Soluções para Desenvolvimento e Ferramentas de Gerenciamento

Filipe Almeida

2023

SUMÁRIO

Capítulo 1.Ferramentas de gerenciamento e configuração de ambientes na AWS

Esse capítulo tem como objetivo apresentar as formas de interação, acesso, gerenciamento e administração de ambientes na AWS. Podemos fazer isso através do tão conhecido console de gerenciamento da AWS, da interface de linha de comando da AWS (AWS CLI), do console mobile e também através dos SDKs existentes para algumas linguagens de programação.

Console de Gerenciamento da AWS

- https://docs.aws.amazon.com/pt_br/IAM/latest/UserGuide/console.html

O AWS Management Console fornece uma interface de usuário baseada na Web que pode ser usada para criar e gerenciar recursos da AWS. Por exemplo, é possível iniciar e interromper instâncias EC2, criar buckets do S3 e bancos de dados do RDS, etc.

Você precisa efetuar login na sua conta da AWS para começar a utilizar o console. Esse login pode ser feito tanto através de usuários do IAM, quanto através do usuário raiz da conta.

Usuário raiz da conta

É aquele usuário que possui as credenciais, e-mail e senha, que foram utilizadas para criar a conta da AWS. É recomendado pela própria AWS que este usuário nunca seja utilizado para gerenciar os seus recursos na AWS. Recomenda-se, inclusive, habilitar o MFA para este usuário e utilizá-lo somente em casos emergenciais e para funções muito específicas no console.

Usuário do IAM

É um usuário criado dentro da sua conta da AWS que recebe permissões de acordo com os privilégios organizacionais que aquela(e) pessoa/objeto deve ter na empresa.

Exemplo:

Se o Neymar é um desenvolvedor e trabalha na empresa PSG, que utiliza a AWS como provedor de computação em nuvem, ele vai receber um usuário na AWS que tenha as permissões adequadas que um desenvolvedor deve ter.

Por outro lado, o Messi, que é recém-chegado na empresa PSG, foi contratado para ser arquiteto de soluções em nuvem e ele precisa de permissões de administrador para gerenciar toda a infraestrutura que a empresa PSG utiliza na AWS. Inclusive, o Messi tem permissões até mesmo para criar outros usuários na AWS, à medida que novas contratações chegam à empresa PSG.

Com os usuários do IAM é possível realizar as operações de gerenciamento da conta da AWS, respeitando, é claro, as permissões concedidas a cada um dos usuários criados.

AWS Console Mobile Application

- https://aws.amazon.com/pt/console/mobile/

O console de gerenciamento para dispositivos Android e iOS permite visualizar e gerenciar alguns recursos da AWS, são eles:

- EC2;
- Load Balancers;
- Zonas Hospedadas do Route53 (Serviço de DNS da AWS);
- Bancos de dados do RDS;
- Grupos de Auto Scaling;
- Elastic Beanstalk;
- Tabelas do DynamoDB;
- CloudFormation;
- OpsWorks;
- CloudWatch.

Entendo o console mobile como sendo um complemento para o console web de gerenciamento da AWS, citado anteriormente.

AWS Command Line Interface

- https://docs.aws.amazon.com/cli/latest/index.html

O AWS CLI é um pacote para linha de comando, disponível para vários sistemas operacionais, que permite interagir com o seu ambiente na AWS através do seu terminal de linha de comando.

O CLI habilita você a controlar e automatizar a criação, alteração e remoção dos seus serviços da AWS utilizando scripts, por exemplo.

Exemplo de comando:

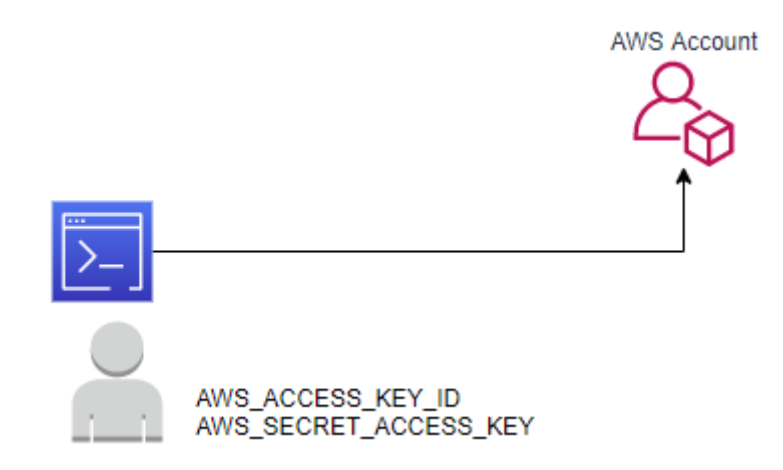
Comando para iniciar uma determinada instância EC2 conhecida pelo seu ID:

< aws ec2 start-instances --instance-ids i-1348636c>

Comando para copiar todo o conteúdo de uma pasta local para um bucket do S3:

< aws s3 cp myfolder s3://mybucket/myfolder --recursive>

Figura 1 – Command Line Interface.



Permissões para o AWS CLI

As permissões para o AWS CLI seguem a mesma lógica de atribuição de permissão para usuários do IAM. Isso porque é necessário utilizar credenciais para o acesso via CLI e estas credenciais são criadas no usuário do IAM que as utilizará para o acesso via CLI, que também chamamos de acesso programático. Vejamos um exemplo para melhorar a compreensão sobre esse assunto:

O usuário Neymar, citado anteriormente, possui as suas credenciais (nome de usuário e senha) para acesso via console web. E ele consegue fazer esse acesso tranquilamente sem nenhum tipo de problema. Através desse acesso, Neymar consegue gerenciar somente os recursos que ele possui permissão para tal.

Como Neymar quer sempre ser melhor no que faz, ele deseja passar a utilizar o acesso à mesma conta da AWS também através do CLI e não somente através do console web. Através desse acesso CLI Neymar conseguirá automatizar alguns recursos de desenvolvimento que ele gerencia. Portanto ele solicita para o administrador da conta que o mesmo crie para ele credenciais programáticas que permitirão com que ele interaja com o ambiente da AWS através da sua linha de comando.

O administrador da conta entrega para Neymar essas credenciais, que são configuradas no seu terminal, e a partir daquele momento, Neymar consegue interagir com os recursos da AWS através do seu terminal. Neymar conseguirá gerenciar apenas os recursos que ele tem permissão para administrar, da mesma forma que acontece ao acessar via console web ou via aplicação mobile.

