Construindo arquiteturas hexagonais em AWS

AWS Orientação prescritiva

AWS Orientação prescritiva: Construindo arquiteturas hexagonais em AWS

Table of Contents

Construindo arquiteturas hexagonais emAWS

Furkan Oruc, Dominik Goby, Darius Kunce e Michal Ploski, Amazon Web Services (AWS)

Junho de 2022 (histórico do documento)

Este guia descreve um modelo mental e uma coleção de padrões para o desenvolvimento de arquiteturas de software. Essas arquiteturas são fáceis de manter, ampliar e escalar em toda a organização à medida que a adoção de produtos cresce. Os hiperescaladores de nuvem, como o Amazon Web Services (AWS), fornecem elementos básicos para pequenas e grandes empresas inovarem e criarem novos produtos de software. O ritmo acelerado dessas novas introduções de serviços e recursos faz com que as partes interessadas da empresa esperem que suas equipes de desenvolvimento criem protótipos de novos produtos mínimos viáveis (MVPs) mais rapidamente, para que novas ideias possam ser testadas e verificadas o mais rápido possível. Muitas vezes, esses MVPs são adotados e se tornam parte do ecossistema de software corporativo. No processo de produção desses MVPs, as equipes às vezes abandonam as regras de desenvolvimento de software e as melhores práticas, como os princípios do SOLID e os testes unitários. Eles presumem que essa abordagem acelerará o desenvolvimento e reduzirá o tempo de comercialização. No entanto, se eles falharem em criar um modelo básico e uma estrutura para arquitetura de software em todos os níveis, será difícil ou mesmo impossível desenvolver novos recursos para o produto. A falta de certeza e a mudança de requisitos também podem atrasar a equipe durante o processo de desenvolvimento.

Este guia apresenta uma arquitetura de software proposta, de uma arquitetura hexagonal de baixo nível a uma decomposição arquitetônica e organizacional de alto nível, que usa design orientado por domínio (DDD) para enfrentar esses desafios. O DDD ajuda a gerenciar a complexidade dos negócios e escalar a equipe de engenharia à medida que novos recursos são desenvolvidos. Ele alinha as partes interessadas comerciais e técnicas aos problemas de negócios, chamados de domínios, usando uma linguagem onipresente. A arquitetura hexagonal é um facilitador técnico dessa abordagem em um domínio muito específico, chamado de contexto limitado. Um contexto limitado é uma subárea altamente coesa e pouco acoplada do problema de negócios. Recomendamos que você adote a arquitetura hexagonal para todos os seus projetos de software corporativo, independentemente de sua complexidade.

A arquitetura hexagonal incentiva a equipe de engenharia a resolver primeiro o problema de negócios, enquanto a arquitetura clássica em camadas desloca o foco da engenharia do domínio para resolver primeiro os problemas técnicos. Além disso, se o software segue uma arquitetura

hexagonal, é mais fácil adotar uma abordagem de desenvolvimento orientada a testes, o que reduz o ciclo de feedback que os desenvolvedores precisam para testar os requisitos de negócios. Por fim, usar comandos e manipuladores de comando é uma forma de aplicar a responsabilidade única e os princípios de abertura e fechamento do SOLID. A adesão a esses princípios produz uma base de código que os desenvolvedores e arquitetos que trabalham no projeto podem navegar e entender com facilidade, além de reduzir o risco de introduzir alterações significativas nas funcionalidades existentes.

Este guia é para arquitetos e desenvolvedores de software interessados em entender os benefícios da adoção da arquitetura hexagonal e do DDD em seus projetos de desenvolvimento de software. Ele inclui um exemplo de design de uma infraestrutura para sua aplicaçãoAWS que suporte a arquitetura hexagonal. Para ver um exemplo de implementação, consulte Estruturar um projeto Python em arquitetura hexagonal usandoAWS Lambda no site da AWS Prescriptive Guidance.

Visão geral

Design orientado por domínio (DDD)

No design orientado por domínio (DDD), um domínio é o núcleo do sistema de software. O modelo de domínio é definido primeiro, antes de você desenvolver qualquer outro módulo, e não depende de outros módulos de baixo nível. Em vez disso, módulos como bancos de dados, camada de apresentação e APIs externas dependem do domínio.

No DDD, os arquitetos decompõem a solução em contextos limitados usando a decomposição baseada na lógica de negócios em vez da decomposição técnica. Os benefícios dessa abordagem são discutidos naResultados de negócios direcionados seção.

O DDD é mais fácil de implementar quando as equipes usam arquitetura hexagonal. Na arquitetura hexagonal, o núcleo do aplicativo é o centro do aplicativo. Ele é desacoplado de outros módulos por meio de portas e adaptadores e não depende de outros módulos. Isso se alinha perfeitamente com o DDD, em que um domínio é o núcleo do aplicativo que resolve um problema de negócios. Este guia propõe uma abordagem em que você modela o núcleo da arquitetura hexagonal como o modelo de domínio de um contexto limitado. A próxima seção descreve a arquitetura hexagonal com mais detalhes.

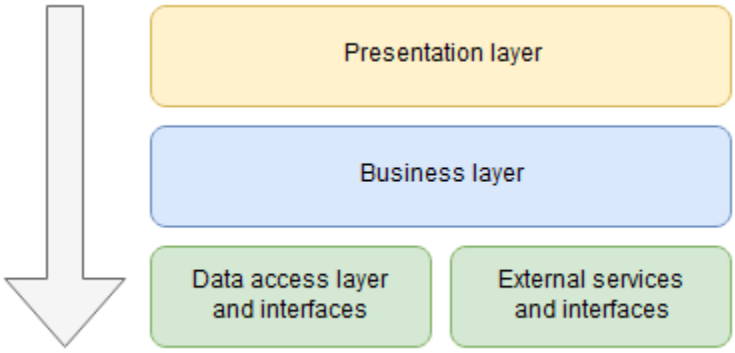
Este guia não aborda todos os aspectos do DDD, que é um tópico muito amplo. Para obter uma melhor compreensão, você pode revisar os recursos do DDD listados no site da Domain Language.

arquiteturas hexagonais

A arquitetura hexagonal, também conhecida como portas e adaptadores ou arquitetura onion, é um princípio de gerenciamento da inversão de dependência em projetos de software. A arquitetura hexagonal promove um forte foco na lógica de negócios do domínio central ao desenvolver software e trata os pontos de integração externos como secundários. A arquitetura hexagonal ajuda os engenheiros de software a adotarem boas práticas, como o desenvolvimento orientado a testes (TDD), que, por sua vez, promove a evolução da arquitetura e ajuda a gerenciar domínios complexos a longo prazo.

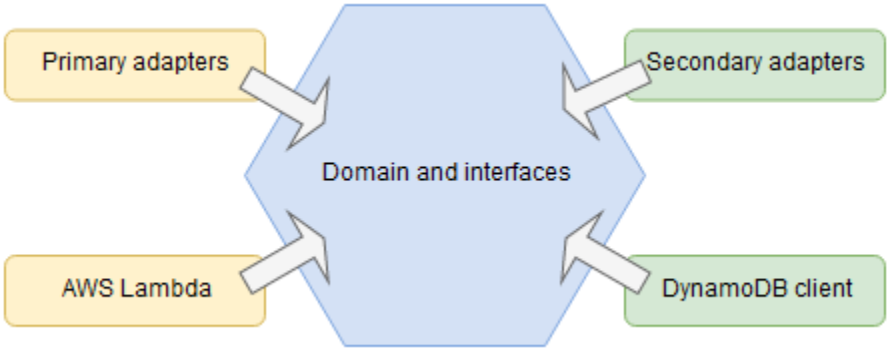
Vamos comparar a arquitetura hexagonal com a arquitetura clássica em camadas, que é a escolha mais popular para modelar projetos de software estruturado. Há diferenças sutis, mas poderosas, entre as duas abordagens.

Na arquitetura em camadas, os projetos de software são estruturados em camadas, que representam preocupações amplas, como lógica de negócios ou lógica de apresentação. Essa arquitetura usa uma hierarquia de dependências, em que as camadas superiores dependem das camadas abaixo delas, mas não o contrário. No diagrama a seguir, a camada de apresentação é responsável pelas interações do usuário, portanto, inclui a interface do usuário, APIs, interfaces de linha de comando e componentes similares. A camada de apresentação depende da camada de negócios, que implementa a lógica do domínio. A camada de negócios, por sua vez, depende da camada de acesso aos dados e de vários serviços externos.



A principal desvantagem dessa configuração é a estrutura de dependências. Por exemplo, se o modelo de armazenamento de dados no banco de dados mudar, isso afetará a interface de acesso aos dados. Qualquer alteração no modelo de dados também afeta a camada de negócios, que depende da interface de acesso aos dados. Como resultado, os engenheiros de software não podem fazer nenhuma alteração na infraestrutura sem afetar a lógica do domínio. Isso, por sua vez, aumenta a probabilidade de erros de regressão.

A arquitetura hexagonal define as relações de dependência de uma forma diferente, conforme ilustrado no diagrama a seguir. Ele concentra a tomada de decisões em torno da lógica de negócios do domínio, que define todas as interfaces. Os componentes externos interagem com a lógica de negócios por meio de interfaces chamadas portas. Portas são abstrações que definem as interações do domínio com o mundo externo. Cada componente da infraestrutura deve implementar essas portas, para que as alterações nesses componentes não afetem mais a lógica do domínio principal.



Os componentes circundantes são chamados de adaptadores. Um adaptador é um proxy entre o mundo externo e o interno e implementa uma porta definida no domínio. Os adaptadores podem ser categorizados em dois grupos: primário e secundário. Os adaptadores primários são os pontos de entrada para o componente de software. Eles permitem que atores, usuários e serviços externos interajam com a lógica central. AWS Lambdaé um bom exemplo de adaptador primário. Ele se integra a váriosAWS serviços que podem invocar as funções do Lambda como pontos de entrada. Os adaptadores secundários são pacotes de bibliotecas de serviços externos que lidam com as comunicações com o mundo externo. Um bom exemplo de adaptador secundário é um cliente Amazon DynamoDB para acesso a dados.

Resultados de negócios direcionados

A arquitetura hexagonal discutida neste guia ajuda você a atingir os seguintes objetivos:

- Reduza o tempo de comercialização melhorando o ciclo de desenvolvimento
- Melhore a qualidade do software
- Adapte-se mais facilmente às mudanças

Esses processos são discutidos em detalhes nas seções a seguir.

Melhorando o ciclo de desenvolvimento

O desenvolvimento de software para a nuvem apresenta novos desafios para os engenheiros de software, porque é muito difícil replicar o ambiente de execução localmente na máquina de desenvolvimento. Uma forma simples de validar o software é implantá-lo na nuvem e testá-lo lá. No entanto, essa abordagem envolve um longo ciclo de feedback, especialmente quando a arquitetura do software contém várias implantações sem servidor. Melhorar esse ciclo de feedback reduz o tempo de desenvolvimento de recursos, o que reduz significativamente o tempo de comercialização.

Testando na nuvem

Testar diretamente na nuvem é a única maneira de garantir que seus componentes arquitetônicos, como gateways no Amazon API Gateway,AWS Lambda funções, tabelas do Amazon DynamoDB e permissõesAWS Identity and Access Management (IAM), estejam configurados corretamente. Também pode ser a única maneira confiável de testar integrações de componentes. Embora algunsAWS serviços (como o DynamoDB) possam ser implantados localmente, a maioria deles não pode ser replicada em uma configuração local. Ao mesmo tempo, ferramentas de terceiros, como o Moto e LocalStackque simulamAWS serviços para fins de teste, podem não refletir os contratos reais de API de serviços com precisão, ou o número de recursos pode ser limitado.

No entanto, a parte mais complexa do software corporativo está na lógica de negócios, não na arquitetura de nuvem. A arquitetura muda com menos frequência do que o domínio, o que deve acomodar os novos requisitos de negócios. Assim, testar a lógica de negócios na nuvem se torna um processo intenso de fazer uma alteração no código, iniciar uma implantação, esperar que o ambiente esteja pronto e validar a alteração. Se uma implantação levar apenas 5 minutos, fazer e testar 10 alterações na lógica de negócios levará uma hora ou mais. Se a lógica de negócios for mais complexa, os testes podem exigir dias de espera pela conclusão das implantações. Se você tiver vários recursos e engenheiros na equipe, o período prolongado rapidamente se tornará perceptível para a empresa.

Testando localmente

Uma arquitetura hexagonal ajuda os desenvolvedores a se concentrarem no domínio em vez dos detalhes técnicos da infraestrutura. Essa abordagem usa testes locais (as ferramentas de teste unitário na estrutura de desenvolvimento escolhida) para cobrir os requisitos de lógica do domínio. Você não precisará perder tempo resolvendo problemas de integração técnica ou implantar seu

software na nuvem para testar a lógica de negócios. Você pode executar testes unitários localmente e reduzir o ciclo de feedback de minutos para segundos. Se uma implantação levar 5 minutos, mas os testes de unidade forem concluídos em 5 segundos, isso significa uma redução significativa no tempo necessário para detectar erros. ATestando primeiro a lógica de negócios seção mais adiante neste guia aborda essa abordagem com mais detalhes.

Paralelização do desenvolvimento

A abordagem de arquitetura hexagonal permite que as equipes de desenvolvimento paralelizem os esforços de desenvolvimento. Os desenvolvedores podem projetar e implementar diferentes componentes do serviço individualmente. Essa paralelização é possível por meio do isolamento de cada componente e das interfaces definidas entre cada componente.

Tempo de comercialização do produto

O teste unitário local melhora o ciclo de feedback do desenvolvimento e reduz o tempo de comercialização de novos produtos ou recursos, especialmente quando eles contêm uma lógica de negócios complexa, conforme explicado anteriormente. Além disso, o aumento da cobertura do código por meio de testes unitários reduz significativamente o risco de introdução de bugs de regressão quando você atualiza ou refatora a base de código. A cobertura de testes unitários também permite que você refatore continuamente a base de código para mantê-la bem organizada, o que acelera o processo de integração de novos engenheiros. Isso é discutido mais adiante naQualidade desde o design seção. Por fim, se a lógica de negócios for bem isolada e testada, ela permitirá que os desenvolvedores se adaptem rapidamente às mudanças nos requisitos funcionais e não funcionais. Isso é explicado mais detalhadamente naAdaptando-se à mudança seção.

Qualidade desde o design

A adoção de uma arquitetura hexagonal ajuda a promover a qualidade da sua base de código desde o início do projeto. É importante criar um processo que ajude você a atender aos requisitos de qualidade esperados desde o início, sem atrasar o processo de desenvolvimento.

Mudanças localizadas e maior legibilidade

O uso da abordagem de arquitetura hexagonal permite que os desenvolvedores alterem o código em uma classe ou componente sem afetar outras classes ou componentes. Esse design promove a coesão dos componentes desenvolvidos. Ao desacoplar o domínio dos adaptadores e usar interfaces conhecidas, você pode aumentar a legibilidade do código. Fica mais fácil identificar problemas e casos secundários.

Essa abordagem também facilita a revisão do código durante o desenvolvimento e limita a introdução de alterações não detectadas ou dívidas técnicas.

Testando primeiro a lógica de negócios

Os testes locais podem ser realizados com a introdução end-to-end, integração e testes unitários no projeto. nd-to-end Os testes eletrônicos abrangem todo o ciclo de vida das solicitações recebidas. Eles normalmente invocam um ponto de entrada do aplicativo e testam para ver se ele atendeu aos requisitos comerciais. Cada projeto de software deve ter pelo menos um cenário de teste que use entradas conhecidas e produza as saídas esperadas. No entanto, adicionar mais cenários secundários pode ser complexo, pois cada teste deve ser configurado para enviar uma solicitação por meio de um ponto de entrada (por exemplo, por meio de uma API REST ou filas), passar por todos os pontos de integração que a ação comercial exige e, em seguida, afirmar o resultado. Configurar o ambiente para o cenário de teste e afirmar os resultados pode levar muito tempo para os desenvolvedores.

