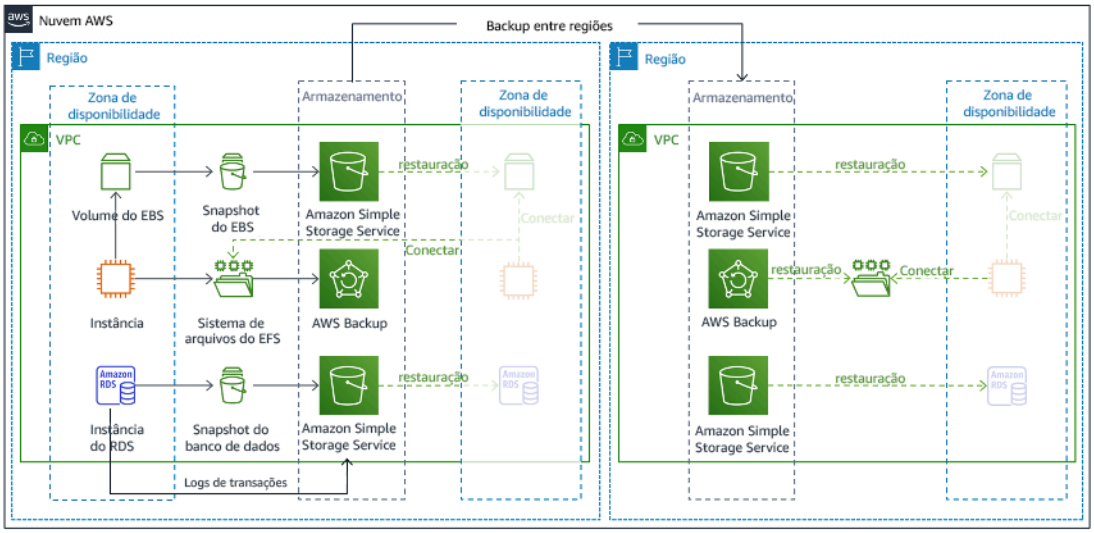


Figura 19: Arquitetura de backup e restauração

Para obter mais detalhes sobre essa estratégia, consulte Arquitetura de recuperação de desastres (DR) na AWS, Parte II: backup e restauração com recuperação rápida.

Luz piloto

Com a abordagem luz piloto, você replica seus dados da região principal 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.

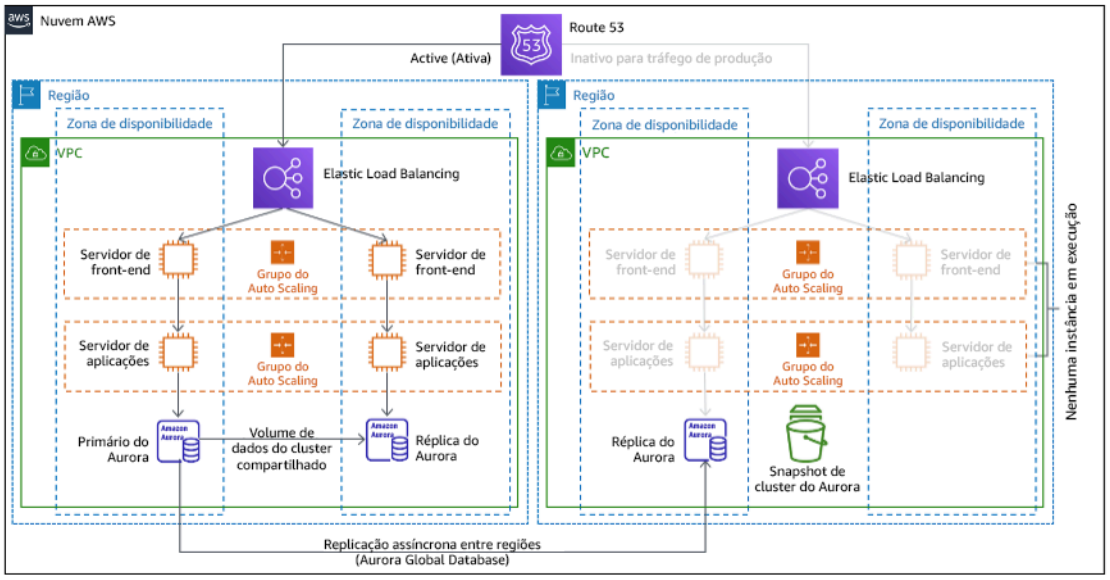


Figura 20: Arquitetura de luz piloto

Para obter mais detalhes sobre essa estratégia, consulte Arquitetura de recuperação de desastres (DR) na AWS, parte III: luz piloto e standby passivo.

Standby passivo

A abordagem de standby passivo envolve garantir que haja uma cópia reduzida, 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 for implantada com capacidade total, isso será conhecido como standby a quente.