Gerenciamento de múltiplas conta da AWS via CLI

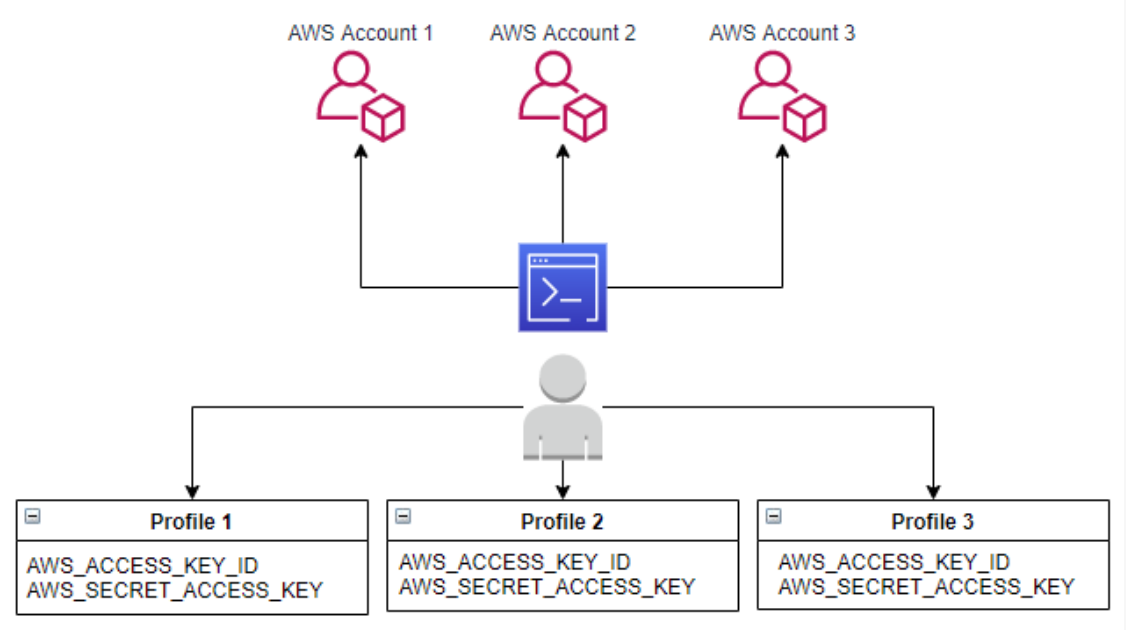
Como vimos, as suas credenciais de acesso via CLI estão vinculadas ao seu usuário do IAM, portanto, também estão vinculadas à sua conta da AWS. Você pode precisar, em algum momento, ter acesso de gerenciamento à múltiplas contas da AWS.

Para isso, você precisará ter um repositório seguro e centralizado onde você armazena as suas credenciais programáticas para acesso a essas múltiplas contas da AWS. A cada vez que você precisar acessar cada uma dessas contas, você deverá copiar as credenciais e configurá-las novamente em seu terminal de linha de comando. No dia a dia, isso pode se tornar uma tarefa repetitiva e chata.

Para facilitar esse acesso a múltiplas contas da AWS via CLI, a própria AWS permite com que você configure perfis para acesso. Com os perfis configurados, as suas credenciais já ficam vinculadas a esses perfis e basta você trocar o perfil que está sendo utilizado naquele momento para acessar uma outra conta da AWS.

A imagem a seguir ilustra isso. Um usuário com três perfis em seu terminal, cada perfil já previamente configurado com a sua respectiva credencial programática do IAM. Esse usuário consegue facilmente trocar os perfis para acessar os recursos de múltiplas contas da AWS. Lembrando que esse acesso será governado pelas permissões que serão atribuídas a esses perfis em cada uma dessas contas da AWS.

Figura 2 – Exemplo de Gerenciamento de múltiplas contas da AWS via CLI.



Através desse link, você consegue abrir a lista de comandos suportados pelo AWS CLI.

AWS SDKs

O AWS SDK, ou Software Development Kit, são um conjunto de bibliotecas fornecidas pela AWS que facilitam a interação com ambientes na AWS utilizando uma das linguagens de programação compatível com o AWS SDK e que sejam da sua preferência.

Linguagens suportadas:

- Java;

- C++;
- Go;
- JavaScript;
- Node.js;
- .NET;
- PHP;
- Python;
- Ruby.

Você consegue escrever scripts de interação com o ambiente da AWS, por exemplo, em sua linguagem de programação preferida (dentro das que estão disponíveis), usando as bibliotecas disponibilizadas pela AWS para cada uma dessas linguagens.

> Capítulo 2

Capítulo 2.Soluções para a IaC na AWS

Nesse capítulo veremos as vantagens de utilizarmos código para gerenciar a nossa infraestrutura criada na AWS, em vez de realizar as operações manualmente através do console de gerenciamento web, por exemplo, ou mesmo através do CLI.

Veremos também quais ferramentas a AWS entrega para gerenciar a infraestrutura através de código e como utilizá-la.

Por último, veremos uma outra solução robusta de mercado para provisionar e gerenciar infraestrutura também através de código, o Terraform.

Por que IaC?

- https://blog.tivit.com/o-que-e-infraestrutura-como-codigo

Talvez nesse momento, sobretudo ao ler os parágrafos de introdução desse capítulo, você esteja se perguntando: “Porque eu deveria desistir de criar os meus recursos de infraestrutura na AWS manualmente e fazer isso através de código? Não seria complexo fazer isso?”. Vamos responder essas perguntas nesse capítulo.

A IaC nasceu como uma metodologia de automação de infraestrutura de TI usada primordialmente para que equipes DevOps (Desenvolvedores e Operação conjunta) possam gerenciar e provisionar infraestrutura por meio de código de forma automatizada, ao invés de ter que recorrer ao acesso físico ao hardware ou até mesmo através de portais ou ferramentas de configuração.

Resumindo, a abordagem IaC nada mais é do que a entrega de uma infraestrutura ágil, utilizando-se de codificação simples e objetiva, sem a

necessidade de diversos passos e processos para se preparar um ambiente, sem perder o poder de controle, segurança, qualidade e disponibilidade.

Isso se deve em grande parte ao fato de que usar "código", para desenvolvedores de software, significa que no lugar de administradores de sistema, um dev pode fornecer e gerenciar os recursos tecnológicos de operações do que quer que seja que ele está criando.