Na arquitetura hexagonal, você testa a lógica de negócios isoladamente e usa testes de integração para testar adaptadores secundários. Você pode usar adaptadores falsos ou falsos em seus testes de lógica de negócios. Você também pode combinar os testes para casos de uso comercial com testes unitários para seu modelo de domínio para manter a alta cobertura com baixo acoplamento. Como boa prática, os testes de integração não devem validar a lógica de negócios. Em vez disso, eles devem verificar se o adaptador secundário chama os serviços externos corretamente.

Idealmente, você pode usar o desenvolvimento orientado por testes (TDD) e começar a definir entidades de domínio ou casos de uso comercial com testes adequados logo no início do desenvolvimento. Escrever os testes primeiro ajuda a criar implementações simuladas das interfaces exigidas pelo domínio. Quando os testes forem bem-sucedidos e as regras de lógica de domínio forem satisfeitas, você poderá implementar os adaptadores reais e implantar o software no ambiente de teste. Nesse ponto, sua implementação da lógica de domínio pode não ser ideal. Em seguida, você pode trabalhar na refatoração da arquitetura existente para evoluí-la introduzindo padrões de design ou reorganizando o código em geral. Ao usar essa abordagem, você pode evitar a introdução de bugs de regressão e melhorar a arquitetura à medida que o projeto cresce. Ao combinar essa abordagem com os testes automáticos que você executa em seu processo de integração contínua, você pode diminuir o número de possíveis bugs antes que eles entrem em produção.

Se você usar implantações sem servidor, poderá provisionar rapidamente uma instância do aplicativo em suaAWS conta para integração e end-to-end testes manuais. Após essas etapas de implementação, recomendamos que você automatize os testes com cada nova alteração enviada ao repositório.

Capacidade de manutenção

A capacidade de manutenção se refere à operação e ao monitoramento de um aplicativo para garantir que ele atenda a todos os requisitos e minimizar a probabilidade de uma falha no sistema. Para tornar o sistema operável, você deve adaptá-lo aos requisitos operacionais ou de tráfego future. Você também deve garantir que ele esteja disponível e seja fácil de implantar, com mínimo ou nenhum impacto nos clientes.

Para entender o estado atual e histórico do seu sistema, você deve torná-lo observável. Você pode fazer isso fornecendo métricas, registros e rastreamentos específicos que os operadores podem usar para garantir que o sistema funcione conforme o esperado e para rastrear bugs. Esses mecanismos também devem permitir que os operadores conduzam a análise da causa raiz sem precisar fazer login na máquina e ler o código.

Uma arquitetura hexagonal visa aumentar a capacidade de manutenção de seus aplicativos web para que seu código exija menos trabalho em geral. Ao separar módulos, localizar alterações e dissociar a lógica de negócios do aplicativo da implementação do adaptador, você pode produzir métricas e registros que ajudam os operadores a obter uma compreensão profunda do sistema e o escopo das alterações específicas feitas nos adaptadores primários ou secundários.

Adaptando-se à mudança

Os sistemas de software tendem a ficar complicados. Uma razão para isso pode ser mudanças frequentes nos requisitos de negócios e pouco tempo para adaptar adequadamente a arquitetura do software. Outro motivo pode ser o investimento insuficiente para configurar a arquitetura de software no início do projeto para se adaptar às mudanças frequentes. Seja qual for o motivo, um sistema de software pode ficar complicado a ponto de ser quase impossível fazer uma mudança. Portanto, é importante criar uma arquitetura de software sustentável desde o início do projeto. Uma boa arquitetura de software pode se adaptar facilmente às mudanças.

Esta seção explica como projetar aplicativos sustentáveis usando uma arquitetura hexagonal que se adapta facilmente aos requisitos não funcionais ou comerciais.

Adaptação aos novos requisitos não funcionais usando portas e adaptadores

Como núcleo do aplicativo, o modelo de domínio define as ações que são exigidas do mundo exterior para atender aos requisitos de negócios. Essas ações são definidas por meio de abstrações, chamadas de portas. Essas portas são implementadas por adaptadores separados. Cada adaptador é responsável por uma interação com outro sistema. Por exemplo, você pode ter um adaptador para o repositório do banco de dados e outro adaptador para interagir com uma API de terceiros. O domínio não está ciente da implementação do adaptador, portanto, é fácil substituir um adaptador por outro. Por exemplo, o aplicativo pode mudar de um banco de dados SQL para um banco de dados NoSQL. Nesse caso, um novo adaptador precisa ser desenvolvido para implementar as portas definidas pelo modelo de domínio. O domínio não tem dependências no repositório do banco de dados e usa abstrações para interagir, portanto, não seria necessário alterar nada no modelo de domínio. Portanto, a arquitetura hexagonal se adapta aos requisitos não funcionais com facilidade.

Adaptação aos novos requisitos de negócios usando comandos e manipuladores de comandos

Na arquitetura clássica em camadas, o domínio depende da camada de persistência. Se você quiser alterar o domínio, também precisará alterar a camada de persistência. Em comparação, na arquitetura hexagonal, o domínio não depende de outros módulos no software. O domínio é o núcleo do aplicativo, e todos os outros módulos (portas e adaptadores) dependem do modelo de domínio.

O domínio usa o princípio de inversão de dependência para se comunicar com o mundo externo por meio de portas. O benefício da inversão de dependência é que você pode alterar o modelo de domínio livremente sem ter medo de quebrar outras partes do código. Como o modelo de domínio reflete o problema de negócios que você está tentando resolver, atualizar o modelo de domínio para se adaptar às mudanças nos requisitos de negócios não é um problema.

Quando você desenvolve software, a separação de preocupações é um princípio importante a ser seguido. Para conseguir essa separação, você pode usar um padrão de comando ligeiramente modificado. Esse é um padrão de design comportamental no qual todas as informações necessárias para concluir uma operação são encapsuladas em um objeto de comando. Essas operações são então processadas por manipuladores de comando. Os manipuladores de comando são métodos que recebem um comando, alteram o estado do domínio e, em seguida, retornam uma resposta ao chamador. Você pode usar clientes diferentes, como APIs síncronas ou filas assíncronas, para executar comandos. Recomendamos que você use comandos e manipuladores de comandos para cada operação no domínio. Seguindo essa abordagem, você pode adicionar novos recursos introduzindo novos comandos e manipuladores de comandos, sem alterar sua lógica comercial existente. Assim, o uso de um padrão de comando facilita a adaptação aos novos requisitos de negócios.

Desacoplamento de componentes usando a fachada de serviço ou o padrão CQRS

Na arquitetura hexagonal, os adaptadores primários são responsáveis por acoplar vagamente as solicitações de leitura e gravação recebidas dos clientes ao domínio. Há duas maneiras de obter esse acoplamento solto: usando um padrão de fachada de serviço ou usando o padrão de segregação de responsabilidade por consulta (CQRS) de comando.

O padrão da fachada de serviço fornece uma interface frontal para atender clientes, como a camada de apresentação ou um microsserviço. Uma fachada de serviços oferece aos clientes várias operações de leitura e gravação. É responsável por transferir as solicitações recebidas para o domínio e mapear a resposta recebida do domínio para os clientes. Usar uma fachada de serviço é fácil para microsserviços que têm uma única responsabilidade com várias operações. No entanto, ao usar a fachada de serviços, é mais difícil seguir a responsabilidade única e os princípios abertos e fechados. O princípio de responsabilidade única afirma que cada módulo deve ser responsável por apenas uma única funcionalidade do software. O princípio aberto-fechado afirma que o código deve ser aberto para extensão e fechado para modificação. À medida que a fachada de serviços se estende, todas as operações são coletadas em uma interface, mais dependências são encapsuladas

nela e mais desenvolvedores começam a modificar a mesma fachada. Portanto, recomendamos usar uma fachada de serviço somente se estiver claro que o serviço não se estenderia muito durante o desenvolvimento.

Outra forma de implementar adaptadores primários na arquitetura hexagonal é usar o padrão CQRS, que separa as operações de leitura e gravação usando consultas e comandos. Conforme explicado anteriormente, os comandos são objetos que contêm todas as informações necessárias para alterar o estado do domínio. Os comandos são executados por métodos de manipulador de comandos. As consultas, por outro lado, não alteram o estado do sistema. Seu único objetivo é devolver dados aos clientes. No padrão CQRS, comandos e consultas são implementados em módulos separados. Isso é especialmente vantajoso para projetos que seguem uma arquitetura orientada por eventos, porque um comando pode ser implementado como um evento processado de forma assíncrona, enquanto uma consulta pode ser executada de forma síncrona usando uma API. Uma consulta também pode usar um banco de dados diferente, otimizado para ela. A desvantagem do padrão CQRS é que ele leva mais tempo para ser implementado do que uma fachada de serviço. Recomendamos usar o padrão CQRS para projetos que você planeja escalar e manter a longo prazo. Comandos e consultas fornecem um mecanismo eficaz para aplicar o princípio de responsabilidade única e desenvolver software fracamente acoplado, especialmente em projetos de grande escala.

O CQRS tem grandes benefícios a longo prazo, mas requer um investimento inicial. Por esse motivo, recomendamos avaliar cuidadosamente o projeto antes de decidir usar o padrão CQRS. No entanto, você pode estruturar seu aplicativo usando comandos e manipuladores de comandos desde o início, sem separar as operações de leitura/gravação. Isso o ajudará a refatorar facilmente seu projeto para o CQRS se você decidir adotar essa abordagem posteriormente.

Escalabilidade organizacional

Uma combinação de arquitetura hexagonal, design orientado por domínio e (opcionalmente) CQRS permite que sua organização escale rapidamente seu produto. De acordo com a Lei de Conway, as arquiteturas de software tendem a evoluir para refletir as estruturas de comunicação de uma empresa. Historicamente, essa observação teve conotações negativas, porque grandes organizações geralmente estruturam suas equipes com base em conhecimentos técnicos, como banco de dados, barramento de serviços corporativos e assim por diante. O problema com essa abordagem é que o desenvolvimento de produtos e recursos sempre envolve questões transversais, como segurança e escalabilidade, que exigem comunicação constante entre as equipes. A estruturação de equipes com base em características técnicas cria silos desnecessários na organização, o que resulta em comunicações deficientes, falta de propriedade e perda de visão

do panorama geral. Eventualmente, esses problemas organizacionais se refletem na arquitetura do software.

A Manobra Inversa de Conway, por outro lado, define a estrutura organizacional com base em domínios que promovem a arquitetura do software. Por exemplo, as equipes multifuncionais são responsáveis por um conjunto específico de contextos limitados, que são identificados usando DDD e evasão de eventos. Esses contextos limitados podem refletir características muito específicas do produto. Por exemplo, a equipe da conta pode ser responsável pelo contexto de pagamento. Cada novo recurso é atribuído a uma nova equipe que tem responsabilidades altamente coesas e vagamente acopladas, para que eles possam se concentrar apenas na entrega desse recurso e reduzir o tempo de lançamento no mercado. As equipes podem ser escaladas de acordo com a complexidade dos recursos, portanto, recursos complexos podem ser atribuídos a mais engenheiros.

Práticas recomendadas

Modele o domínio de negócios

Volte do domínio comercial ao design do software para garantir que o software que você está escrevendo atenda às necessidades comerciais.

Use metodologias de design orientado por domínio (DDD), como a invasão de eventos, para modelar o domínio de negócios. O evento Storming tem um formato de workshop flexível. Durante o workshop, especialistas em domínio e software exploram a complexidade do domínio de negócios de forma colaborativa. Os especialistas em software usam os resultados do workshop para iniciar o processo de design e desenvolvimento de componentes de software.

Escreva e execute testes desde o início

Use o desenvolvimento orientado por testes (TDD) para verificar a exatidão do software que você está desenvolvendo. O TDD funciona melhor em um nível de teste unitário. O desenvolvedor projeta um componente de software escrevendo primeiro um teste, que invoca esse componente. Esse componente não tem implementação no início, portanto, o teste falha. Como próxima etapa, o desenvolvedor implementa a funcionalidade do componente, usando dispositivos de teste com objetos simulados para simular o comportamento de dependências externas ou portas. Quando o teste for bem-sucedido, o desenvolvedor poderá continuar implementando adaptadores reais. Essa abordagem melhora a qualidade do software e resulta em um código mais legível, porque os desenvolvedores entendem como os usuários usariam os componentes. A arquitetura hexagonal suporta a metodologia TDD separando o núcleo do aplicativo. Os desenvolvedores escrevem testes unitários que se concentram no comportamento central do domínio. Eles não precisam escrever adaptadores complexos para executar seus testes; em vez disso, eles podem usar objetos e acessórios simulados simples.

Use o desenvolvimento orientado por comportamento (BDD) para garantir a end-to-end aceitação em um nível de recurso. No BDD, os desenvolvedores definem cenários para recursos e os verificam com as partes interessadas da empresa. Os testes de BDD usam o máximo de linguagem natural possível para conseguir isso. A arquitetura hexagonal suporta a metodologia BDD com seu conceito de adaptadores primários e secundários. Os desenvolvedores podem criar adaptadores primários e secundários que podem ser executados localmente sem chamar serviços externos. Eles configuram o conjunto de testes do BDD para usar o adaptador primário local para executar o aplicativo.

Execute automaticamente cada teste no pipeline de integração contínua para avaliar constantemente a qualidade do sistema.

Defina o comportamento do domínio

Decomponha o domínio em entidades, objetos de valor e agregados (leia sobre a implementação do design orientado por domínio) e defina seu comportamento. Implemente o comportamento do domínio para que os testes que foram escritos no início do projeto sejam bem-sucedidos. Defina comandos que invocam o comportamento dos objetos de domínio. Defina eventos que os objetos de domínio emitem após concluírem um comportamento.

Defina interfaces que os adaptadores possam usar para interagir com o domínio.

Automatize os testes e a implantação

Depois de uma prova inicial de conceito, recomendamos que você invista tempo implementando DevOps práticas. Por exemplo, pipelines de integração contínua e entrega contínua (CI/CD) e ambientes de teste dinâmico ajudam você a manter a qualidade do código e evitar erros durante a implantação.

- Execute seus testes de unidade dentro do processo de CI e teste seu código antes que ele seja mesclado.
- Crie um processo de CD para implantar seu aplicativo em um ambiente estático de desenvolvimento/teste ou em ambientes criados dinamicamente que suportem integração e end-to-end testes automáticos.
- Automatize o processo de implantação para ambientes dedicados.

Dimensione seu produto usando microsserviços e CQRS

Se seu produto for bem-sucedido, escale seu produto decompondo seu projeto de software em microsserviços. Utilize a portabilidade que a arquitetura hexagonal oferece para melhorar o desempenho. Divida os serviços de consulta e os manipuladores de comandos em APIs síncronas e assíncronas separadas. Considere adotar o padrão de segregação de responsabilidade por consultas de comando (CQRS) e a arquitetura orientada por eventos.

Se você receber muitas solicitações de novos recursos, considere escalar sua organização com base nos padrões de DDD. Estruture suas equipes de forma que elas possuam um ou mais recursos

como contextos limitados, conforme discutido anteriormente naEscalabilidade organizacional seção. Essas equipes podem então implementar a lógica de negócios usando a arquitetura hexagonal.

Projete uma estrutura de projeto que mapeie conceitos de arquitetura hexagonal

A infraestrutura como código (IaC) é uma prática amplamente adotada no desenvolvimento em nuvem. Ele permite que você defina e mantenha seus recursos de infraestrutura (como redes, balanceadores de carga, máquinas virtuais e gateways) como código-fonte. Dessa forma, você pode acompanhar todas as alterações em sua arquitetura usando um sistema de controle de versão. Além disso, você pode criar e mover a infraestrutura facilmente para fins de teste. Recomendamos que você mantenha o código do aplicativo e o código de infraestrutura no mesmo repositório ao desenvolver seus aplicativos em nuvem. Essa abordagem facilita a manutenção da infraestrutura para sua aplicação.

Recomendamos que você divida seu aplicativo em três pastas ou projetos que correspondam aos conceitos de arquitetura hexagonal:`entrypoints` (adaptadores primários),`domain` (domínio e interfaces) `eadapters` (adaptadores secundários).

