Aula 9 – Infraestrutura como Código: AWS SAM (Serverless Application Model)

Bem-vindo à nona aula do nosso curso de Computação Serverless! Se você já se aventurou no mundo do desenvolvimento de aplicações sem servidor, sabe que a promessa de escalar automaticamente e pagar apenas pelo uso é incrivelmente atraente. No entanto, à medida que suas aplicações crescem e se tornam mais complexas, gerenciar todas as funções, APIs, bancos de dados e outros recursos manualmente pode se transformar em um verdadeiro desafio. É como tentar construir uma casa sem um projeto detalhado, adicionando tijolos e fiação conforme a necessidade surge – o resultado é, no mínimo, caótico.

É nesse cenário que a **Infraestrutura como Código (IaC)** entra em cena, revolucionando a forma como lidamos com a complexidade da nuvem. Em vez de configurar recursos clicando em interfaces gráficas, nós os descrevemos em arquivos de texto, que podem ser versionados, revisados e automatizados. Para o universo serverless da AWS, uma das ferramentas mais poderosas e nativas para essa tarefa é o **AWS Serverless Application Model (SAM)**. Ele atua como um arquiteto especializado, traduzindo suas intenções de aplicação serverless em um plano de construção executável.

Nesta aula, nosso objetivo é desvendar o AWS SAM, compreendendo sua estrutura, suas vantagens e como ele se posiciona frente a outras ferramentas populares. Ao final, você será capaz de entender a importância do IaC para aplicações serverless, identificar a estrutura de um template SAM, comparar o SAM com o Serverless Framework e, o mais importante, utilizar a **SAM CLI** para construir, testar e implantar suas próprias aplicações serverless de forma eficiente e automatizada. Prepare-se para elevar o nível da sua gestão de infraestrutura na nuvem!

O Desafio da Gestão de Infraestrutura Serverless e a Solução IaC

Imagine que você está construindo um castelo de Lego. No início, com poucas peças, é fácil montá-lo e desmontá-lo. Mas e se o castelo crescer, com centenas de peças, torres, pontes e até um fosso? Reconstruí-lo manualmente, garantindo que cada peça esteja no lugar certo, seria exaustivo e propenso a erros. No mundo serverless, a situação é semelhante: uma aplicação pode envolver dezenas de funções Lambda, APIs Gateway, tabelas DynamoDB, filas SQS e muito mais. Gerenciar tudo isso manualmente, através do console da AWS, torna-se rapidamente inviável e arriscado.

Inconsistências entre Ambientes

Replicar configurações manualmente em desenvolvimento, teste e produção aumenta riscos de falhas

Documentação Desatualizada

Configurações manuais raramente são documentadas adequadamente, dificultando manutenção

Gargalo Operacional

A agilidade prometida pelo serverless se transforma em complexidade de gestão

É aqui que a **Infraestrutura como Código (IaC)** surge como uma solução elegante e poderosa. Em vez de configurar recursos de nuvem manualmente, você os descreve em arquivos de texto, usando uma linguagem declarativa. Esses arquivos se tornam a "planta baixa" do seu castelo serverless, detalhando cada componente e sua interconexão. Com essa abordagem, a infraestrutura se torna versionável, auditável e replicável, assim como o código da sua aplicação. Você pode usar ferramentas de controle de versão como o Git para gerenciar as mudanças na sua infraestrutura, permitindo revisões, reversões e colaboração de forma transparente.

AWS SAM: O Modelo de Aplicação Serverless da AWS

CloudFormation

Ferramenta nativa e fundamental para IaC na AWS. Incrivelmente poderoso, permite definir praticamente qualquer recurso da AWS em um template.

Desafio: Verbosidade pode ser um obstáculo, especialmente para aplicações serverless. É como programar em Assembly - controle total, mas muitos detalhes.

AWS SAM

Extensão do CloudFormation com sintaxe mais concisa e de alto nível para recursos serverless. Atua como um "atalho" ou "biblioteca" de componentes pré-configurados.

