Escolhendo uma infraestrutura como ferramenta de código para sua organização

AWS Recomendações

AWS Recomendações: Escolhendo uma infraestrutura como ferramenta de código para sua organização

Table of Contents

Escolhendo uma infraestrutura como ferramenta de código para sua organização

Amazon Web Services (colaboradores)

Fevereiro de 2026 (histórico do documento)

A infraestrutura como código (IaC) é o processo de provisionamento e gerenciamento da infraestrutura de um aplicativo 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. É um componente essencial do Agile e de DevOps práticas, como controle de versão, integração contínua e implantação contínua.

A escolha de uma ferramenta de infraestrutura como código (IaC) é considerada uma decisão estratégica para uma organização. Essa decisão afeta todas as equipes que criam infraestrutura, aplicativos e serviços para a empresa. Cada ferramenta tem prós e contras e, portanto, não existe um one-size-fits-all modelo.

No passado, gerenciar e provisionar a infraestrutura era um processo manual repleto de erros. O IaC simplifica essas tarefas por meio de código e se tornou uma solução confiável para a implantação de infraestrutura. As ferramentas de IaC capacitam os desenvolvedores a definir e implantar a infraestrutura usando linguagens de programação. Isso não apenas aumenta a agilidade dos negócios, mas também acelera o crescimento e a velocidade da inovação. Além disso, o IaC melhora significativamente a segurança porque permite que sua organização escaneie o código antes da implantação, verificando se a infraestrutura é confiável e segura. Em última análise, a ferramenta certa de IaC não é apenas uma decisão técnica, mas estratégica que afeta diretamente o sucesso geral do negócio.

Este guia explora cinco ferramentas de IaC diferentes que podem ser usadas para provisionar AWS recursos: AWS CloudFormation, AWS Serverless Application Model (AWS SAM), AWS Cloud Development Kit (AWS CDK), HashiCorp Terraform e Pulumi. Ele compara essas ferramentas e orienta você no processo de escolha de uma que atenda às necessidades de sua equipe, organização e talento na nuvem. A chave é alinhar a ferramenta de IaC escolhida às metas de sua organização e às habilidades de seus desenvolvedores. Por exemplo, se sua equipe é proficiente em JavaScript, você pode escolher AWS CDK com a ferramenta principal de IaC, TypeScript pois ela otimiza seu fluxo de trabalho de desenvolvimento.

Benefícios da implementação do IaC

Ao escolher a ferramenta de IaC certa para sua organização, você pode obter os seguintes benefícios:

- Velocidade — Um objetivo do IaC é tornar as coisas mais rápidas, eliminando processos manuais. Uma abordagem baseada em código torna mais fácil fazer mais em menos tempo. Configurar manualmente cada ambiente de TI é muito caro e requer engenheiros e arquitetos dedicados para configurar o hardware e o software. Ao aumentar a velocidade, as organizações são mais capazes de se adaptar rapidamente às mudanças nas necessidades dos clientes e nas condições do mercado. Ele também permite que você escreva seu código IaC uma vez e implante esse mesmo código em centenas de ambientes em uma fração do tempo necessário para criá-los manualmente.
- Escalabilidade — o IaC facilita a adição de recursos à infraestrutura existente. Os upgrades são provisionados rapidamente para que você possa se expandir rapidamente durante os períodos de pico de uso. Por exemplo, organizações que executam serviços on-line podem se expandir para acompanhar as demandas dos usuários durante períodos de alta demanda, como a Black Friday.
- Segurança aprimorada — o IaC é excelente em permitir segurança e conformidade para organizações. As organizações podem definir políticas e configurações básicas de segurança e aplicá-las por meio de pipelines, executando testes automatizados em seu IaC. Se o IaC violar as regras de conformidade, o pipeline falhará e fornecerá feedback automaticamente ao desenvolvedor. Isso pode impedir a criação de infraestrutura insegura.
- Consistência — A consistência é outro benefício vital do IaC. Quando vários engenheiros estão implantando configurações manualmente, as inconsistências são inevitáveis. Com o tempo, fica difícil rastrear e reproduzir os mesmos ambientes. Essas inconsistências geralmente levam a diferenças críticas entre ambientes de desenvolvimento, controle de qualidade e produção.
- Eficiência — O IaC melhora a eficiência e a produtividade em todo o ciclo de vida do desenvolvimento. Os programadores criam ambientes de sandbox para desenvolver isoladamente. As operações podem provisionar rapidamente a infraestrutura para testes de segurança. Os engenheiros de controle de qualidade têm cópias exatas dos ambientes de produção durante os testes. Quando estão prontos para a implantação, os desenvolvedores colocam a infraestrutura e o código em produção em uma única etapa. Também pode permitir um maior nível de colaboração e trabalho em equipe. As equipes podem criar padrões seguros para aplicativos e depois compartilhar esses modelos ou construções com outras equipes, o que reduz o trabalho duplicado.