A estrutura do projeto a seguir fornece um exemplo dessa abordagem ao projetar uma API noAWS. O projeto mantém o código do aplicativo (`app`) e o código de infraestrutura (`infra`) no mesmo repositório, conforme recomendado anteriormente.

```
app/  # application code
|--- adapters/  # implementation of the ports defined in the domain
     |--- tests/  # adapter unit tests
|--- entrypoints/  # primary adapters, entry points
     |--- api/  # api entry point
          |--- model/  # api model
          |--- tests/  # end to end api tests
|--- domain/  # domain to implement business logic using hexagonal architecture
     |--- command_handlers/  # handlers used to run commands on the domain
     |--- commands/  # commands on the domain
     |--- events/  # events emitted by the domain
     |--- exceptions/  # exceptions defined on the domain
     |--- model/  # domain model
     |--- ports/  # abstractions used for external communication
     |--- tests/  # domain tests
infra/  # infrastructure code
```

Conforme discutido anteriormente, o domínio é o núcleo do aplicativo e não depende de nenhum outro módulo. Recomendamos que você estruture a`domain` pasta para incluir as seguintes subpastas:

- `command handlers`contém os métodos ou classes que executam comandos no domínio.
- `commands`contém os objetos de comando que definem as informações necessárias para realizar uma operação no domínio.
- `events`contém os eventos que são emitidos por meio do domínio e, em seguida, roteados para outros microsserviços.
- `exceptions`contém os erros conhecidos definidos no domínio.
- `model`contém entidades de domínio, objetos de valor e serviços de domínio.
- `ports`contém as abstrações por meio das quais o domínio se comunica com bancos de dados, APIs ou outros componentes externos.
- `tests`contém os métodos de teste (como testes de lógica de negócios) que são executados no domínio.

Os adaptadores primários são os pontos de entrada para o aplicativo, conforme representado pela`entrypoints` pasta. Este exemplo usa a`api` pasta como adaptador primário. Essa pasta contém uma API`model`, que define a interface que o adaptador primário requer para se comunicar com os clientes. A`tests` pasta contém end-to-end testes para a API. Esses são testes superficiais que validam se os componentes do aplicativo estão integrados e funcionam em harmonia.

Os adaptadores secundários, conforme representados pela`adapters` pasta, implementam as integrações externas exigidas pelas portas do domínio. Um repositório de banco de dados é um ótimo exemplo de adaptador secundário. Quando o sistema de banco de dados muda, você pode escrever um novo adaptador usando a implementação definida pelo domínio. Não há necessidade de alterar o domínio ou a lógica comercial. A`tests` subpasta contém testes de integração externa para cada adaptador.

Exemplos de infraestrutura emAWS

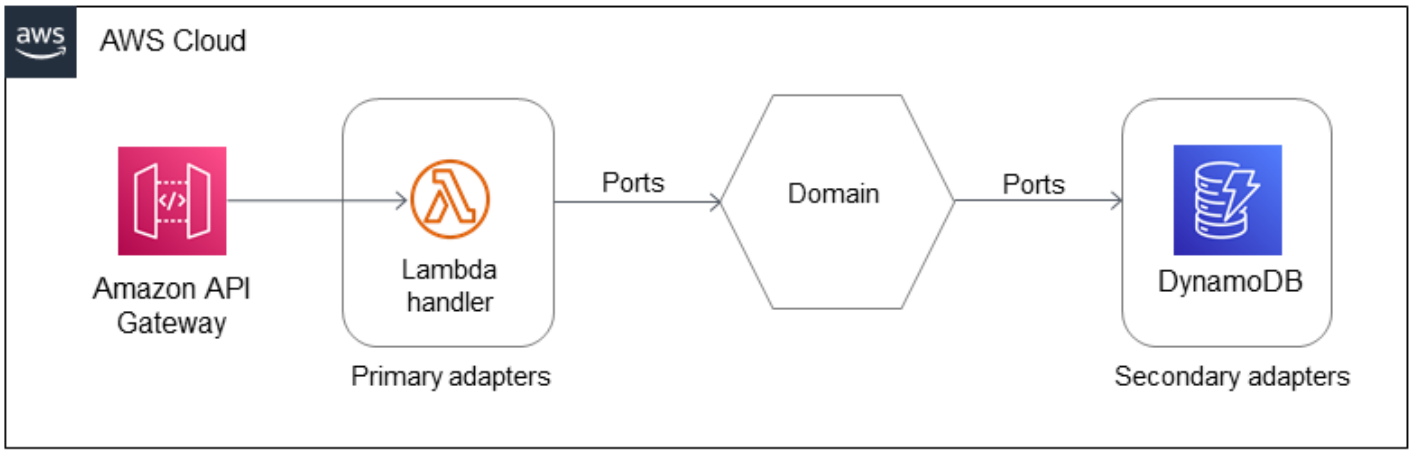
Esta seção fornece exemplos para projetar uma infraestrutura para seu aplicativo naAWS qual você pode usar para implementar uma arquitetura hexagonal. Recomendamos que você comece com uma arquitetura simples para criar um produto mínimo viável (MVP). A maioria dos microsserviços precisa de um único ponto de entrada para lidar com as solicitações do cliente, uma camada de computação para executar o código e uma camada de persistência para armazenar dados. OsAWS serviços a seguir são ótimos candidatos para uso como clientes, adaptadores primários e adaptadores secundários em arquitetura hexagonal:

- Clientes: Amazon API Gateway, Amazon Simple Queue Service (Amazon SQS), Elastic Load Balancing, Amazon EventBridge
- Adaptadores primários:AWS Lambda Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Compute Service (Amazon EC2), Amazon Elastic Compute Service (Amazon EC2), Amazon Elastic Compute Cloud (Amazon EC2)
- Adaptadores secundários: Amazon DynamoDB, Amazon Relational Database Service (Amazon RDS), Amazon Aurora Amazon API Gateway, Amazon SQS, Amazon Elastic Load Balancing RDS EventBridge, Amazon RDS Simple Notification Service (Amazon SNS) RDS

As seções a seguir analisam esses serviços mais detalhadamente.

Comece de forma simples

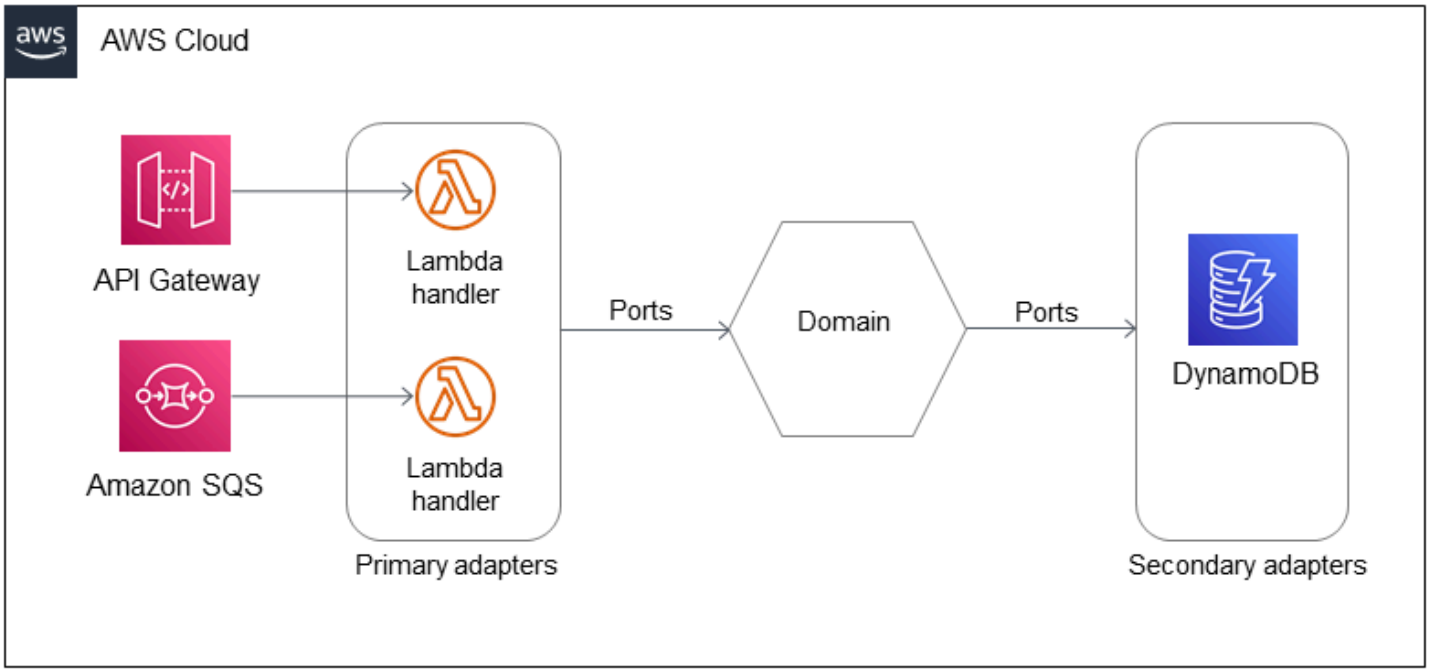
Recomendamos que você comece de forma simples ao arquitetar um aplicativo usando arquitetura hexagonal. Neste exemplo, o API Gateway é usado como cliente (API REST), o Lambda é usado como adaptador primário (computação) e o DynamoDB é usado como adaptador secundário (persistência). O cliente do gateway chama o ponto de entrada, que, nesse caso, é um manipulador Lambda.



Essa arquitetura é totalmente sem servidor e oferece ao arquiteto um bom ponto de partida. Recomendamos que você use o padrão de comando no domínio porque ele facilita a manutenção do código e se adapta aos novos requisitos comerciais e não funcionais. Essa arquitetura pode ser suficiente para criar microsserviços simples com algumas operações.

Aplique o padrão CQRS

Recomendamos que você mude para o padrão CQRS se o número de operações no domínio for escalar. Você pode aplicar o padrão CQRS como uma arquitetura totalmente sem servidor noAWS usando o exemplo a seguir.

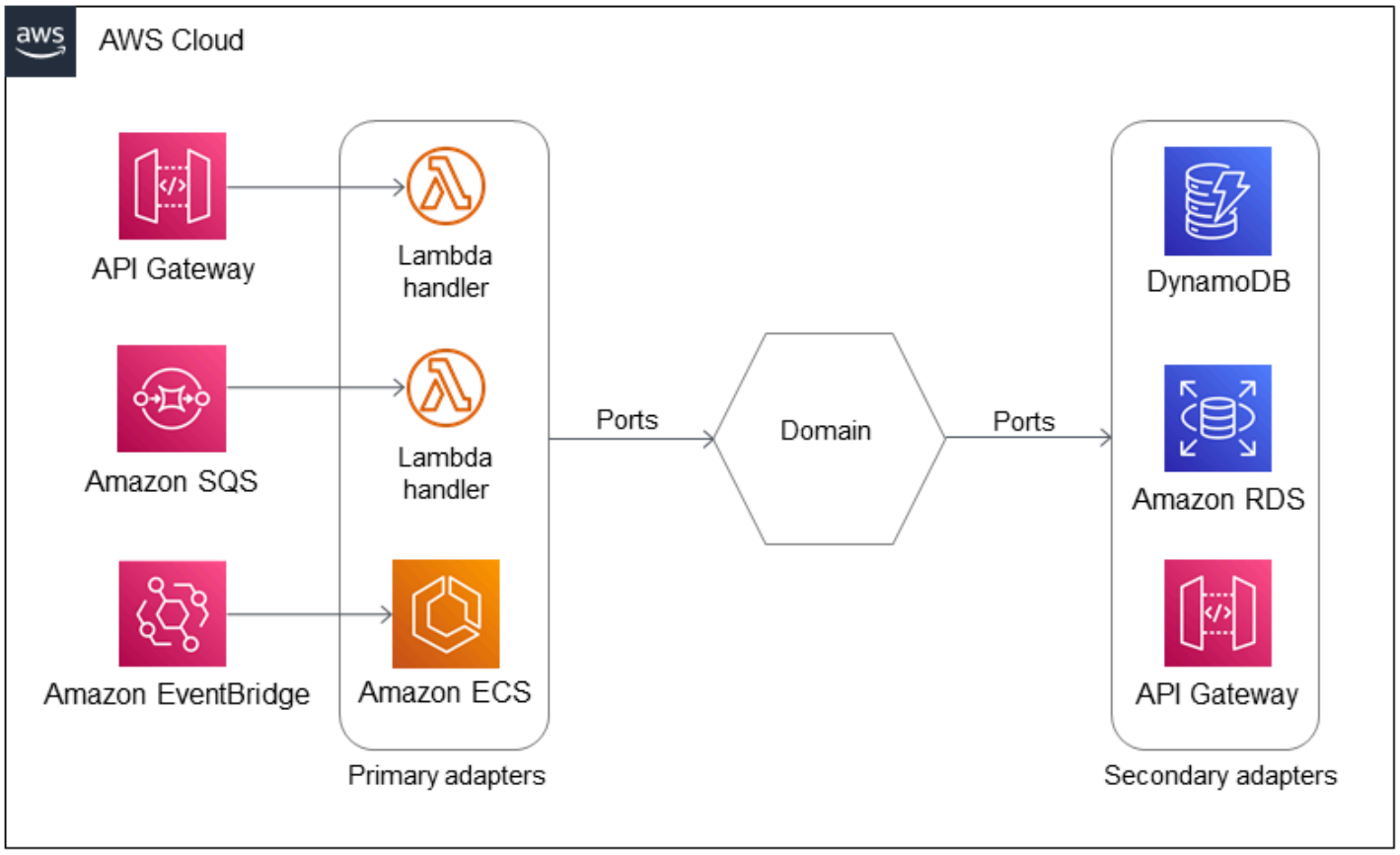


Este exemplo usa dois manipuladores do Lambda, um para consultas e outro para comandos. As consultas são executadas de forma síncrona usando um gateway de API como cliente. Os comandos são executados de forma assíncrona usando o Amazon SQS como cliente.

Essa arquitetura inclui vários clientes (API Gateway e Amazon SQS) e vários adaptadores primários (Lambda), que são chamados por seus pontos de entrada correspondentes (manipuladores do Lambda). Todos os componentes pertencem ao mesmo contexto limitado, portanto, estão dentro do mesmo domínio.

Desenvolva a arquitetura adicionando contêineres, um banco de dados relacional e uma API externa

Os contêineres são uma boa opção para tarefas de longa duração. Talvez você também queira usar um banco de dados relacional se tiver um esquema de dados predefinido e quiser se beneficiar do poder da linguagem SQL. Além disso, o domínio precisaria se comunicar com APIs externas. Você pode desenvolver o exemplo da arquitetura para dar suporte a esses requisitos, conforme mostrado no diagrama a seguir.

