Ambientes de Desenvolvimento

Aula 05 – Computação Serverless: Lambda

Rodolfo Riyoei Goya
rodolfo.goya@faculdadeimpacta.com.br

Sumário

- Processamento Serverless
- AWS Lambda

Módulo 13: Criar microsserviços e arquiteturas sem servidor

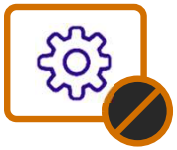
Seção 4: Introdução a arquiteturas sem servidor

aws academy

O que significa sem servidor?

Uma maneira de criar e executar aplicações e serviços sem se preocupar com servidores

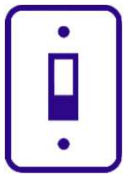
Princípios de arquiteturas sem servidor



Sem provisionamento de infraestrutura, sem gerenciamento

Escalabilidade automática

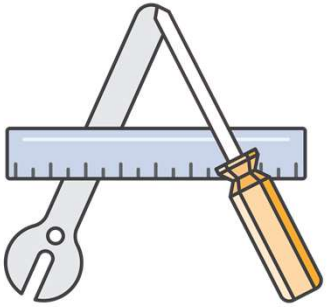
Pagamento pelo valor



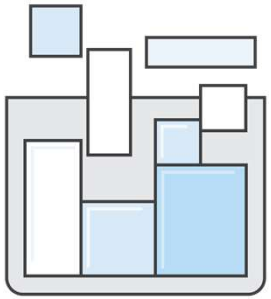
Altamente disponível e protegido

Benefícios de ser sem servidor

Custo total de propriedade menor



Concentre-se na sua aplicação, não na configuração



Criar aplicações de microsserviços

Ofertas sem servidor da AWS

aws academy

Computação

AWS Lambda e Lambda@Edge

AWS Fargate

Armazenamento

Amazon S3

Amazon EFS

Armazenamentos de dados

Amazon DynamoDB

Amazon Aurora

Amazon RDS Proxy

Proxy de API

Amazon API Gateway

Integração de aplicações

Amazon SNS

AWS AppSync

Amazon SQS

Amazon EventBridge

Orquestração

AWS Step Functions

Análise de dados

Amazon Kinesis

Amazon Athena

Principais lições da Seção 4

- A computação sem servidor permite criar e executar aplicações e serviços sem provisionar ou gerenciar servidores
- As arquiteturas sem servidor oferecem os seguintes benefícios:
  - Custo total de propriedade (TCO) menor
  - Você pode se concentrar em sua aplicação
  - Você pode usá-las para criar aplicações de microsserviços

Módulo 13: Criar microsserviços e arquiteturas sem servidor

Seção 5: Criar arquiteturas sem servidor com o AWS Lambda

aws academy

AWS Lambda

AWS
Lambda

- É um serviço de computação totalmente gerenciado
- Executa seu código em uma programação ou em resposta a eventos (por exemplo, alterações em um bucket do Amazon S3 ou em uma tabela do Amazon DynamoDB)
- Compatível com API de tempo de execução, Java, Go, PowerShell, Node.js, C#, Python e Ruby
- Pode ser executado em pontos de presença mais próximos de seus usuários

Como o AWS Lambda funciona

Origem do evento

Publicar

Resposta
(opcional)

AWS Lambda

Chamar

Serviços

Executar

Função do Lambda
(código personalizado)