- Custos reduzidos — o IaC reduz os custos de desenvolvimento de software. Não há necessidade de gastar recursos na configuração manual dos ambientes. Você paga apenas pelos recursos que está usando ativamente, portanto, não há sobrecarga desnecessária.
- Maior confiabilidade — Se sua infraestrutura for grande, fica fácil configurar incorretamente um recurso ou provisionar serviços na ordem errada. Com o IaC, os recursos são sempre provisionados e configurados exatamente conforme declarado. Como a criação manual de recursos está sujeita a erros, você pode ter um alto grau de confiança de que sua infraestrutura será criada conforme o esperado ao usar a IaC.
- Detecção de desvio de configuração — O desvio de configuração ocorre quando a configuração que provisionou seu ambiente não corresponde mais ao ambiente real. Muitas ferramentas de IaC ajudam você a detectar desvios e corrigi-los. Por exemplo, se alguém modificar incorretamente seus recursos manualmente, você poderá usar a ferramenta IaC para detectar o que foi alterado e restaurá-lo ao estado pretendido.
- Experimentação — Como a IaC torna o provisionamento de uma nova infraestrutura muito mais rápido e fácil, você pode fazer e testar mudanças experimentais sem investir muito tempo e recursos. Se você gostar dos resultados, poderá ampliar rapidamente a nova infraestrutura para produção.

Analisando as ferramentas de IaC para o Nuvem AWS

Este guia explora cinco ferramentas diferentes de IaC que podem ser usadas para AWS provisionar recursos. Ele compara essas ferramentas e orienta você no processo de escolha de uma que atenda às necessidades de sua equipe, organização e talento na nuvem. A chave é alinhar a ferramenta de IaC escolhida às metas de sua organização e às habilidades de seus desenvolvedores.

Nesta seção, você aprenderá mais sobre como cada ferramenta funciona e sobre suas vantagens e desvantagens. Por exemplo, algumas ferramentas só podem ser usadas no Nuvem AWS e não são adequadas para implantações em várias nuvens ou em nuvem híbrida. Outros exigem conhecimento de linguagens de programação especializadas e outros oferecem suporte a linguagens de programação comuns. Avalie as vantagens e desvantagens de cada ferramenta e considere qual seria a mais adequada para sua organização.

Esta seção aborda as seguintes ferramentas de IaC:

- AWS CloudFormation
- AWS Serverless Application Model (AWS SAM)
- AWS Cloud Development Kit (AWS CDK)
- HashiCorp Terraforma
- Pulumi

Usando AWS CloudFormation como uma ferramenta IaC

AWS CloudFormationé um AWS service (Serviço da AWS) que usa arquivos de modelo para automatizar o provisionamento de recursos. AWS Você cria um modelo que descreve todos os AWS recursos que você deseja implantar e CloudFormation provisiona e configura esses recursos para você.

CloudFormation os modelos são escritos usando JSON ou YAML. Uma CloudFormation pilha é a implementação dos recursos definidos em seu modelo. Você pode gerenciar suas CloudFormation pilhas por meio do Console de gerenciamento da AWS, programaticamente por meio do CloudFormation SDK ou por meio do (). AWS Command Line Interface AWS CLI Para obter mais informações sobre como CloudFormation funciona, consulte AWS CloudFormation conceitos e Como AWS CloudFormation funciona na CloudFormation documentação.

