



Construindo arquiteturas hexagonais em AWS

AWS Recomendações



AWS Recomendações: Construindo arquiteturas hexagonais em AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

Introdução	1
Visão geral	3
Design orientado por domínio (DDD)	3
Arquitetura hexagonal	3
Resultados de negócios desejados	5
Melhorando o ciclo de desenvolvimento	6
Testes na nuvem	6
Testes no local	6
Paralelização do desenvolvimento	7
Tempo de comercialização do produto	7
Qualidade por design	8
Alterações localizadas e maior legibilidade	8
Testando primeiro a lógica de negócios	8
Manutenibilidade	9
Adaptando-se à mudança	10
Adaptação aos novos requisitos não funcionais usando portas e adaptadores	10
Adaptando-se aos novos requisitos de negócios usando comandos e manipuladores de comandos	10
Desacoplamento de componentes usando a fachada de serviço ou o padrão CQRS	11
Escalabilidade organizacional	12
Práticas recomendadas	14
Modele o domínio comercial	14
Escreva e execute testes desde o início	14
Defina o comportamento do domínio	15
Automatize os testes e a implantação	15
Dimensione seu produto usando microsserviços e CQRS	15
Crie uma estrutura de projeto que mapeie conceitos de arquitetura hexagonal	16
Exemplos de infraestrutura	18
Comece de forma simples	18
Aplique o padrão CQRS	19
Evolua a arquitetura adicionando contêineres, um banco de dados relacional e uma API externa	20
Adicionar mais domínios (diminuir o zoom)	21
Perguntas frequentes	23

Por que eu deveria usar uma arquitetura hexagonal?	23
Por que eu deveria usar o design orientado por domínio?	23
Posso praticar o desenvolvimento orientado a testes sem arquitetura hexagonal?	23
Posso escalar meu produto sem arquitetura hexagonal e design orientado por domínio?	23
Quais tecnologias devo usar para implementar a arquitetura hexagonal?	23
Estou desenvolvendo um produto mínimo viável. Faz sentido gastar tempo pensando em arquitetura de software?	24
Estou desenvolvendo um produto mínimo viável e não tenho tempo para escrever testes.	24
Quais padrões de design adicionais posso usar com a arquitetura hexagonal?	24
Próximas etapas	25
Recursos	26
Histórico do documento	28
Glossário	29
#	29
A	30
B	33
C	35
D	39
E	43
F	45
G	47
H	48
eu	50
L	52
M	54
O	58
P	61
Q	64
R	64
S	67
T	71
U	73
V	73
W	74
Z	75
.....	lxxvi

Construindo arquiteturas hexagonais em AWS

Furkan Oruc, Dominik Goby, Darius Kuncce e Michal Ploski, da 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 do produto 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 introduções de novos serviços e recursos faz com que as partes interessadas do negócio esperem que suas equipes de desenvolvimento criem protótipos de novos produtos mínimos viáveis (MVPs) com mais rapidez, para que novas ideias possam ser testadas e verificadas o mais rápido possível. Muitas vezes, eles MVPs são adotados e se tornam parte do ecossistema de software corporativo. No processo de produção deles MVPs, as equipes às vezes abandonam as regras e as melhores práticas de desenvolvimento de software, como [os princípios SOLID e os testes unitários](#). Eles presumem que essa abordagem acelerará o desenvolvimento e reduzirá o tempo de lançamento no mercado. 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 as mudanças nos requisitos também podem atrasar a equipe durante o processo de desenvolvimento.

Este guia aborda uma arquitetura de software proposta, desde uma arquitetura hexagonal de baixo nível até 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 a escalar a equipe de engenharia à medida que novos recursos são desenvolvidos. Ele alinha as partes interessadas comerciais e técnicas aos problemas comerciais, 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 desvia 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 comandos é uma forma de aplicar a responsabilidade única e os princípios de aberto-fechado do SOLID](#). A adesão a esses princípios produz uma base de código que os desenvolvedores e arquitetos que trabalham no projeto podem facilmente navegar e entender, além de reduzir o risco de introduzir alterações significativas na funcionalidade existente.

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 seu aplicativo AWS que ofereça suporte à arquitetura hexagonal. Para ver um exemplo de implementação, consulte [Estruturar um projeto Python em arquitetura hexagonal usando](#) o site AWS Prescriptive AWS Lambda 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 externos APIs 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 na [Resultados de negócios desejados](#) 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 ao DDD, em que um domínio é o núcleo do aplicativo que resolve um problema comercial. 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.

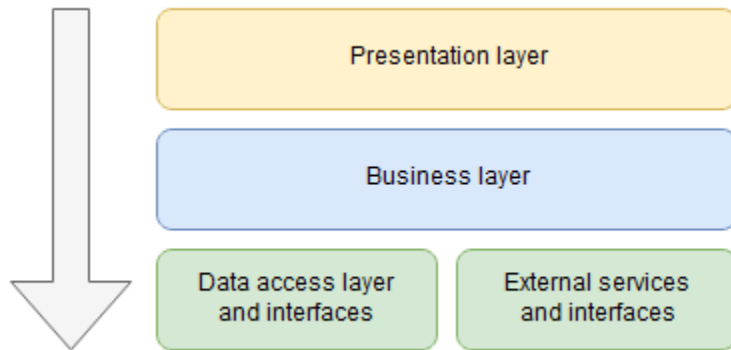
Este guia não aborda todos os aspectos do DDD, que é um tópico muito amplo. Para entender melhor, você pode revisar os recursos do DDD listados no site da [Domain Language](#).

Arquitetura hexagonal

A arquitetura hexagonal, também conhecida como portas e adaptadores ou arquitetura onion, é um princípio de gerenciamento da inversão de dependências 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 adotar boas práticas, como o desenvolvimento orientado a testes (TDD), que, por sua vez, promove a [evolução da arquitetura](#) e ajuda você a gerenciar domínios complexos a longo prazo.