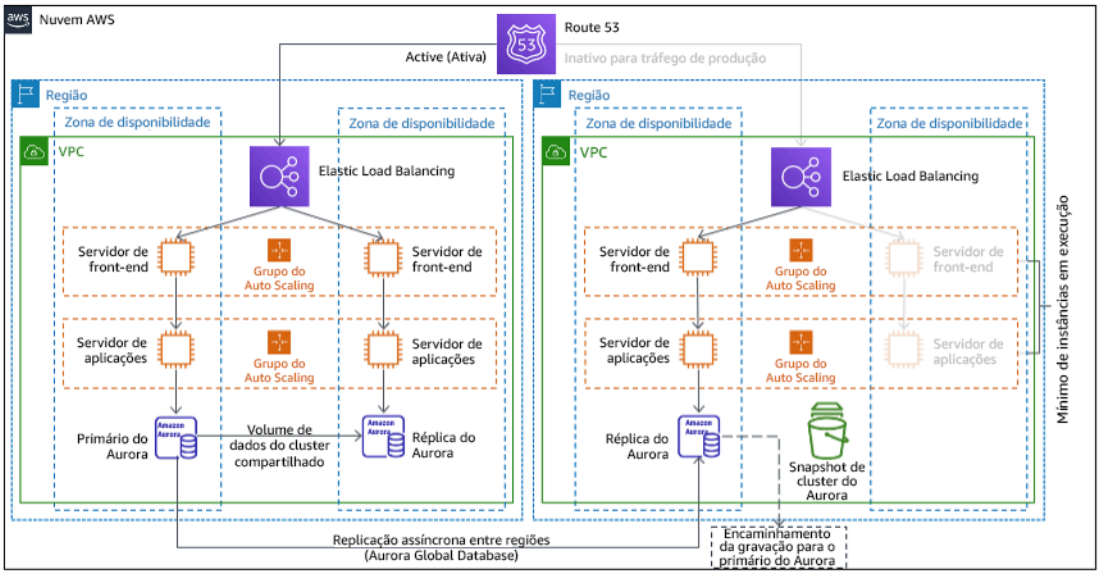


Figura 21: Arquitetura de standby passivo

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. Quando o AWS Lambda é usado, a simultaneidade provisionada pode fornecer ambientes de runtime para que eles estejam preparados para responder imediatamente às invocações da sua função.

Para obter mais detalhes sobre essa estratégia, consulte Arquitetura de recuperação de desastres (DR) na AWS, parte III: luz piloto e standby passivo.

Multissite ativa/ativa

Você pode executar sua workload simultaneamente em várias regiões como parte de uma estratégia multissite ativa/ativa. A estratégia multissite ativa-ativa atende ao tráfego de todas as regiões onde está implantada. Os clientes podem selecionar essa estratégia para outros fins além da recuperação de desastres. 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. A multissite ativa/ativa é a estratégia de DR mais complexa operacionalmente e deve ser selecionada apenas quando os requisitos empresariais exigirem.

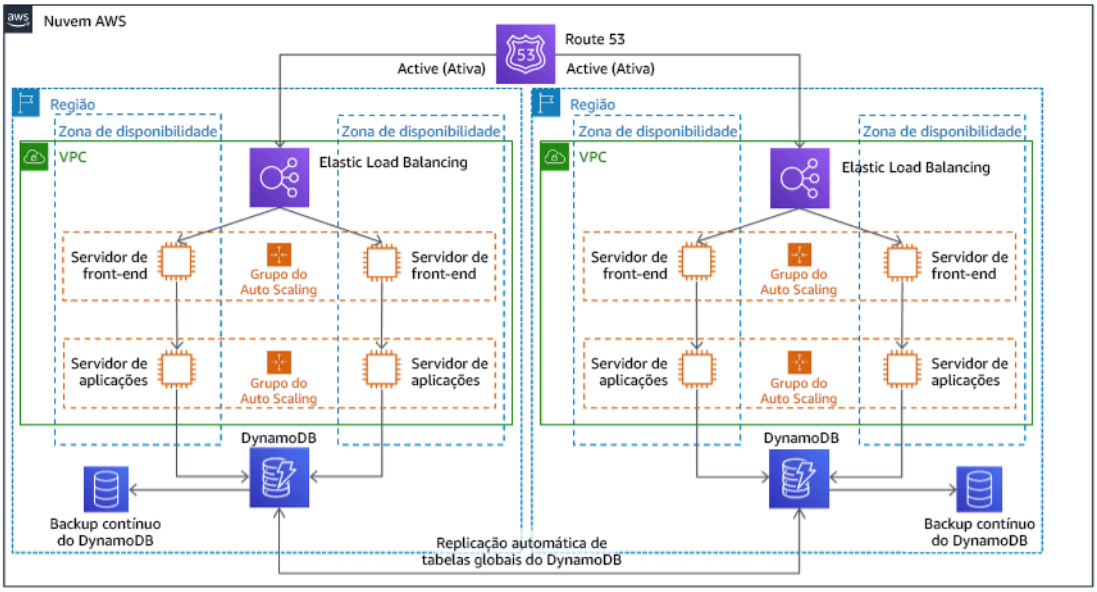


Figura 22: Arquitetura multissite ativa/ativa

Para obter mais detalhes sobre essa estratégia, consulte [Arquitetura de recuperação de desastres (DR) na AWS, Parte IV: multissite ativa/ativa]

AWS Elastic Disaster Recovery

Se você está considerando a estratégia de luz piloto ou standby passivo para recuperação de desastres, o AWS Elastic Disaster Recovery pode fornecer uma abordagem alternativa com benefícios aprimorados. O Elastic Disaster Recovery pode oferecer uma meta de RPO e RTO semelhante ao standby passivo, mas mantendo a abordagem de baixo custo da luz piloto. O Elastic Disaster Recovery replica os dados da sua região primária para sua região de recuperação usando proteção contínua de dados para obter 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. Quando o Elastic Disaster Recovery é usado, o serviço coordena e orquestra a recuperação de recursos de computação quando iniciado como parte de um failover ou de uma simulação.