Vantagem: Permite declarar funções Lambda, APIs e tabelas DynamoDB com muito menos código que CloudFormation puro.

A grande sacada do SAM é que ele **se transforma** em CloudFormation. Quando você implanta um template SAM, ele é primeiro convertido para um template CloudFormation completo, que então é usado para provisionar os recursos na sua conta AWS. Isso significa que você obtém a simplicidade do SAM sem perder a robustez e a capacidade de gerenciamento do CloudFormation. É como ter um assistente que pega suas instruções simplificadas e as traduz para um manual técnico detalhado que a AWS entende perfeitamente. Essa integração nativa com o ecossistema AWS é uma das suas maiores forças, garantindo compatibilidade e acesso às últimas funcionalidades da plataforma.

Anatomia de um Template SAM (template.yaml)

Agora que entendemos o que é o AWS SAM, vamos mergulhar na sua estrutura. Um template SAM é um arquivo YAML (ou JSON, mas YAML é mais comum pela legibilidade) que descreve os recursos da sua aplicação serverless. Ele é o coração da sua definição de infraestrutura, e entender seus componentes é fundamental para começar a construir. A estrutura básica de um template SAM é bastante familiar para quem já trabalhou com CloudFormation, mas com algumas adições e simplificações importantes.

01

AWSTemplateFormatVersion

Especifica a versão do formato do template

02

Transform: AWS::Serverless-2016-10-31

A mágica que ativa o "modo serverless" do CloudFormation

03

Resources

Onde você define todos os componentes da aplicação

No topo de qualquer template SAM, você encontrará a seção AWSTemplateFormatVersion, que especifica a versão do formato do template. Logo abaixo, e crucial para o SAM, está a propriedade Transform: AWS::Serverless-2016-10-31. Esta linha é a mágica que diz ao CloudFormation para interpretar o template usando a sintaxe simplificada do SAM. Sem ela, o CloudFormation não saberia como expandir os recursos serverless concisos que você vai definir. É como ligar um "modo serverless" para o seu template.

Tipos de Recursos SAM

- **AWS::Serverless::Function** - Funções Lambda com permissões e eventos
- **AWS::Serverless::Api** - API Gateway configurado
- **AWS::Serverless::SimpleTable** - Tabelas DynamoDB simplificadas

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Um template SAM simples para uma função Lambda.

Resources:
  MinhaFuncaoServerless:
    Type: AWS::Serverless::Function
    Properties:
      Handler: app.lambda_handler
      Runtime: python3.9
      CodeUri: s3://seu-bucket-de-codigo/minha-funcao.zip
      MemorySize: 128
      Timeout: 30
      Events:
        MinhaApi:
          Type: Api
          Properties:
            Path: /hello
            Method: get
```

Neste exemplo, MinhaFuncaoServerless é uma função Lambda Python que responde a requisições GET no caminho /hello através de um API Gateway, tudo definido em poucas linhas.

Detalhando os Recursos SAM: Funções e APIs

Aprofundando na seção Resources, vamos focar nos tipos de recursos mais comuns e essenciais para a maioria das aplicações serverless: as funções Lambda e as APIs Gateway. Estes são os pilares sobre os quais muitas arquiteturas sem servidor são construídas, e o SAM oferece uma maneira elegante de defini-los. Compreender como configurar esses elementos no seu template é o primeiro passo para construir aplicações robustas e escaláveis.

AWS::Serverless::Function

O tipo de recurso AWS::Serverless::Function é a estrela do show. Ele encapsula uma função AWS Lambda, mas vai além, permitindo que você defina não apenas o código e o runtime, mas também as permissões de execução (IAM Role), a quantidade de memória, o tempo limite e, crucialmente, os eventos que a disparam. Por exemplo, você pode configurar um evento Api para integrar a função com o API Gateway, um evento SQS para processar mensagens de uma fila, ou um evento S3 para reagir a uploads de arquivos. Essa capacidade de definir a função e seus gatilhos em um único bloco é o que torna o SAM tão poderoso e conciso.

Handler

Ponto de entrada do código

Runtime

Linguagem (nodejs18.x, python3.9)

CodeUri

Local do código (S3 ou caminho local)

Events

Gatilhos que invocam a função

Dentro das Properties de uma AWS::Serverless::Function, você encontrará chaves como Handler (o ponto de entrada do seu código), Runtime (a linguagem de programação, como nodejs18.x ou python3.9), CodeUri (o local do seu código, que pode ser um caminho local ou um bucket S3), e Events. A seção Events é onde a mágica da integração acontece. Ao definir um Type: Api dentro de Events, o SAM automaticamente provisiona um API Gateway e configura a rota e o método HTTP para invocar sua função Lambda. É como ter um assistente que não só constrói a sala, mas também instala a porta e a campainha para você.