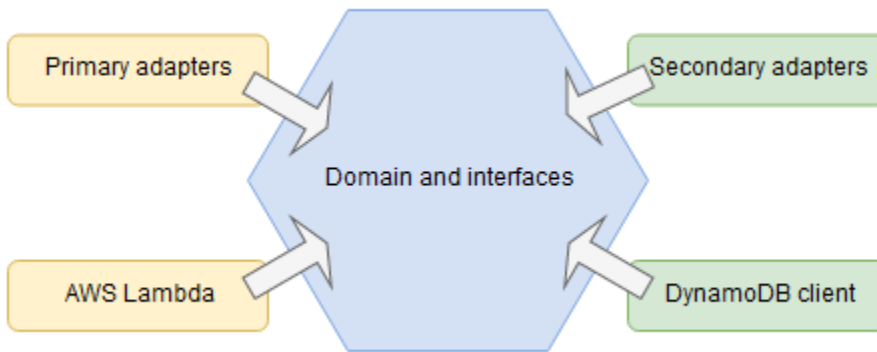
Vamos comparar a arquitetura hexagonal com a arquitetura clássica em camadas, que é a escolha mais popular para modelar projetos de software estruturados. Há diferenças sutis, mas poderosas, entre as duas abordagens.

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 têm dependências nas 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 de domínio. A camada de negócios, por sua vez, tem dependências na camada de acesso a dados e em vários serviços externos.



A principal desvantagem dessa configuração é a estrutura de dependências. Por exemplo, se o modelo para armazenar dados no banco de dados for alterado, 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 os relacionamentos de dependência de uma maneira 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. As 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 mundo 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 externos, usuários e serviços interajam com a lógica central. AWS Lambda é um bom exemplo de adaptador primário. Ele se integra a vários AWS 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 desejados

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

- [Reduza o tempo de lançamento no mercado 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, pois 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 lançamento no mercado.

Testes 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 permissões (IAM) AWS Identity and Access Management, estejam configurados corretamente. Também pode ser a única maneira confiável de testar integrações de componentes. Embora alguns AWS 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](#), [LocalStack](#) que simulam AWS serviços para fins de teste, podem não refletir com precisão os contratos reais de API de serviços 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, 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 mudanças na lógica de negócios levará uma hora ou mais. Se a lógica de negócios for mais complexa, os testes podem exigir dias de espera até que as implantações sejam concluídas. Se você tem vários recursos e engenheiros na equipe, o período prolongado rapidamente se torna perceptível para a empresa.

Testes no local

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 de domínio.

Você não precisará perder tempo resolvendo problemas de integração técnica ou implantando 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 representa uma redução significativa no tempo necessário para detectar erros. A [Testando 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 de unidade local melhora o ciclo de feedback do desenvolvimento e reduz o tempo de lançamento de novos produtos ou recursos no mercado, especialmente quando eles contêm uma lógica comercial complexa, conforme explicado anteriormente. Além disso, o aumento da cobertura de código por meio de testes unitários reduz significativamente o risco de introduzir bugs de regressão ao atualizar ou refatorar a base de código. A cobertura do teste unitário 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 detalhadamente na [Qualidade por 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 na [Adaptando-se à mudança](#) seção.

Qualidade por 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.

Alterações 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 isolados.

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

Testando primeiro a lógica de negócios

Os testes locais podem ser realizados introduzindo end-to-end, integrando e testando unidades no projeto. End-to-endos testes 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 saídas esperadas. No entanto, adicionar mais cenários secundários pode se tornar complexo, porque 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 exigidos pela ação comercial e, em seguida, afirmar o resultado. Configurar o ambiente para o cenário de teste e afirmar os resultados pode levar muito tempo dos desenvolvedores.

Na arquitetura hexagonal, você testa a lógica de negócios isoladamente e usa testes de integração para testar adaptadores secundários. Você pode usar adaptadores simulados ou falsos em seus testes de lógica comercial. Você também pode combinar os testes para casos de uso comercial com testes unitários para seu modelo de domínio para manter uma 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 a testes (TDD) e começar a definir entidades de domínio ou casos de uso de negócios com testes adequados logo no início do desenvolvimento. Escrever os testes primeiro ajuda você a criar implementações simuladas das interfaces exigidas pelo domínio. Quando os testes são bem-sucedidos e as regras de lógica de domínio são satisfeitas, você pode 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 cheguem à produção.

Se você usa implantações sem servidor, pode provisionar rapidamente uma instância do aplicativo em sua AWS conta para integração e teste manuais. end-to-end Após essas etapas de implementação, recomendamos que você automatize os testes a cada nova alteração enviada ao repositório.

Manutenibilidade

A capacidade de manutenção se refere à operação e 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 futuros. 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 da web para que seu código exija menos trabalho em geral. Ao separar módulos, localizar alterações e desacoplar a lógica de negócios do aplicativo da implementação do adaptador, você pode produzir métricas e registros que ajudem os operadores a obter uma compreensão profunda do sistema e entender o escopo de 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 um 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 alteração. 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 de negócios.

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 externo 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, então 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 haveria necessidade de alterar nada no modelo de domínio. Portanto, a arquitetura hexagonal se adapta facilmente aos requisitos não funcionais.

Adaptando-se 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. A vantagem da inversão de dependências 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 comercial que você está tentando resolver, atualizar o modelo de domínio para se adaptar às mudanças nos requisitos comerciais não é um problema.

Quando você desenvolve software, a separação de interesses 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 pelos 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 filas síncronas APIs ou assíncronas, para executar comandos. Recomendamos que você use comandos e manipuladores de comando para cada operação no domínio. Ao seguir 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 livremente as solicitações de leitura e gravação recebidas dos clientes ao domínio. Há duas maneiras de obter esse acoplamento frouxo: usando um padrão de fachada de serviço ou usando o padrão de segregação de responsabilidade por consulta de comando (CQRS).

O [padrão de fachada de serviço](#) fornece uma interface frontal para atender clientes, como a camada de apresentação ou um microsserviço. Uma fachada de serviço 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 em várias operações. No entanto, ao usar a fachada de serviço, é mais difícil seguir os princípios de [responsabilidade única e de aberto-fechado](#). 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 serviço se estende, todas as operações são reunidas 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 manipuladores 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 a 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 que seja 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 que você avalie seu projeto cuidadosamente 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 read/write as operações. 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 as 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 dessa abordagem é que o desenvolvimento de produtos e recursos sempre envolve questões transversais, como segurança e escalabilidade, que exigem comunicação constante entre as equipes. Estruturar equipes com base em características técnicas cria silos desnecessários na organização, o que resulta em comunicações precárias, 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 de software. [Por exemplo, equipes multifuncionais são responsáveis por um conjunto específico de contextos limitados, que são identificados usando DDD e event storming.](#) 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 pouco unidas, para que 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, para que recursos complexos possam ser atribuídos a mais engenheiros.