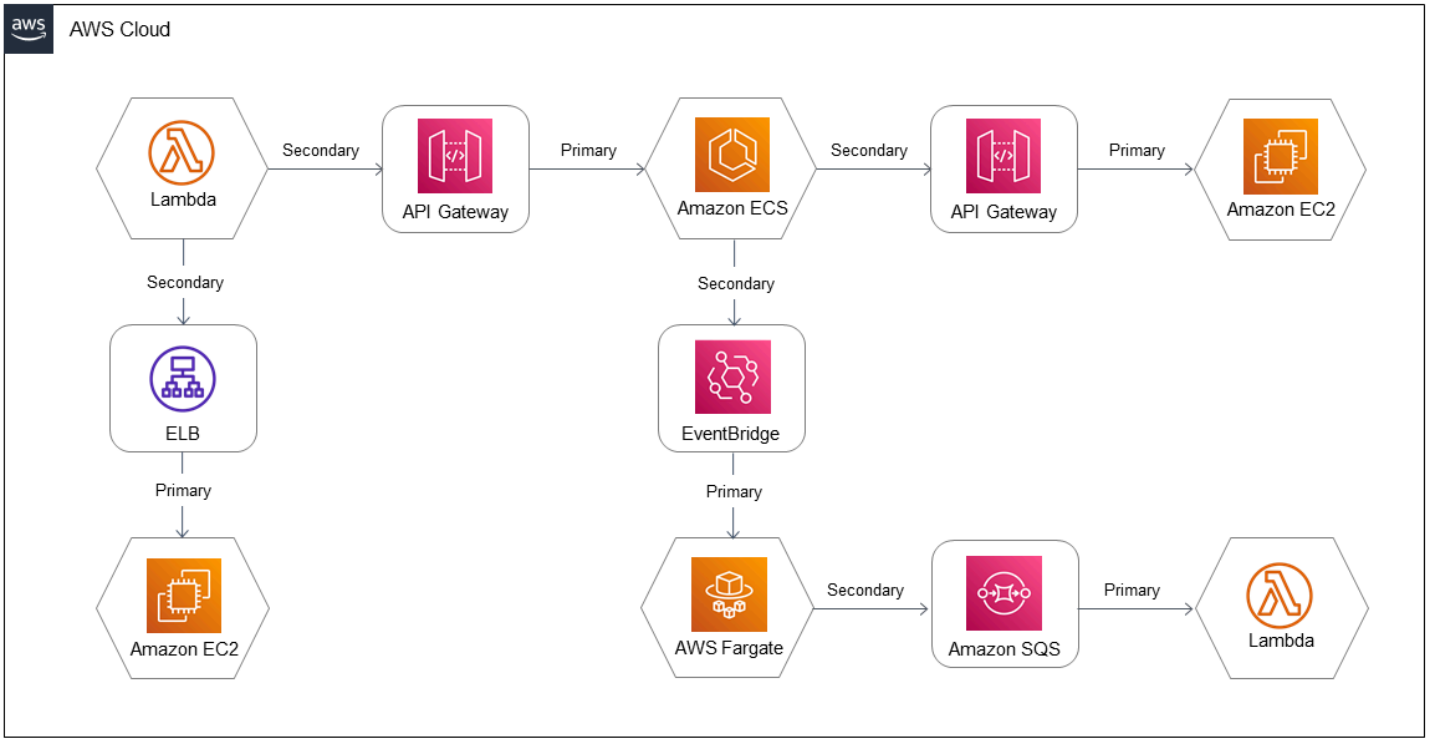


Este exemplo usa o Amazon ECS como adaptador principal para iniciar tarefas de longa duração no domínio. A Amazon EventBridge (cliente) inicia uma tarefa do Amazon ECS (ponto de entrada) quando acontece um evento específico. A arquitetura inclui o Amazon RDS como outro adaptador secundário para armazenar dados relacionais. Ele também adiciona outro gateway de API como adaptador secundário para invocar uma chamada de API externa. Como resultado, a arquitetura usa vários adaptadores primários e secundários que dependem de diferentes camadas de computação subjacentes em um domínio de negócios.

O domínio está sempre vagamente acoplado a todos os adaptadores primários e secundários por meio de abstrações chamadas portas. O domínio define o que ele exige do mundo externo usando portas. Como é responsabilidade do adaptador implementar a porta, a troca de um adaptador para outro não afeta o domínio. Por exemplo, você pode mudar do Amazon DynamoDB para o Amazon RDS criando um novo adaptador, sem afetar o domínio.

Adicionar mais domínios (diminuir o zoom)

A arquitetura hexagonal se alinha bem com os princípios de uma arquitetura de microsserviços. Os exemplos de arquitetura mostrados até agora continham um único domínio (ou contexto limitado). Os aplicativos normalmente incluem vários domínios, que precisam se comunicar por meio de adaptadores primários e secundários. Cada domínio representa um microsserviço e está vagamente acoplado a outros domínios.



Nessa arquitetura, cada domínio usa um conjunto diferente de ambientes de computação. (Cada domínio também pode ter vários ambientes de computação, como no exemplo anterior.) Cada domínio define suas interfaces necessárias para se comunicar com outros domínios por meio de portas. As portas são implementadas usando adaptadores primários e secundários. Dessa forma, o domínio não será afetado se houver uma alteração no adaptador. Além disso, os domínios são dissociados uns dos outros.

No exemplo de arquitetura mostrado no diagrama anterior, Lambda, Amazon EC2, Amazon ECS eAWS Fargate são usados como adaptadores primários. O API Gateway, o Elastic Load Balancing e o Amazon SQS são usados como adaptadores secundários. EventBridge

Perguntas frequentes

Por que devo usar uma arquitetura hexagonal?

A arquitetura hexagonal muda o foco dos desenvolvedores para a lógica do domínio, simplifica a automação de testes e melhora a qualidade e a adaptabilidade do código. Essas melhorias resultam em um tempo de comercialização mais rápido e em um escalonamento técnico e organizacional mais fácil.

Por que devo usar design orientado por domínio?

O design orientado por domínio (DDD) permite que você crie componentes e construções de software usando uma linguagem comum entre as partes interessadas da empresa e os engenheiros. O DDD ajuda você a gerenciar a complexidade do software e é uma estratégia eficaz para manter produtos de software a longo prazo.

Posso praticar o desenvolvimento orientado por testes sem arquitetura hexagonal?

Sim. O desenvolvimento orientado por testes (TDD) não se limita a padrões específicos de design de software. No entanto, a arquitetura hexagonal facilita a prática do TDD.

Posso escalar meu produto sem arquitetura hexagonal e design orientado por domínio?

Sim. A escalabilidade técnica e organizacional do produto pode ser alcançada com a maioria dos padrões de design. No entanto, a arquitetura hexagonal e o DDD facilitam o dimensionamento e são mais eficazes para grandes projetos a longo prazo.

Quais tecnologias devo usar para implementar a arquitetura hexagonal?

A arquitetura hexagonal não se limita a uma pilha de tecnologia específica. Recomendamos que você escolha uma tecnologia que ofereça suporte à inversão de dependência e ao teste de unidade.

Estou desenvolvendo um produto mínimo viável. Faz sentido gastar tempo pensando em arquitetura de software?

Sim. Recomendamos usar padrões de design que sejam familiares para MVPs. Recomendamos que você experimente praticar a arquitetura hexagonal até que seus engenheiros se sintam confortáveis com ela. Estabelecer uma arquitetura hexagonal para novos projetos não exige um investimento de tempo significativamente maior do que começar sem nenhuma arquitetura.

Estou desenvolvendo um produto mínimo viável e não tenho tempo para escrever testes.

Se o seu MVP contém lógica de negócios, é altamente recomendável escrever testes automatizados para ele. Isso reduzirá o ciclo de feedback e economizará tempo.

Quais padrões de design adicionais posso usar com a arquitetura hexagonal?

Use o padrão CQRS para apoiar o dimensionamento do sistema geral. Use o padrão de repositório para armazenar e restaurar seu modelo de domínio. Use a unidade de padrão de trabalho para gerenciar as etapas do processo transacional. Use composição em vez de herança para modelar agregados de domínio, entidades e objetos de valor. Não crie hierarquias de objetos complexas.

Next steps (Próximas etapas)

- Familiarize-se ainda mais com os conceitos de design orientado por domínios lendo os links coletados naRecursos seção.
- Se você estiver implementando um novo projeto, use o modelo de estrutura do projeto fornecido neste guia e implemente alguns recursos.
- Se você estiver implementando um projeto existente, identifique o código que pode ser dividido entre operações somente de leitura e somente de gravação. Abstrata o código somente para leitura nos serviços de consulta e coloque o código somente para gravação nos manipuladores de comando.
- Quando a estrutura básica do projeto estiver implementada, escreva testes unitários, estabeleça a integração contínua (CI) com a automação de testes e siga as práticas de desenvolvimento orientado a testes (TDD).

Recursos

Referências

- Estruture um projeto Python em arquitetura hexagonal usandoAWS Lambda (padrão de orientaçãoAWS prescritiva)
- Equipes ágeis (site do Scaled Agile Framework)
- Padrões de arquitetura com Python, de Harry Percival e Bob Gregory (O'Reilly Media, 31 de março de 2020), especificamente os seguintes capítulos:
  - Comandos e manipulador de comandos
  - Segregação de responsabilidade por consulta de comando (CQRS)
  - Padrão de repositório
- Event Storming: a abordagem mais inteligente de colaboração além dos limites dos silos, por Alberto Brandolini (site Event Storming)
- Desmistificando a Lei de Conway, de Sam Newman (site da Thoughtworks, 30 de junho de 2014)
- Desenvolvendo arquitetura evolutiva comAWS Lambda, por James Beswick (AWSCompute Blog, 8 de julho de 2021)
- Idioma de domínio: enfrentando a complexidade no coração do software (site em idioma de domínio)
- Facade, de Dive Into Design Patterns de Alexander Shvets (ebook, 5 de dezembro de 2018)
- GivenWhenThen, de Martin Fowler (21 de agosto de 2013)
- Implementando o design orientado por domínio, por Vaughn Vernon (Addison-Wesley Professional, fevereiro de 2013)
- Manobra inversa de Conway (site da Thoughtworks, 8 de julho de 2014)
- Padrão do mês: Red Green Refactor (site da DZone, 2 de junho de 2017)
- Princípios de design do SOLID explicados: princípio de inversão de dependência com exemplos de código, por Thorben Janssen (site Stackify, 7 de maio de 2018)
- Princípios SOLID: explicação e exemplos, por Simon Hoiberg (site da ITNEXT, 1º de janeiro de 2019)
- A arte do desenvolvimento ágil: desenvolvimento orientado a testes, de James Shore e Shane Warden (O'Reilly Media, 25 de março de 2010)

- Os princípios SOLID da programação orientada a objetos explicados em inglês simples, por Yigit Kemal Erinc (postagens sobre programaçãofreeCodeCamp orientada a objetos, 20 de agosto de 2020)
- O que é uma arquitetura orientada a eventos? (AWSsite)

Serviços da AWS

- Amazon API Gateway
- Amazon Aurora
- Amazon DynamoDB
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Elastic Load Balancing
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon Relational Database Service (Amazon RDS)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)

Outras ferramentas

- Moto
- LocalStack

Histórico do documento

A tabela a seguir descreve alterações significativas neste guia. Se quiser ser notificado sobre future atualizações, você pode assinar um feed RSS.

Alteração	Descrição	Data
Publicação inicial	—	15 de junho de 2022

AWS Glossário de orientação prescritiva

A seguir estão os termos comumente usados em estratégias, guias e padrões fornecidos pela Orientação AWS Prescritiva. Para sugerir entradas, use o link Fornecer feedback no final do glossário.

Números

7 Rs

Sete estratégias comuns de migração para mover aplicações para a nuvem. Essas estratégias baseiam-se nos 5 Rs identificados pela Gartner em 2011 e consistem em:

- Refatorar/rearquitetar: mova uma aplicação e modifique sua arquitetura aproveitando ao máximo os recursos nativos de nuvem para melhorar a agilidade, a performance e a escalabilidade. Isso normalmente envolve a portabilidade do sistema operacional e do banco de dados. Exemplo: migre seu banco de dados Oracle local para a edição compatível com o Amazon Aurora PostgreSQL.
- Redefinir a plataforma (mover e redefinir [mover e redefinir (lift-and-reshape]): mova uma aplicação para a nuvem e introduza algum nível de otimização a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Amazon Relational Database Service (Amazon RDS) for Oracle no. Nuvem AWS
- Recomprar (drop and shop): mude para um produto diferente, normalmente migrando de uma licença tradicional para um modelo SaaS. Exemplo: migre seu sistema de gerenciamento de relacionamento com o cliente (CRM) para a Salesforce.com.
- Redefinir a hospedagem (mover sem alterações [lift-and-shift])mover uma aplicação para a nuvem sem fazer nenhuma alteração a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Oracle em uma instância do EC2 no. Nuvem AWS
- Realocar (mover o hipervisor sem alterações [hypervisor-level lift-and-shift]): mover a infraestrutura para a nuvem sem comprar novo hardware, reescrever aplicações ou modificar suas operações existentes. Você migra servidores de uma plataforma local para um serviço em nuvem para a mesma plataforma. Exemplo: migrar um Microsoft Hyper-V aplicativo para o. AWS
- Reter (revisitar): mantenha as aplicações em seu ambiente de origem. Isso pode incluir aplicações que exigem grande refatoração, e você deseja adiar esse trabalho para um

momento posterior, e aplicações antigas que você deseja manter porque não há justificativa comercial para migrá-las.

- Retirar: desative ou remova aplicações que não são mais necessárias em seu ambiente de origem.

A

ABAC

Consulte controle de acesso baseado em atributos.

serviços abstratos

Veja os serviços gerenciados.

ACID

Veja atomicidade, consistência, isolamento, durabilidade.

migração ativa-ativa

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia (por meio de uma ferramenta de replicação bidirecional ou operações de gravação dupla), e ambos os bancos de dados lidam com transações de aplicações conectadas durante a migração. Esse método oferece suporte à migração em lotes pequenos e controlados, em vez de exigir uma substituição única. É mais flexível, mas exige mais trabalho do que a migração ativa-passiva.

migração ativa-passiva

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia, mas somente o banco de dados de origem manipula as transações das aplicações conectadas enquanto os dados são replicados no banco de dados de destino. O banco de dados de destino não aceita nenhuma transação durante a migração.

função agregada

Uma função SQL que opera em um grupo de linhas e calcula um único valor de retorno para o grupo. Exemplos de funções agregadas incluem `SUM` e. `MAX`

AI

Veja inteligência artificial.

AIOps

Veja as operações de inteligência artificial.

anonimização

O processo de excluir permanentemente informações pessoais em um conjunto de dados. A anonimização pode ajudar a proteger a privacidade pessoal. Dados anônimos não são mais considerados dados pessoais.

antipadrões

Uma solução frequentemente usada para um problema recorrente em que a solução é contraproducente, ineficaz ou menos eficaz do que uma alternativa.

controle de aplicativos

Uma abordagem de segurança que permite o uso somente de aplicativos aprovados para ajudar a proteger um sistema contra malware.

portfólio de aplicações

Uma coleção de informações detalhadas sobre cada aplicação usada por uma organização, incluindo o custo para criar e manter a aplicação e seu valor comercial. Essas informações são fundamentais para o processo de descoberta e análise de portfólio e ajudam a identificar e priorizar as aplicações a serem migradas, modernizadas e otimizadas.

inteligência artificial (IA)

O campo da ciência da computação que se dedica ao uso de tecnologias de computação para desempenhar funções cognitivas normalmente associadas aos humanos, como aprender, resolver problemas e reconhecer padrões. Para obter mais informações, consulte O que é inteligência artificial?

operações de inteligência artificial (AIOps)

O processo de usar técnicas de machine learning para resolver problemas operacionais, reduzir incidentes operacionais e intervenção humana e aumentar a qualidade do serviço. Para obter mais informações sobre como as AIOps são usadas na estratégia de migração para a AWS , consulte o guia de integração de operações.

criptografia assimétrica

Um algoritmo de criptografia que usa um par de chaves, uma chave pública para criptografia e uma chave privada para descriptografia. É possível compartilhar a chave pública porque ela não é usada na descriptografia, mas o acesso à chave privada deve ser altamente restrito.

atomicidade, consistência, isolamento, durabilidade (ACID)

Um conjunto de propriedades de software que garantem a validade dos dados e a confiabilidade operacional de um banco de dados, mesmo no caso de erros, falhas de energia ou outros problemas.

controle de acesso por atributo (ABAC)

A prática de criar permissões minuciosas com base nos atributos do usuário, como departamento, cargo e nome da equipe. Para obter mais informações, consulte ABAC AWS na documentação AWS Identity and Access Management (IAM).

fonte de dados autorizada

Um local onde você armazena a versão principal dos dados, que é considerada a fonte de informações mais confiável. Você pode copiar dados da fonte de dados autorizada para outros locais com o objetivo de processar ou modificar os dados, como anonimizá-los, redigi-los ou pseudonimizá-los.

Availability Zone (zona de disponibilidade)

Um local distinto dentro de um Região da AWS que está isolado de falhas em outras zonas de disponibilidade e fornece conectividade de rede barata e de baixa latência a outras zonas de disponibilidade na mesma região.