Vantagens de usar CloudFormation:

- CloudFormation os conjuntos de alterações permitem que você visualize as alterações em uma pilha em execução antes de implantá-las. Os conjuntos de alterações resumem as alterações propostas nos recursos em execução em uma pilha existente. Isso pode ajudá-lo a identificar conflitos ou consequências não intencionais antes da implantação. Por exemplo, se você alterar o nome de uma instância de banco de dados do Amazon Relational Database Service (Amazon RDS) CloudFormation , criará um novo banco de dados e excluirá o antigo. Você perderia os dados no banco de dados antigo, a menos que já tenha feito backup deles. Se você gerar um conjunto de alterações, verá que sua alteração fará com que seu banco de dados seja substituído e poderá planejar adequadamente antes de atualizar sua pilha.
- Se ocorrer um erro durante a implantação de um conjunto de alterações, CloudFormation reverte automaticamente para o último estado de funcionamento conhecido.
- Você pode usar conjuntos de CloudFormation pilhas para implantar recursos em vários Contas da AWS e. Regiões da AWS
- Não há cobrança adicional pelo uso CloudFormation com provedores de recursos nos seguintes namespaces: AWS: :*, Alexa: :* e Custom: :*. Nesses casos, você paga somente pelos AWS recursos provisionados, como se os tivesse provisionado manualmente.
- CloudFormation gerencia o estado para você. Isso significa que CloudFormation faz chamadas de serviço subjacentes AWS para provisionar e configurar seus recursos conforme definido em seus CloudFormation modelos.
- CloudFormation fornece ferramentas para detectar e corrigir desvios de configuração. Para obter mais informações, consulte Detecção de alterações de configuração não gerenciadas em pilhas e recursos na documentação. CloudFormation
- Você pode usar CloudFormation para criar recursos personalizados. Você pode escrever uma lógica de provisionamento personalizada em modelos que são CloudFormation executados sempre que você cria, atualiza ou exclui pilhas.
- CloudFormation oferece suporte à modelagem, provisionamento e gerenciamento de recursos de aplicativos de terceiros com CloudFormation registro.
- CloudFormation suporta a importação de recursos existentes para o CloudFormation gerenciamento.

Desvantagens de usar CloudFormation:

- Se você não estiver familiarizado com a sintaxe JSON ou YAML, pode levar algum tempo para se acostumar. O JSON não foi projetado para ser legível por humanos e não permite que você faça comentários embutidos. O YAML permite que você faça comentários e é mais fácil de ler. No entanto, sua sintaxe é baseada em tabulações e espaços, portanto, pode ser fácil cometer erros de indentação.
- CloudFormation não oferece suporte a implantações em várias nuvens.
- Você deve usar uma implementação de nível superior, como a AWS Cloud Development Kit (AWS CDK), para criar construções reutilizáveis e outros códigos modularizados.

Usando o AWS Serverless Application Model (AWS SAM) como uma ferramenta de IaC

O AWS Serverless Application Model (AWS SAM) é um kit de ferramentas que se estende AWS CloudFormation. Ele inclui recursos adicionais projetados para ajudar você a criar aplicativos sem servidor com mais rapidez. Quando você implanta um AWS SAM modelo, ele é CloudFormation convertido para criar os recursos definidos. AWS SAM consiste em duas partes, a especificação do AWS SAM modelo e a Interface de Linha de AWS SAM Comando (AWS SAM CLI). Embora você possa usar a CloudFormation sintaxe diretamente no AWS SAM modelo, AWS SAM oferece sua própria sintaxe exclusiva que se concentra especificamente em acelerar o desenvolvimento sem servidor. Essa sintaxe abreviada permite definições otimizadas de IaC para recursos sem servidor, como Amazon API Gateway, e recursos. AWS Lambda AWS Step Functions A AWS SAM CLI é uma ferramenta de desenvolvedor que inclui recursos para ajudá-lo a testar AWS Lambda funções localmente, criar pipelines de integração contínua e entrega contínua (CI/CD) e executar comandos para implantar aplicativos sem servidor.

Vantagens de usar AWS SAM:

- AWS SAM tem as mesmas vantagens que CloudFormation.
- Em comparação com CloudFormation, você pode usar com mais facilidade AWS SAM para criar aplicativos e recursos sem servidor, como um Amazon API Gateway que é apoiado por. AWS Lambda
- Usando a AWS SAM CLI, você pode testar AWS Lambda funções localmente. Ao invocar localmente uma função Lambda no modo de depuração, você pode então anexar um depurador a ela. Com o depurador, você pode percorrer seu código linha por linha, ver os valores de várias variáveis e corrigir problemas da mesma forma que faria com qualquer outro aplicativo.

Desvantagens de usar AWS SAM:

- AWS SAM tem as mesmas desvantagens que CloudFormation.
- AWS SAM não pode ser usado fora do AWS.