Práticas recomendadas

Modele o domínio comercial

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

Use metodologias de design orientado por domínio (DDD), como o [evento storming](#), para modelar o domínio comercial. O Event Storming tem um formato de workshop flexível. Durante o workshop, especialistas em domínio e software exploram a complexidade do domínio comercial 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 a 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 principal 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 do domínio. Defina os eventos que os objetos do 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 dinâmicos ajudam você a manter a qualidade do código e evitar erros durante a implantação.

- Execute seus testes de unidade dentro do seu processo de CI e teste seu código antes que ele seja mesclado.
- Crie um processo de CD para implantar seu aplicativo em um dev/test ambiente estático 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 comando em síncronos e assíncronos separados. APIs Considere adotar o padrão de segregação de responsabilidade por consulta de comando (CQRS) e a arquitetura orientada a eventos.

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

como contextos limitados, conforme discutido anteriormente na [Escalabilidade organizacional](#) seção. Essas equipes podem então implementar a lógica de negócios usando a arquitetura hexagonal.

Crie uma estrutura de projeto que mapeie conceitos de arquitetura hexagonal

A infraestrutura como código (IaC) é uma prática amplamente adotada no desenvolvimento de 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 da infraestrutura no mesmo repositório ao desenvolver seus aplicativos na nuvem. Essa abordagem facilita a manutenção da infraestrutura do seu aplicativo.

Recomendamos que você divida seu aplicativo em três pastas ou projetos que mapeiam os conceitos da arquitetura hexagonal: `entrypoints` (adaptadores primários), `domain` (domínio e interfaces) e `adapters` (adaptadores secundários).

A estrutura do projeto a seguir fornece um exemplo dessa abordagem ao projetar uma API no AWS. O projeto mantém o código do aplicativo (`app`) e o código da 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 pelo domínio e, em seguida, roteados para outros microsserviços.
- `exceptions` contém os erros conhecidos definidos no domínio.
- `model` contém as 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 ou outros componentes externos. APIs
- `tests` contém os métodos de teste (como testes de lógica de negócios) que são executados no domínio.

Os adaptadores principais são os pontos de entrada do aplicativo, conforme representados pela `entrypoints` pasta. Este exemplo usa a `api` pasta como adaptador primário. Essa pasta contém uma `APIModel`, 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 do 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 de negócios. A `tests` subpasta contém testes de integração externa para cada adaptador.

Exemplos de infraestrutura em AWS

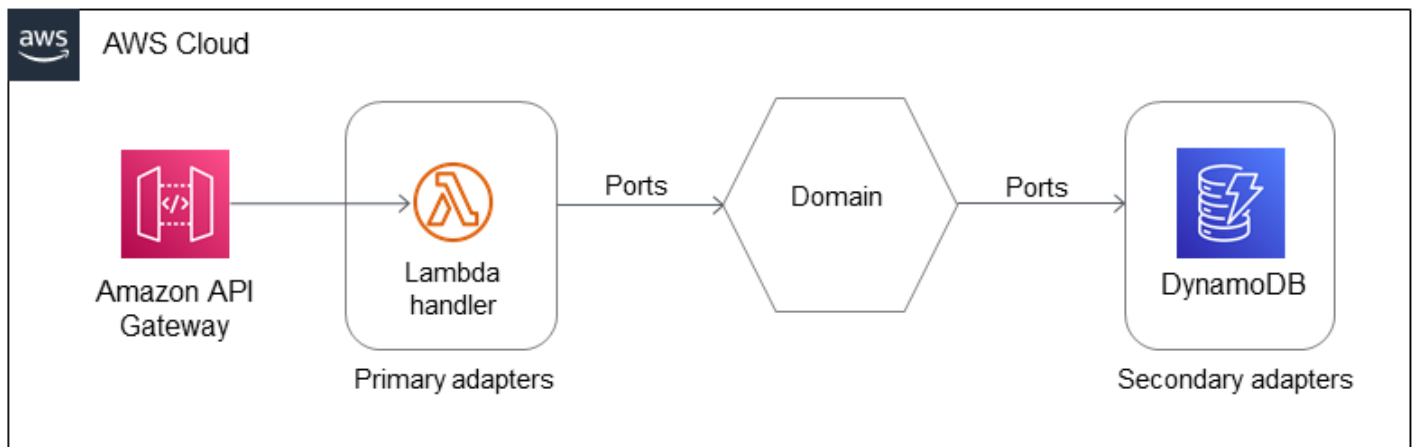
Esta seção fornece exemplos para projetar uma infraestrutura para seu aplicativo AWS que você pode usar para implementar uma arquitetura hexagonal. Recomendamos que você comece com uma arquitetura simples para criar um produto mínimo viável (MVP). A maioria dos microsserviços precisa de um único ponto 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. Os AWS serviços a seguir são ótimos candidatos para uso como clientes, adaptadores primários e adaptadores secundários na arquitetura hexagonal:

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

As seções a seguir discutem esses serviços no contexto da arquitetura hexagonal com mais detalhes.