AWS Estrutura de adoção da nuvem (AWS CAF)

Uma estrutura de diretrizes e melhores práticas AWS para ajudar as organizações a desenvolver um plano eficiente e eficaz para migrar com sucesso para a nuvem. AWS O CAF organiza a orientação em seis áreas de foco chamadas perspectivas: negócios, pessoas, governança, plataforma, segurança e operações. As perspectivas de negócios, pessoas e governança têm como foco habilidades e processos de negócios; as perspectivas de plataforma, segurança e operações concentram-se em habilidades e processos técnicos. Por exemplo, a perspectiva das pessoas tem como alvo as partes interessadas que lidam com recursos humanos (RH), funções de pessoal e gerenciamento de pessoal. Nessa perspectiva, o AWS CAF fornece orientação para desenvolvimento, treinamento e comunicação de pessoas para ajudar a preparar a organização para a adoção bem-sucedida da nuvem. Para obter mais informações, consulte o site da AWS CAF e o whitepaper da AWS CAF.

AWS Estrutura de qualificação da carga de trabalho (AWS WQF)

Uma ferramenta que avalia as cargas de trabalho de migração do banco de dados, recomenda estratégias de migração e fornece estimativas de trabalho. AWS O WQF está incluído com AWS

Schema Conversion Tool ()AWS SCT. Ela analisa esquemas de banco de dados e objetos de código, código de aplicações, dependências e características de performance, além de fornecer relatórios de avaliação.

B

bot ruim

Um bot destinado a perturbar ou causar danos a indivíduos ou organizações.

BCP

Veja o planejamento de continuidade de negócios.

gráfico de comportamento

Uma visualização unificada e interativa do comportamento e das interações de recursos ao longo do tempo. É possível usar um gráfico de comportamento com o Amazon Detective para examinar tentativas de login malsucedidas, chamadas de API suspeitas e ações similares. Para obter mais informações, consulte Dados em um gráfico de comportamento na documentação do Detective.

sistema big-endian

Um sistema que armazena o byte mais significativo antes. Veja também endianness.

classificação binária

Um processo que prevê um resultado binário (uma de duas classes possíveis). Por exemplo, seu modelo de ML pode precisar prever problemas como “Este e-mail é ou não é spam?” ou “Este produto é um livro ou um carro?”

filtro de bloom

Uma estrutura de dados probabilística e eficiente em termos de memória que é usada para testar se um elemento é membro de um conjunto.

blue/green deployment (implantação azul/verde)

Uma estratégia de implantação em que você cria dois ambientes separados, mas idênticos. Você executa a versão atual do aplicativo em um ambiente (azul) e a nova versão do aplicativo no outro ambiente (verde). Essa estratégia ajuda você a reverter rapidamente com o mínimo de impacto.

bot

Um aplicativo de software que executa tarefas automatizadas pela Internet e simula a atividade ou interação humana. Alguns bots são úteis ou benéficos, como rastreadores da Web que indexam informações na Internet. Alguns outros bots, conhecidos como bots ruins, têm como objetivo perturbar ou causar danos a indivíduos ou organizações.

botnet

Redes de bots infectadas por malware e sob o controle de uma única parte, conhecidas como pastor de bots ou operador de bots. As redes de bots são o mecanismo mais conhecido para escalar bots e seu impacto.

ramo

Uma área contida de um repositório de código. A primeira ramificação criada em um repositório é a ramificação principal. Você pode criar uma nova ramificação a partir de uma ramificação existente e, em seguida, desenvolver recursos ou corrigir bugs na nova ramificação. Uma ramificação que você cria para gerar um recurso é comumente chamada de ramificação de recurso. Quando o recurso estiver pronto para lançamento, você mesclará a ramificação do recurso de volta com a ramificação principal. Para obter mais informações, consulte Sobre filiais (GitHub documentação).

acesso em vidro quebrado

Em circunstâncias excepcionais e por meio de um processo aprovado, um meio rápido para um usuário obter acesso a um Conta da AWS que ele normalmente não tem permissão para acessar. Para obter mais informações, consulte o indicador Implementar procedimentos de quebra de vidro na orientação do Well-Architected AWS .

estratégia brownfield

A infraestrutura existente em seu ambiente. Ao adotar uma estratégia brownfield para uma arquitetura de sistema, você desenvolve a arquitetura de acordo com as restrições dos sistemas e da infraestrutura atuais. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

cache do buffer

A área da memória em que os dados acessados com mais frequência são armazenados.

capacidade de negócios

O que uma empresa faz para gerar valor (por exemplo, vendas, atendimento ao cliente ou marketing). As arquiteturas de microsserviços e as decisões de desenvolvimento podem

ser orientadas por recursos de negócios. Para obter mais informações, consulte a seção Organizados de acordo com as capacidades de negócios do whitepaper Executar microsserviços conteinerizados na AWS.

planejamento de continuidade de negócios (BCP)

Um plano que aborda o impacto potencial de um evento disruptivo, como uma migração em grande escala, nas operações e permite que uma empresa retome as operações rapidamente.

C

CAF

Consulte Estrutura de adoção da AWS nuvem.

implantação canária

O lançamento lento e incremental de uma versão para usuários finais. Quando estiver confiante, você implanta a nova versão e substituirá a versão atual em sua totalidade.

CCoE

Veja o Centro de Excelência em Nuvem.

CDC

Veja a captura de dados de alterações.

captura de dados de alterações (CDC)

O processo de rastrear alterações em uma fonte de dados, como uma tabela de banco de dados, e registrar metadados sobre a alteração. É possível usar o CDC para várias finalidades, como auditar ou replicar alterações em um sistema de destino para manter a sincronização.

engenharia do caos

Introduzir intencionalmente falhas ou eventos disruptivos para testar a resiliência de um sistema. Você pode usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estressam suas AWS cargas de trabalho e avaliar sua resposta.

CI/CD

Veja a integração e a entrega contínuas.

classificação

Um processo de categorização que ajuda a gerar previsões. Os modelos de ML para problemas de classificação predizem um valor discreto. Os valores discretos são sempre diferentes uns dos outros. Por exemplo, um modelo pode precisar avaliar se há ou não um carro em uma imagem.

criptografia no lado do cliente

Criptografia de dados localmente, antes que o alvo os AWS service (Serviço da AWS) receba.

Centro de Excelência da Nuvem (CCoE)

Uma equipe multidisciplinar que impulsiona os esforços de adoção da nuvem em toda a organização, incluindo o desenvolvimento de práticas recomendas de nuvem, a mobilização de recursos, o estabelecimento de cronogramas de migração e a liderança da organização em transformações em grande escala. Para obter mais informações, consulte as postagens do CCoE no blog de estratégia Nuvem AWS corporativa.

computação em nuvem

A tecnologia de nuvem normalmente usada para armazenamento de dados remoto e gerenciamento de dispositivos de IoT. A computação em nuvem geralmente está conectada à tecnologia de computação de ponta.

modelo operacional em nuvem

Em uma organização de TI, o modelo operacional usado para criar, amadurecer e otimizar um ou mais ambientes de nuvem. Para obter mais informações, consulte Criar seu modelo operacional de nuvem.

estágios de adoção da nuvem

As quatro fases pelas quais as organizações normalmente passam quando migram para o Nuvem AWS:

- Projeto: executar alguns projetos relacionados à nuvem para fins de prova de conceito e aprendizado
- Fundação: realizar investimentos fundamentais para escalar sua adoção da nuvem (por exemplo, criar uma zona de pouso, definir um CCoE, estabelecer um modelo de operações)
- Migração: migrar aplicações individuais
- Reinvenção: otimizar produtos e serviços e inovar na nuvem

Esses estágios foram definidos por Stephen Orban na postagem do blog The Journey Toward Cloud-First & the Stages of Adoption no blog de estratégia Nuvem AWS empresarial. Para obter

informações sobre como eles se relacionam com a estratégia de AWS migração, consulte o guia de preparação para migração.

CMDB

Consulte o banco de dados de gerenciamento de configuração.

repositório de código

Um local onde o código-fonte e outros ativos, como documentação, amostras e scripts, são armazenados e atualizados por meio de processos de controle de versão. Os repositórios de nuvem comuns incluem GitHub ou AWS CodeCommit. Cada versão do código é chamada de ramificação. Em uma estrutura de microsserviços, cada repositório é dedicado a uma única peça de funcionalidade. Um único pipeline de CI/CD pode usar vários repositórios.

cache frio

Um cache de buffer que está vazio, não está bem preenchido ou contém dados obsoletos ou irrelevantes. Isso afeta a performance porque a instância do banco de dados deve ler da memória principal ou do disco, um processo que é mais lento do que a leitura do cache do buffer.

dados frios

Dados que raramente são acessados e geralmente são históricos. Ao consultar esse tipo de dados, consultas lentas geralmente são aceitáveis. Mover esses dados para níveis ou classes de armazenamento de baixo desempenho e menos caros pode reduzir os custos.

visão computacional (CV)

Um campo da IA que usa aprendizado de máquina para analisar e extrair informações de formatos visuais, como imagens e vídeos digitais. Por exemplo, AWS Panorama oferece dispositivos que adicionam CV às redes de câmeras locais, e a Amazon SageMaker fornece algoritmos de processamento de imagem para CV.

desvio de configuração

Para uma carga de trabalho, uma alteração de configuração em relação ao estado esperado. Isso pode fazer com que a carga de trabalho se torne incompatível e, normalmente, é gradual e não intencional.

banco de dados de gerenciamento de configuração (CMDB)

Um repositório que armazena e gerencia informações sobre um banco de dados e seu ambiente de TI, incluindo componentes de hardware e software e suas configurações. Normalmente, os dados de um CMDB são usados no estágio de descoberta e análise do portfólio da migração.

pacote de conformidade

Um conjunto de AWS Config regras e ações de remediação que você pode montar para personalizar suas verificações de conformidade e segurança. Você pode implantar um pacote de conformidade como uma entidade única em uma Conta da AWS região ou em uma organização usando um modelo YAML. Para obter mais informações, consulte Pacotes de conformidade na documentação. AWS Config

integração contínua e entrega contínua (CI/CD)

O processo de automatizar os estágios de origem, criação, teste, preparação e produção do processo de lançamento do software. O CI/CD é comumente descrito como um pipeline. O CI/CD pode ajudar você a automatizar processos, melhorar a produtividade, melhorar a qualidade do código e entregar com mais rapidez. Para obter mais informações, consulte Benefícios da entrega contínua. CD também pode significar implantação contínua. Para obter mais informações, consulte Entrega contínua versus implantação contínua.

CV

Veja visão computacional.

D

dados em repouso

Dados estacionários em sua rede, por exemplo, dados que estão em um armazenamento.

classificação de dados

Um processo para identificar e categorizar os dados em sua rede com base em criticalidade e confidencialidade. É um componente crítico de qualquer estratégia de gerenciamento de riscos de segurança cibernética, pois ajuda a determinar os controles adequados de proteção e retenção para os dados. A classificação de dados é um componente do pilar de segurança no AWS Well-Architected Framework. Para obter mais informações, consulte Classificação de dados.

desvio de dados

Uma variação significativa entre os dados de produção e os dados usados para treinar um modelo de ML ou uma alteração significativa nos dados de entrada ao longo do tempo. O desvio de dados pode reduzir a qualidade geral, a precisão e a imparcialidade das previsões do modelo de ML.

dados em trânsito

Dados que estão se movendo ativamente pela sua rede, como entre os recursos da rede.

malha de dados

Uma estrutura arquitetônica que fornece propriedade de dados distribuída e descentralizada com gerenciamento e governança centralizados.

minimização de dados

O princípio de coletar e processar apenas os dados estritamente necessários. Praticar a minimização de dados no Nuvem AWS pode reduzir os riscos de privacidade, os custos e a pegada de carbono de sua análise.

perímetro de dados

Um conjunto de proteções preventivas em seu AWS ambiente que ajudam a garantir que somente identidades confiáveis acessem recursos confiáveis das redes esperadas. Para obter mais informações, consulte Construindo um perímetro de dados em. AWS

pré-processamento de dados

A transformação de dados brutos em um formato que seja facilmente analisado por seu modelo de ML. O pré-processamento de dados pode significar a remoção de determinadas colunas ou linhas e o tratamento de valores ausentes, inconsistentes ou duplicados.

proveniência dos dados

O processo de rastrear a origem e o histórico dos dados ao longo de seu ciclo de vida, por exemplo, como os dados foram gerados, transmitidos e armazenados.

titular dos dados

Um indivíduo cujos dados estão sendo coletados e processados.

data warehouse

Um sistema de gerenciamento de dados que oferece suporte à inteligência comercial, como análises. Os data warehouses geralmente contêm grandes quantidades de dados históricos e geralmente são usados para consultas e análises.

linguagem de definição de dados (DDL)

Instruções ou comandos para criar ou modificar a estrutura de tabelas e objetos em um banco de dados.

linguagem de manipulação de dados (DML)

Instruções ou comandos para modificar (inserir, atualizar e excluir) informações em um banco de dados.

DDL

Consulte a linguagem de definição de banco de dados.

deep ensemble

A combinação de vários modelos de aprendizado profundo para gerar previsões. Os deep ensembles podem ser usados para produzir uma previsão mais precisa ou para estimar a incerteza nas previsões.

Aprendizado profundo

Um subcampo do ML que usa várias camadas de redes neurais artificiais para identificar o mapeamento entre os dados de entrada e as variáveis-alvo de interesse.

defense-in-depth

Uma abordagem de segurança da informação na qual uma série de mecanismos e controles de segurança são cuidadosamente distribuídos por toda a rede de computadores para proteger a confidencialidade, a integridade e a disponibilidade da rede e dos dados nela contidos. Ao adotar essa estratégia AWS, você adiciona vários controles em diferentes camadas da AWS Organizations estrutura para ajudar a proteger os recursos. Por exemplo, uma defense-in-depth abordagem pode combinar autenticação multifatorial, segmentação de rede e criptografia.

administrador delegado

Em AWS Organizations, um serviço compatível pode registrar uma conta de AWS membro para administrar as contas da organização e gerenciar as permissões desse serviço. Essa conta é chamada de administrador delegado para esse serviço Para obter mais informações e uma lista de serviços compatíveis, consulte Serviços que funcionam com o AWS Organizations na documentação do AWS Organizations .

implantação

O processo de criar uma aplicação, novos recursos ou correções de código disponíveis no ambiente de destino. A implantação envolve a implementação de mudanças em uma base de código e, em seguida, a criação e execução dessa base de código nos ambientes da aplicação

ambiente de desenvolvimento

Veja o ambiente.

controle detectivo

Um controle de segurança projetado para detectar, registrar e alertar após a ocorrência de um evento. Esses controles são uma segunda linha de defesa, alertando você sobre eventos de segurança que contornaram os controles preventivos em vigor. Para obter mais informações, consulte Controles detectivos em Como implementar controles de segurança na AWS.

mapeamento do fluxo de valor de desenvolvimento (DVSM)

Um processo usado para identificar e priorizar restrições que afetam negativamente a velocidade e a qualidade em um ciclo de vida de desenvolvimento de software. O DVSM estende o processo de mapeamento do fluxo de valor originalmente projetado para práticas de manufatura enxuta. Ele se concentra nas etapas e equipes necessárias para criar e movimentar valor por meio do processo de desenvolvimento de software.

gêmeo digital

Uma representação virtual de um sistema real, como um prédio, fábrica, equipamento industrial ou linha de produção. Os gêmeos digitais oferecem suporte à manutenção preditiva, ao monitoramento remoto e à otimização da produção.

tabela de dimensões

Em um esquema em estrela, uma tabela menor que contém atributos de dados sobre dados quantitativos em uma tabela de fatos. Os atributos da tabela de dimensões geralmente são campos de texto ou números discretos que se comportam como texto. Esses atributos são comumente usados para restringir consultas, filtrar e rotular conjuntos de resultados.

desastre

Um evento que impede que uma workload ou sistema cumpra seus objetivos de negócios em seu local principal de implantação. Esses eventos podem ser desastres naturais, falhas técnicas ou o resultado de ações humanas, como configuração incorreta não intencional ou ataque de malware.