Alterações no estado dos dados

Solicitações de endpoints

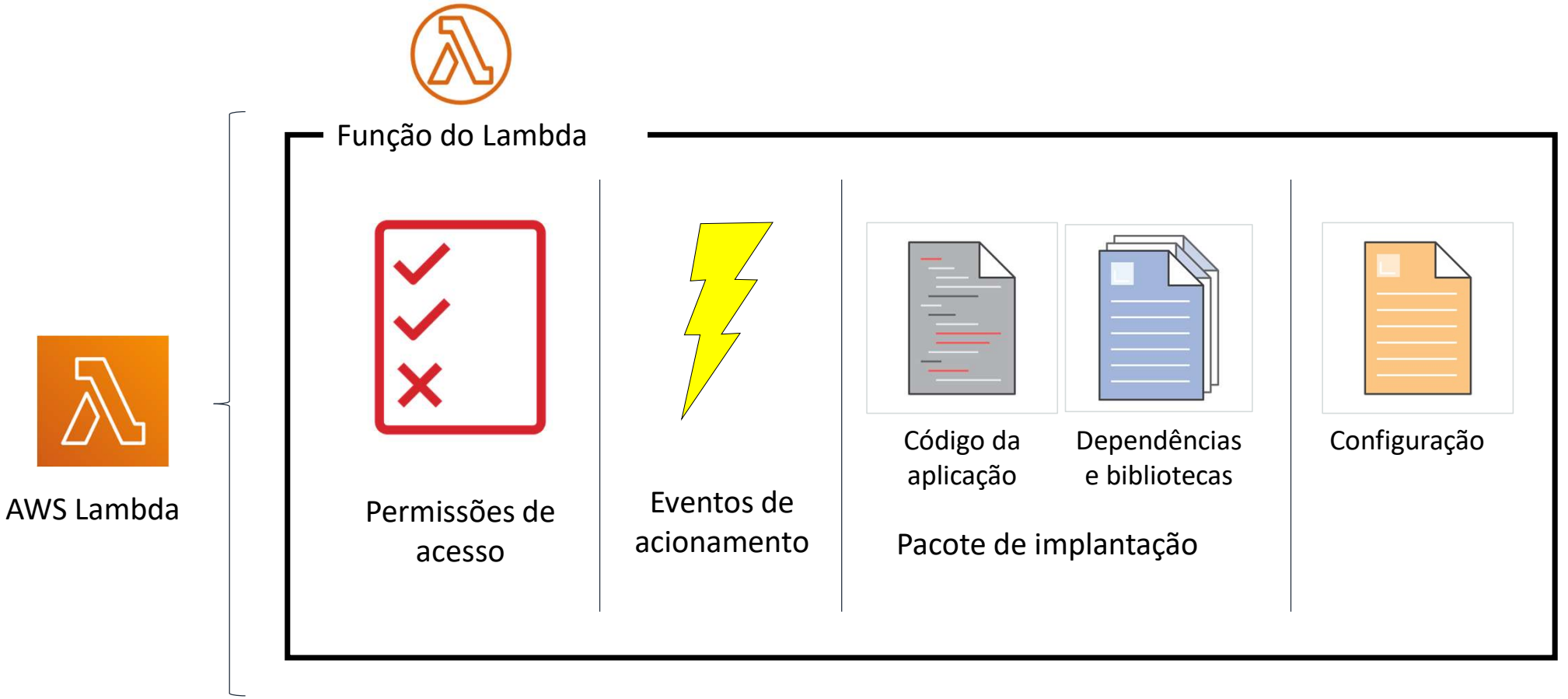
Alterações no estado do recurso

Eventos programados

Idiomas aceitos:

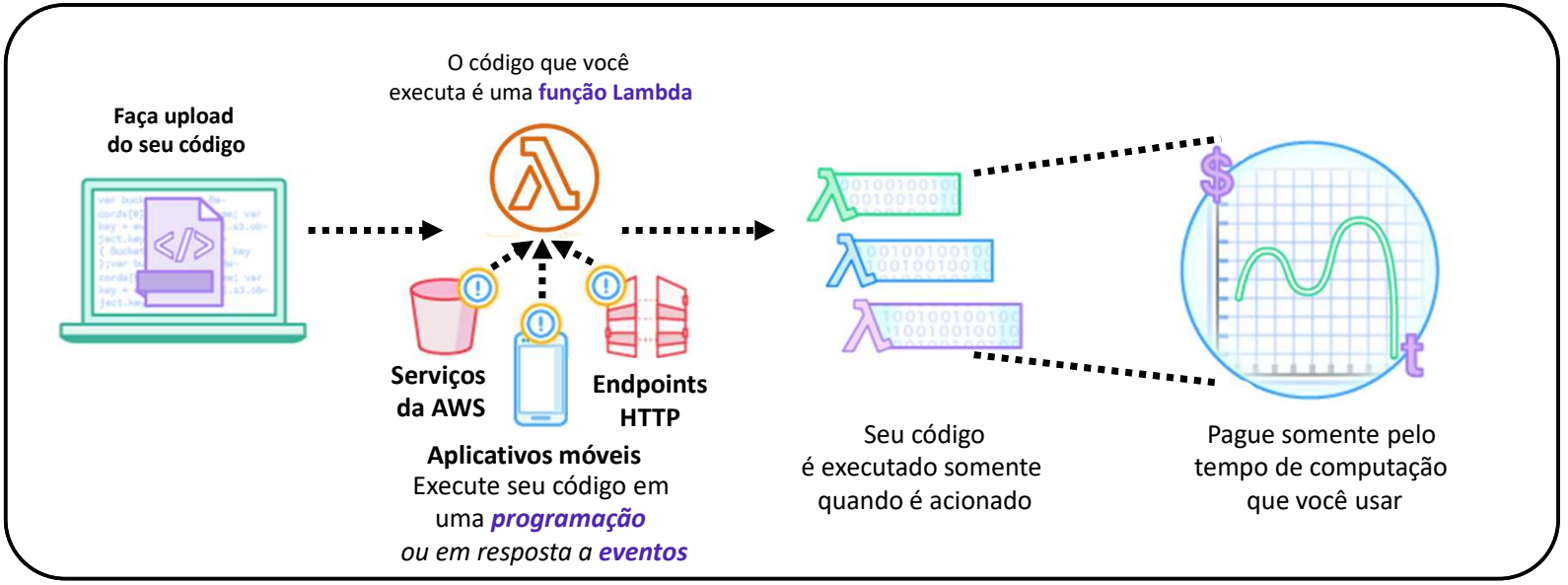
- Node.js
- Python
- Java
- C#
- Go
- Ruby
- PowerShell
- API de tempo de execução

Funções do Lambda



AWS Lambda: execute código sem servidores

O AWS Lambda é um serviço de computação **sem servidor** .