Comece de forma simples

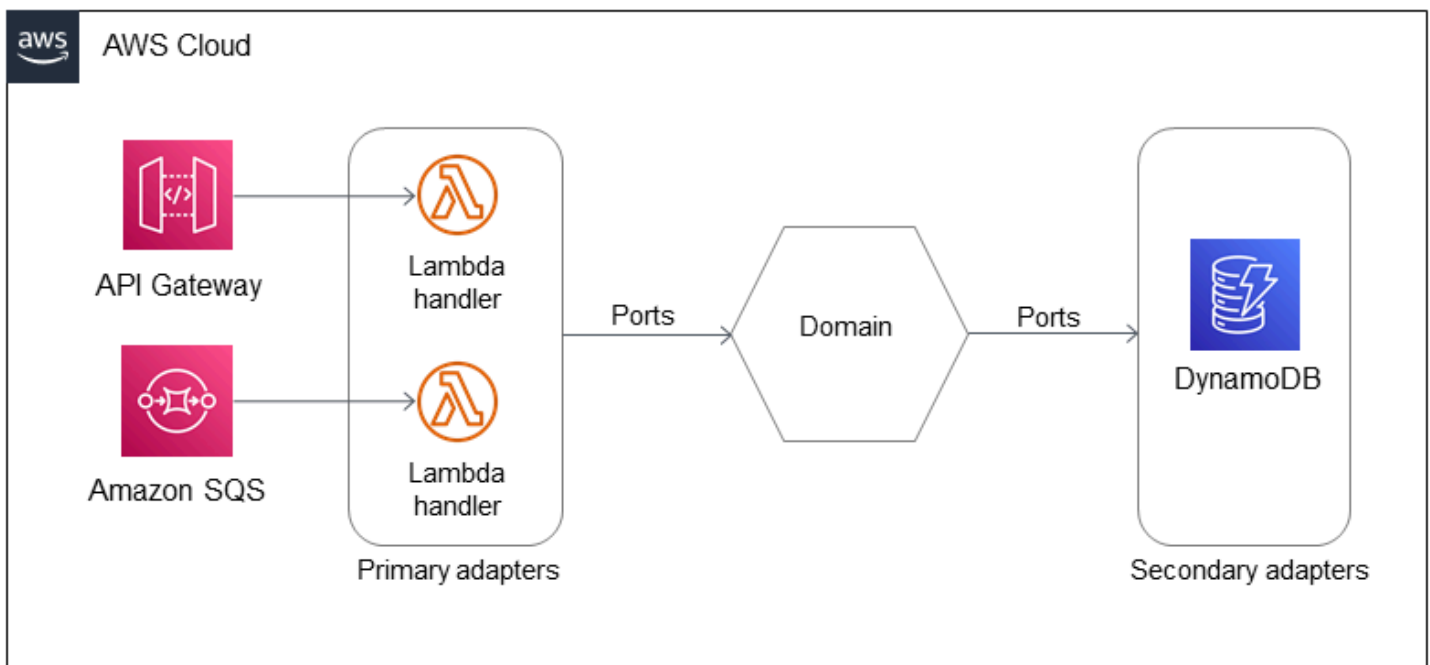
Recomendamos que você comece de forma simples ao arquitetar um aplicativo usando a arquitetura hexagonal. Neste exemplo, o API Gateway é usado como cliente (API REST), o Lambda é usado como adaptador primário (computação) e o DynamoDB é usado como adaptador secundário (persistência). O cliente do gateway 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 quiser aumentar. Você pode aplicar o padrão CQRS como uma arquitetura totalmente sem servidor AWS usando o exemplo a seguir.

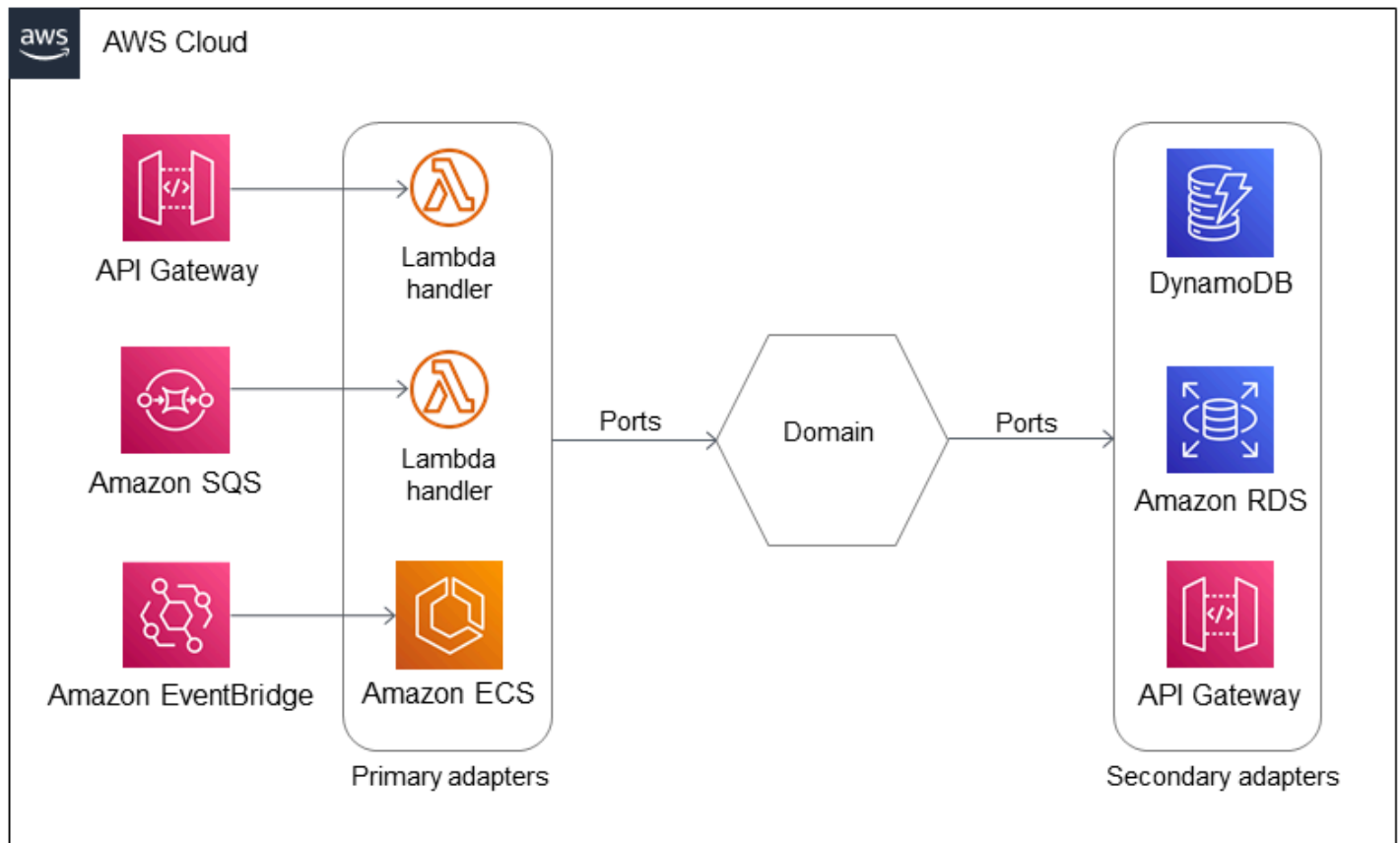


Este exemplo usa dois manipuladores Lambda, um para consultas e outro para comandos. As consultas são executadas de forma síncrona usando um gateway de API como cliente. Os comandos são executados de forma assíncrona usando o Amazon SQS como cliente.