Usando o AWS CDK como uma ferramenta de IaC

AWS Cloud Development Kit (AWS CDK)É uma estrutura de desenvolvimento de software de código aberto que permite definir seus recursos de aplicativos em nuvem usando linguagens de programação conhecidas. Os AWS CDK suportes JavaScript são Python, Java, C# e Go. TypeScript O AWS CDK provisiona seus recursos de forma segura e repetível. AWS CloudFormation Quando você sintetiza seu AWS CDK código, o resultado é um CloudFormation modelo. Isso AWS CDK fornece abstrações de alto nível que simplificam o processo de definição AWS de recursos.

AWS CDK Ele usa o conceito de construções. Uma construção é um componente dentro do seu aplicativo que representa um ou mais CloudFormation recursos e suas configurações, como um bucket do Amazon Simple Storage Service (Amazon S3). As construções podem ser compostas e personalizadas para criar uma infraestrutura mais complexa. Para obter mais informações, consulte Níveis de construção na AWS CDK documentação. O AWS CDK gera CloudFormation modelos com base no código escrito pelos desenvolvedores. Isso elimina a necessidade de criação manual CloudFormation de modelos. Muitas organizações personalizam, compartilham e reutilizam construções dentro de uma comunidade, assim como qualquer outra biblioteca de software. O compartilhamento de construções ajuda os desenvolvedores a programar com mais rapidez e a incorporar as melhores práticas por padrão.

AWS CDK aspectos podem ajudar as organizações a aplicar padrões a todas as construções dentro de um determinado escopo. O aspecto pode modificar as construções, por exemplo, adicionando tags. Ou poderia verificar algo sobre o estado das construções.

AWS CDK Isso permite que os desenvolvedores usem suas habilidades e conhecimentos de programação existentes para definir a infraestrutura de nuvem. Ao usar linguagens de programação conhecidas, os desenvolvedores podem aplicar seus conhecimentos para descrever AWS recursos, facilitando a transição do desenvolvimento de aplicativos para o provisionamento de infraestrutura. Além disso, AWS CDK eles podem agilizar a criação de AWS infraestrutura. Isso acelera o ciclo de vida do desenvolvimento em comparação com a criação manual de modelos. CloudFormation

Vantagens de usar o AWS CDK:

- O AWS CDK suporta linguagens de programação conhecidas.
- As linguagens de uso geral permitem o uso de construções lógicas, como for-loops, objetos, tipos fortes e outras técnicas de programação. Isso ajuda os desenvolvedores a declarar a infraestrutura de forma concisa e sem erros. Essa abordagem também possibilita o uso de um ambiente de desenvolvimento integrado (IDE) e ferramentas relacionadas para ajudar a gerenciar a complexidade da declaração de um grande número de recursos.
- AWS CDK as construções são compartilháveis e ajudam você a atender aos requisitos de governança e conformidade.
- As AWS CDK construções podem diminuir o tempo e o esforço de desenvolvimento. Para obter mais informações, consulte a Referência da API Construct Library.
- O AWS CDK é baseado em CloudFormation. Se você estiver familiarizado com CloudFormation seus conceitos, os AWS CDK conceitos serão mais fáceis de entender.
- AWS CDK Isso ajuda você a realizar testes de unidade e testes instantâneos.
- Se um recurso não for suportado nativamente no AWS CDK, você pode usar uma construção de nível 1 e substituições brutas. Como alternativa, você pode usar um recurso CloudFormation personalizado que chame a API diretamente.
- Você pode limpar os recursos de forma eficiente excluindo as CloudFormation pilhas.

Desvantagens de usar o AWS CDK:

- AWS CDK Isso requer um ambiente inicializado em cada um. Conta da AWS O bootstrapping é uma ação única que você deve executar em cada ambiente em que você implanta recursos.
- O AWS CDK pode ser usado para implantar o IaC somente no Nuvem AWS.

Usando o Terraform como uma ferramenta IaC para o Nuvem AWS

HashiCorp O Terraform é uma ferramenta de infraestrutura como código (IaC) que ajuda você a gerenciar sua infraestrutura em nuvem. Usando o Terraform, você pode definir recursos locais e na nuvem em arquivos de configuração que podem ser versionados, reutilizados e compartilhados. Em seguida, você pode usar um fluxo de trabalho consistente para provisionar e gerenciar toda a sua infraestrutura durante todo o ciclo de vida.