Anatomia de uma função do Lambda

Handler()

Função a ser executada após a invocação

event

Objeto com dados enviados durante a invocação da função do Lambda

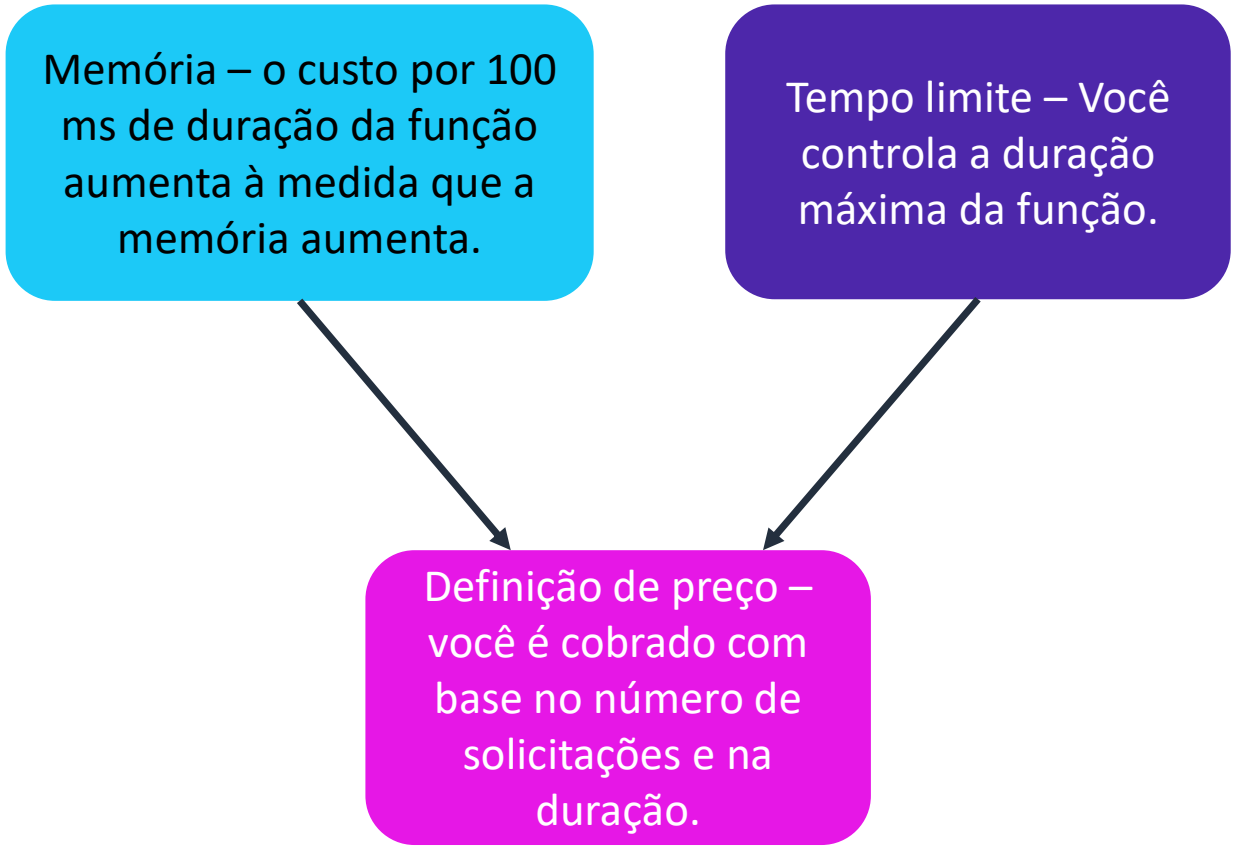
context

Objeto com métodos disponíveis para interagir com informações de tempo de execução (ID da solicitação, grupo de logs, mais)

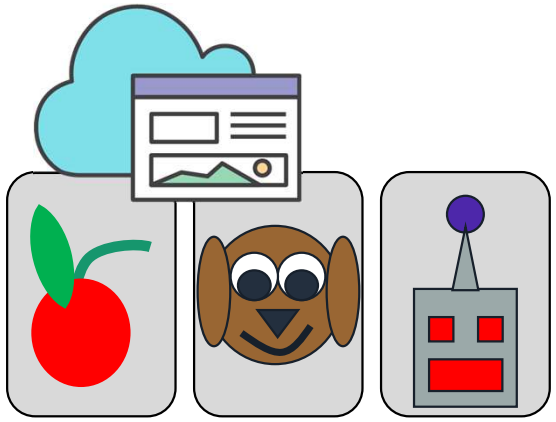
```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello World')
    }
```