```
Resources:
  MinhaFuncaoDeSaudacao:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/app.handler
      Runtime: nodejs18.x
      CodeUri: src/
      MemorySize: 128
      Timeout: 10
      Environment:
        Variables:
          NOME_APP: SaudacaoApp
      Events:
        ApiGatewayEvent:
          Type: Api
          Properties:
            Path: /saudacao
            Method: get
            RestApiId: !Ref MinhaApiPrincipal

  MinhaApiPrincipal:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      DefinitionBody:
        swagger: '2.0'
        info:
          title: "MinhaApiPrincipal"
        paths:
          /saudacao:
            get:
              x-amazon-apigateway-integration:
                httpMethod: POST
                type: aws_proxy
                uri: !Sub
"arn:${AWS::Partition}:apigateway:${AWS::Region}:lambda:path/2015-
03-31/functions/${MinhaFuncaoDeSaudacao.Arn}/invocations"
```

Neste exemplo, a função MinhaFuncaoDeSaudacao é exposta via API Gateway no caminho /saudacao. O SAM cuida de toda a complexidade de conectar esses serviços, permitindo que você se concentre na lógica de negócios da sua aplicação.

Serverless Framework vs. AWS SAM: Uma Escolha Estratégica

Ao explorar o universo da Infraestrutura como Código para aplicações serverless, é quase inevitável se deparar com duas ferramentas dominantes: o AWS SAM e o Serverless Framework. Ambas visam simplificar o desenvolvimento e a implantação de aplicações sem servidor, mas o fazem com abordagens e filosofias distintas. Entender essas diferenças é crucial para tomar uma decisão estratégica sobre qual ferramenta melhor se alinha às suas necessidades e ao contexto do seu projeto.

AWS SAM

Pense nessas ferramentas como dois tipos de veículos para uma viagem. O **AWS SAM** é como um carro de corrida de alta performance, otimizado e construído especificamente para as estradas da AWS. Ele é nativo, profundamente integrado com o CloudFormation e o restante do ecossistema AWS. Isso significa que ele tende a ser mais simples para quem já está totalmente imerso na AWS, aproveitando ao máximo as funcionalidades da plataforma sem a necessidade de configurações adicionais. Sua curva de aprendizado para quem já conhece CloudFormation é mais suave, e ele se beneficia diretamente das inovações da AWS.

Serverless Framework