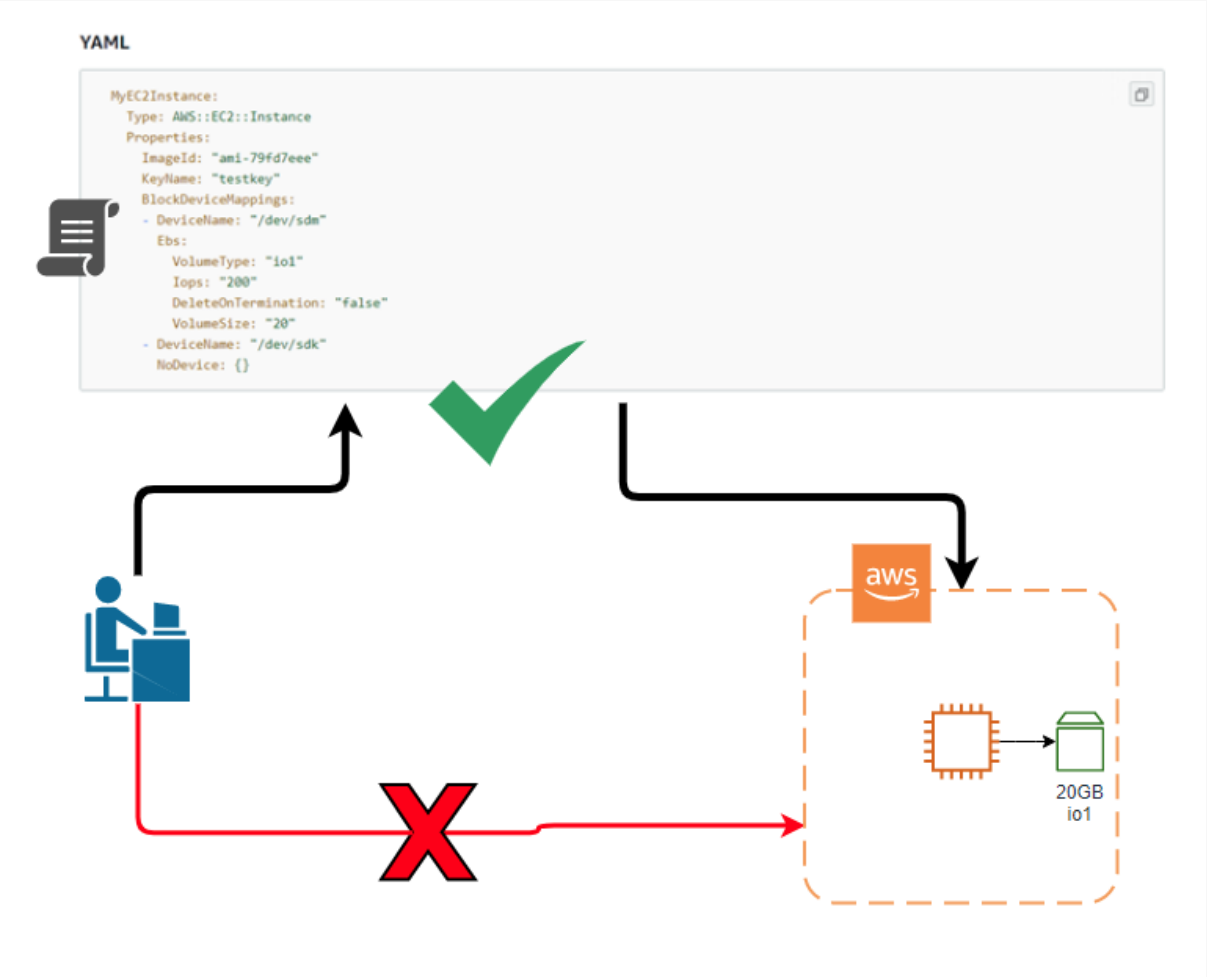
Essa tecnologia é muito semelhante aos scripts de programação que são usados para automatizar processos - porém, nesta modalidade, os scripts são usados para automatizar etapas de configuração que devem ser repetidas várias vezes em vários servidores.

Esse tipo de automação, além de trazer um ganho de velocidade nas entregas - beneficiando fortemente organizações que precisam de rapidez em suas respostas -, também diminui a incidência de erros que poderiam ser causados por um administrador.

A documentação de cada processo também se torna muito mais simples e confiável, uma vez que o próprio script se torna o registro documental de todas as suas próprias modificações. Assim, a credibilidade e a rastreabilidade de cada projeto são exponencialmente maiores.

Por se ter muito mais controle do ambiente e previsibilidade programável de variação dos recursos, a gestão financeira é outro ponto de alto ganho com a aplicação dessa tecnologia. Ao determinar que um certo volume de recursos precisa estar ativo ou pode ser desligado dinamicamente baseado no interesse de tráfego ou carga de processamento, atinge-se o estado da arte da gestão financeira de sistemas e rodando em nuvens.

Figura 3 - Ambiente sendo criado via IaC e não manualmente.



Cloud Formation – A Solução de IaC nativa da AWS

- https://aws.amazon.com/pt/cloudformation/

Como ferramenta de IaC da própria AWS para provisionamento de recursos, temos o CloudFormation, que é capaz de criar e gerenciar recursos somente dentro da nuvem da AWS e de alguns parceiros terceiros que estão no registro público do CloudFormation.

Pode ser escrito em duas linguagens:

- YAML;
- JSON.

Figura 4 - Exemplo de código em JSON.

JSON

```
"MyEC2Instance" : {
   "Type" : "AWS::EC2::Instance",
   "Properties" : {
      "ImageId" : "ami-79fd7eee",
      "KeyName" : "testkey",
      "BlockDeviceMappings" : [
         {
            "DeviceName" : "/dev/sdm",
            "Ebs" : {
              "VolumeType" : "io1",
              "Iops" : "200",
              "DeleteOnTermination" : "false",
              "VolumeSize" : "20"
            }
          },
          {
            "DeviceName" : "/dev/sdk",
            "NoDevice" : {}
          }
      ]
   }
}
```

Figura 5 - Exemplo de código em YAML.

YAML

```
MyEC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: "ami-79fd7eee"
    KeyName: "testkey"
    BlockDeviceMappings:
    - DeviceName: "/dev/sdm"
      Ebs:
        VolumeType: "io1"
        Iops: "200"
        DeleteOnTermination: "false"
        VolumeSize: "20"
    - DeviceName: "/dev/sdk"
      NoDevice: {}
```

Fonte: https://docs.aws.amazon.com/pt_br/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html.

Para executar um script de CloudFormation na AWS você pode escrever o código e fazer upload para o S3 e executar utilizando o serviço de CloudFormation através do console de gerenciamento da AWS.

Você pode também ir diretamente no serviço de CloudFormation através de console de gerenciamento da AWS e fazer upload do código escrito diretamente por lá.

Terraform – Uma solução de IaC agnóstica

O terraform é uma ferramenta de provisionamento de recursos através de código, mantido pela Hashicorp, que é uma empresa de bastante destaque e relevância no meio DevOps. Ao contrário do CloudFormation, essa ferramenta diversos provedores de serviços de computação tais como AWS, OpenStack, Azure, GCP etc.

Uma das características mais relevantes do terraform é a sua impotência, termo muito utilizado na matemática ou em ciência da computação para indicar a propriedade que algumas operações têm de poderem ser aplicadas várias vezes sem que o valor do resultado se altere após a aplicação inicial. Ou seja, uma vez aplicado o seu código terraform, você poderá aplicá-lo quantas vezes desejar e nenhuma alteração será feita em sua infraestrutura, a menos que você tenha de fato alterado algo em seu código.

O terraform utiliza uma linguagem própria de alto nível e fácil de se reutilizar. Para executar um código de terrafom, você precisa ter um ambiente para instalá-lo. Os procedimentos para instalação variam de acordo com o SO onde esse código será executado, mas de maneira geral, a instalação é fácil de ser feita e você pode encontrar diversas documentações para auxiliá-lo nesse sentido, inclusive documentações oficiais da própria hashicorp.

- ✔ Baixar e Instalar o terraform: https://learn.hashicorp.com/tutorials/terraform/install-cli

Permissões para execução do Terraform

No ambiente onde você instalou o terraform e a partir do qual ele será executado, você precisa ter também as credenciais da sua conta da AWS de destino configuradas. Chamamos de conta de destino a conta onde a infraestrutura escrita nos códigos do terraform será provisionada. É a conta da AWS mostrada na imagem que vimos no começo desse tópico

sobre IaC. Conforme vimos no primeiro capítulo desse modulo, você precisa ter o AWS CLI instalado e configurado com credenciais programáticas do IAM com privilégios suficientes para o terraform ser executado e provisionar a infraestrutura necessária.