Configuração e faturamento da função do Lambda



Exemplo do AWS Lambda: Jogo de navegador de máquina caça-níquel simulado



Script do navegador da Web

Chama métodos de serviço

Retorna resposta

AWS SDK para JavaScript

Invoca função

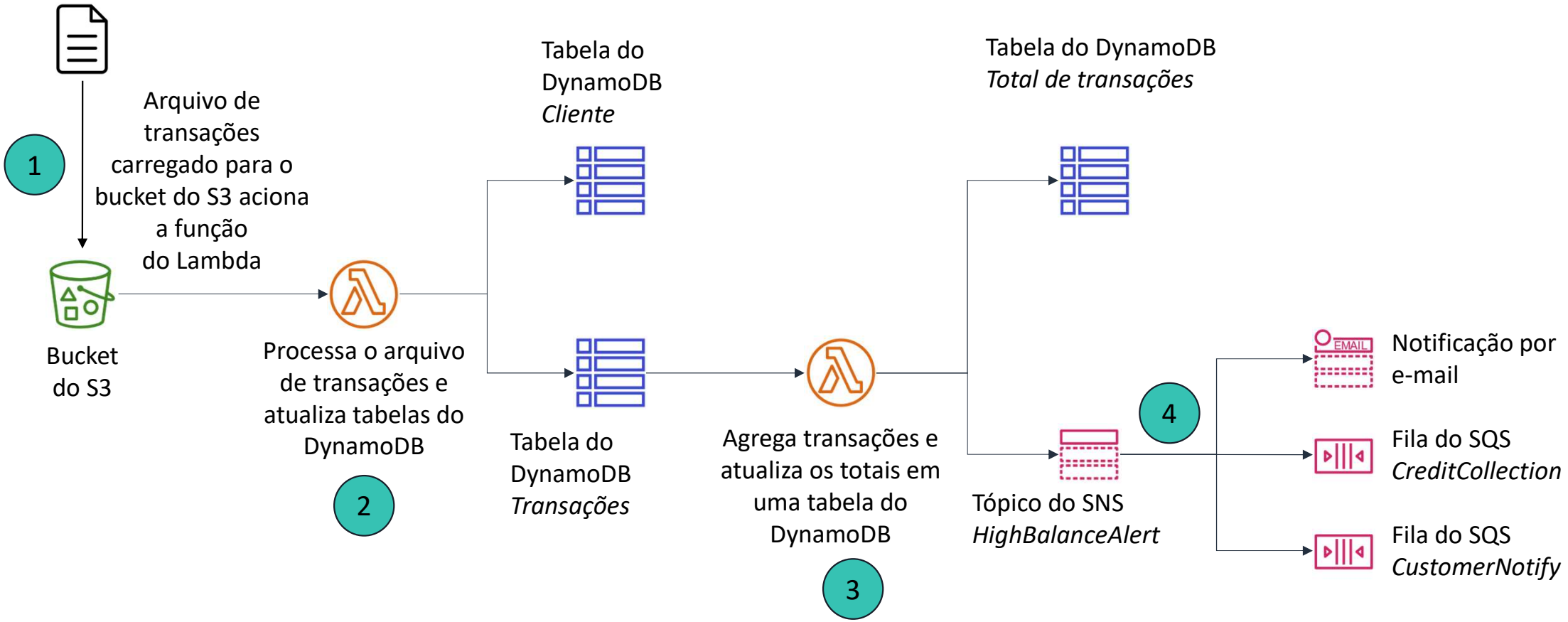
Retorna o resultado

AWS Lambda

```
lambda.invoke(pullParams, function(error, data) {
 if (error) {
 prompt(error);
 } else {
 pullResults = JSON.parse(data.Payload);
 }
});
```

```
{
 isWinner: false,
 leftWheelImage : {S : 'cherry.png'},
 midWheelImage : {S : 'puppy.png'},
 rightWheelImage : {S : 'robot.png'}
}
```

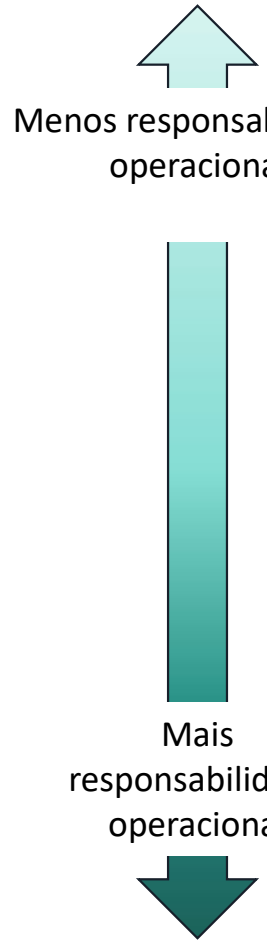
Exemplo de função do Lambda com base em eventos: processamento de pedidos