Por outro lado, o **Serverless Framework** é como um veículo utilitário robusto e versátil, capaz de trafegar em diferentes tipos de terreno. Ele é uma solução multi-cloud, o que significa que você pode usá-lo para implantar aplicações serverless não apenas na AWS, mas também no Google Cloud Platform (GCP), Azure e outras plataformas. Sua força reside na sua flexibilidade, no vasto ecossistema de plugins e na grande comunidade. Se você tem uma estratégia multi-cloud ou precisa de funcionalidades muito específicas que podem ser adicionadas via plugins, o Serverless Framework oferece essa liberdade. No entanto, essa flexibilidade pode vir com uma curva de aprendizado um pouco mais íngreme e a necessidade de gerenciar mais dependências.

Como Escolher?

Vá de SAM se: Você está 100% comprometido com a AWS e busca a maior simplicidade e integração nativa.

Escolha Serverless Framework se: Você precisa de flexibilidade multi-cloud, um ecossistema de plugins rico ou já tem experiência com ele.

Característica	AWS SAM	Serverless Framework
Natureza	Extensão do AWS CloudFormation	Abstração multi-cloud (Node.js CLI)
Provedores Suportados	Exclusivamente AWS	AWS, Azure, GCP, OpenWhisk, etc.
Curva de Aprendizado	Mais fácil para AWS-only, CloudFormation	Mais flexível, mas pode exigir plugins
Ecossistema	Nativo AWS, ferramentas da AWS	Grande comunidade, vasto ecossistema de plugins
Foco Principal	Simplificar o desenvolvimento serverless na AWS	Abstrair a complexidade da nuvem para serverless

A SAM CLI: Seu Companheiro de Desenvolvimento Serverless

Definir sua infraestrutura em um template SAM é apenas o primeiro passo. Para transformar esse plano em uma aplicação real na nuvem, você precisa de uma ferramenta que possa interpretar o template, empacotar seu código e interagir com a AWS. É aí que entra a **AWS SAM Command Line Interface (CLI)**. A SAM CLI é a ferramenta essencial para desenvolvedores que trabalham com o AWS SAM, atuando como a ponte entre o seu código local e a nuvem da AWS.

Pense na SAM CLI como o "controle remoto" da sua aplicação serverless. Com ela, você pode inicializar novos projetos, construir seu código, testar suas funções Lambda localmente (simulando o ambiente da AWS!) e, finalmente, implantar sua aplicação na nuvem. Ela simplifica tarefas complexas que, de outra forma, exigiriam múltiplos comandos do AWS CLI ou interações com o console. A SAM CLI é construída sobre o AWS CLI, mas adiciona comandos específicos e otimizados para o fluxo de trabalho serverless, tornando sua experiência de desenvolvimento muito mais fluida e produtiva.

sam init

Gerador de projetos que cria estrutura inicial com template SAM e código de exemplo para diferentes runtimes

sam build

Prepara aplicação para implantação, empacotando código e resolvendo dependências