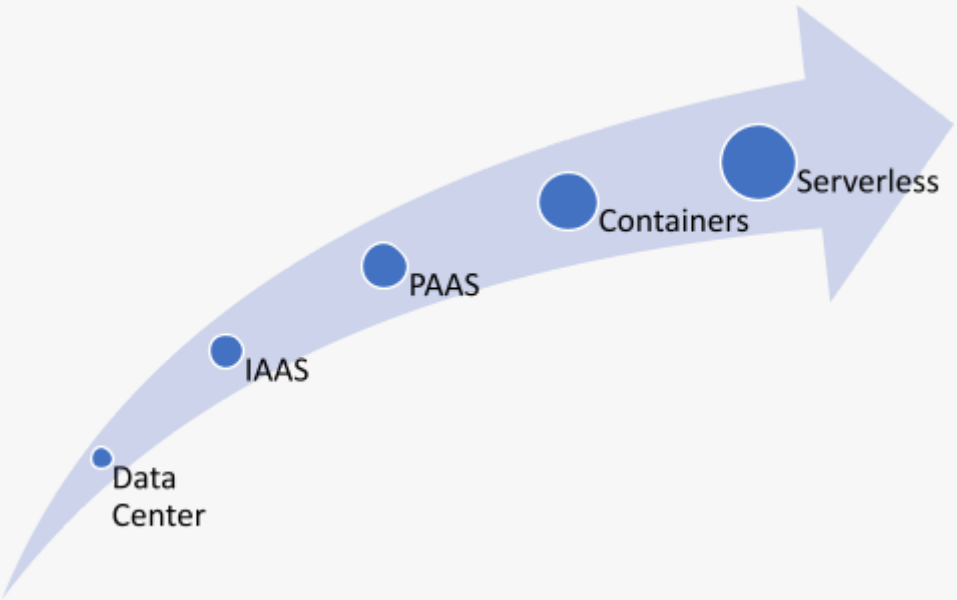
> Capítulo 3

Capítulo 3.AWS Lambda e outras soluções serverless

Introdução ao conceito de serverless e suas vantagens

O que é serverless?

O conceito de serverless está fortemente ligado ao surgimento da computação em nuvem e sem ela o conceito de serverless não existiria. Alguns serviços básicos de computação em nuvem, como o EC2 e o RDS, já permitem com que os administradores de infraestrutura tenham bem menos responsabilidades a respeito daquela infraestrutura provisionada em comparação ao que teriam se provisionassem essa infraestrutura em seus tradicionais datacenters físicos. Esses administradores por exemplo não precisam se preocupar com energia, refrigeração, conexões de internet e segurança física de onde as suas instâncias EC2 estão sendo executadas. Essa responsabilidade é transferida 100% para o provedor de nuvem, que para o nosso caso, é a AWS.



Ainda assim, o administrador de infra precisa se preocupar com a segurança do sistema operacional da instância que provisionou, precisa se

preocupar com atualizações desse sistema operacional, com as portas de acesso que estão liberadas nessa instância, com a disponibilidade desse servidor na AWS e com as limitações de hardware que possam existir ao utilizar instâncias para suportar determinada aplicação, por exemplo.

Os serviços que são de caráter serverless eliminam até mesmo a necessidade de se preocupar com coisas desse tipo. Tais serviços abstraem das mãos dos usuários de nuvem qualquer necessidade de preocupação com servidores.

Com a computação sem servidor, as tarefas de gerenciamento de infraestrutura, como provisionamento de capacidade e correção, são gerenciadas pela AWS, para que você possa se concentrar apenas em seu negócio. Os serviços sem servidor, como o AWS Lambda, vêm com dimensionamento automático, alta disponibilidade integrada e um modelo de cobrança de valor pago.

Então podemos definir os principais benefícios do serverless:

- Escalabilidade – Os recursos escalam de acordo com a demanda sem necessidade de provisionamento direto de recursos. A AWS gerencia isso automaticamente;
- Disponibilidade – Os recursos ficam disponíveis durante todo o tempo;
- Diminuição dos riscos de ameaças diretas a infraestrutura;
- Vantagem competitiva significativa;
- Custo operacional reduzido significativamente;
- Rápida inovação.

AWS Lambda – Criando sua primeira função na AWS

O AWS Lambda é um desses serviços de caráter serverless da AWS, que permite você a executar códigos em resposta a eventos. Esses eventos podem ser desde agendamentos que você faz para que uma função Lambda seja executada periodicamente, até eventos que se originam de mudanças no seu ambiente da AWS.

Por exemplo, pode ser que você deseje executar uma função lambda que verifique os arquivos que chegam até um bucket do S3 na sua conta da AWS. Cada vez que chegar um arquivo, uma função Lambda é acionada e ela faz certo tipo de validação e conferência no arquivo que chegou até o seu bucket. Se encontrar algum problema naquele arquivo, outra função Lambda pode ser acionada para deletar imediatamente aquele arquivo do bucket.

Você apenas escreve o código e o submete para o serviço do Lambda, e a AWS cuida do ambiente onde esse código irá rodar. A sua preocupação é apenas no seu código, no seu negócio.

Linguagens de programação nativamente suportadas:

- Node.js
- Python;
- Java;
- Go;
- PowerShell;
- C#;
- Ruby.

Além dessas linguagens nativamente suportadas, o Lambda oferece uma API em tempo de execução que permite com que você utilize qualquer linguagem de programação da sua escolha. Então, podemos dizer que indiretamente, o Lambda fornece suporte a todas as linguagens de programação.

Custo do AWS Lambda

Criar uma função lambda em si não gera custos na sua conta da AWS. Somente quando a função é executada que haverá custo sob o tempo de duração de execução daquela instância. A quantidade de memória utilizada para a execução daquela função também pode influenciar o seu custo de execução.

Um outro fator que influencia o custo de uma função Lambda é a quantidade de requisições para aquela determinada função. O primeiro milhão de requisições é gratuito e depois é cobrado $ 0.20 pelo próximo milhão de requisições.

Lambda em Ação

Toda vez que você está em contato com um dispositivo Echo da AWS, conversando com a Alexa, na verdade, você está interagindo com funções Lambda

Funções Lambda são utilizadas por diversos tipos de dispositivos de IA
Fonte: Imagem retirada da Internet.

> Capítulo 4

Capítulo 4.AWS API Gateway e Soluções AWS para IoT

Introdução ao API Gateway

- https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml

O que é uma API?

API significa, em inglês, *Application Programming Interface*, ou interface de programação de aplicação. Uma API é formada por um conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos.

Por meio de APIs, desenvolvedores podem criar softwares e aplicativos capazes de se comunicar com outras plataformas. Por exemplo: caso um desenvolvedor queira criar um aplicativo de fotos para Android, ele poderá ter acesso à câmera do celular através da API do sistema operacional, sem ter a necessidade de criar uma interface de câmera do zero.

O mesmo acontece com aplicativos que utilizam os serviços de mapas por meio da API do Google Maps ou, ainda, nas integrações entre apps, como o Spotify e o Instagram, que possibilita compartilhar faixas nos Stories.

O serviço de API da AWS

O Amazon API Gateway é um serviço da AWS para criação, publicação, manutenção, monitoramento e proteção de APIs REST e WebSocket em qualquer escala. Os desenvolvedores de API podem criar APIs que acessem a AWS ou outros web services, bem como dados armazenados na Nuvem AWS. Como um desenvolvedor de APIs do API Gateway, é possível criar APIs para uso em suas próprias aplicações cliente.

Ou você pode disponibilizar suas APIs para desenvolvedores de aplicativos de terceiros.

O API Gateway cria APIs RESTful que:

- São baseadas em HTTP.
- Habilitam a comunicação cliente-servidor sem estado.
- Implementam os métodos HTTP padrão, como GET, POST, PUT, PATCH e DELETE.