Camadas do Lambda - Layers

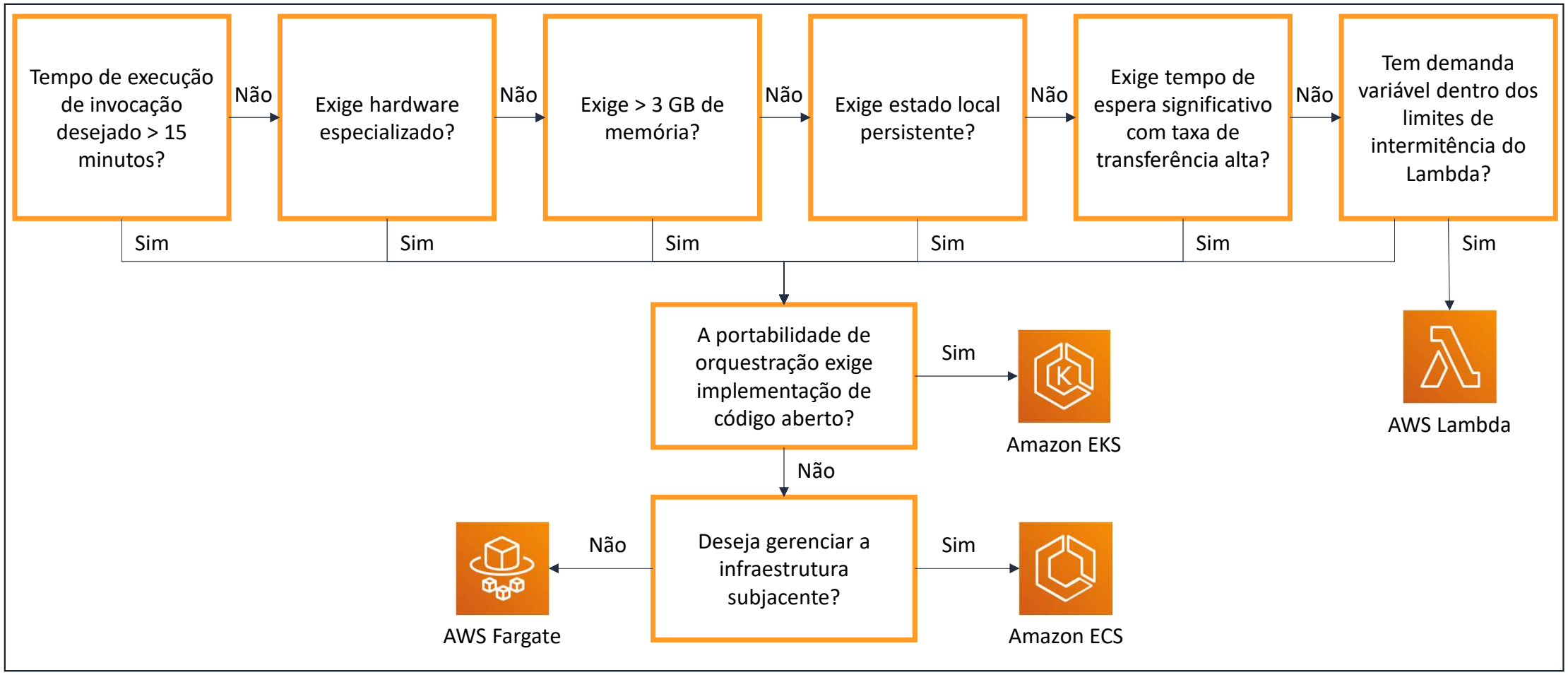
- Ativar funções para compartilhar código facilmente – você pode fazer upload de uma camada uma vez e referenciá-la em qualquer função
- Promover a separação de responsabilidades – os desenvolvedores podem iterar mais rapidamente na gravação da lógica de negócios
- Permitir que você mantenha seus pacotes de implantação pequenos
- Limites:
  - Até cinco camadas
  - 250 MB

Comparação da responsabilidade operacional para arquiteturas de contêineres e sem servidor



	A AWS gerencia	O cliente gerencia
Funções sem servidor do AWS Lambda	• Integrações de fontes de dados • Hardware físico, software, rede e instalações • Provisionamento	• Código da aplicação
Contêineres sem servidor do AWS Fargate	• Orquestração e provisionamento de contêineres • Dimensionamento de clusters • Hardware físico, kernel/sistema operacional do host, rede e instalações	• Código da aplicação • Integrações de fontes de dados • Configuração e atualizações de segurança, configuração de rede e tarefas de gerenciamento
Gerenciamento de contêineres como serviço do Amazon ECS e do Amazon EKS	• Plano de controle de orquestração de contêineres • Software de hardware físico, rede e instalações	• Código da aplicação • Integrações de fontes de dados • Clusters de trabalho • Configuração e atualizações de segurança, configuração de rede, firewall e tarefas de gerenciamento