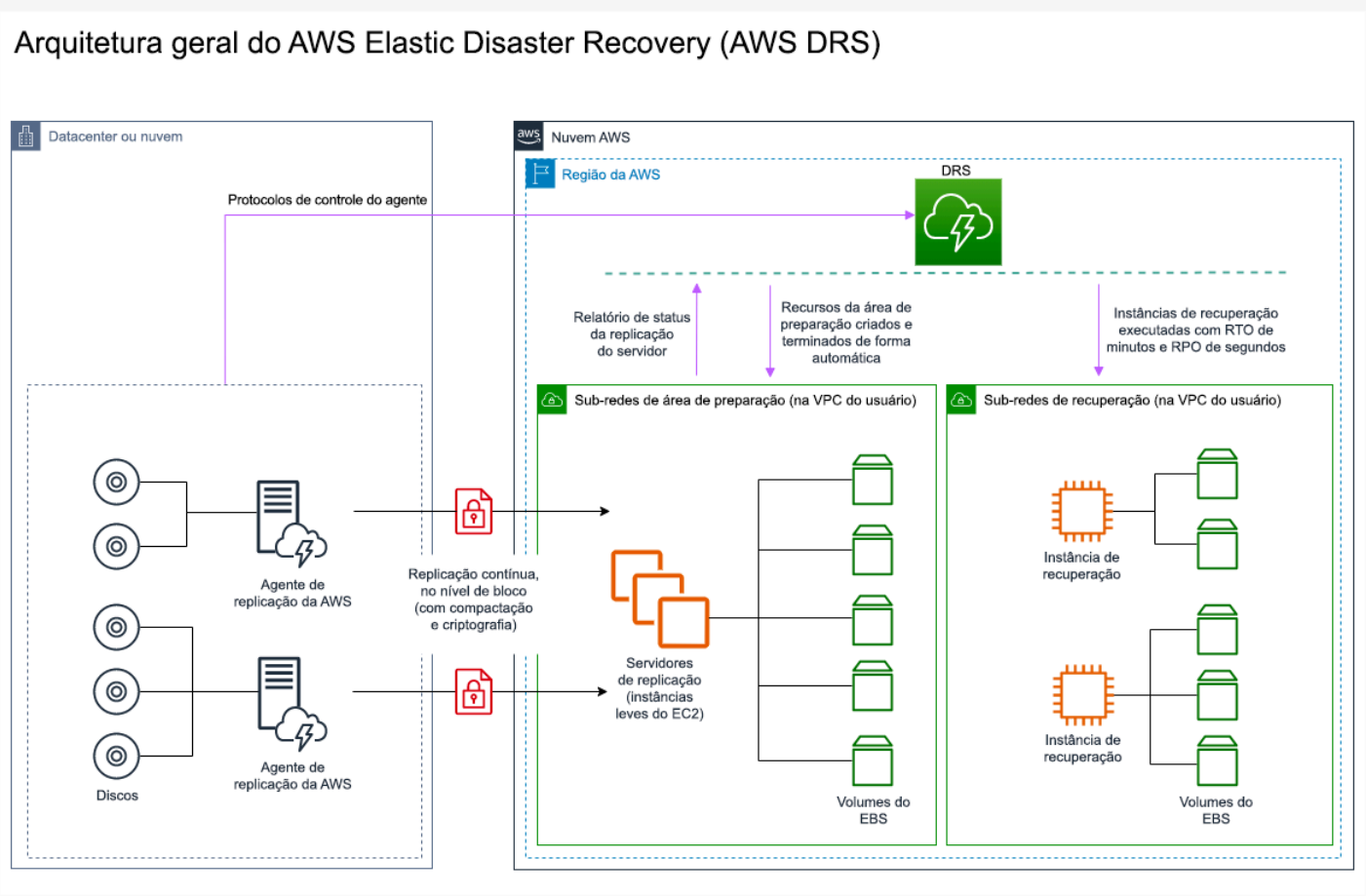


Figura 23: Arquitetura do AWS Elastic Disaster Recovery

Práticas adicionais para proteger dados

Com todas as estratégias, você também deve mitigar as consequências de um desastre de dados. A replicação contínua de dados protege você contra alguns tipos de desastre, mas não contra corrompimento 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.

Usar 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 faz uso de uma multissite ativa/ativa, já que as instâncias do Amazon EC2 e o Elastic Load Balancer têm recursos implantados

em várias AZs, processando ativamente as solicitações. A arquitetura também demonstra o standby a quente, em que, se a instância primária do Amazon RDS falhar (ou a própria AZ falhar), a instância em espera será promovida para primária.