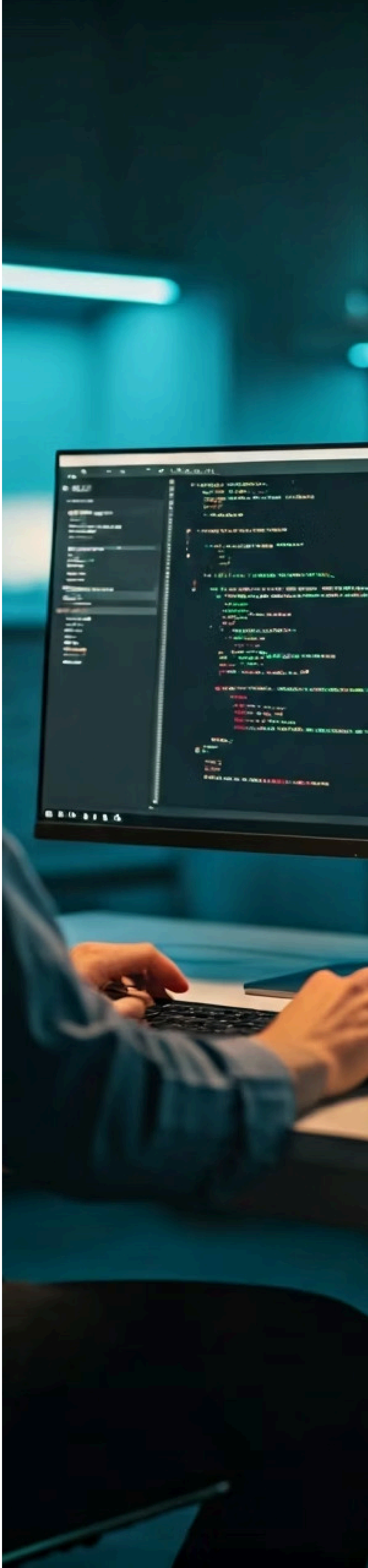
sam local

Testa funções localmente, simulando ambiente AWS no seu computador

sam deploy

Implanta aplicação na AWS, criando ou atualizando stack CloudFormation

Um dos primeiros comandos que você provavelmente usará é sam init. Este comando é um gerador de projetos, permitindo que você comece rapidamente com um template básico para diferentes runtimes e tipos de aplicações serverless. Ele cria uma estrutura de diretórios com um template SAM (template.yaml) e um código de exemplo, dando-lhe um ponto de partida sólido. Em seguida, o comando sam build é crucial. Ele prepara sua aplicação para implantação, empacotando seu código, resolvendo dependências (como módulos Node.js ou Python) e criando um artefato de implantação que pode ser enviado para a AWS. É como um "compilador" para sua aplicação serverless, garantindo que tudo esteja pronto para subir para a nuvem.



Teste Local e Deploy com SAM CLI

A capacidade de testar sua aplicação serverless localmente antes de implantá-la na nuvem é um divisor de águas no desenvolvimento. Evitar ciclos de feedback lentos, onde cada teste requer um deploy completo, economiza tempo e recursos. A SAM CLI oferece ferramentas robustas para simular o ambiente AWS no seu próprio computador, permitindo que você itere rapidamente e depure seu código de forma eficiente.

Desenvolvimento Local

Escreva e teste código no seu ambiente

Teste com SAM Local

Simule Lambda e API Gateway localmente

Validação

Verifique comportamento antes do deploy

Deploy na AWS

Implante com confiança na nuvem

Comandos de Teste Local

sam local invoke

Simula execução de função Lambda, passando evento de teste (JSON). Verifica lógica sem conexão AWS.

sam local start-api

Emula API Gateway localmente. Faça requisições HTTP para sua máquina e veja funções Lambda em tempo real.

Deploy Automatizado

Uma vez que sua aplicação esteja testada e pronta, o comando sam deploy é o responsável por levá-la para a nuvem. Este comando automatiza o processo de empacotar seu código (se você não usou sam build explicitamente), fazer upload dos artefatos para um bucket S3, e então criar ou atualizar uma stack CloudFormation com base no seu template SAM. O sam deploy --guided é particularmente útil para a primeira implantação, pois ele o guiará através das configurações necessárias, como o nome da stack, a região da AWS e as permissões. É um processo que transforma seu template e código em uma aplicação serverless funcional e escalável na AWS, com apenas alguns comandos.