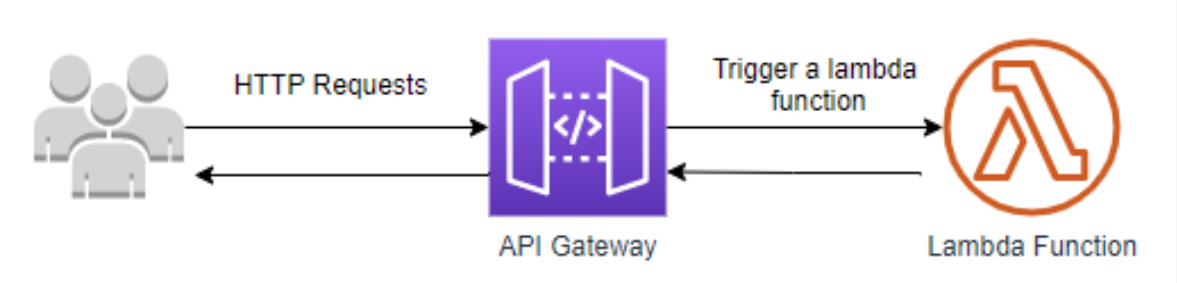
O API Gateway cria APIs WebSocket que:

- Seguem o protocolo WebSocket, que permite a comunicação full-duplex entre cliente e servidor com estado.
- Roteiam mensagens recebidas com base no conteúdo da mensagem.

Este é um serviço também de caráter serverless. Podemos construir funções Lambda que respondem a requisições HTTP, por exemplo, que chegam através do API Gateway. Dessa forma o API Gateway serve como uma interface para uma aplicação simples construída em Lambda. Inclusive, é isso que vamos ver na parte prática dessa seção.

Construindo uma API 100% Serverless

Figura 6 - Diagrama de arquitetura do ambiente simples que será construído.



Ambiente de API Gateway + Lambda.

O conceito de IoT

- https://maplink.global/blog/o-que-e-iot/

Para introduzirmos este assunto nada melhor do que te propor um desafio. Você consegue ficar um dia sem nenhum tipo de tecnologia? Será que consegue? Apostaria com você que não!

Lembre-se, não estamos falando apenas do seu smatphone ou computador. Você não poderia usar o caixa do mercado, o reconhecimento digital do estacionamento do condomínio, babá eletrônica, câmera de segurança, seu smartwatch e nem sua Alexa!

É... agora, vai concordar comigo que não vai dar, né?

O que queremos te mostrar é que o nosso cotidiano é coberto de equipamentos e objetos que dependem da internet e da tecnologia para desempenharem suas funções. Hoje, é praticamente impossível ter uma rotina sem envolver inúmeros aparelhos conectados à internet, seja no wi-fi ou 3G.

E essa conectividade é ainda mais forte e complexa dentro das empresas. Com a evolução da tecnologia e equipamentos cada vez mais modernos e conectados, surgiu o termo a Internet das Coisas, e reinventou muito do que nós já considerávamos tecnológico.

Agora que podemos conectar esses objetos do cotidiano – eletrodomésticos, carros, termostatos, babás eletrônicas – à internet por meio de dispositivos incorporados, é possível criar uma rede inteligente de comunicação e processamento dos dados que são gerados por esses dispositivos que chamamos de “coisas”.

E a computação em nuvem, certamente, foi uma evolução tecnológica fundamental para a adoção da IoT de maneira geral como está acontecendo agora e ainda se intensificará ao longo dos próximos anos.

Principais Tecnologias e evolução tecnológicas que tornaram a IoT possível:

- **Conectividade em larga escala** – Praticamente hoje todo dispositivo fabricado possui algum meio para se conectar à internet, que também está presente em cada canto do mundo;
- **Plataformas de computação em nuvem** – Os provedores de nuvem fazem com os recursos computacionais necessários para a coleta e o processamento dos dados que chegam dos dispositivos conectados fiquem disponíveis quase que o tempo todo e tenham resiliência e robustez para a sua operação;
- **Acesso à tecnologia de sensores de baixo custo e baixa potência** – Tornou-se mais barato a fabricação de sensores confiáveis no mercado;
- **Machine Learning e análise avançada** – A grande massa de dados que chega até os ambientes em nuvem é processada graças ao estágio de evolução em Machine Learning e análise avançada em que nós nos encontramos hoje;
- **Inteligência Artificial conversacional** – Os avanços das redes neurais trouxeram o processamento de linguagem natural (NLP) até muitos dos dispositivos IoT existentes hoje no mercado e os tornaram atraentes, acessíveis, viáveis e principalmente uteis para o nosso uso, doméstica e corporativamente dizendo.

IoT na AWS

A AWS prove serviços que permitem com que criemos um ambiente de recepção e processamento de dados provenientes de dispositivos IoT. Esses dados são capturados e podem ser armazenados e processados com o auxílio de diversos outros serviços da AWS.

Principais características do AWS IoT:

- É completamente gerenciado pela AWS;
- Os dispositivos conseguem interagir de maneira segura com aplicações construídas na AWS e também com outros dispositivos IoT;
- O AWS IoT consegue suportar bilhões de dispositivos e trilhões de mensagens provenientes desses dispositivos;
- Essas mensagens podem ser roteadas para outros serviços da AWS e para outros dispositivos.

IoT e o BigData

São termos que, na maioria dos casos, andam juntos. Isso acontece pelo fato de um ambiente de IoT gerar quantidades gigantescas de dados que precisam ser compreendidos. E é aí que entra o conceito de BigData.

O IoT consegue analisar dados em tempo real e fornecer recursos e funcionalidades para armazenar essa massa de dados. Tudo isso sem a necessidade de se preocupar com escalabilidade, disponibilidade e gerenciamento de infraestrutura.

O ecossistema do IoT é grande e pode ser integrado com diversos serviços da AWS, tais como:

- ElasticSearch;
- Kinesis;
- DynamoDB;
- Amazon SageMaker;
- S3;

- SQS;
- SNS;
- Lambda.

> Capítulo 5

Capítulo 5.Ferramentas de DevOps da AWS

Neste capítulo vamos falar sobre os serviços da AWS que contemplam a esteira de desenvolvimento de uma aplicação, desde a fase de controle de origem através de um repositório Git até a gestão completa de um pipeline de desenvolvimento. A AWS fornece serviços que suportam todo o fluxo de entrega de uma aplicação.

Para entendermos melhor sobre a utilização desses serviços, vamos compreender melhor sobre o conceito de CI/CD, que significa *Continuous Integrations* & Continuous Delivery/Deployment. Podemos entender o CI/CD como sendo um conjunto de boas práticas do desenvolvimento de software. Promove as pequenas mudanças e a automação sobre a entrega desse software para a ambiente onde será executado. Esse conceito e cultura tem sido implantado nos últimos anos em grandes empresas como AWS, Netflix, Google e Facebook, que inclusive serviram como pioneiras para esse tipo de cultura.

A automação, que é uma das principais características dessa abordagem de desenvolvimento de software, permite o deploy ágil das aplicações, elimina erros manuais que são bem comuns quando temos softwares sendo construídos manualmente. Basicamente, queremos que o nosso software seja compilado e implantado seguindo uma série de comandos pré-definidos que vão nos levar ao resultado correto e esperado.