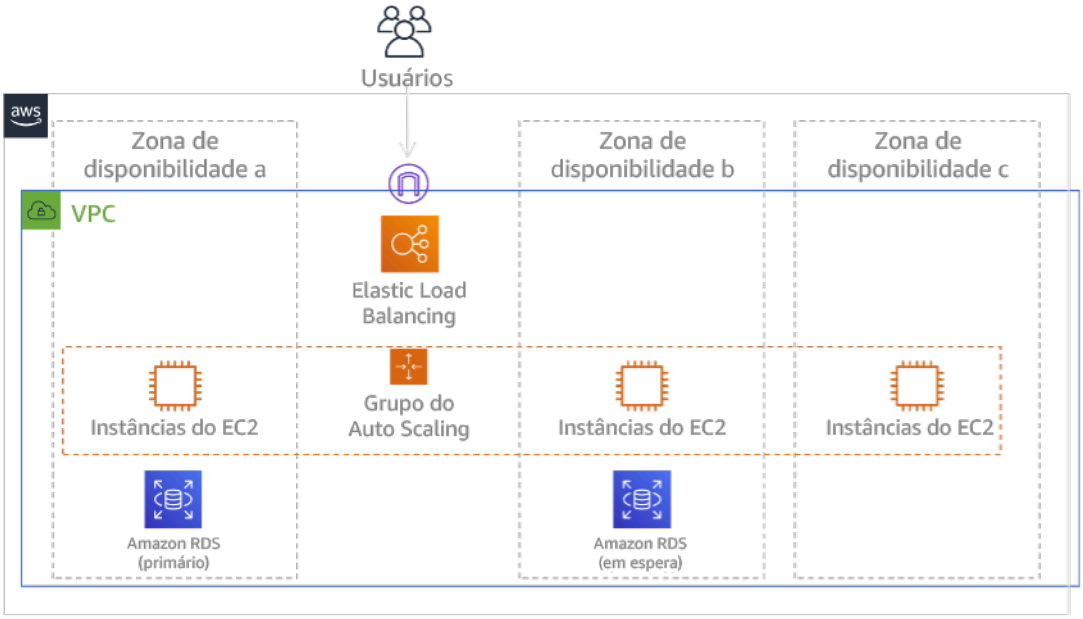


Figura 24: Arquitetura Multi-AZ

Além da arquitetura de alta disponibilidade, é necessário 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, copie os backups de dados para outra Região da AWS como uma camada adicional de proteção.

Uma abordagem alternativa menos comum para uma única região, a recuperação de desastres multi-AZ é ilustrada na publicação do blog Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack. Aqui, a estratégia é manter o máximo de isolamento possível entre as AZs, da mesma forma que as regiões operam. Ao usar esta estratégia alternativa, você pode escolher uma abordagem ativa/ativa ou ativa/passiva.

> **Note**
>
> Algumas workloads têm requisitos regulatórios de residência de dados. Se isso se aplicar à sua workload em uma localidade que atualmente tem apenas uma Região da AWS, a opção de multirregiões não atenderá às suas necessidades empresariais. As estratégias Multi-AZ fornecem boa proteção contra a maioria dos desastres.

3. Avalie os recursos da sua workload e qual será sua configuração na região de recuperação antes do failover (durante a operação normal).

   Para infraestrutura e recursos da AWS, use 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, é possível usar o AWS CloudFormation StackSets. Para estratégias multissite ativa/ativa 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 os parâmetros e a lógica condicional do CloudFormation, é possível controlar se uma pilha implantada está ativa ou em espera com um único modelo. Quando o Elastic Disaster Recovery é usado, o serviço replica e orquestra a restauração de configurações da aplicação e os recursos de computação.

   Todas as estratégias de recuperação de desastres exigem que as fontes de dados sejam copiadas para backup dentro da Região da AWS e que esses backups sejam copiados para a região de recuperação. O AWS Backup fornece uma visão centralizada na qual é possível configurar, programar e monitorar backups desses recursos. Para luz piloto, standby passivo e multissite ativa/ativa, também é necessário replicar dados da região principal 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 esta série de blogs sobre como Criar aplicações multirregiões com serviços da AWS.

4. Determine e implemente como você preparará sua região de recuperação para failover quando necessário (durante um evento de desastre).

   Para multissite ativa/ativa, 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, talvez seja necessário promover instâncias somente leitura de bancos de dados para transformá-las na 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. Consulte 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), sub-redes e grupos de segurança necessários. É possível automatizar grande parte do processo de restauração. Para saber como, consulte esta postagem do blog.

5. Determine e implemente como você preparará sua região de recuperação 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, o failover iniciado manualmente é usado com frequência. Nesse caso, você ainda deve automatizar as etapas para failover para que a inicialização manual ocorra com o apertar um botão.

   Há várias opções de gerenciamento de tráfego a serem consideradas ao usar os serviços 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 o failover iniciado manualmente, é possível usar o Amazon Application Recovery Controller, que fornece uma API de plano de dados altamente disponível destinada a redirecionar 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. Crie um plano de como será failback da workload.

   O failback é quando a operação da workload retorna 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 datastores 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 permaneça atualizada.

Alguns serviços da AWS fazem isso automaticamente. Se estiver usando tabelas globais do Amazon DynamoDB, mesmo que a tabela na região primária tenha se tornado indisponível, o DynamoDB retomará a propagação de todas as gravações pendentes assim que ela retornar online. Se estiver usando o Amazon Aurora Global Database e usando failover planejado gerenciado, a topologia de 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.

Nos casos em que isso não é automático, será necessário 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.

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 praticam uma rotação agendada, 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 playbook 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 Como executar um 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
- Criar uma solução de backend multirregiões ativa-ativa sem servidor em uma hora
- Backend multirregiões sem servidor: recarregado
- RDS: como replicar uma réplica de leitura entre regiões
- Route 53: configurar o failover de DNS
- S3: replicação entre regiões
- O que é o AWS Backup?
- What is Amazon Application Recovery Controller?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: introdução - AWS
- Parceiro da 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 multirregiões (ARC209-R2)
- Introdução ao AWS Elastic Disaster Recovery | Amazon Web Services

REL13-BP03 Testar a implementação da recuperação de desastres para validá-la

Teste regularmente o failover para o local de recuperação para verificar se a operação está correta e se o RTO e o RPO são cumpridos.