Recuperação de desastres (RD)

A estratégia e o processo que você usa para minimizar o tempo de inatividade e a perda de dados causados por um desastre. Para obter mais informações, consulte Recuperação de desastres de cargas de trabalho em AWS: Recuperação na nuvem no AWS Well-Architected Framework.

DML

Consulte linguagem de manipulação de banco de dados.

design orientado por domínio

Uma abordagem ao desenvolvimento de um sistema de software complexo conectando seus componentes aos domínios em evolução, ou principais metas de negócios, atendidos por cada componente. Esse conceito foi introduzido por Eric Evans em seu livro, Design orientado por domínio: lidando com a complexidade no coração do software (Boston: Addison-Wesley Professional, 2003). Para obter informações sobre como usar o design orientado por domínio com o padrão strangler fig, consulte Modernizar incrementalmente os serviços web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

DR

Veja a recuperação de desastres.

detecção de deriva

Rastreando desvios de uma configuração básica. Por exemplo, você pode usar AWS CloudFormation para detectar desvios nos recursos do sistema ou AWS Control Tower para detectar mudanças em seu landing zone que possam afetar a conformidade com os requisitos de governança.

DVSM

Veja o mapeamento do fluxo de valor do desenvolvimento.

E

EDA

Veja a análise exploratória de dados.

computação de borda

A tecnologia que aumenta o poder computacional de dispositivos inteligentes nas bordas de uma rede de IoT. Quando comparada à computação em nuvem, a computação de ponta pode reduzir a latência da comunicação e melhorar o tempo de resposta.

Criptografia

Um processo de computação que transforma dados de texto simples, legíveis por humanos, em texto cifrado.

chave de criptografia

Uma sequência criptográfica de bits aleatórios que é gerada por um algoritmo de criptografia. As chaves podem variar em tamanho, e cada chave foi projetada para ser imprevisível e exclusiva.

endianismo

A ordem na qual os bytes são armazenados na memória do computador. Os sistemas big-endian armazenam o byte mais significativo antes. Os sistemas little-endian armazenam o byte menos significativo antes.

endpoint

Veja o endpoint do serviço.

serviço de endpoint

Um serviço que pode ser hospedado em uma nuvem privada virtual (VPC) para ser compartilhado com outros usuários. Você pode criar um serviço de endpoint com AWS PrivateLink e conceder permissões a outros diretores Contas da AWS ou a AWS Identity and Access Management (IAM). Essas contas ou entidades principais podem se conectar ao serviço de endpoint de maneira privada criando endpoints da VPC de interface. Para obter mais informações, consulte Criar um serviço de endpoint na documentação do Amazon Virtual Private Cloud (Amazon VPC).

planejamento de recursos corporativos (ERP)

Um sistema que automatiza e gerencia os principais processos de negócios (como contabilidade, MES e gerenciamento de projetos) para uma empresa.

criptografia envelopada

O processo de criptografar uma chave de criptografia com outra chave de criptografia. Para obter mais informações, consulte Criptografia de envelope na documentação AWS Key Management Service (AWS KMS).

environment (ambiente)

Uma instância de uma aplicação em execução. Estes são tipos comuns de ambientes na computação em nuvem:

- ambiente de desenvolvimento: uma instância de uma aplicação em execução que está disponível somente para a equipe principal responsável pela manutenção da aplicação. Ambientes de desenvolvimento são usados para testar mudanças antes de promovê-las para ambientes superiores. Esse tipo de ambiente às vezes é chamado de ambiente de teste.

- ambientes inferiores: todos os ambientes de desenvolvimento para uma aplicação, como aqueles usados para compilações e testes iniciais.
- ambiente de produção: uma instância de uma aplicação em execução que os usuários finais podem acessar. Em um pipeline de CI/CD, o ambiente de produção é o último ambiente de implantação.
- ambientes superiores: todos os ambientes que podem ser acessados por usuários que não sejam a equipe principal de desenvolvimento. Isso pode incluir um ambiente de produção, ambientes de pré-produção e ambientes para testes de aceitação do usuário.

epic

Em metodologias ágeis, categorias funcionais que ajudam a organizar e priorizar seu trabalho. Os epics fornecem uma descrição de alto nível dos requisitos e das tarefas de implementação. Por exemplo, os épicos de segurança AWS da CAF incluem gerenciamento de identidade e acesso, controles de detetive, segurança de infraestrutura, proteção de dados e resposta a incidentes. Para obter mais informações sobre epics na estratégia de migração da AWS , consulte o guia de implementação do programa.

ERP

Consulte planejamento de recursos corporativos.

análise exploratória de dados (EDA)

O processo de analisar um conjunto de dados para entender suas principais características. Você coleta ou agrega dados e, em seguida, realiza investigações iniciais para encontrar padrões, detectar anomalias e verificar suposições. O EDA é realizado por meio do cálculo de estatísticas resumidas e da criação de visualizações de dados.

F

tabela de fatos

A tabela central em um esquema em estrela. Ele armazena dados quantitativos sobre operações comerciais. Normalmente, uma tabela de fatos contém dois tipos de colunas: aquelas que contêm medidas e aquelas que contêm uma chave externa para uma tabela de dimensões.

falham rapidamente

Uma filosofia que usa testes frequentes e incrementais para reduzir o ciclo de vida do desenvolvimento. É uma parte essencial de uma abordagem ágil.

limite de isolamento de falhas

No Nuvem AWS, um limite, como uma zona de disponibilidade, Região da AWS um plano de controle ou um plano de dados, que limita o efeito de uma falha e ajuda a melhorar a resiliência das cargas de trabalho. Para obter mais informações, consulte Limites de isolamento de AWS falhas.

ramificação de recursos

Veja a filial.

recursos

Os dados de entrada usados para fazer uma previsão. Por exemplo, em um contexto de manufatura, os recursos podem ser imagens capturadas periodicamente na linha de fabricação.

importância do recurso

O quanto um recurso é importante para as previsões de um modelo. Isso geralmente é expresso como uma pontuação numérica que pode ser calculada por meio de várias técnicas, como Shapley Additive Explanations (SHAP) e gradientes integrados. Para obter mais informações, consulte Interpretabilidade do modelo de aprendizado de máquina com:AWS.

transformação de recursos

O processo de otimizar dados para o processo de ML, incluindo enriquecer dados com fontes adicionais, escalar valores ou extrair vários conjuntos de informações de um único campo de dados. Isso permite que o modelo de ML se beneficie dos dados. Por exemplo, se a data “2021-05-27 00:15:37” for dividida em “2021”, “maio”, “quinta” e “15”, isso poderá ajudar o algoritmo de aprendizado a aprender padrões diferenciados associados a diferentes componentes de dados.

FGAC

Veja o controle de acesso refinado.

Controle de acesso refinado (FGAC)

O uso de várias condições para permitir ou negar uma solicitação de acesso.

migração flash-cut

Um método de migração de banco de dados que usa replicação contínua de dados por meio da captura de dados alterados para migrar dados no menor tempo possível, em vez de usar uma abordagem em fases. O objetivo é reduzir ao mínimo o tempo de inatividade.

G

bloqueio geográfico

Veja as restrições geográficas.

restrições geográficas (bloqueio geográfico)

Na Amazon CloudFront, uma opção para impedir que usuários em países específicos acessem distribuições de conteúdo. É possível usar uma lista de permissões ou uma lista de bloqueios para especificar países aprovados e banidos. Para obter mais informações, consulte Restringir a distribuição geográfica do seu conteúdo na CloudFront documentação.

Fluxo de trabalho do GitFlow

Uma abordagem na qual ambientes inferiores e superiores usam ramificações diferentes em um repositório de código-fonte. O fluxo de trabalho do Gitflow é considerado legado, e o fluxo de trabalho baseado em troncos é a abordagem moderna e preferida.

estratégia greenfield

A ausência de infraestrutura existente em um novo ambiente. Ao adotar uma estratégia greenfield para uma arquitetura de sistema, é possível selecionar todas as novas tecnologias sem a restrição da compatibilidade com a infraestrutura existente, também conhecida como brownfield. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

barreira de proteção

Uma regra de alto nível que ajuda a gerenciar recursos, políticas e conformidade em todas as unidades organizacionais (UOs). Barreiras de proteção preventivas impõem políticas para garantir o alinhamento a padrões de conformidade. Elas são implementadas usando políticas de controle de serviço e limites de permissões do IAM. Barreiras de proteção detectivas detectam violações de políticas e problemas de conformidade e geram alertas para remediação. Eles são implementados usando AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector e verificações personalizadas AWS Lambda .

H

HA

Veja a alta disponibilidade.

migração heterogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que usa um mecanismo de banco de dados diferente (por exemplo, Oracle para Amazon Aurora). A migração heterogênea geralmente faz parte de um esforço de redefinição da arquitetura, e converter o esquema pode ser uma tarefa complexa. O AWS fornece o AWS SCT para ajudar nas conversões de esquemas.

alta disponibilidade (HA)

A capacidade de uma workload operar continuamente, sem intervenção, em caso de desafios ou desastres. Os sistemas AH são projetados para realizar o failover automático, oferecer consistentemente desempenho de alta qualidade e lidar com diferentes cargas e falhas com impacto mínimo no desempenho.

modernização de historiador

Uma abordagem usada para modernizar e atualizar os sistemas de tecnologia operacional (OT) para melhor atender às necessidades do setor de manufatura. Um historiador é um tipo de banco de dados usado para coletar e armazenar dados de várias fontes em uma fábrica.

migração homogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que compartilha o mesmo mecanismo de banco de dados (por exemplo, Microsoft SQL Server para Amazon RDS para SQL Server). A migração homogênea geralmente faz parte de um esforço de redefinição da hospedagem ou da plataforma. É possível usar utilitários de banco de dados nativos para migrar o esquema.

dados quentes

Dados acessados com frequência, como dados em tempo real ou dados translacionais recentes. Esses dados normalmente exigem uma camada ou classe de armazenamento de alto desempenho para fornecer respostas rápidas às consultas.

hotfix

Uma correção urgente para um problema crítico em um ambiente de produção. Devido à sua urgência, um hotfix geralmente é feito fora do fluxo de trabalho típico de uma DevOps versão.

período de hipercuidados

Imediatamente após a substituição, o período em que uma equipe de migração gerencia e monitora as aplicações migradas na nuvem para resolver quaisquer problemas. Normalmente,

a duração desse período é de 1 a 4 dias. No final do período de hipercuidados, a equipe de migração normalmente transfere a responsabilidade pelas aplicações para a equipe de operações de nuvem.

I

IaC

Veja a infraestrutura como código.

Política baseada em identidade

Uma política anexada a um ou mais diretores do IAM que define suas permissões no Nuvem AWS ambiente.

aplicação ociosa

Uma aplicação que tem um uso médio de CPU e memória entre 5 e 20% em um período de 90 dias. Em um projeto de migração, é comum retirar essas aplicações ou retê-las on-premises.

IIoT

Veja a Internet das Coisas industrial.

infraestrutura imutável

Um modelo que implanta uma nova infraestrutura para cargas de trabalho de produção em vez de atualizar, corrigir ou modificar a infraestrutura existente. Infraestruturas imutáveis são inerentemente mais consistentes, confiáveis e previsíveis do que infraestruturas mutáveis. Para obter mais informações, consulte as melhores práticas de implantação usando infraestrutura imutável no Well-Architected AWS Framework.

VPC de entrada (admissão)

Em uma arquitetura de AWS várias contas, uma VPC que aceita, inspeciona e roteia conexões de rede de fora de um aplicativo. A Arquitetura de referência de segurança da AWS recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

migração incremental

Uma estratégia de substituição na qual você migra a aplicação em pequenas partes, em vez de realizar uma única substituição completa. Por exemplo, é possível mover inicialmente

apenas alguns microsserviços ou usuários para o novo sistema. Depois de verificar se tudo está funcionando corretamente, mova os microsserviços ou usuários adicionais de forma incremental até poder descomissionar seu sistema herdado. Essa estratégia reduz os riscos associados a migrações de grande porte.

Indústria 4.0

Um termo que foi introduzido por Klaus Schwab em 2016 para se referir à modernização dos processos de fabricação por meio de avanços em conectividade, dados em tempo real, automação, análise e IA/ML.

infraestrutura

Todos os recursos e ativos contidos no ambiente de uma aplicação.

Infraestrutura como código (IaC)

O processo de provisionamento e gerenciamento da infraestrutura de uma aplicação por meio de um conjunto de arquivos de configuração. A IaC foi projetada para ajudar você a centralizar o gerenciamento da infraestrutura, padronizar recursos e escalar rapidamente para que novos ambientes sejam reproduzíveis, confiáveis e consistentes.

Internet das Coisas Industrial (IIoT)

O uso de sensores e dispositivos conectados à Internet nos setores industriais, como manufatura, energia, automotivo, saúde, ciências biológicas e agricultura. Para obter mais informações, consulte Construir uma estratégia de transformação digital para a Internet das Coisas Industrial (IIoT).

VPC de inspeção

Em uma arquitetura de AWS várias contas, uma VPC centralizada que gerencia as inspeções do tráfego de rede entre VPCs (na mesma ou em diferentes Regiões da AWS), a Internet e as redes locais. A Arquitetura de referência de segurança da AWS recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

Internet das Coisas (IoT)

A rede de objetos físicos conectados com sensores ou processadores incorporados que se comunicam com outros dispositivos e sistemas pela Internet ou por uma rede de comunicação local. Para obter mais informações, consulte O que é IoT?

interpretabilidade

Uma característica de um modelo de machine learning que descreve o grau em que um ser humano pode entender como as previsões do modelo dependem de suas entradas. Para obter mais informações, consulte Interpretabilidade do modelo de machine learning com a AWS.

IoT

Consulte Internet das Coisas.

Biblioteca de informações de TI (ITIL)

Um conjunto de práticas recomendadas para fornecer serviços de TI e alinhar esses serviços a requisitos de negócios. A ITIL fornece a base para o ITSM.

Gerenciamento de serviços de TI (ITSM)

Atividades associadas a design, implementação, gerenciamento e suporte de serviços de TI para uma organização. Para obter informações sobre a integração de operações em nuvem com ferramentas de ITSM, consulte o guia de integração de operações.

ITIL

Consulte a biblioteca de informações de TI.

ITSM

Veja o gerenciamento de serviços de TI.

L

controle de acesso baseado em etiqueta (LBAC)

Uma implementação do controle de acesso obrigatório (MAC) em que os usuários e os dados em si recebem explicitamente um valor de etiqueta de segurança. A interseção entre a etiqueta de segurança do usuário e a etiqueta de segurança dos dados determina quais linhas e colunas podem ser vistas pelo usuário.

zona de pouso

Uma landing zone é um AWS ambiente bem arquitetado, com várias contas, escalável e seguro. Um ponto a partir do qual suas organizações podem iniciar e implantar rapidamente workloads e aplicações com confiança em seu ambiente de segurança e infraestrutura. Para obter mais

informações sobre zonas de pouso, consulte Configurar um ambiente da AWS com várias contas seguro e escalável.

migração de grande porte

Uma migração de 300 servidores ou mais.

LBAC

Veja controle de acesso baseado em etiquetas.

privilégio mínimo

A prática recomendada de segurança de conceder as permissões mínimas necessárias para executar uma tarefa. Para obter mais informações, consulte Aplicar permissões de privilégios mínimos na documentação do IAM.

mover sem alterações (lift-and-shift)

Veja 7 Rs.

sistema little-endian

Um sistema que armazena o byte menos significativo antes. Veja também endianness.

ambientes inferiores

Veja o ambiente.

M

machine learning (ML)

Um tipo de inteligência artificial que usa algoritmos e técnicas para reconhecimento e aprendizado de padrões. O ML analisa e aprende com dados gravados, por exemplo, dados da Internet das Coisas (IoT), para gerar um modelo estatístico baseado em padrões. Para obter mais informações, consulte Machine learning.

ramificação principal

Veja a filial.

malware

Software projetado para comprometer a segurança ou a privacidade do computador. O malware pode interromper os sistemas do computador, vazar informações confidenciais ou obter acesso

não autorizado. Exemplos de malware incluem vírus, worms, ransomware, cavalos de Tróia, spyware e keyloggers.

serviços gerenciados

Serviços da AWS para o qual AWS opera a camada de infraestrutura, o sistema operacional e as plataformas, e você acessa os endpoints para armazenar e recuperar dados. O Amazon Simple Storage Service (Amazon S3) e o Amazon DynamoDB são exemplos de serviços gerenciados. Eles também são conhecidos como serviços abstratos.

sistema de execução de manufatura (MES)

Um sistema de software para rastrear, monitorar, documentar e controlar processos de produção que convertem matérias-primas em produtos acabados no chão de fábrica.