Os desenvolvedores usam uma linguagem de configuração de alto nível chamada linguagem Terraform. A sintaxe nativa de baixo nível da linguagem Terraform é a Linguagem de HashiCorpConfiguração (HCL). A linguagem Terraform foi projetada para ser fácil para os humanos

lerem e escreverem. Você usa a linguagem Terraform para descrever o estado final desejado da nuvem ou da infraestrutura local. O Terraform então gera um plano para atingir esse estado final e você executa o plano para provisionar a infraestrutura.

Vantagens de usar o Terraform:

- O Terraform é independente de plataforma. Você pode usá-lo com qualquer provedor de serviços em nuvem. Você pode configurar, testar e implantar infraestrutura em AWS vários outros provedores de nuvem. Se sua organização usa vários provedores de nuvem, o Terraform pode ser uma solução única, unificada e consistente para gerenciar a infraestrutura em nuvem. Para obter mais informações sobre o suporte a várias nuvens, consulte Provisionamento em várias nuvens no site do Terraform.
- O Terraform não tem agente. Não é necessário instalar nenhum software na infraestrutura gerenciada.
- Os módulos do Terraform são uma maneira poderosa de reutilizar código e seguir o princípio Don't Repeat Yourself (DRY). Por exemplo, você pode ter uma configuração específica para um aplicativo que contém uma instância do Amazon Elastic Compute Cloud (Amazon EC2), volumes do Amazon Elastic Block Store (Amazon EBS) e outros recursos agrupados logicamente. Se precisar criar várias cópias dessa configuração ou aplicativo, você pode empacotar os recursos em um módulo do Terraform e criar várias instâncias do módulo em vez de copiar o código inteiro várias vezes. Esses módulos podem ajudá-lo a organizar, encapsular e reutilizar configurações. Eles também fornecem consistência e garantem as melhores práticas.
- O Terraform é capaz de detectar e gerenciar desvios (postagem no blog do Terraform) em sua infraestrutura. Por exemplo, se os recursos gerenciados pelo Terraform forem modificados fora do Terraform, você poderá detectar o desvio e restaurá-los ao estado desejado usando a CLI do Terraform.

Desvantagens de usar o Terraform:

- Support para novos recursos ou novos recursos relacionados a qualquer provedor de nuvem pode não estar disponível.
- O Terraform não gerencia automaticamente seu estado da mesma forma AWS CloudFormation. Ele é armazenado por padrão em um arquivo local, mas você também pode armazená-lo remotamente em um bucket do Amazon S3 ou por meio do Terraform Enterprise.
- O estado do Terraform pode conter dados confidenciais, como senhas de bancos de dados, o que pode causar problemas de segurança. É uma prática recomendada criptografar seu arquivo de

estado, armazená-lo remotamente, ativar o controle de versão de arquivos nele e usar o mínimo de privilégios para operações de leitura e gravação nele. Para obter mais informações, consulte Protegendo dados confidenciais usando o AWS Secrets Manager HashiCorp Terraform.

- Em agosto de 2023, a Hashicorp anunciou que não seria mais licenciada como código aberto sob a Licença Pública Mozilla. Em vez disso, agora está licenciado sob a Business Source License.

Usando o Pulumi como uma ferramenta IaC para o Nuvem AWS

O Pulumi é uma infraestrutura de código aberto como ferramenta de código. Ele oferece suporte a linguagens de programação comuns existentes TypeScript, como Python JavaScript, Go, .NET, Java e YAML. Ele também usa seu ecossistema nativo para interagir com os recursos da nuvem por meio do SDK Pulumi. Uma CLI disponível para download, um tempo de execução, bibliotecas e um serviço hospedado trabalham juntos para provisionar, atualizar e gerenciar a infraestrutura em nuvem.

Você pode usar o Pulumi SDK para criar e implantar software em nuvem que usa contêineres, funções sem servidor, serviços hospedados e infraestrutura em qualquer nuvem.

Opcionalmente, você pode emparelhar o Pulumi com o Pulumi Cloud. O Pulumi Cloud é um serviço gerenciado que armazena seu estado e segredos e gerencia as implantações de sua infraestrutura Pulumi.