Práticas comuns que devem ser evitadas:

- Nunca praticar failovers na produção.

Benefícios de implementar esta prática recomendada: testar regularmente seu plano de recuperação de desastres garantirá que ele funcione quando necessário e que sua equipe saiba como executar a estratégia.

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

Orientação para implementação

Um padrão que deve ser evitado é o desenvolvimento de caminhos de recuperação que raramente são executados. Por exemplo, você pode ter um datastore secundário utilizado para consultas somente leitura. Quando você grava em um datastore e o datastore 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 datastore 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 testado 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 praticar regularmente essa falha na produção para se convencer de que o caminho de recuperação funciona. No exemplo que acabamos de discutir, você deve realizar o failover para o standby regularmente, não importa a necessidade.

Etapas de implementação

1. Projete suas workloads 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 baseadas no Amazon EC2, use o AWS Elastic Disaster Recovery para implementar e lançar instâncias de simulação para sua estratégia de DR. A AWS Elastic Disaster

Recovery fornece a capacidade de realizar exercícios 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 da 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: como se preparar para o failover
- O projeto de computação orientado por recuperação de Berkeley/Stanford
- O que é o AWS Fault Injection Simulator?

Vídeos relacionados:

- AWS re:Invent 2018: Padrões de arquitetura para aplicações ativas-ativas multirregiões (ARC209-R2)
- AWS re:Invent 2019: Soluções de backup e restauração e recuperação de desastres com a AWS

REL13-BP04 Gerenciar o desvio de configuração no local ou na região de recuperação de desastres

Para realizar um procedimento bem-sucedido de recuperação de desastres (DR), a workload deve ser capaz de retomar as operações normais em tempo hábil, sem perda relevante de funcionalidade ou dados, assim que o ambiente de DR ficar on-line. Para atingir essa meta, é essencial manter a infraestrutura, os dados e as configurações consistentes entre o ambiente de DR e o ambiente primário.

Resultado desejado: a configuração e os dados do local de recuperação de desastres estão em paridade com o local primário, o que facilita a recuperação rápida e completa quando necessário.

Práticas comuns que devem ser evitadas:

- Não atualizar os locais de recuperação quando são feitas alterações nos locais primários, o que resulta em configurações desatualizadas que podem prejudicar os esforços de recuperação.
- Não considerar possíveis limitações, como diferenças de serviço entre locais primários e de recuperação, o que pode levar a falhas inesperadas durante o failover.
- Depender de processos manuais para atualizar e sincronizar o ambiente de DR, o que aumenta o risco de erro humano e inconsistência.
- Não conseguir detectar desvio na configuração, o que leva a uma falsa sensação de prontidão do local de DR antes de um incidente.

Benefícios de implementar essa prática recomendada: a consistência entre o ambiente de DR e o ambiente primário melhora significativamente a probabilidade de uma recuperação bem-sucedida após um incidente e reduz o risco de falha no procedimento de recuperação.

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

Orientação para implementação

Uma abordagem abrangente ao gerenciamento de configuração e preparação para failover pode ajudar você a verificar se o local de DR está constantemente atualizado e preparado para assumir o controle em caso de falha no local primário.

Para obter consistência entre os ambientes primário e de recuperação de desastres (DR), valide se os pipelines de entrega distribuem aplicações tanto para os locais primários quanto para os locais de DR. Implemente as alterações nos locais de DR após um período de avaliação apropriado (também conhecido como implantações progressivas) para detectar problemas no local primário e interromper a implantação antes que eles se espalhem. Implemente o monitoramento para detectar desvios na configuração e rastrear as alterações e a conformidade dos ambientes. Execute a remediação automatizada no local de DR para mantê-lo totalmente consistente e pronto para assumir o controle no caso de um incidente.

Etapas de implementação

1. Valide se a região de DR contém os recursos e serviços da AWS necessários para uma execução bem-sucedida do seu plano de DR.
2. Use a infraestrutura como código (IaC). Mantenha a infraestrutura de produção e modelos de configuração de aplicações precisos e aplique-os regularmente ao ambiente de recuperação de desastres. O AWS CloudFormation pode detectar desvios entre o que os modelos do CloudFormation especificam e o que é realmente implantado.

3. Configure pipelines de CI/CD para implantar atualizações de aplicações e infraestrutura em todos os ambientes, incluindo locais primários e de DR. Soluções de CI/CD, como o AWS CodePipeline, podem automatizar o processo de implantação, o que reduz o risco de desvio na configuração.
4. Faça implantações progressivas entre os ambientes primário e de DR. Essa abordagem permite que as atualizações sejam inicialmente implantadas e testadas no ambiente primário, o que isola os problemas no local primário antes que eles sejam propagados para o local de DR. Essa abordagem evita que defeitos sejam enviados simultaneamente para a produção e para o local de DR e mantém a integridade do ambiente de DR.
5. Monitore continuamente as configurações de recursos nos ambientes primário e de DR. Soluções como o AWS Config podem ajudar a impor a conformidade da configuração e detectar desvios, o que ajuda a manter as configurações consistentes entre os ambientes.
6. Implemente mecanismos de alerta para rastrear e notificar sobre qualquer desvio de configuração ou interrupção ou atraso na replicação de dados.
7. Automatize a correção do desvio de configuração detectado.
8. Agende auditorias regulares e verificações de conformidade para verificar o alinhamento contínuo entre as configurações primária e de DR. As revisões periódicas ajudam você a manter a conformidade com as regras definidas e a identificar quaisquer discrepâncias que precisem ser resolvidas.
9. Verifique se há incompatibilidades na capacidade provisionada pela AWS, nas cotas de serviço, nos controles de utilização e nas discrepâncias de configuração e versão.