As ferramentas básicas da AWS que permitem com que o fluxo de desenvolvimento que segue a esses determinados padrões seja implantado são:

- Code Commit
- Code Build

- Code Deploy
- Code Pipeline

Vamos falar sobre cada uma dessas ferramentas nesse capítulo. Divirtam-se!

Code Commit

O AWS CodeCommit é um serviço de controle de origem gerenciado seguro e altamente dimensionável que hospeda repositórios privados do Git. Com o CodeCommit, você não precisa gerenciar seu próprio sistema de controle de origem nem se preocupar com o dimensionamento da sua infraestrutura. O CodeCommit pode ser usado para armazenar qualquer item, desde código a binários. E por ser compatível com a funcionalidade padrão do Git, ele funciona perfeitamente com suas ferramentas existentes baseadas em Git.

Principais benefícios do CodeCommit:

- **Completamente gerenciado** - O AWS CodeCommit elimina a necessidade de hospedar, manter, fazer backup e escalar seus próprios servidores de controle de origem. O serviço ajusta automaticamente a escalabilidade para atender às necessidades crescentes de seu projeto.
- **Segurança** - O AWS CodeCommit criptografa automaticamente seus arquivos em trânsito e em repouso. O CodeCommit é integrado ao AWS Identity and Access Management (IAM), permitindo que você personalize o acesso de usuários específicos aos seus repositórios.
- **Alta disponibilidade** - O AWS CodeCommit tem uma arquitetura altamente escalável, redundante e durável. O

serviço foi projetado para manter seus repositórios altamente disponíveis e acessíveis.

- **Colaboração:** O AWS CodeCommit ajuda você a promover a colaboração nos códigos com seus colegas por meio de solicitações pull, ramificação e mesclagem. Você pode implantar fluxos de trabalho que incluem avaliações e comentários de códigos por padrão, e controlar quem tem permissão para fazer alterações em ramificações específicas.

Em resumo, o CodeCommit é o serviço de armazenamento e controle de códigos da AWS, similar ao GitHub, BitBucket, GitLab etc.

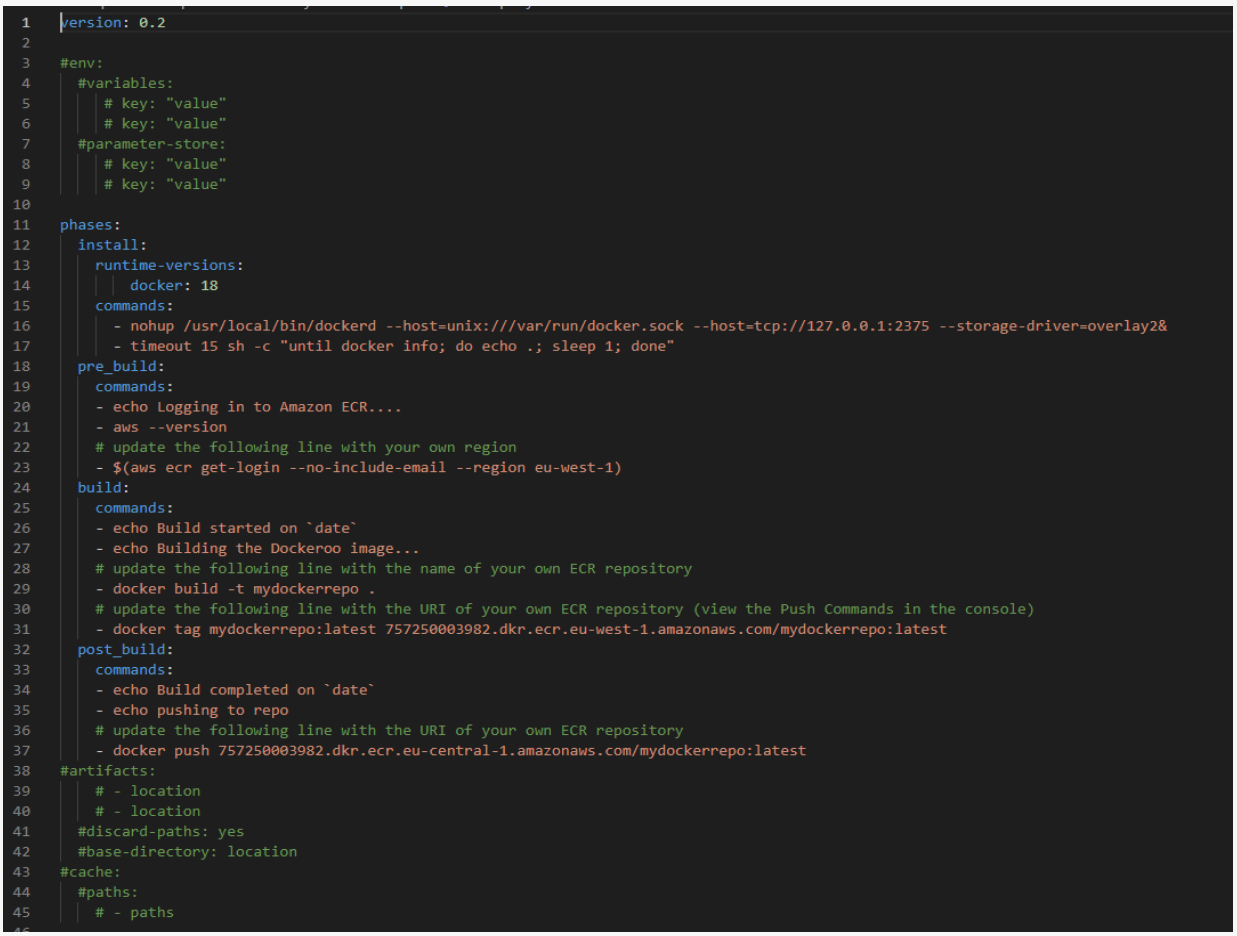
Code Build

O AWS CodeBuild é um serviço de integração contínua totalmente gerenciado que compila o código-fonte, realiza testes e produz pacotes de software prontos para implantação. O CodeBuild escala continuamente e processa múltiplas compilações ao mesmo tempo, o que evita que elas fiquem esperando em uma fila. Você pode começar a usar rapidamente com ambientes de compilação pré-definidos ou criar ambientes de compilação personalizados com suas próprias ferramentas de compilação. Com o CodeBuild, os recursos computacionais usados são cobrados por minuto.

O CodeBuild exige um arquivo de configuração para definição dos comandos de build, o buildspec.yml.Esse é o arquivo de referência dos comandos que serão executados durante a compilação do código. Normalmente esse arquivo fica na raiz do repositório da aplicação que está sendo compilada, mas os comandos podem ser informados manualmente nas configurações do projeto de compilação.

Abaixo está um exemplo de um arquivo buildspec.yml.