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 Lambda). Todos os componentes pertencem ao mesmo contexto limitado, portanto, estão dentro do mesmo domínio.

Evolua 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 o externo APIs. Você pode desenvolver o exemplo de arquitetura para dar suporte a esses requisitos, conforme mostrado no diagrama a seguir.

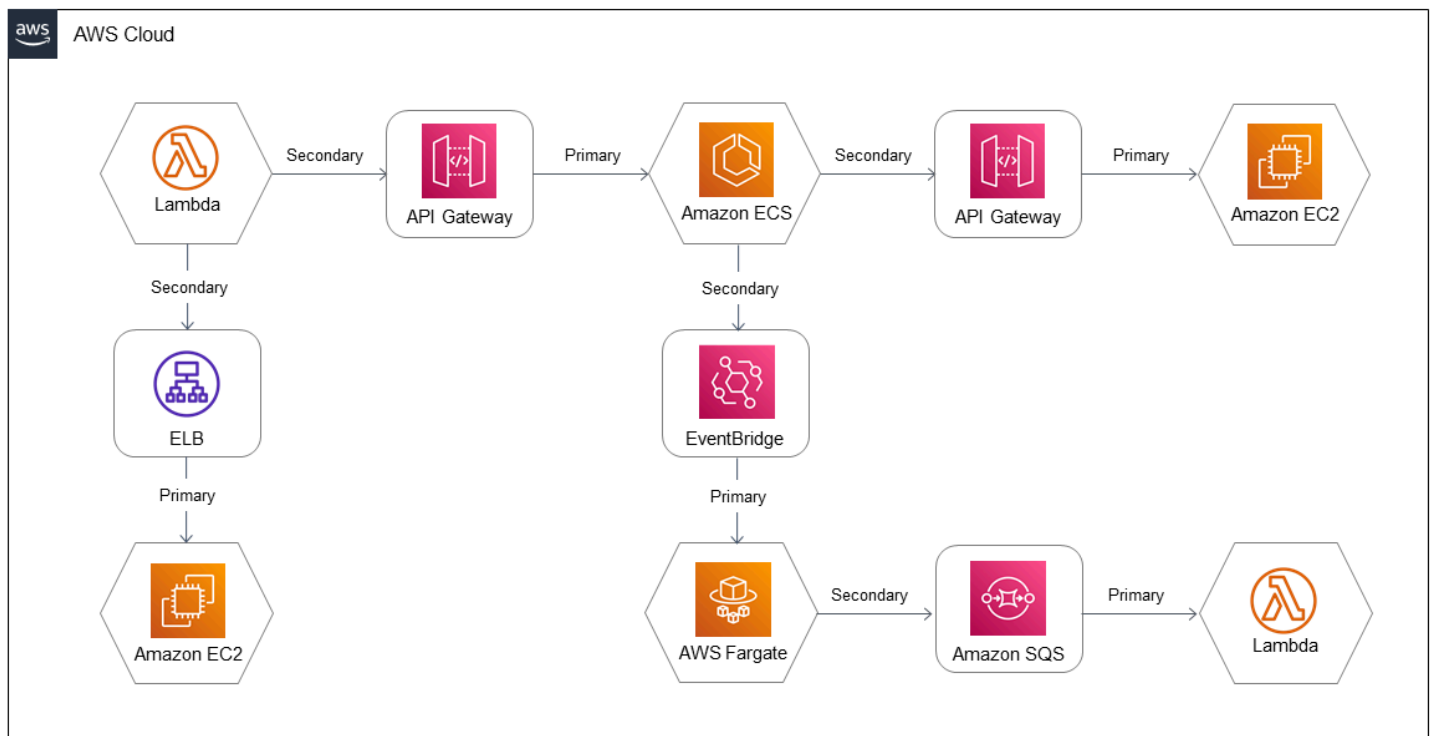


Este exemplo usa o Amazon ECS como o adaptador principal para iniciar tarefas de longa execução no domínio. A Amazon EventBridge (cliente) inicia uma tarefa do Amazon ECS (ponto de entrada) quando um evento específico acontece. A arquitetura inclui o Amazon RDS como outro adaptador secundário para armazenar 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 comercial.

O domínio está sempre fracamente 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 mudança de um adaptador para outro não afeta o domínio. Por exemplo, você pode mudar do Amazon DynamoDB para o Amazon RDS escrevendo um novo adaptador, sem afetar o domínio.

Adicionar mais domínios (diminuir o zoom)

A arquitetura hexagonal se alinha bem aos princípios de uma arquitetura de microsserviços. Os exemplos de arquitetura mostrados até agora continham um único domínio (ou contexto limitado). Os aplicativos geralmente 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 computacionais. (Cada domínio também pode ter vários ambientes computacionais, 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 desacoplados uns dos outros.

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

Perguntas frequentes

Por que eu deveria 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 lançamento mais rápido no mercado e em um escalonamento técnico e organizacional mais fácil.

Por que eu deveria usar o 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 os produtos de software a longo prazo.

Posso praticar o desenvolvimento orientado a testes sem arquitetura hexagonal?

Sim. O desenvolvimento orientado a 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. O dimensionamento técnico e organizacional do produto pode ser alcançado com a maioria dos padrões de design. No entanto, a arquitetura hexagonal e o DDD facilitam a escalabilidade 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 dependências e ao teste de unidade.

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

Sim. Recomendamos que você use padrões de design que lhe sejam familiares 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 oferecer suporte ao dimensionamento do sistema geral. Use o [padrão de repositório](#) para armazenar e restaurar seu modelo de domínio. Use o padrão de unidade 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 complexas de objetos.