```
# Exemplo de fluxo de trabalho com SAM CLI

# 1. Inicializa um novo projeto (se ainda não tiver um)
# sam init --runtime nodejs18.x --name minha-app-serverless --app-template hello-world

# 2. Constrói a aplicação (empacota dependências, etc.)
sam build

# 3. Testa a função localmente com um evento de exemplo
# (Crie um arquivo event.json com o payload da sua requisição)
sam local invoke MinhaFuncaoDeSaudacao -e event.json

# 4. Inicia um API Gateway local para testar via HTTP
sam local start-api

# 5. Implanta a aplicação na AWS (primeira vez com --guided)
sam deploy --guided
```

Este fluxo de trabalho integrado da SAM CLI acelera significativamente o ciclo de desenvolvimento, permitindo que você se concentre na inovação, e não na complexidade da infraestrutura.

Tendências e o Futuro da Infraestrutura Serverless com SAM

O cenário da computação serverless está em constante evolução, com inovações surgindo a um ritmo acelerado. O AWS SAM, como uma ferramenta nativa e fundamental para a AWS, precisa se adaptar e incorporar essas tendências para permanecer relevante e eficaz. Compreender essas direções futuras nos ajuda a posicionar nossas habilidades e projetos para o sucesso a longo prazo.

Evolução do FaaS

Funções Lambda mais robustas com tempos de execução mais longos, gerenciamento de estado sofisticado e capacidade para cargas intensivas

Serverless Containers

Fargate e Cloud Run borram linhas entre serverless e contêineres, oferecendo flexibilidade com simplicidade operacional

Automação Avançada

Ferramentas IaC evoluem para atender arquiteturas serverless cada vez mais complexas e difundidas

Uma das tendências mais notáveis é a evolução do **Function-as-a-Service (FaaS)**. Inicialmente, as funções Lambda eram vistas como pequenos blocos de código para tarefas rápidas e sem estado. No entanto, a demanda por funções mais robustas e com maior capacidade está crescendo. Isso se traduz em suporte a tempos de execução mais longos, gerenciamento de estado mais sofisticado (muitas vezes via integração com serviços como Step Functions ou DynamoDB) e a capacidade de lidar com cargas de trabalho mais intensivas. O SAM, ao simplificar a orquestração desses serviços auxiliares, facilita a construção de aplicações FaaS mais complexas e duradouras.

Outra área de crescimento explosivo são os **Serverless Containers**. Tecnologias como AWS Fargate e Google Cloud Run estão borrando as linhas entre serverless e contêineres, oferecendo a flexibilidade dos contêineres (permitindo qualquer runtime e dependências personalizadas) com a simplicidade operacional e o modelo de pagamento por uso do serverless. Embora o SAM seja focado principalmente em FaaS, a AWS tem investido em integrar contêineres com Lambda (via imagens de contêiner), e o SAM já suporta essa abordagem. No futuro, podemos esperar que o SAM continue a evoluir para abranger mais plenamente esses paradigmas de contêineres serverless, permitindo que os desenvolvedores definam e implantem cargas de trabalho baseadas em contêineres com a mesma facilidade que hoje definem funções Lambda.

A contínua evolução das ferramentas de Infraestrutura como Código, como o próprio SAM e o Serverless Framework, é um reflexo da necessidade de automação e padronização. À medida que as arquiteturas serverless se tornam mais difundidas e complexas, a capacidade de definir, gerenciar e implantar toda a infraestrutura de forma programática não é mais um luxo, mas uma necessidade. O SAM, com sua forte integração com a AWS, está bem posicionado para continuar sendo uma ferramenta de escolha para quem busca otimizar o desenvolvimento serverless na plataforma da Amazon.