Recursos

Práticas recomendadas relacionadas:

- REL01-BP01 Conhecer as cotas e restrições de serviços
- REL01-BP02 Gerenciar cotas de serviço em várias contas e regiões
- REL01-BP04 Monitorar e gerenciar cotas
- REL13-BP03 Testar a implementação da recuperação de desastres para validá-la

Documentos relacionados:

- Como remediar recursos não compatíveis da AWS pelo Regras do AWS Config
- AWS Systems Manager Automation
- AWS CloudFormationDetectar alterações de configuração não gerenciadas em pilhas e recursos

- AWS CloudFormation: detectar desvios em uma pilha inteira do CloudFormation
- 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?
- Como remediar recursos não compatíveis da AWS pelo Regras do AWS Config

Vídeos relacionados:

- AWS re:Invent 2018: Padrões de arquitetura para aplicações ativas-ativas multirregiões (ARC209-R2)

Exemplos relacionados:

- Registro do AWS CloudFormation
- Monitor de cotas para AWS
- Implement automatic drift remediation for AWS CloudFormation using Amazon CloudWatch and AWS Lambda
- Blog de arquitetura da AWS: série de recuperação de desastres
- AWS Marketplace: produtos que podem ser usados para recuperação de desastres
- Automatizar implantações seguras e sem intervenção manual

REL13-BP05 Automatizar a recuperação

Implemente mecanismos de recuperação testados e automatizados que sejam confiáveis, observáveis e reproduzíveis para reduzir o risco e o impacto comercial da falha.

Resultado desejado: você implementou um fluxo de trabalho de automação bem documentado, padronizado e completamente testado para processos de recuperação. Sua automação de recuperação corrige automaticamente pequenos problemas que apresentam baixo risco de perda ou indisponibilidade de dados. Você pode invocar rapidamente processos de recuperação para incidentes graves, observar o comportamento de remediação enquanto eles operam e encerrar os processos se observar situações perigosas ou falhas.

Práticas comuns que devem ser evitadas:

- Você depende de componentes ou mecanismos que estão em estado de falha ou degradação como parte do seu plano de recuperação.
- Seus processos de recuperação exigem intervenção manual, como acesso ao console (também conhecido como operações de clique).
- Você inicia automaticamente os procedimentos de recuperação em situações que apresentam um alto risco de perda ou indisponibilidade de dados.
- Você não inclui um mecanismo para abortar um procedimento de recuperação (como um cabo Andon ou um grande botão vermelho de parada) que não está funcionando ou que apresenta riscos adicionais.

Benefícios de implementar essa prática recomendada:

- Maior confiabilidade, previsibilidade e consistência das operações de recuperação.
- Capacidade de atingir objetivos de recuperação mais rigorosos, incluindo Objetivo de Tempo de Recuperação (RTO) e Objetivo de Ponto de Recuperação (RPO).
- Probabilidade reduzida de falha na recuperação durante um incidente.
- Risco reduzido de falhas associadas a processos de recuperação manual propensos a erros humanos.

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

Orientação para implementação

Para implementar a recuperação automatizada, você precisa de uma abordagem abrangente que use serviços da AWS e práticas recomendadas. Para começar, identifique componentes críticos e possíveis pontos de falha em sua workload. Desenvolva processos automatizados que possam recuperar suas workloads e dados de falhas sem intervenção humana.

Desenvolva a automação da recuperação usando princípios de infraestrutura como código (IaC). Isso torna seu ambiente de recuperação consistente com o ambiente de origem e permite o controle de versão de seus processos de recuperação. Para orquestrar fluxos de trabalho de recuperação complexos, considere soluções como o AWS Systems Manager Automation ou o AWS Step Functions.

A automação de processos de recuperação oferece benefícios significativos e pode facilitar o alcance do objetivo de tempo de recuperação (RTO) e do objetivo de ponto de recuperação (RPO).

No entanto, podem ocorrer situações inesperadas que podem provocar falhas ou criar novos riscos, como tempo de inatividade adicional e perda de dados. Para mitigar esse risco, forneça a capacidade de interromper rapidamente uma automação de recuperação em andamento. Uma vez interrompida, você pode investigar e tomar medidas corretivas.

Para workloads compatíveis, considere soluções como o AWS Elastic Disaster Recovery (AWS DRS) para fornecer failover automatizado. AWS O DRS replica continuamente suas máquinas (incluindo o sistema operacional, a configuração do estado do sistema, bancos de dados, aplicações e arquivos) em uma área de armazenamento de baixo custo em sua Conta da AWS de destino e região preferida. Se ocorrer um incidente, o AWS DRS automatiza a conversão de seus servidores replicados em workloads totalmente provisionadas em sua região de recuperação na. AWS

A manutenção e o aprimoramento da recuperação automatizada são um processo contínuo. Teste e refine continuamente os procedimentos de recuperação com base nas lições aprendidas e mantenha-se atualizado sobre novos serviços e recursos da AWS que podem aprimorar os recursos de recuperação.