Escolher uma plataforma de computação: contêineres x AWS Lambda

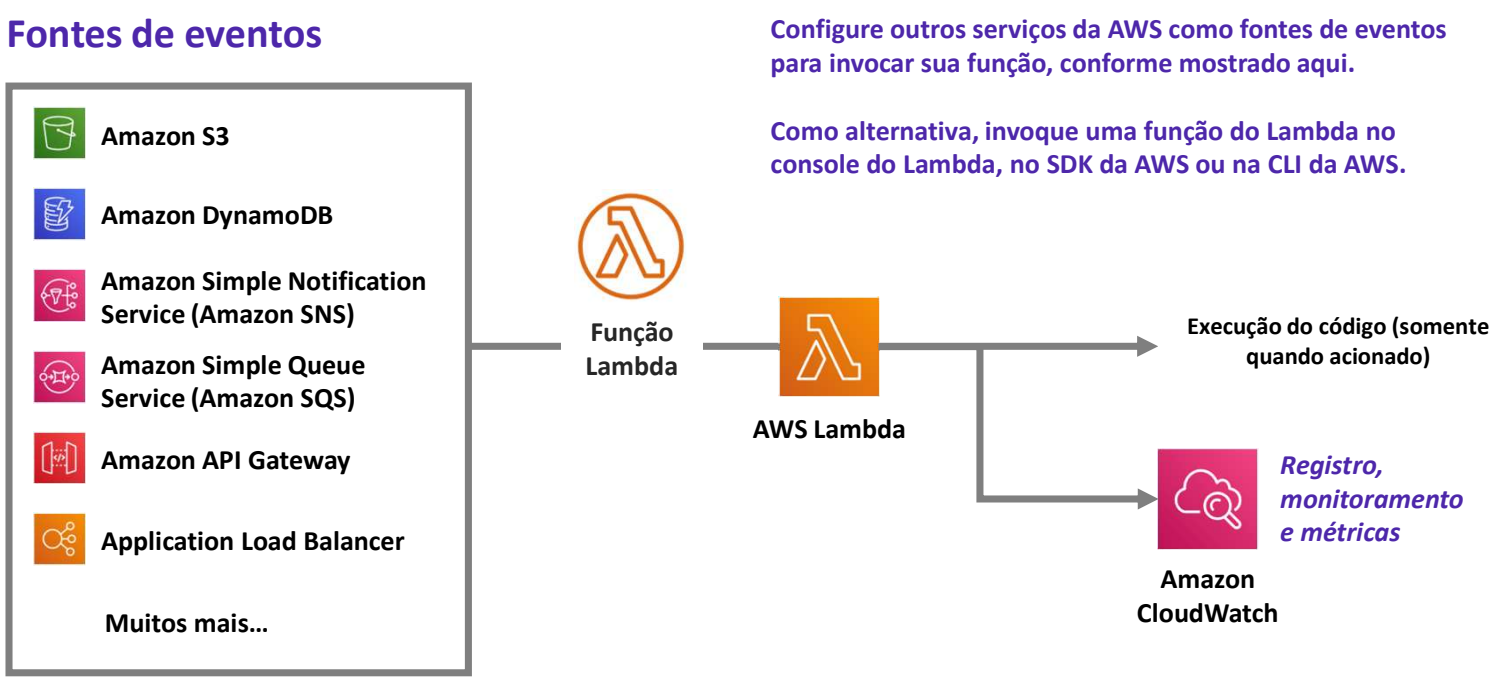


Benefícios do Lambda

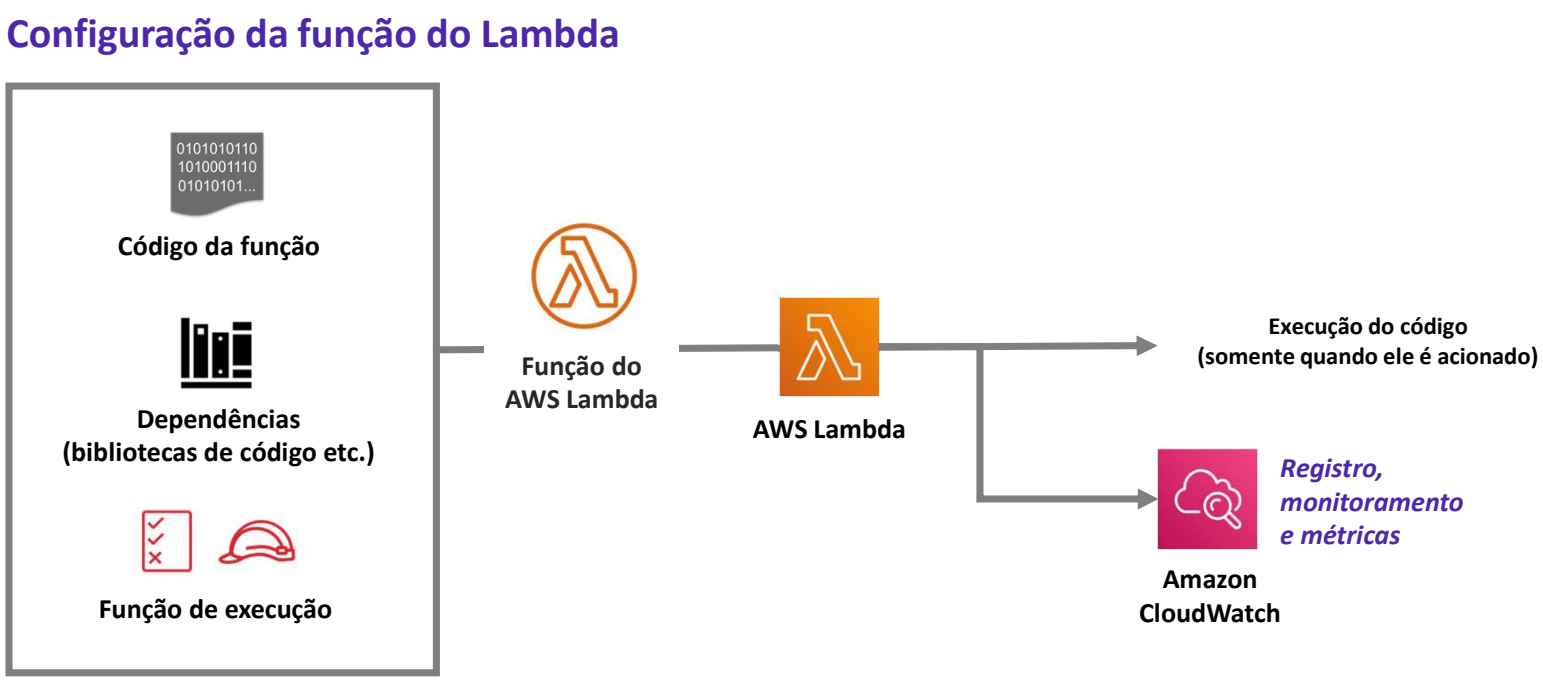
AWS Lambda

- Oferece suporte a várias linguagens de programação
- Administração totalmente automatizada
- Tolerância a falhas integrada
- Oferece suporte à orquestração de várias funções
- Pay-per-use a definição de preço