Próximas etapas

- Familiarize-se ainda mais com os conceitos de design orientado por domínio lendo os links coletados na seção. [Recursos](#)
- 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 para leitura e somente gravação. Abstraia 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 pronta, 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 usando AWS Lambda \(padrão de orientação prescritiva\)](#) AWS
- [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 para 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 com AWS Lambda](#), de James Beswick (AWS Compute Blog, 8 de julho de 2021)
- [Linguagem de domínio: enfrentando a complexidade no coração do software](#) (site da linguagem de domínio)
- [Facade](#), de Dive Into Design Patterns de Alexander Shvets (e-book, 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](#) (DZone site, 2 de junho de 2017)
- [Princípios de design do SOLID explicados: Princípio de inversão de dependência com exemplos de código](#), de Thorben Janssen (site da Stackify, 7 de maio de 2018)
- [Princípios SOLID: explicação e exemplos](#), de Simon Hoiberg (site 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 de programação orientada a freeCodeCamp objetos, 20 de agosto de 2020)

- [O que é uma arquitetura orientada a eventos? \(AWS site\)](#)

AWS serviços

- [Amazon API Gateway](#)
- [Amazon Aurora](#)
- [Amazon DynamoDB](#)
- [Nuvem de computação elástica da Amazon \(Amazon EC2\)](#)
- [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 feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em 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:

- **Refactor/re-architect** — mova um aplicativo e modifique sua arquitetura aproveitando ao máximo os recursos nativos da nuvem para melhorar a agilidade, o desempenho 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 Amazon PostgreSQL-Compatible Aurora Edition.
- **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: migrar seu banco de dados Oracle on-premises para o Amazon Relational Database Service (Amazon RDS) para Oracle na 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 o 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: migrar seu banco de dados Oracle on-premises para o Oracle em uma instância do EC2 na 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 on-premises para um serviço de 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

A2A () Agent-to-Agent

Um protocolo com estado para colaboração entre agentes, apoiando a delegação de tarefas e a transferência de estados.

ABAC

Consulte [controle de acesso baseado em atributo](#).

serviços abstraídos

Veja [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 em que 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.

Agente

Um sistema de IA que pode raciocinar, planejar e realizar ações de forma autônoma usando ferramentas para atingir metas.

Agente Ops

Práticas operacionais para criar, testar, implantar e executar agentes de IA na produção em grande escala.

AGGREGATE FUNCTION

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 [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 aplicações

Uma abordagem de segurança que permite o uso somente de aplicações aprovadas 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 descryptografia. É possível compartilhar a chave pública porque ela não é usada na descryptografia, 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.

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 malicioso

Um [bot](#) destinado a causar interrupção ou danos a indivíduos ou organizações.

BCP

Veja [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 implantação

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

bot

Uma aplicação de software que executa tarefas automatizadas na internet e simula a atividade ou interação humana. Alguns bots são úteis ou benéficos, como crawlers da web que indexam informações na internet. Outros bots, conhecidos como bots maliciosos, têm como objetivo causar interrupção ou danos a indivíduos ou organizações.

botnet

Redes de [bots](#) infectadas por [malware](#) e sob o controle de uma única parte, conhecidas como bot herder ou operador de bots. Os botnets 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 de emergência

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 vidros](#) na AWS Well-Architected orientação.

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 containerizados 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

Veja [AWS Cloud Adoption Framework](#).

implantação canário

O lançamento lento e incremental de uma versão para usuários finais. Quando estiver confiante, você implanta a nova versão e substitui a versão atual por completo.

CCoE

Veja [Centro de Excelência da Nuvem](#).

CDC

Veja [captura de dados de alteração](#).

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 [integração e 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.

Desenvolvedor cidadão

Um usuário corporativo que cria aplicativos de IA usando plataformas sem code/low código sem habilidades técnicas especializadas.

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 recomendadas 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 é normalmente conectada à tecnologia de [computação de borda](#).

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 ao migrar para a 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
- Re-invention — Otimizando produtos e serviços e inovando na nuvem

Esses estágios foram definidos por Stephen Orban na postagem do blog [The Journey Toward Cloud-First & the Stages of Adoption](#) no blog Nuvem AWS Enterprise Strategy. Para obter informações sobre como eles se relacionam com a estratégia de AWS migração, consulte o [guia de preparação para migração](#).

CMDB

Veja [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 o GitHub ou o Bitbucket Cloud. 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 CI/CD pipeline 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 de [IA](#) que usa machine learning para analisar e extrair informações de formatos visuais, como vídeos e imagens digitais. Por exemplo, a Amazon SageMaker AI fornece algoritmos de processamento de imagem para CV.

desvio de configuração

Em uma workload, uma alteração de configuração em relação ao estado esperado. Isso pode fazer com que a workload 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. CI/CD é comumente descrito como um pipeline. CI/CD pode ajudá-lo 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 na AWS Well-Architected Estrutura. 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.

data mesh

Um framework de arquitetura 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 compatível com business intelligence, como analytics. 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

Veja [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.

defesa completa

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 abordagem de defesa aprofundada 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 [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 normalmente são 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](#) na AWS Well-Architected estrutura.

DML

Veja [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 Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Para obter informações sobre como você pode usar o design orientado por domínio com o padrão strangler fig, consulte Modernizando os [serviços web legados da Microsoft ASP.NET \(ASMX\) de forma incremental usando](#) contêineres e o Amazon API Gateway.

DR

Veja [recuperação de desastres](#).

Detecção da oscilação

Rastreamento de desvios de uma configuração de linha de base. 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 [mapeamento do fluxo de valor de desenvolvimento](#).

E

EDA

Veja [análise exploratória de dados](#).

EDI

Veja [intercâmbio eletrônico de dados](#).

computação de borda

A tecnologia que aumenta o poder computacional de dispositivos inteligentes nas bordas de uma rede de IoT. Quando comparada com a [computação em nuvem](#), a computação de borda pode reduzir a latência da comunicação e melhorar o tempo de resposta.

intercâmbio eletrônico de dados (EDI)

A troca automatizada de documentos comerciais entre organizações. Para obter mais informações, consulte [O que é EDI \(Intercâmbio eletrônico de dados\)?](#).

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. Big-endian os sistemas armazenam primeiro o byte mais significativo. Little-endian os sistemas armazenam primeiro o byte menos significativo.

endpoint

Veja [endpoint de 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 empresariais (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).

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 CI/CD pipeline, 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

Veja [planejamento de recursos empresariais](#).

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](#). Ela armazena dados quantitativos sobre as operações comerciais. Normalmente, uma tabela de fatos contém dois tipos de colunas: as que contêm medidas e as que contêm uma chave externa para uma tabela de dimensões.

Antecipar-se à falha

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

delimitação de isolamento contra 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 [AWS Fault Isolation Boundaries](#).

ramificação de recursos

Veja [ramificação](#).

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.

prompt few shot

Fornecer a um [LLM](#) um pequeno número de exemplos que demonstram a tarefa e o resultado desejado antes de solicitar que ele execute uma tarefa semelhante. Essa técnica é uma aplicação do aprendizado contextual, em que os modelos aprendem com exemplos (fotos) incorporados aos prompts. Few-shot a solicitação pode ser eficaz para tarefas que exigem formatação, raciocínio ou conhecimento de domínio específicos. Veja também [prompts zero-shot](#).