MAP

Consulte Migration Acceleration Program.

mecanismo

Um processo completo no qual você cria uma ferramenta, impulsiona a adoção da ferramenta e, em seguida, inspeciona os resultados para fazer ajustes. Um mecanismo é um ciclo que se reforça e se aprimora à medida que opera. Para obter mais informações, consulte Construindo mecanismos no AWS Well-Architected Framework.

conta-membro

Todos, Contas da AWS exceto a conta de gerenciamento, que fazem parte de uma organização em AWS Organizations. Uma conta só pode ser membro de uma organização de cada vez.

MES

Veja o sistema de execução de manufatura.

Transporte de telemetria de enfileiramento de mensagens (MQTT)

Um protocolo de comunicação leve machine-to-machine (M2M), baseado no padrão de publicação/assinatura, para dispositivos de IoT com recursos limitados.

microsserviço

Um serviço pequeno e independente que se comunica por meio de APIs bem definidas e normalmente pertence a equipes pequenas e autônomas. Por exemplo, um sistema de seguradora pode incluir microsserviços que mapeiam as capacidades comerciais, como vendas ou marketing, ou subdomínios, como compras, reclamações ou análises. Os benefícios dos

microsserviços incluem agilidade, escalabilidade flexível, fácil implantação, código reutilizável e resiliência. Para obter mais informações, consulte Integração de microsserviços usando serviços sem AWS servidor.

arquitetura de microsserviços

Uma abordagem à criação de aplicações com componentes independentes que executam cada processo de aplicação como um microsserviço. Esses microsserviços se comunicam por meio de uma interface bem definida usando APIs leves. Cada microsserviço nessa arquitetura pode ser atualizado, implantado e escalado para atender à demanda por funções específicas de uma aplicação. Para obter mais informações, consulte Implementação de microsserviços em. AWS

Programa de Aceleração da Migração (MAP)

Um AWS programa que fornece suporte de consultoria, treinamento e serviços para ajudar as organizações a criar uma base operacional sólida para migrar para a nuvem e ajudar a compensar o custo inicial das migrações. O MAP inclui uma metodologia de migração para executar migrações legadas de forma metódica e um conjunto de ferramentas para automatizar e acelerar cenários comuns de migração.

migração em escala

O processo de mover a maior parte do portfólio de aplicações para a nuvem em ondas, com mais aplicações sendo movidas em um ritmo mais rápido a cada onda. Essa fase usa as práticas recomendadas e lições aprendidas nas fases anteriores para implementar uma fábrica de migração de equipes, ferramentas e processos para agilizar a migração de workloads por meio de automação e entrega ágeis. Esta é a terceira fase da estratégia de migração para a AWS.

fábrica de migração

Equipes multifuncionais que simplificam a migração de workloads por meio de abordagens automatizadas e ágeis. As equipes da fábrica de migração geralmente incluem operações, analistas e proprietários de negócios, engenheiros de migração, desenvolvedores e DevOps profissionais que trabalham em sprints. Entre 20 e 50% de um portfólio de aplicações corporativas consiste em padrões repetidos que podem ser otimizados por meio de uma abordagem de fábrica. Para obter mais informações, consulte discussão sobre fábricas de migração e o guia do Cloud Migration Factory neste conjunto de conteúdo.

metadados de migração

As informações sobre a aplicação e o servidor necessárias para concluir a migração. Cada padrão de migração exige um conjunto de metadados de migração diferente. Exemplos de metadados de migração incluem a sub-rede, o grupo de segurança e AWS a conta de destino.

padrão de migração

Uma tarefa de migração repetível que detalha a estratégia de migração, o destino da migração e a aplicação ou o serviço de migração usado. Exemplo: rehospede a migração para o Amazon EC2 AWS com o Application Migration Service.

Avaliação de Portfólio para Migração (MPA)

Uma ferramenta on-line que fornece informações para validar o caso de negócios para migrar para o. Nuvem AWS O MPA fornece avaliação detalhada do portfólio (dimensionamento correto do servidor, preços, comparações de TCO, análise de custos de migração), bem como planejamento de migração (análise e coleta de dados de aplicações, agrupamento de aplicações, priorização de migração e planejamento de ondas). A ferramenta MPA (requer login) está disponível gratuitamente para todos os AWS consultores e consultores parceiros da APN.

Avaliação de Preparação para Migração (MRA)

O processo de obter insights sobre o status de prontidão de uma organização para a nuvem, identificar pontos fortes e fracos e criar um plano de ação para fechar as lacunas identificadas, usando o CAF. AWS Para mais informações, consulte o guia de preparação para migração. A MRA é a primeira fase da estratégia de migração para a AWS.

estratégia de migração

A abordagem usada para migrar uma carga de trabalho para o. Nuvem AWS Para obter mais informações, consulte a entrada de 7 Rs neste glossário e consulte Mobilize sua organização para acelerar migrações em grande escala.

ML

Veja o aprendizado de máquina.

modernização

Transformar uma aplicação desatualizada (herdada ou monolítica) e sua infraestrutura em um sistema ágil, elástico e altamente disponível na nuvem para reduzir custos, ganhar eficiência e aproveitar as inovações. Para obter mais informações, consulte Estratégia para modernizar aplicativos no Nuvem AWS.

avaliação de preparação para modernização

Uma avaliação que ajuda a determinar a preparação para modernização das aplicações de uma organização. Ela identifica benefícios, riscos e dependências e determina o quão bem a organização pode acomodar o estado futuro dessas aplicações. O resultado da avaliação é um

esquema da arquitetura de destino, um roteiro que detalha as fases de desenvolvimento e os marcos do processo de modernização e um plano de ação para abordar as lacunas identificadas. Para obter mais informações, consulte Avaliação da prontidão para modernização de aplicativos no. Nuvem AWS

aplicações monolíticas (monólitos)

Aplicações que são executadas como um único serviço com processos fortemente acoplados. As aplicações monolíticas apresentam várias desvantagens. Se um recurso da aplicação apresentar um aumento na demanda, toda a arquitetura deverá ser escalada. Adicionar ou melhorar os recursos de uma aplicação monolítica também se torna mais complexo quando a base de código cresce. Para resolver esses problemas, é possível criar uma arquitetura de microsserviços. Para obter mais informações, consulte Decompor monólitos em microsserviços.

MAPA

Consulte Avaliação do portfólio de migração.

MQTT

Consulte Transporte de telemetria de enfileiramento de mensagens.

classificação multiclasse

Um processo que ajuda a gerar previsões para várias classes (prevendo um ou mais de dois resultados). Por exemplo, um modelo de ML pode perguntar “Este produto é um livro, um carro ou um telefone?” ou “Qual categoria de produtos é mais interessante para este cliente?”

infraestrutura mutável

Um modelo que atualiza e modifica a infraestrutura existente para cargas de trabalho de produção. Para melhorar a consistência, confiabilidade e previsibilidade, o AWS Well-Architected Framework recomenda o uso de infraestrutura imutável como uma prática recomendada.

O

OAC

Veja o controle de acesso de origem.

CARVALHO

Veja a identidade de acesso de origem.

OCM

Veja o gerenciamento de mudanças organizacionais.

migração offline

Um método de migração no qual a workload de origem é desativada durante o processo de migração. Esse método envolve tempo de inatividade prolongado e geralmente é usado para workloads pequenas e não críticas.

OI

Veja a integração de operações.

OLA

Veja o contrato em nível operacional.

migração online

Um método de migração no qual a workload de origem é copiada para o sistema de destino sem ser colocada offline. As aplicações conectadas à workload podem continuar funcionando durante a migração. Esse método envolve um tempo de inatividade nulo ou mínimo e normalmente é usado para workloads essenciais para a produção.

OPC-UA

Consulte Comunicação de processo aberto — Arquitetura unificada.

Comunicação de processo aberto — Arquitetura unificada (OPC-UA)

Um protocolo de comunicação machine-to-machine (M2M) para automação industrial. O OPC-UA fornece um padrão de interoperabilidade com esquemas de criptografia, autenticação e autorização de dados.

acordo de nível operacional (OLA)

Um acordo que esclarece o que os grupos funcionais de TI prometem oferecer uns aos outros para apoiar um acordo de serviço (SLA).

análise de prontidão operacional (ORR)

Uma lista de verificação de perguntas e melhores práticas associadas que ajudam você a entender, avaliar, prevenir ou reduzir o escopo de incidentes e possíveis falhas. Para obter mais informações, consulte Operational Readiness Reviews (ORR) no Well-Architected AWS Framework.

tecnologia operacional (OT)

Sistemas de hardware e software que funcionam com o ambiente físico para controlar operações, equipamentos e infraestrutura industriais. Na manufatura, a integração dos sistemas OT e de tecnologia da informação (TI) é o foco principal das transformações da Indústria 4.0.

integração de operações (OI)

O processo de modernização das operações na nuvem, que envolve planejamento de preparação, automação e integração. Para obter mais informações, consulte o guia de integração de operações.

trilha organizacional

Uma trilha criada por ela AWS CloudTrail registra todos os eventos de todos Contas da AWS em uma organização em AWS Organizations. Essa trilha é criada em cada Conta da AWS que faz parte da organização e monitora a atividade em cada conta. Para obter mais informações, consulte Criação de uma trilha para uma organização na CloudTrail documentação.

gerenciamento de alterações organizacionais (OCM)

Uma estrutura para gerenciar grandes transformações de negócios disruptivas de uma perspectiva de pessoas, cultura e liderança. O OCM ajuda as organizações a se prepararem e fazerem a transição para novos sistemas e estratégias, acelerando a adoção de alterações, abordando questões de transição e promovendo mudanças culturais e organizacionais. Na estratégia de AWS migração, essa estrutura é chamada de aceleração de pessoas, devido à velocidade de mudança exigida nos projetos de adoção da nuvem. Para obter mais informações, consulte o guia do OCM.

controle de acesso de origem (OAC)

Em CloudFront, uma opção aprimorada para restringir o acesso para proteger seu conteúdo do Amazon Simple Storage Service (Amazon S3). O OAC oferece suporte a todos os buckets S3 Regiões da AWS, criptografia do lado do servidor com AWS KMS (SSE-KMS) e solicitações dinâmicas ao bucket S3. `PUT` `DELETE`

Identidade do acesso de origem (OAI)

Em CloudFront, uma opção para restringir o acesso para proteger seu conteúdo do Amazon S3. Quando você usa o OAI, CloudFront cria um principal com o qual o Amazon S3 pode se autenticar. Os diretores autenticados podem acessar o conteúdo em um bucket do S3 somente por meio de uma distribuição específica. CloudFront Veja também OAC, que fornece um controle de acesso mais granular e aprimorado.

OU

Veja a análise de prontidão operacional.

NÃO

Veja a tecnologia operacional.

VPC de saída (egresso)

Em uma arquitetura de AWS várias contas, uma VPC que gerencia conexões de rede que são iniciadas de dentro de um aplicativo. A Arquitetura de referência de segurança da AWS recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

P

limite de permissões

Uma política de gerenciamento do IAM anexada a entidades principais do IAM para definir as permissões máximas que o usuário ou perfil podem ter. Para obter mais informações, consulte Limites de permissões na documentação do IAM.

Informações de identificação pessoal (PII)

Informações que, quando visualizadas diretamente ou combinadas com outros dados relacionados, podem ser usadas para inferir razoavelmente a identidade de um indivíduo. Exemplos de PII incluem nomes, endereços e informações de contato.

PII

Veja informações de identificação pessoal.

manual

Um conjunto de etapas predefinidas que capturam o trabalho associado às migrações, como a entrega das principais funções operacionais na nuvem. Um manual pode assumir a forma de scripts, runbooks automatizados ou um resumo dos processos ou etapas necessários para operar seu ambiente modernizado.

PLC

Consulte controlador lógico programável.

AMEIXA

Veja o gerenciamento do ciclo de vida do produto.

política

Um objeto que pode definir permissões (consulte a política baseada em identidade), especificar as condições de acesso (consulte a política baseada em recursos) ou definir as permissões máximas para todas as contas em uma organização em AWS Organizations (consulte a política de controle de serviços).

persistência poliglota

Escolher de forma independente a tecnologia de armazenamento de dados de um microsserviço com base em padrões de acesso a dados e outros requisitos. Se seus microsserviços tiverem a mesma tecnologia de armazenamento de dados, eles poderão enfrentar desafios de implementação ou apresentar baixa performance. Os microsserviços serão implementados com mais facilidade e alcançarão performance e escalabilidade melhores se usarem o armazenamento de dados mais bem adaptado às suas necessidades. Para obter mais informações, consulte Habilitar a persistência de dados em microsserviços.

avaliação do portfólio

Um processo de descobrir, analisar e priorizar o portfólio de aplicações para planejar a migração. Para obter mais informações, consulte Avaliar a preparação para a migração.

predicado

Uma condição de consulta que retorna `true` ou `false`, normalmente localizada em uma `WHERE` cláusula.

pressão de predicados

Uma técnica de otimização de consulta de banco de dados que filtra os dados na consulta antes da transferência. Isso reduz a quantidade de dados que devem ser recuperados e processados do banco de dados relacional e melhora o desempenho das consultas.

controle preventivo

Um controle de segurança projetado para evitar que um evento ocorra. Esses controles são a primeira linha de defesa para ajudar a evitar acesso não autorizado ou alterações indesejadas em sua rede. Para obter mais informações, consulte Controles preventivos em Como implementar controles de segurança na AWS.

principal (entidade principal)

Uma entidade AWS que pode realizar ações e acessar recursos. Essa entidade geralmente é um usuário raiz para um Conta da AWS, uma função do IAM ou um usuário. Para obter mais informações, consulte Entidade principal em Termos e conceitos de perfis na documentação do IAM.

Privacidade por design

Uma abordagem em engenharia de sistemas que leva em consideração a privacidade em todo o processo de engenharia.

zonas hospedadas privadas

Um contêiner que armazena informações sobre como você quer que o Amazon Route 53 responda a consultas ao DNS para um domínio e seus subdomínios dentro de uma ou mais VPCs. Para obter mais informações, consulte Como trabalhar com zonas hospedadas privadas na documentação do Route 53.

controle proativo

Um controle de segurança projetado para impedir a implantação de recursos não compatíveis. Esses controles examinam os recursos antes de serem provisionados. Se o recurso não estiver em conformidade com o controle, ele não será provisionado. Para obter mais informações, consulte o guia de referência de controles na AWS Control Tower documentação e consulte Controles proativos em Implementação de controles de segurança em AWS.

gerenciamento do ciclo de vida do produto (PLM)

O gerenciamento de dados e processos de um produto em todo o seu ciclo de vida, desde o design, desenvolvimento e lançamento, passando pelo crescimento e maturidade, até o declínio e a remoção.

ambiente de produção

Veja o ambiente.

controlador lógico programável (PLC)

Na fabricação, um computador altamente confiável e adaptável que monitora as máquinas e automatiza os processos de fabricação.

pseudonimização

O processo de substituir identificadores pessoais em um conjunto de dados por valores de espaço reservado. A pseudonimização pode ajudar a proteger a privacidade pessoal. Os dados pseudonimizados ainda são considerados dados pessoais.

publicar/assinar (pub/sub)

Um padrão que permite comunicações assíncronas entre microsserviços para melhorar a escalabilidade e a capacidade de resposta. Por exemplo, em um MES baseado em microsserviços, um microsserviço pode publicar mensagens de eventos em um canal no qual outros microsserviços possam se inscrever. O sistema pode adicionar novos microsserviços sem alterar o serviço de publicação.