Figura 7 – Exemplo de um arquivo buildspec.yml.



```
version: 0.2

#env:
  #variables:
    # key: "value"
    # key: "value"
  #parameter-store:
    # key: "value"
    # key: "value"

phases:
  install:
    runtime-versions:
      docker: 18
    commands:
      - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --host=tcp://127.0.0.1:2375 --storage-driver=overlay2&
      - timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
  pre_build:
    commands:
    - echo Logging in to Amazon ECR....
    - aws --version
    # update the following line with your own region
    - $(aws ecr get-login --no-include-email --region eu-west-1)
  build:
    commands:
    - echo Build started on `date`
    - echo Building the Dockeroo image...
    # update the following line with the name of your own ECR repository
    - docker build -t mydockerrepo .
    # update the following line with the URI of your own ECR repository (view the Push Commands in the console)
    - docker tag mydockerrepo:latest 757250003982.dkr.ecr.eu-west-1.amazonaws.com/mydockerrepo:latest
  post_build:
    commands:
    - echo Build completed on `date`
    - echo pushing to repo
    # update the following line with the URI of your own ECR repository
    - docker push 757250003982.dkr.ecr.eu-central-1.amazonaws.com/mydockerrepo:latest
#artifacts:
    # - location
    # - location
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

Code Build e o Docker – Um exemplo prático

O CodeBuild pode se tornar um grande aliado no processo de construção/compilação de imagens Docker antes delas se tornarem containers, de fato, e irem para o ambiente de execução através do Deploy.

O CodeBuild, por ser uma ferramenta de compilação, consegue executar todos os comandos de build de uma imagem do Docker e salvar essa imagem em um registry de imagens Docker como é o ECR, da própria AWS. Todos os comandos de build e tag de imagens do Docker que são feitos manualmente, podem ser automatizados utilizando o CodeBuild através do arquivo buildspec.yml

Nas aulas práticas, veremos exemplos disso!

Code Deploy

O AWS CodeDeploy é um serviço totalmente gerenciado de implantação que automatiza implantações de software em diversos serviços de computação como Amazon EC2, AWS Fargate, AWS Lambda e servidores locais.

O AWS CodeDeploy facilita o lançamento rápido de novos recursos, ajuda a evitar tempo de inatividade durante a implantação de aplicativos e lida com a complexidade de atualizá-los. Você pode usar o AWS CodeDeploy para automatizar implantações de software e eliminar a necessidade de operações manuais propensas a erros. O serviço é dimensionado para estar de acordo com as suas necessidades de implantação.

Os métodos de Deploy:

- In-Place (Rolling Update) – A aplicação é interrompida em cada instância e a release é instalada.
- Blue/Green – Novas instâncias são provisionadas e a release é instalada nessas novas instâncias. O Blue representa o deployment ativo, enquanto o green representa a nova versão da aplicação.

Características e vantagens de cada método:

In-Place Deployment	Blue/Green
A capacidade computacional total é reduzida durante o deployment.	Sem redução de capacidade computacional total.
Não é permitido para Lambda.	O ambiente “green” pode ser criado previamente.
O Roll-back envolve fazer o Deploy novamente.	É fácil fazer o “switch” entre os ambientes “green” e “blue”.

É uma ótima solução para um primeiro Deploy.	Você paga pelos dois ambientes até que você termine o ambiente antigo.

O arquivo AppSpec

O arquivo para definir as instruções do deployment é chamado de appspec. Este arquivo é um arquivo de configuração de Deploy e nele estarão os comandos e parâmetros a serem executados durante o deployment. Para deployments em isntâncias EC2 e/ou ambientes on-premises, esse arquivo pode ser escrito somente usando o YAML. Para deploy de funções Lambda, esse arquivo pode ser escrito ou em YAML ou em JSON.

Exemplo de um arquivo appspec para deploy em EC2/on-premise.

Figura 8 – Exemplo de um arquivo appspec.yml.

```yaml
version: 0.0
os: linux
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

O arquivo appspec deve estar no diretório raiz do projeto em questão, caso contrário o Deploy falhará.

Code Pipeline

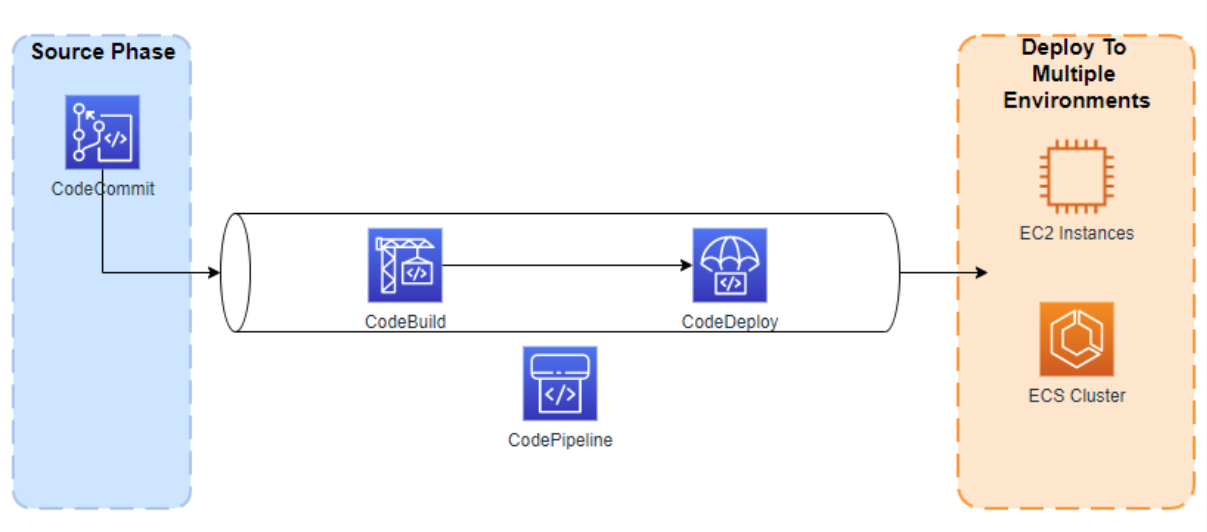
O AWS CodePipeline é um serviço gerenciado de entrega contínua que ajuda a automatizar pipelines de liberação para oferecer atualizações rápidas e confiáveis de aplicativos e infraestruturas. O CodePipeline automatiza as fases de compilação, teste e implantação do processo de liberação sempre que ocorre uma mudança no código, de acordo com o modelo de liberação que você definiu.

Você pode integrar facilmente o AWS CodePipeline com serviços de terceiros como GitHub ou com o seu próprio plug-in personalizado. Com o AWS CodePipeline, você paga somente pelo que utiliza. Não há custos iniciais e não há compromissos de longo prazo.

O CodePipeline será o grande maestro do nosso processo de entrega de software. Ele é capaz de reconhecer uma mudança no código, armazenado no CodeCommit, por exemplo, e acionar um projeto de compilação do CodeBuild que irá compilar a aplicação e enviá-la para o seu ambiente produtivo final.

O CodeDeploy pode ser acionado também para cumprir o seu papel de fazer o Deploy de uma aplicação em um determinado ambiente. Dessa forma, teremos o processo de condução do software, desde o local onde é armazenado, o repositório Git desse código, até o local onde será consumido pelos usuários, como um ambiente em ECS, EKS ou utilizando instâncias EC2, por exemplo.

Figura 9 – Pipeline automatizado de DevOps com o Code Pipeline fazendo o papel de maestro.