FGAC

Veja [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 via [captura de dados de alteração](#) para migrar os dados no menor tempo possível, em vez de usar uma abordagem em fases. O objetivo é reduzir ao mínimo o tempo de inatividade.

FM

Veja [modelo de base](#).

modelo de base (FM)

Uma grande rede neural de aprendizado profundo que treina em grandes conjuntos de dados generalizados e não rotulados. Os FMs são capazes de realizar uma ampla variedade de tarefas gerais, como entender a linguagem, gerar texto e imagens e conversar em linguagem natural. Para obter mais informações, consulte [O que são modelos de base?](#).

Gateway FM

[Um intermediário centralizado que controla e normaliza o acesso aos modelos de fundação.](#)

Também conhecido como gateway LLM.

G

IA generativa

Um subconjunto de modelos de [IA](#) que foram treinados em grandes quantidades de dados e que podem usar um simples prompt de texto para criar novos artefatos e conteúdo, como imagens, vídeos, texto e áudio. Para obter mais informações, consulte [O que é IA generativa?](#).

bloqueio geográfico

Veja [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 trunk-based](#) é a abordagem moderna e preferencial.

golden image

Um snapshot de um sistema ou software usado como modelo para implantar novas instâncias desse sistema ou software. Por exemplo, na manufatura, uma golden image pode ser usada para

provisionar software em vários dispositivos e ajudar a melhorar a velocidade, a escalabilidade e a produtividade nas operações de fabricação de dispositivos.

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 CSPM, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector e verificações personalizadas AWS Lambda .

grades de proteção (IA)

Mecanismos de segurança que filtram, validam e restringem as entradas e saídas dos [agentes](#) para ajudar a garantir um comportamento de IA responsável e seguro.

H

HA

Veja [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.

dados de hold-out

Uma parte dos dados históricos rotulados que são retidos de um conjunto de dados usado para treinar um modelo de [machine learning](#). Você pode usar dados de hold-out para avaliar a performance do modelo comparando as previsões do modelo com os dados de retenção.

humano no circuito (HiTL)

Um padrão de fluxo de trabalho em que a execução do [agente](#) é pausada para análise e aprovação humana em pontos críticos de decisão.

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.

eu

laC

Veja [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.

IloT

Veja [Internet das Coisas Industrial](#).

infraestrutura imutável

Um modelo que implanta uma nova infraestrutura para workloads de produção em vez de atualizar, aplicar patches ou modificar a infraestrutura existente. Infraestruturas imutáveis são inerentemente mais consistentes, confiáveis e preditivas do que [infraestruturas mutáveis](#). Para obter mais informações, consulte as melhores práticas de [implantação usando infraestrutura imutável](#) na AWS Well-Architected Estrutura.

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 na conectividade, dados em tempo real, automação, análise e. AI/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 aprendizado de máquina com AWS](#).

IoT

Veja [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

Veja [biblioteca de informações de TI](#).

ITSM

Veja [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](#).

grande modelo de linguagem (LLM)

Um modelo de [IA](#) de aprendizado profundo pré-treinado em uma grande quantidade de dados. Um LLM pode realizar várias tarefas, como responder a perguntas, resumir documentos, traduzir texto para outros idiomas e completar frases. Para obter mais informações, consulte [O que é grande modelo de linguagem \(LLM\)?](#).

migração de grande porte

Uma migração de 300 servidores ou mais.

LBAC

Veja [controle de acesso baseado em rótulo](#).

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](#).

LLM

Veja [grande modelo de linguagem](#).

ambientes inferiores

Veja [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 [ramificação](#).

Malware

Software projetado para comprometer a segurança ou a privacidade do computador. O malware pode interromper os sistemas do computador, vaziar informações sensíveis ou obter acesso não autorizado. Exemplos de malware incluem vírus, worms, ransomware, cavalos de Troia, 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 abstraídos.

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

Veja [Programa de Aceleração da Migração](#).

MCP

Consulte [Protocolo de contexto do modelo](#).

Protocolo de contexto para modelos (MCP)

Um protocolo sem estado para comunicação entre [agentes](#) e [ferramentas](#).

Servidor MCP

Um serviço que expõe uma ou mais [ferramentas](#) por meio do [Model Context Protocol](#).

mecanismo

Um processo completo em que 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 [Criação de mecanismos](#) na AWS Well-Architected estrutura.

conta de 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 [sistema de execução de manufatura](#).

Transporte de Telemetria de Enfileiramento de Mensagens (MQTT)

[Um protocolo de comunicação leve, máquina a máquina \(M2M\), baseado no padrão, para dispositivos de IoT com recursos publish/subscribelimitados.](#)

microserviço

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 microserviços que mapeiam as capacidades comerciais, como vendas ou marketing, ou subdomínios, como compras, reclamações ou análises. Os benefícios dos microserviços incluem agilidade, escalabilidade flexível, fácil implantação, código reutilizável e resiliência. Para obter mais informações, consulte [Integração de microserviços usando serviços sem AWS servidor](#).

arquitetura de microserviços

Uma abordagem à criação de aplicações com componentes independentes que executam cada processo de aplicação como um microserviço. Esses microserviços se comunicam por meio de uma interface bem definida usando APIs leves. Cada microserviço 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 microserviços 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

Cross-functional equipes que simplificam a migração de cargas de trabalho 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: rehoste 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 a 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 workload para a Nuvem AWS. Para obter mais informações, veja a entrada [7 Rs](#) neste glossário e consulte [Mobilize sua organização para acelerar migrações em grande escala](#).

ML

Veja [machine learning](#).

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 [Strategy for modernizing applications in the 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 [Evaluating modernization readiness for applications in the 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](#).

MPA

Veja [Avaliação do Portfólio para Migração](#).

MQTT

Veja [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 workloads de produção. Para melhorar a consistência, confiabilidade e previsibilidade, a AWS Well-Architected Estrutura recomenda o uso de [infraestrutura imutável](#) como uma prática recomendada.

O

OAC

Veja [controle de acesso de origem](#).

OAI

Veja [identidade de acesso de origem](#).

OCM

Veja [gerenciamento de alterações 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 [integração de operações](#).