Vantagens de usar o Pulumi:

- A Pulumi fornece infraestrutura de mais de cinquenta provedores de nuvem e software como serviço (SaaS).
- Ele oferece uma interface completa e consistente, projetada para reduzir a complexidade da nuvem.

Desvantagens de usar o Pulumi:

- Embora o Pulumi tenha uma comunidade ativa que oferece suporte, ele é menor do que a comunidade do Terraform.
- Pulumi tem uma curva de aprendizado mais acentuada.

Escolhendo uma ferramenta IaC

Então, qual ferramenta você deve escolher?

Com tantas opções de ferramentas diferentes e requisitos comerciais variados, não há nenhuma one-size-fits-all abordagem. Além das vantagens e desvantagens de cada ferramenta discutida neste guia, considere as seguintes recomendações para seus requisitos de negócios e modelo operacional:

- Se você estiver gerenciando ou implantando uma AWS solução sem servidor com dependência ou dependentes mínimos, AWS Serverless Application Model (AWS SAM) pode ser uma boa opção para você. Ele tem todos os mesmos recursos do AWS CloudFormation. Também simplifica o teste e a implantação de aplicativos sem servidor no. Nuvem AWS
- Se você está gerenciando sua infraestrutura totalmente ativada AWS, então, AWS CloudFormation e AWS Cloud Development Kit (AWS CDK) essas são boas opções. Eles fornecem gerenciamento de out-of-the-box estado e você também pode usar novos recursos ou AWS recursos de forma nativa.
- Se você deseja um utilitário de vários fornecedores, especialmente para gerenciar infraestrutura de várias nuvens ou de nuvem híbrida, o Terraform pode ser uma boa escolha porque é independente de plataforma. Com o Terraform, você também pode usar uma ampla variedade de plug-ins e ele tem uma grande comunidade com opções de suporte corporativo.
- Se você tiver uma distribuição de cima para baixo com as melhores práticas e se tiver uma orquestração na qual você cria, publica e distribui módulos reutilizáveis usando linguagens de programação comuns, essa AWS CDK pode ser uma boa opção.
- Se sua organização pode tolerar um alto nível de risco e precisa oferecer suporte a ambientes multinuvem ou de nuvem híbrida, considere usar o Pulumi.

Próximas etapas

Este guia discutiu as vantagens e desvantagens de várias ferramentas de infraestrutura como código (IaC) que você pode usar para criar, implantar e gerenciar AWS recursos e infraestrutura. Com base na ferramenta que você escolher, recomendamos que você visite os seguintes recursos para começar.

AWS Cloud Development Kit (AWS CDK)

- AWS CDK oficina de introdução

AWS CloudFormation

- AWS CloudFormation laboratório
- AWS CloudFormation Workshop

HashiCorp Terraforma

- AWS Workshop de Terraform
- CDK para repositório Terraform () GitHub

Pulumi

- Repositório Pulumi () GitHub
- AWS modernização com a oficina Pulumi

Recursos

AWS documentação

- Melhores práticas para usar o AWS CDK in TypeScript para criar projetos de IaC (orientação AWS prescritiva)
- Verifique as melhores práticas em AWS CDK aplicativos ou CloudFormation modelos usando pacotes de regras cdk-nag (orientação prescritiva)AWS
- Introdução a DevOps on AWS: Infraestrutura como código (AWS white paper)
- Documentação da AWS CloudFormation

Outra documentação

- Introdução à infraestrutura como código (HashiCorpsite)
- HashiCorpDocumentação do Terraform
- Documentação do Pulumi
- Pulumi: O que é infraestrutura como código (site Pulumi)

Ferramentas

- cdk-bag () GitHub
- cfn-bag () GitHub
- Terratest

Colaboradores

Autoria

- Siamak Heshmati, arquiteto sênior, DevOps AWS
- Mason Cahill, consultor sênior DevOps , AWS
- Sandip Gangapadhyay, arquiteto sênior, DevOps AWS
- Sandeep Gawande, consultor líder sênior, AWS
- Rajneesh Tyagi, consultor principal, AWS

Analisando

- David Howerton, gerente sênior de engajamento, AWS
- Steven Guggenheimer, arquiteto sênior de aplicativos em nuvem, AWS

Redação técnica

- Lilly AbouHarb, redatora técnica sênior, AWS

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
Atualizações do Pulumi	Atualizamos o capítulo Pulumi.	17 de fevereiro de 2026
Publicação inicial	—	23 de fevereiro de 2024

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.