Consolidação e Próximos Passos

Chegamos ao fim da nossa jornada pela Infraestrutura como Código com AWS SAM. Vimos como a gestão manual de recursos serverless pode ser um gargalo e como o IaC, com o SAM, oferece uma solução elegante e automatizada. Exploramos a estrutura de um template SAM, detalhamos a definição de funções e APIs, e comparamos o SAM com o Serverless Framework, destacando suas particularidades. Por fim, mergulhamos na SAM CLI, a ferramenta que nos permite construir, testar localmente e implantar nossas aplicações serverless com eficiência.

Em prática

A partir de agora, ao desenvolver suas aplicações serverless na AWS, priorize a abordagem de Infraestrutura como Código utilizando o AWS SAM. Isso não só garantirá que sua infraestrutura seja consistente e replicável, mas também acelerará seu ciclo de desenvolvimento e facilitará a colaboração em equipe. Use a SAM CLI para testar suas funções localmente antes de qualquer deploy, economizando tempo e recursos.

Autoavaliação

1. Qual é o principal propósito da Infraestrutura como Código (IaC) no contexto de aplicações serverless?
   - a) Apenas documentar a infraestrutura de forma visual.
   - b) Automatizar o provisionamento e gerenciamento de recursos de nuvem através de arquivos de configuração.
   - c) Substituir completamente o uso de serviços de nuvem por soluções on-premise.
   - d) Reduzir a necessidade de testes em ambientes de desenvolvimento.
2. Qual propriedade é essencial em um template SAM para que o CloudFormation o interprete corretamente como um modelo serverless?
   - a) AWSTemplateFormatVersion
   - b) Description
   - c) Transform: AWS::Serverless-2016-10-31
   - d) Resources
3. Qual dos seguintes comandos da SAM CLI é utilizado para simular a execução de uma função Lambda localmente, sem a necessidade de implantá-la na AWS?
   - a) sam deploy
   - b) sam build
   - c) sam init
   - d) sam local invoke
4. Em comparação com o Serverless Framework, qual é uma característica marcante do AWS SAM?
   - a) É uma solução multi-cloud, suportando diversos provedores.
   - b) Possui um vasto ecossistema de plugins para estender funcionalidades.
   - c) É uma extensão nativa do AWS CloudFormation, focada exclusivamente na AWS.
   - d) Exige uma curva de aprendizado mais íngreme para desenvolvedores AWS.
5. Você está iniciando um novo projeto serverless na AWS e precisa decidir entre usar AWS SAM ou Serverless Framework. Descreva um cenário em que o AWS SAM seria a escolha mais vantajosa e justifique sua resposta.

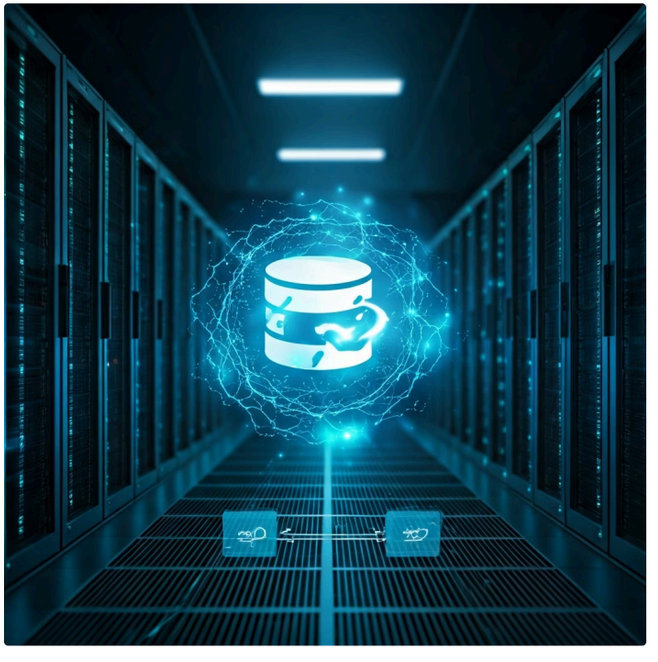
Gabarito

1. b) **2.** c) **3.** d) **4.** c)

Próxima Jornada: Bancos de Dados Serverless

Conexão com a Próxima Aula

Nesta aula, focamos em como construir e gerenciar a infraestrutura de computação serverless. Mas onde esses dados são armazenados? Na **Aula 10 – Bancos de Dados Serverless: NoSQL com DynamoDB**, exploraremos como os bancos de dados sem servidor, como o DynamoDB da AWS, se integram perfeitamente a essas arquiteturas, oferecendo escalabilidade e performance sem a complexidade de gerenciar servidores de banco de dados.



Recursos Adicionais

- **Documentação Oficial do AWS SAM:** Para aprofundar nos detalhes de cada recurso e comando.
- **Repositório de Exemplos do AWS SAM:** Para ver exemplos práticos de templates e aplicações.
- **Blog da AWS Serverless:** Para se manter atualizado sobre as últimas tendências e funcionalidades.

NOTA IMPORTANTE: As informações regulatórias/legais/técnicas desta aula estão atualizadas até 2025. Consulte sempre fontes oficiais para verificar alterações.