Ola

Veja [acordo de 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

Veja [Open Process Communications - Unified Architecture](#).

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

Um protocolo de comunicação máquina a máquina (M2M) para automação industrial. 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 práticas recomendadas 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 \(ORR\)](#) na AWS Well-Architected Estrutura.

tecnologia operacional (TO)

Sistemas de hardware e software que trabalham com o ambiente físico para controlar operações, equipamentos e infraestrutura industriais. Na manufatura, a integração dos sistemas de tecnologia da informação (TI) e tecnologia operacional (TO) 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 necessária 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 do S3 Regiões da AWS, à criptografia do lado do servidor com AWS KMS (SSE-KMS) e à dinâmica PUT e DELETE às solicitações ao bucket do S3.

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.

ORR

Veja [análise de prontidão operacional](#).

OT

Veja [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

Veja [controlador lógico programável](#).

PLM

Veja [gerenciamento do ciclo de vida do produto](#).

política

Um objeto que pode definir permissões (veja [política baseada em identidade](#)), especificar condições de acesso (veja [política baseada em recurso](#)) ou definir as permissões máximas para todas as contas em uma organização no AWS Organizations (veja [política de controle de serviços](#)).

persistência poliglota

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

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 cláusula `WHERE`.

pushdown de predicados

Uma técnica de otimização de consultas 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 a performance 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 desenvolvimento.

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](#) desenvolvido para evitar a implantação de recursos não conformes. Esses controles verificam 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 a concepção, o desenvolvimento e o lançamento, passando pelo crescimento e maturidade, até o declínio e a remoção.

ambiente de produção

Veja [ambiente](#).

controlador lógico programável (PLC)

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

encadeamento de prompts

Uso da saída de um prompt do [LLM](#) como entrada para o próximo prompt para gerar respostas melhores. Essa técnica é usada para dividir uma tarefa complexa em subtarefas, ou para refinar ou expandir iterativamente uma resposta preliminar. Isso ajuda a melhorar a precisão e a relevância das respostas de um modelo e permite resultados mais granulares e personalizados.

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.

publish/subscribe (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 em que outros microsserviços possam assinar. 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, aprovador, consultado, informado \(RACI\)](#).

RAG

Veja [geração aumentada via recuperação](#).

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, aprovador, consultado, informado \(RACI\)](#).

RCAC

Veja [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.

Redefinir arquitetura

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 informações, consulte [Specify which Regiões da AWS your account can use](#).

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 uma aplicação de resistir ou se recuperar de interrupções. [Alta disponibilidade](#) e [recuperação de desastres](#) são considerações comuns ao planejar a resiliência na Nuvem AWS. Para obter mais informações, consulte [Nuvem AWS Resilience](#).

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](#).

Retirada

Veja [7 Rs](#).

Geração Aumentada de Recuperação (RAG)

Uma tecnologia de [IA generativa](#) em que um [LLM](#) faz referência a uma fonte de dados autorizada que está fora de suas fontes de dados de treinamento antes de gerar uma resposta. Por exemplo, um modelo RAG pode realizar uma pesquisa semântica na base de conhecimento ou nos dados personalizados de uma organização. Para obter mais informações, consulte [O que é RAG \(geração aumentada via recuperação\)?](#).

alternância

O processo de atualizar periodicamente um [segredo](#) para dificultar o acesso de um invasor às credenciais.

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 [objetivo de ponto de recuperação](#).

RTO

Veja [objetivo de 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 no Console de gerenciamento da AWS 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 [política de controle de serviço](#).

secret

Em AWS Secrets Manager informações confidenciais ou restritas, como uma senha ou credenciais de usuário, que você armazena de forma criptografada. 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 [What's in a Secrets Manager secret?](#) na documentação do Secrets Manager.

segurança desde a concepção

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

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 primários de controles de segurança: [preventivos](#), [detectivos](#), [responsivos](#) e [proativos](#).

hardening 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 aplicação de patches em uma instância do Amazon EC2 ou a alternância 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 avaliação de um aspecto de performance de um serviço, como taxa de erro, disponibilidade ou throughput.

objetivo de nível de serviço (SLO)

Uma métrica alvo que representa a integridade de um serviço, conforme avaliado 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](#).

Inteligência artificial sombria

Aplicativos de [IA](#) não autorizados criados ou usados fora dos canais controlados dentro de uma organização.

SIEM

Veja [sistema de gerenciamento de eventos e informações de segurança](#).

ponto único de falha (SPOF)

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

SLA

Veja [acordo de serviço](#).

SLI

Veja [indicador de nível de serviço](#).

SLO

Veja [objetivo de nível de serviço](#).

modelo dividir e semear

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 [Phased approach to modernizing applications in the Nuvem AWS](#).

SPOF

Veja [ponto único de falha](#).

esquema em estrela

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 ser usada 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 [Modernizando os serviços web legados da Microsoft ASP.NET \(ASMX\) de forma incremental usando contêineres e o Amazon API Gateway](#).

sub-rede

Um intervalo de endereços IP na VPC. Cada sub-rede fica alocada em uma única zona de disponibilidade.

controle supervisorio 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 a performance. Você pode usar o [Amazon CloudWatch Synthetics](#) para criar esses testes.

prompt do sistema

Uma técnica para fornecer contexto, instruções ou orientações a um [LLM](#) a fim de direcionar seu comportamento. Os prompts do sistema ajudam a definir o contexto e a estabelecer regras para interações com os usuários.

T

tags

Key-value pares que atuam como metadados para organizar seus AWS recursos. As tags podem ajudar você a gerenciar, identificar, organizar, pesquisar e filtrar recursos da . 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 [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.

ferramenta

Uma função ou API que um [agente](#) pode invocar para realizar operações em sistemas externos.

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.

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 [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 backend.

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.

WORM

Veja [gravação única e várias leituras](#).

WQF

Veja [AWS Workload Qualification Framework](#).

gravação única e várias leituras (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, normalmente malware, que tira proveito de uma [vulnerabilidade zero-day](#).

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.

prompt zero shot

Fornecer a um [LLM](#) instruções para realizar uma tarefa, mas sem exemplos (shots) que possam ajudar a orientá-lo. O LLM deve usar seu conhecimento pré-treinado para lidar com a tarefa. A eficácia dos prompts zero-shot depende da complexidade da tarefa e da qualidade do prompt. Veja também [prompts few-shot](#).

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.