Etapas de implementação

1. Planeje a recuperação automatizada
    a. Faça uma análise completa da arquitetura, dos componentes e das dependências da workload para identificar e planejar mecanismos de recuperação automatizados. Categorize as dependências da workload em dependências rígidas e flexíveis. Dependências rígidas são aquelas sem as quais a workload não pode operar e para as quais nenhum substituto pode ser fornecido. Dependências flexíveis são aquelas que a workload normalmente usa, mas que são substituíveis por sistemas ou processos substitutos temporários ou que podem ser tratadas por meio de uma degradação gradual.
    b. Estabeleça processos para identificar e recuperar dados perdidos ou corrompidos.
    c. Defina as etapas para confirmar um estado estável recuperado após a conclusão das ações de recuperação.
    d. Considere todas as ações necessárias para preparar o sistema recuperado para o serviço completo, como pré-aquecimento e preenchimento de caches.
    e. Considere os problemas que podem ser encontrados durante o processo de recuperação e como detectá-los e corrigi-los.
    f. Considere cenários em que o local primário e o respectivo ambiente de gerenciamento estejam inacessíveis. Verifique se as ações de recuperação podem ser executadas de forma independente, sem depender do local primário. Considere soluções como o Controlador de

Recuperação de Aplicações (ARC) da Amazon para redirecionar o tráfego sem a necessidade de alterar manualmente os registros DNS.

2. Desenvolva um processo de recuperação automatizado
   a. Implemente mecanismos automatizados de detecção de falhas e failover para recuperação sem usar as mãos. Crie painéis, como com o Amazon CloudWatch, para relatar o progresso e a integridade dos procedimentos automatizados de recuperação. Inclua procedimentos para validar a recuperação bem-sucedida. Forneça um mecanismo para abortar uma recuperação em andamento.
   b. Crie manuais como um processo alternativo para falhas que não podem ser recuperadas automaticamente e leve em consideração seu plano de recuperação de desastres.
   c. Teste os processos de recuperação conforme discutido em REL13-BP03.
3. Prepare-se para a recuperação
   a. Avalie o estado do seu local de recuperação e implante componentes essenciais nele com antecedência. Para conferir mais detalhes, consulte REL13-BP04.
   b. Defina funções, responsabilidades e processos de tomada de decisão claros para operações de recuperação, envolvendo partes interessadas e equipes relevantes em toda a organização.
   c. Defina as condições para iniciar seus processos de recuperação.
   d. Crie um plano para reverter o processo de recuperação e retornar ao local primário, se necessário ou depois que ele for considerado seguro.

Recursos

Práticas recomendadas relacionadas:

- REL07-BP01 Usar automação ao obter ou escalar recursos
- REL11-BP01 Monitorar todos os componentes da workload para detectar falhas
- 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 no local ou na região de recuperação de desastres

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)
- Orchestrate Disaster Recovery Automation using Amazon Route 53 ARC and AWS Step Functions
- Build AWS Systems Manager Automation runbooks using AWS CDK
- AWS Marketplace: produtos que podem ser usados para recuperação de desastres
- AWS Systems Manager Automation
- AWS Elastic Disaster Recovery
- Usar o Elastic Disaster Recovery para failover e failback
- Recursos do AWS Elastic Disaster Recovery
- Parceiro da APN: parceiros que podem ajudar com a recuperação de desastres

Vídeos relacionados:

- AWS re:Invent 2018: Padrões de arquitetura para aplicações ativas-ativas multirregiões (ARC209-R2)
- AWS re:Invent 2022: AWS On Air ft. AWS Failback for AWS Elastic Disaster Recovery

Conclusão

Seja você novo nos tópicos de disponibilidade e confiabilidade ou um veterano experiente buscando informações para maximizar sua disponibilidade de workload de missão crítica, esperamos que este whitepaper tenha feito você 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 a 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 altos níveis de disponibilidade foram atingidos na AWS. Contate sua equipe de conta ou use o link Entre em contato conosco em nosso site.

Colaboradores

Os colaboradores deste documento incluem:

- Michael Fischer, arquiteto líder de soluções, Amazon Web Services
- Seth Eliot, defensor líder de desenvolvedores, Amazon Web Services
- Mahanth Jayadeva, arquiteto de soluções, Well-Architected, Amazon Web Services
- James Devine, arquiteto líder de soluções, Amazon Web Services
- Jason DiDomenico, arquiteto sênior de soluções, Cloud Foundations, Amazon Web Services
- Marcin Bednarz, arquiteto líder de soluções, Amazon Web Services
- Tyler Applebaum, arquiteto sênior de soluções, Amazon Web Services
- Rodney Lester, arquiteto líder de soluções, Amazon Web Services
- Joe Chapman, arquiteto sênior de soluções, Amazon Web Services
- Adrian Hornsby, engenheiro líder de desenvolvimento de sistemas, Amazon Web Services
- Kevin Miller, vice-presidente, S3, Amazon Web Services
- Shannon Richards, gerente líder de programas técnicos, Amazon Web Services
- Laurent Domb, tecnólogo líder, Fed Fin, Amazon Web Services
- Kevin Schwarz, arquiteto sênior de soluções, Amazon Web Services
- Rob Martell, arquiteto líder de resiliência na nuvem, Amazon Web Services
- Priyam Reddy, gerente sênior de arquitetura de soluções, DR, Amazon Web Services
- Jeff Ferris, tecnólogo líder, Amazon Web Services
- Matias Battaglia, arquiteto sênior de soluções, Amazon Web Services

Outras fontes de leitura

Para obter informações adicionais, consulte:

- Framework Well-Architected da AWS
- Centro de Arquitetura do AWS

Revisões do documento

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

Alteração	Descrição	Data
Orientação atualizada sobre práticas recomendadas	As práticas recomendadas foram atualizadas com novas orientações nas seguintes áreas: REL 1, REL 2, REL 4, REL 6, REL 7, REL 8, REL 10, REL 12 e REL 13. A orientação foi ampliada e esclarecida em todo o pilar. A REL10-BP02 e a REL12-BP03 tiveram suas orientações incorporadas a outras práticas recomendadas. Os recursos do pilar do foram atualizados.	6 de novembro de 2024
Orientação atualizada sobre práticas recomendadas	Pequenas atualizações nas práticas recomendadas em REL 2, 4, 5, 6, 7 e 8.	27 de junho de 2024
Orientação atualizada sobre práticas recomendadas	As práticas recomendadas foram atualizadas com novas diretrizes 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 recursos da workload, Projetar a workload para se adaptar às mudanças na demanda, Implementar	6 de dezembro de 2023

	mudanças e Testar a confiabilidade.	
Orientação atualizada sobre práticas recomendadas	As práticas recomendadas foram atualizadas com novas diretrizes nas seguintes áreas: Monitorar recursos da workload e Projetar a workload para resistir a falhas de componentes.	3 de outubro de 2023
Orientação atualizada sobre práticas recomendadas	As práticas recomendadas foram atualizadas com novas diretrizes nas seguintes áreas: Projetar a arquitetura de serviços da workload, Projetar interações em um sistema distribuído para mitigar ou resistir a falhas e Monitorar recursos da workload.	13 de julho de 2023
Atualização secundária	Remoção de linguagem não inclusiva.	13 de abril de 2023
Atualizações para o novo Framework	Atualizações nas práticas recomendadas com recomendações e adição de novas práticas recomendadas.	10 de abril de 2023
Whitepaper atualizado	Práticas recomendadas atualizadas com novas orientações para implementação.	15 de dezembro de 2022
Atualizações menores	Correção de números das figuras e alterações secundárias em geral.	17 de novembro de 2022

Whitepaper atualizado	Práticas recomendadas ampliadas e planos de melhoria adicionados.	20 de outubro de 2022
Whitepaper atualizado	Adição de duas novas práticas recomendadas ao pilar Confiabilidade nas seções Usar o isolamento de falhas para proteger sua workload e Projetar sua workload para resistir a falhas de components.	05 de maio de 2022
Whitepaper atualizado	Atualização das orientações sobre recuperação de desastres para incluir o Route 53 Application Recovery Controller. Adição de referências ao DevOps Guru. Atualização de vários links de recursos e outras alterações editoriais secundárias.	26 de outubro de 2021
Atualização secundária	Adição de informações sobre o AWS Fault Injection Service (AWS FIS).	15 de março de 2021
Atualização secundária	Atualização de texto secundária.	4 de janeiro de 2021

Whitepaper atualizado	Atualização do Apêndice A para: atualizar as seções Meta de design de disponibilidade do Amazon SQS, Amazon SNS e Amazon MQ; reordenar as linhas em tabelas para maior facilidade de pesquisa; melhorar a explicação das diferenças entre disponibilidade e recuperação de desastres e como elas contribuem para a resiliência; expandir a cobertura de arquiteturas de várias regiões (para disponibilidade) e as estratégias de varias regiões (para recuperação 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 game days.	7 de dezembro de 2020
Atualização secundária	Atualização do Apêndice A para atualizar a meta de design de disponibilidade do AWS Lambda.	27 de outubro de 2020
Atualização secundária	Atualização do Apêndice A para adicionar a meta de design de disponibilidade do AWS Global Accelerator.	24 de julho de 2020

Atualizações para o novo Framework	Atualizações substanciais e conteúdo novo/revisado, incluindo: adição da seção de práticas recomendadas de "Arquitetura da 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 princípios de design, deslocamento de Recuperação automática de falhas para antes de Testar procedimentos de recuperaç ão, atualização de diagramas e formatos para equações, remoção das seções de Serviços-chave e integraçã o de referências a serviços-chave da AWS nas práticas recomendadas.	8 de julho de 2020

Atualização secundária	Correção de link quebrado	1 de outubro de 2019
Whitepaper atualizado	Atualização do Apêndice A	1 de abril de 2019
Whitepaper atualizado	Adição de recomendações de rede do AWS Direct Connect específicas e mais metas de design de serviço	1º de setembro de 2018
Whitepaper atualizado	Adicionadas as seções de Princípios de design e Gerenciamento de limites. Links atualizados, removida a ambiguidade da terminolo gia upstream/downstream e adicionadas referências explícitas aos tópicos de Pilar de confiabilidade restantes nos cenários de disponibi lidade.	1º de junho de 2018
Whitepaper atualizado	Alterada a solução DynamoDB entre regiões para tabelas globais do DynamoDB. Adição de metas de design de serviço	1º de março de 2018
Atualizações menores	Pequena correção ao cálculo de disponibilidade para incluir a disponibilidade da aplicação	1.º de dezembro de 2017
Whitepaper atualizado	Atualização para fornecer orientação sobre designs de alta disponibilidade, incluindo conceitos, práticas recomenda das e implementações de exemplo.	1 de novembro de 2017

Publicação inicial	Publicação do pilar Confiabilidade: AWS Well Architected Framework.	1º de novembro de 2016

Avisos

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

Glossário da AWS

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