Q

plano de consulta

Uma série de etapas, como instruções, usadas para acessar os dados em um sistema de banco de dados relacional SQL.

regressão de planos de consultas

Quando um otimizador de serviço de banco de dados escolhe um plano menos adequado do que escolhia antes de uma determinada alteração no ambiente de banco de dados ocorrer. Isso pode ser causado por alterações em estatísticas, restrições, configurações do ambiente, associações de parâmetros de consulta e atualizações do mecanismo de banco de dados.

R

Matriz RACI

Veja responsável, responsável, consultado, informado (RACI).

ransomware

Um software mal-intencionado desenvolvido para bloquear o acesso a um sistema ou dados de computador até que um pagamento seja feito.

Matriz RASCI

Veja responsável, responsável, consultado, informado (RACI).

RCAC

Veja o controle de acesso por linha e coluna.

réplica de leitura

Uma cópia de um banco de dados usada somente para leitura. É possível encaminhar consultas para a réplica de leitura e reduzir a carga no banco de dados principal.

rearquiteta

Veja 7 Rs.

objetivo de ponto de recuperação (RPO).

O máximo período 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.

objetivo de tempo de recuperação (RTO)

O máximo atraso aceitável entre a interrupção e a restauração do serviço.

refatorar

Veja 7 Rs.

Região

Uma coleção de AWS recursos em uma área geográfica. Cada um Região da AWS é isolado e independente dos outros para fornecer tolerância a falhas, estabilidade e resiliência. Para obter mais informações, consulte Especificar o que Regiões da AWS sua conta pode usar.

regressão

Uma técnica de ML que prevê um valor numérico. Por exemplo, para resolver o problema de “Por qual preço esta casa será vendida?” um modelo de ML pode usar um modelo de regressão linear para prever o preço de venda de uma casa com base em fatos conhecidos sobre a casa (por exemplo, a metragem quadrada).

redefinir a hospedagem

Veja 7 Rs.

versão

Em um processo de implantação, o ato de promover mudanças em um ambiente de produção.

realocar

Veja 7 Rs.

redefinir a plataforma

Veja 7 Rs.

recomprar

Veja 7 Rs.

resiliência

A capacidade de um aplicativo de resistir ou se recuperar de interrupções. Alta disponibilidade e recuperação de desastres são considerações comuns ao planejar a resiliência no. Nuvem AWS Para obter mais informações, consulte Nuvem AWS Resiliência.

política baseada em recurso

Uma política associada a um recurso, como um bucket do Amazon S3, um endpoint ou uma chave de criptografia. Esse tipo de política especifica quais entidades principais têm acesso permitido, ações válidas e quaisquer outras condições que devem ser atendidas.

matriz responsável, accountable, consultada, informada (RACI)

Uma matriz que define as funções e responsabilidades de todas as partes envolvidas nas atividades de migração e nas operações de nuvem. O nome da matriz é derivado dos tipos de responsabilidade definidos na matriz: responsável (R), responsabilizável (A), consultado (C) e informado (I). O tipo de suporte (S) é opcional. Se você incluir suporte, a matriz será chamada de matriz RASCI e, se excluir, será chamada de matriz RACI.

controle responsivo

Um controle de segurança desenvolvido para conduzir a remediação de eventos adversos ou desvios em relação à linha de base de segurança. Para obter mais informações, consulte Controles responsivos em Como implementar controles de segurança na AWS.

reter

Veja 7 Rs.

aposentar-se

Veja 7 Rs.

rotação

O processo de atualizar periodicamente um segredo para dificultar o acesso das credenciais por um invasor.

controle de acesso por linha e coluna (RCAC)

O uso de expressões SQL básicas e flexíveis que tenham regras de acesso definidas. O RCAC consiste em permissões de linha e máscaras de coluna.

RPO

Veja o objetivo do ponto de recuperação.

RTO

Veja o objetivo do tempo de recuperação.

runbook

Um conjunto de procedimentos manuais ou automatizados necessários para realizar uma tarefa específica. Eles são normalmente criados para agilizar operações ou procedimentos repetitivos com altas taxas de erro.

S

SAML 2.0

Um padrão aberto que muitos provedores de identidade (IdPs) usam. Esse recurso permite o login único federado (SSO), para que os usuários possam fazer login AWS Management Console ou chamar as operações da AWS API sem que você precise criar um usuário no IAM para todos em sua organização. Para obter mais informações sobre a federação baseada em SAML 2.0, consulte Sobre a federação baseada em SAML 2.0 na documentação do IAM.

SCADA

Veja controle de supervisão e aquisição de dados.

SCP

Veja a política de controle de serviços.

secret

Em AWS Secrets Manager, informações confidenciais ou restritas, como uma senha ou credenciais de usuário, que você armazena de forma criptografada. Ele consiste no valor secreto

e em seus metadados. O valor secreto pode ser binário, uma única string ou várias strings. Para obter mais informações, consulte O que há em um segredo do Secrets Manager? na documentação do Secrets Manager.

controle de segurança

Uma barreira de proteção técnica ou administrativa que impede, detecta ou reduz a capacidade de uma ameaça explorar uma vulnerabilidade de segurança. Existem quatro tipos principais de controles de segurança: preventivos, detectivos, responsivos e proativos.

fortalecimento da segurança

O processo de reduzir a superfície de ataque para torná-la mais resistente a ataques. Isso pode incluir ações como remover recursos que não são mais necessários, implementar a prática recomendada de segurança de conceder privilégios mínimos ou desativar recursos desnecessários em arquivos de configuração.

sistema de gerenciamento de eventos e informações de segurança (SIEM)

Ferramentas e serviços que combinam sistemas de gerenciamento de informações de segurança (SIM) e gerenciamento de eventos de segurança (SEM). Um sistema SIEM coleta, monitora e analisa dados de servidores, redes, dispositivos e outras fontes para detectar ameaças e violações de segurança e gerar alertas.

automação de resposta de segurança

Uma ação predefinida e programada projetada para responder ou remediar automaticamente um evento de segurança. Essas automações servem como controles de segurança responsivos ou detectivos que ajudam você a implementar as melhores práticas AWS de segurança. Exemplos de ações de resposta automatizada incluem a modificação de um grupo de segurança da VPC, a correção de uma instância do Amazon EC2 ou a rotação de credenciais.

Criptografia do lado do servidor

Criptografia dos dados em seu destino, por AWS service (Serviço da AWS) quem os recebe.

política de controle de serviços (SCP)

Uma política que fornece controle centralizado sobre as permissões de todas as contas em uma organização no AWS Organizations. As SCPs definem barreiras de proteção ou estabelecem limites para as ações que um administrador pode delegar a usuários ou perfis. É possível usar SCPs como listas de permissão ou de negação para especificar quais serviços ou ações são permitidos ou proibidos. Para obter mais informações, consulte Políticas de controle de serviço na AWS Organizations documentação.

service endpoint (endpoint de serviço)

O URL do ponto de entrada para um AWS service (Serviço da AWS). Você pode usar o endpoint para se conectar programaticamente ao serviço de destino. Para obter mais informações, consulte Endpoints do AWS service (Serviço da AWS) na Referência geral da AWS.

acordo de serviço (SLA)

Um acordo que esclarece o que uma equipe de TI promete fornecer aos clientes, como tempo de atividade e performance do serviço.

indicador de nível de serviço (SLI)

Uma medida de um aspecto de desempenho de um serviço, como taxa de erro, disponibilidade ou taxa de transferência.

objetivo de nível de serviço (SLO)

Uma métrica alvo que representa a integridade de um serviço, conforme medida por um indicador de nível de serviço.

modelo de responsabilidade compartilhada

Um modelo que descreve a responsabilidade com a qual você compartilha AWS pela segurança e conformidade na nuvem. AWS é responsável pela segurança da nuvem, enquanto você é responsável pela segurança na nuvem. Para obter mais informações, consulte o Modelo de responsabilidade compartilhada.

SIEM

Veja informações de segurança e sistema de gerenciamento de eventos.

ponto único de falha (SPOF)

Uma falha em um único componente crítico de um aplicativo que pode interromper o sistema.

SLA

Veja o contrato de nível de serviço.

ESGUIO

Veja o indicador de nível de serviço.

SLO

Veja o objetivo do nível de serviço.

split-and-seed modelo

Um padrão para escalar e acelerar projetos de modernização. À medida que novos recursos e lançamentos de produtos são definidos, a equipe principal se divide para criar novas equipes de produtos. Isso ajuda a escalar os recursos e os serviços da sua organização, melhora a produtividade do desenvolvedor e possibilita inovações rápidas. Para obter mais informações, consulte Abordagem em fases para modernizar aplicativos no. Nuvem AWS

CUSPE

Veja um único ponto de falha.

esquema de estrelas

Uma estrutura organizacional de banco de dados que usa uma grande tabela de fatos para armazenar dados transacionais ou medidos e usa uma ou mais tabelas dimensionais menores para armazenar atributos de dados. Essa estrutura foi projetada para uso em um data warehouse ou para fins de inteligência comercial.

padrão strangler fig

Uma abordagem à modernização de sistemas monolíticos que consiste em reescrever e substituir incrementalmente a funcionalidade do sistema até que o sistema herdado possa ser desativado. Esse padrão usa a analogia de uma videira que cresce e se torna uma árvore estabelecida e, eventualmente, supera e substitui sua hospedeira. O padrão foi apresentado por Martin Fowler como forma de gerenciar riscos ao reescrever sistemas monolíticos. Para ver um exemplo de como aplicar esse padrão, consulte Modernizar incrementalmente os serviços Web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

sub-rede

Um intervalo de endereços IP na VPC. Uma sub-rede deve residir em uma única zona de disponibilidade.

controle de supervisão e aquisição de dados (SCADA)

Na manufatura, um sistema que usa hardware e software para monitorar ativos físicos e operações de produção.

symmetric encryption (criptografia simétrica)

Um algoritmo de criptografia que usa a mesma chave para criptografar e descriptografar dados.

testes sintéticos

Testar um sistema de forma que simule as interações do usuário para detectar possíveis problemas ou monitorar o desempenho. Você pode usar o Amazon CloudWatch Synthetics para criar esses testes.

T

tags

Pares de valores-chave que atuam como metadados para organizar seus recursos. AWS As tags podem ajudar você a gerenciar, identificar, organizar, pesquisar e filtrar recursos. Para obter mais informações, consulte Marcar seus recursos do AWS.

variável-alvo

O valor que você está tentando prever no ML supervisionado. Ela também é conhecida como variável de resultado. Por exemplo, em uma configuração de fabricação, a variável-alvo pode ser um defeito do produto.

lista de tarefas

Uma ferramenta usada para monitorar o progresso por meio de um runbook. Uma lista de tarefas contém uma visão geral do runbook e uma lista de tarefas gerais a serem concluídas. Para cada tarefa geral, ela inclui o tempo estimado necessário, o proprietário e o progresso.

ambiente de teste

Veja o ambiente.

treinamento

O processo de fornecer dados para que seu modelo de ML aprenda. Os dados de treinamento devem conter a resposta correta. O algoritmo de aprendizado descobre padrões nos dados de treinamento que mapeiam os atributos dos dados de entrada no destino (a resposta que você deseja prever). Ele gera um modelo de ML que captura esses padrões. Você pode usar o modelo de ML para obter previsões de novos dados cujo destino você não conhece.

gateway de trânsito

Um hub de trânsito de rede que pode ser usado para interconectar as VPCs e as redes on-premises. Para obter mais informações, consulte O que é um gateway de trânsito na AWS Transit Gateway documentação.

fluxo de trabalho baseado em troncos

Uma abordagem na qual os desenvolvedores criam e testam recursos localmente em uma ramificação de recursos e, em seguida, mesclam essas alterações na ramificação principal. A ramificação principal é então criada para os ambientes de desenvolvimento, pré-produção e produção, sequencialmente.

Acesso confiável

Conceder permissões a um serviço que você especifica para realizar tarefas em sua organização AWS Organizations e em suas contas em seu nome. O serviço confiável cria um perfil vinculado ao serviço em cada conta, quando esse perfil é necessário, para realizar tarefas de gerenciamento para você. Para obter mais informações, consulte Usando AWS Organizations com outros AWS serviços na AWS Organizations documentação.

tuning (ajustar)

Alterar aspectos do processo de treinamento para melhorar a precisão do modelo de ML. Por exemplo, você pode treinar o modelo de ML gerando um conjunto de rótulos, adicionando rótulos e repetindo essas etapas várias vezes em configurações diferentes para otimizar o modelo.

equipe de duas pizzas

Uma pequena DevOps equipe que você pode alimentar com duas pizzas. Uma equipe de duas pizzas garante a melhor oportunidade possível de colaboração no desenvolvimento de software.

U

incerteza

Um conceito que se refere a informações imprecisas, incompletas ou desconhecidas que podem minar a confiabilidade dos modelos preditivos de ML. Há dois tipos de incertezas: a incerteza epistêmica é causada por dados limitados e incompletos, enquanto a incerteza aleatória é causada pelo ruído e pela aleatoriedade inerentes aos dados. Para obter mais informações, consulte o guia Como quantificar a incerteza em sistemas de aprendizado profundo.

tarefas indiferenciadas

Também conhecido como trabalho pesado, trabalho necessário para criar e operar um aplicativo, mas que não fornece valor direto ao usuário final nem oferece vantagem competitiva. Exemplos de tarefas indiferenciadas incluem aquisição, manutenção e planejamento de capacidade.

ambientes superiores

Veja o ambiente.

V

aspiração

Uma operação de manutenção de banco de dados que envolve limpeza após atualizações incrementais para recuperar armazenamento e melhorar a performance.

controle de versões

Processos e ferramentas que rastreiam mudanças, como alterações no código-fonte em um repositório.

emparelhamento de VPC

Uma conexão entre duas VPCs que permite rotear tráfego usando endereços IP privados. Para ter mais informações, consulte O que é emparelhamento de VPC? na documentação da Amazon VPC.

Vulnerabilidade

Uma falha de software ou hardware que compromete a segurança do sistema.

W

cache quente

Um cache de buffer que contém dados atuais e relevantes que são acessados com frequência. A instância do banco de dados pode ler do cache do buffer, o que é mais rápido do que ler da memória principal ou do disco.

dados mornos

Dados acessados raramente. Ao consultar esse tipo de dados, consultas moderadamente lentas geralmente são aceitáveis.

função de janela

Uma função SQL que executa um cálculo em um grupo de linhas que se relacionam de alguma forma com o registro atual. As funções de janela são úteis para processar tarefas, como calcular uma média móvel ou acessar o valor das linhas com base na posição relativa da linha atual.

workload

Uma coleção de códigos e recursos que geram valor empresarial, como uma aplicação voltada para o cliente ou um processo de back-end.

workstreams

Grupos funcionais em um projeto de migração que são responsáveis por um conjunto específico de tarefas. Cada workstream é independente, mas oferece suporte aos outros workstreams do projeto. Por exemplo, o workstream de portfólio é responsável por priorizar aplicações, planejar ondas e coletar metadados de migração. O workstream de portfólio entrega esses ativos ao workstream de migração, que então migra os servidores e as aplicações.

MINHOCA

Veja escrever uma vez, ler muitas.

WQF

Consulte o AWS Workload Qualification Framework.

escreva uma vez, leia muitas (WORM)

Um modelo de armazenamento que grava dados uma única vez e evita que os dados sejam excluídos ou modificados. Os usuários autorizados podem ler os dados quantas vezes forem necessárias, mas não podem alterá-los. Essa infraestrutura de armazenamento de dados é considerada imutável.

Z

exploração de dia zero

Um ataque, geralmente malware, que tira proveito de uma vulnerabilidade de dia zero.

vulnerabilidade de dia zero

Uma falha ou vulnerabilidade não mitigada em um sistema de produção. Os agentes de ameaças podem usar esse tipo de vulnerabilidade para atacar o sistema. Os desenvolvedores frequentemente ficam cientes da vulnerabilidade como resultado do ataque.

aplicação zumbi

Uma aplicação que tem um uso médio de CPU e memória inferior a 5%. Em um projeto de migração, é comum retirar essas aplicações.

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