> Capítulo 6

Capítulo 6.AWS AppSync

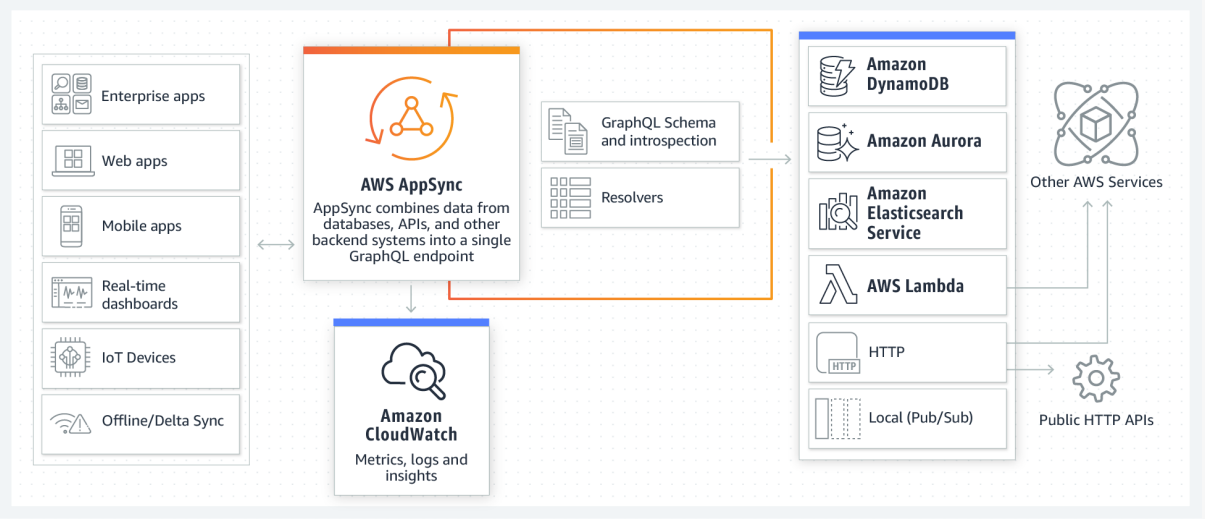
Introdução ao AppSync

O AppSync é um serviço gerenciado da AWS que facilita o desenvolvimento de APIs GraphQL ao lidar com o trabalho pesado de se conectar com segurança a fontes de dados como o AWS DynamoDB, Lambda e muito mais. Depois de implantado, o AWS AppSync escala automaticamente seu mecanismo de execução da API GraphQL para cima e para baixo para atender aos volumes de solicitação da API.

Principais benefícios:

- Conecte-se a diversas origens de dados utilizando um recurso de solicitação de rede apenas;
- Proteção dos dados utilizando vários modos de autenticação simultâneos;
- Faz cache de dados;
- Atualizações de dados em tempo real por websockets para milhões de clientes;
- Para aplicações Web e Mobile, o AppSync ainda fornece acesso aos dados locais quando esses dispositivos ficam offline, sincronização dos dados e resolução de conflitos quando os dispositivos estão online novamente;
- Serverless.

Figura 10 – Arquitetura padrão de um ambiente em GraphQL.



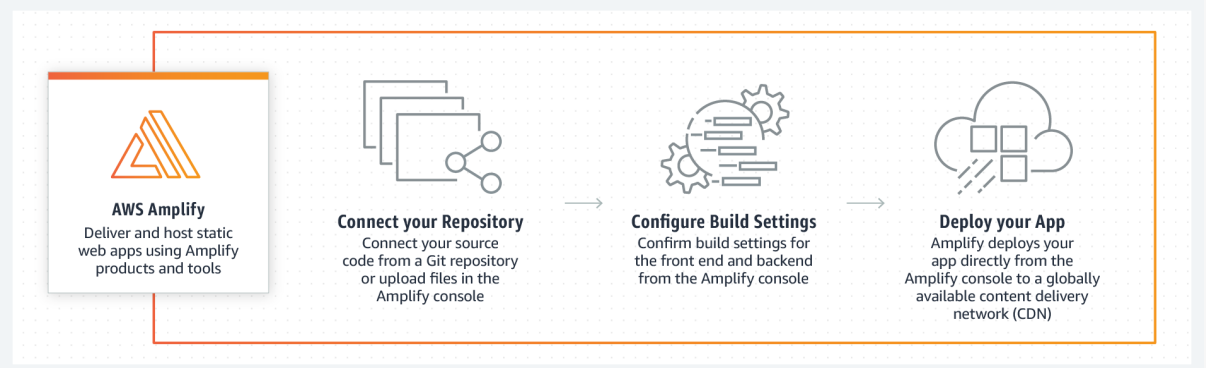
> Capítulo 7

Capítulo 7.AWS Amplify

Introdução ao AWS Amplify

O AWS Amplify é um seviço completamente gerenciado da AWS que permite a implantação descomplicada de aplicações web estáticas. O Amplify utiliza a rede de distribuição de conteúdo da AWS com centenas de pontos de presença em todo o mundo para entregar esse conteúdo estático com uma experiência melhorada para os usuários.

No que diz respeito a automatização o Amplify gerencia os fluxos de trabalho de CI/CD incorporados que aceleram o ciclo de lançamento de sua aplicação. Basta conectar o repositório de código da aplicação no console do Amplify e as alterações no front-end e no back-end serão implantadas em um único fluxo de trabalho em cada confirmação de código.



> Capítulo 8

Capítulo 8.AWS Device Farm

Introdução ao DeviceFarm

O AWS Device Farm é um serviço de teste de aplicações que permite melhorar a qualidade de suas aplicações móveis e da Web, testando-os em uma ampla variedade de navegadores de desktop e dispositivos móveis reais baseados em Android e iOS, sem precisar provisionar e gerenciar qualquer infraestrutura de teste.

O serviço permite que você execute seus testes simultaneamente em vários navegadores de desktop ou dispositivos reais para acelerar a execução do seu conjunto de testes e gera vídeos e logs para ajudá-lo a identificar rapidamente problemas com seu aplicativo.

De maneira resumida, o DeviceFarm permite com que você teste o seu aplicativo mobile que está sendo desenvolvido utilizando dispositivos moveis virtuais dentro da nuvem da AWS. Você consegue interagir com esse dispositivo móvel remotamente e gesticular, deslizar tela e interagir normalmente em tempo real, diretamente do seu navegador da web.

Referências

AMARAL, Armando. O que é infraestrutura como Código? **Tivit Blog**, São Paulo, 20 set. 2018. Disponível em: https://blog.tivit.com/o-que-e-infraestrutura-como-codigo. Acesso em: 17 mar. 2022.

APLICATIVO móvel do AWS Console. [*S.l.*], 2021. AWS. Disponível em: https://aws.amazon.com/pt/console/mobile/. Acesso em: 17 mar. 2022.

AWS CLI Command Reference. [*S.l.*], 2021. AWS CLI Documment. Disponível em: https://docs.aws.amazon.com/cli/latest/index.html. Acesso em: 17 mar. 2022.

FABRO, Clara. O que é API e para que serve? Cinco perguntas e respostas. **TechTudo**, Rio de Janeiro, 15 jun. 2020. Disponível em: https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml. Acesso em: 17 mar. 2022.

INSTALL Terraform. [*S.l.*], [202-?]. HashiCorp Learn. Disponível em: https://learn.hashicorp.com/tutorials/terraform/install-cli. Acesso em: 17 mar. 2022.

O que é o AWS Device Farm? [S.l.], 2021. AWS Documentação. Disponível em: https://docs.aws.amazon.com/pt_br/devicefarm/latest/developerguide/welcome.html. Acesso em: 17 mar. 2022.

O que é o IAM? [*S.l.*], 2021. AWS Documentação. Disponível em: https://docs.aws.amazon.com/pt_br/IAM/latest/UserGuide/console.html. Acesso em: 17 mar. 2022.