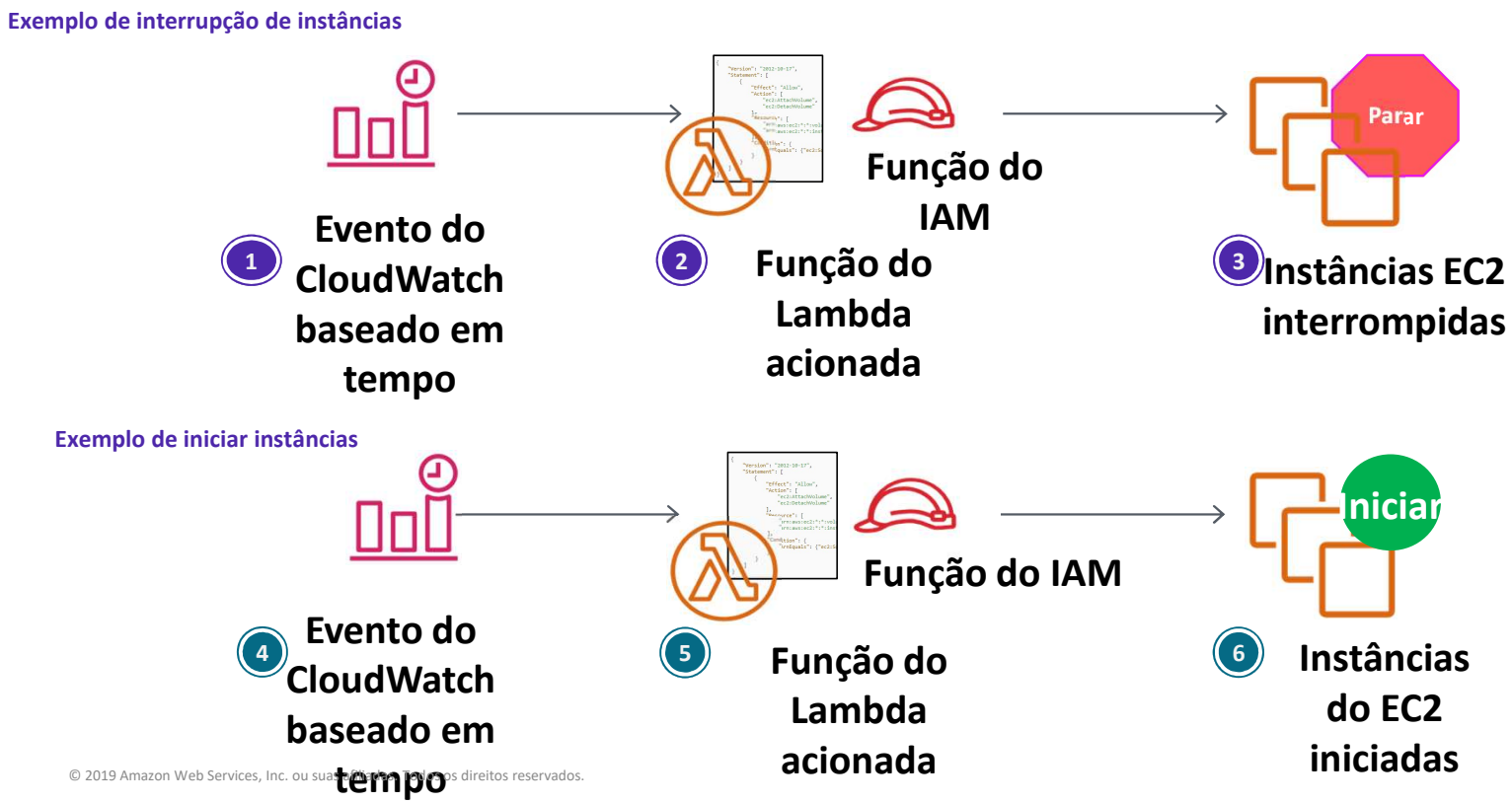
Fontes de eventos do AWS Lambda



Configuração da função do AWS Lambda



Exemplo de função Lambda baseada em programação: Iniciar e interromper EC2



Exemplo de função Lambda baseada em eventos: Criar imagens em miniatura

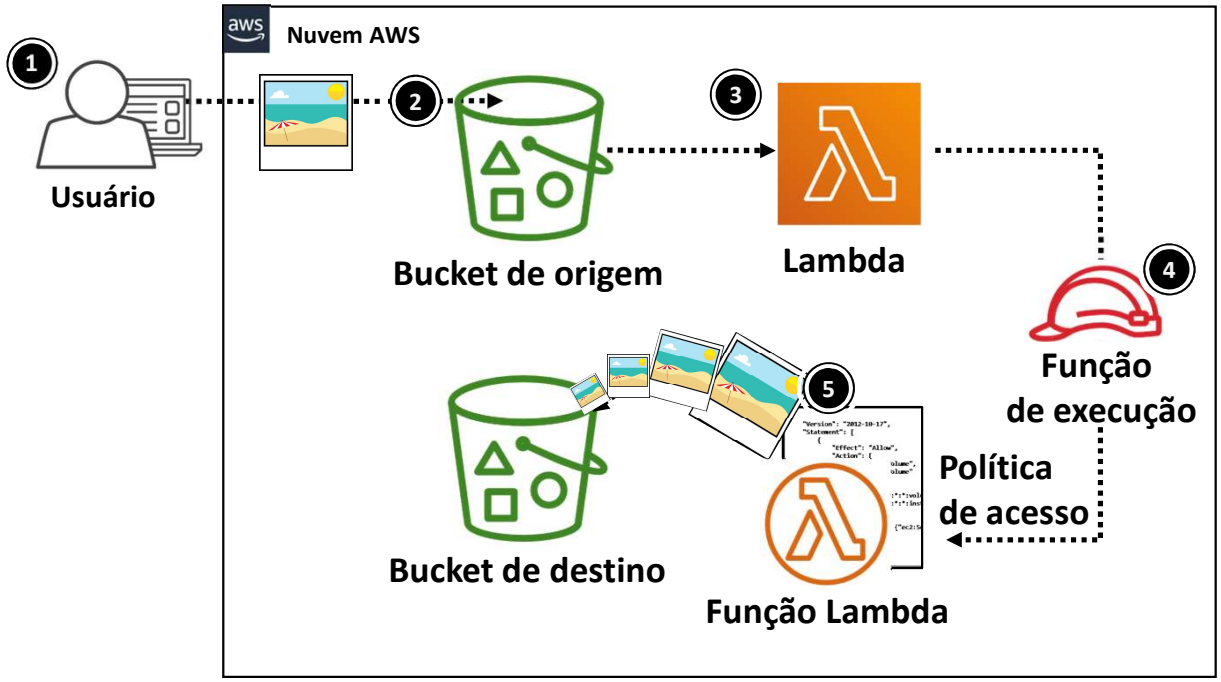


Diagrama de demonstração

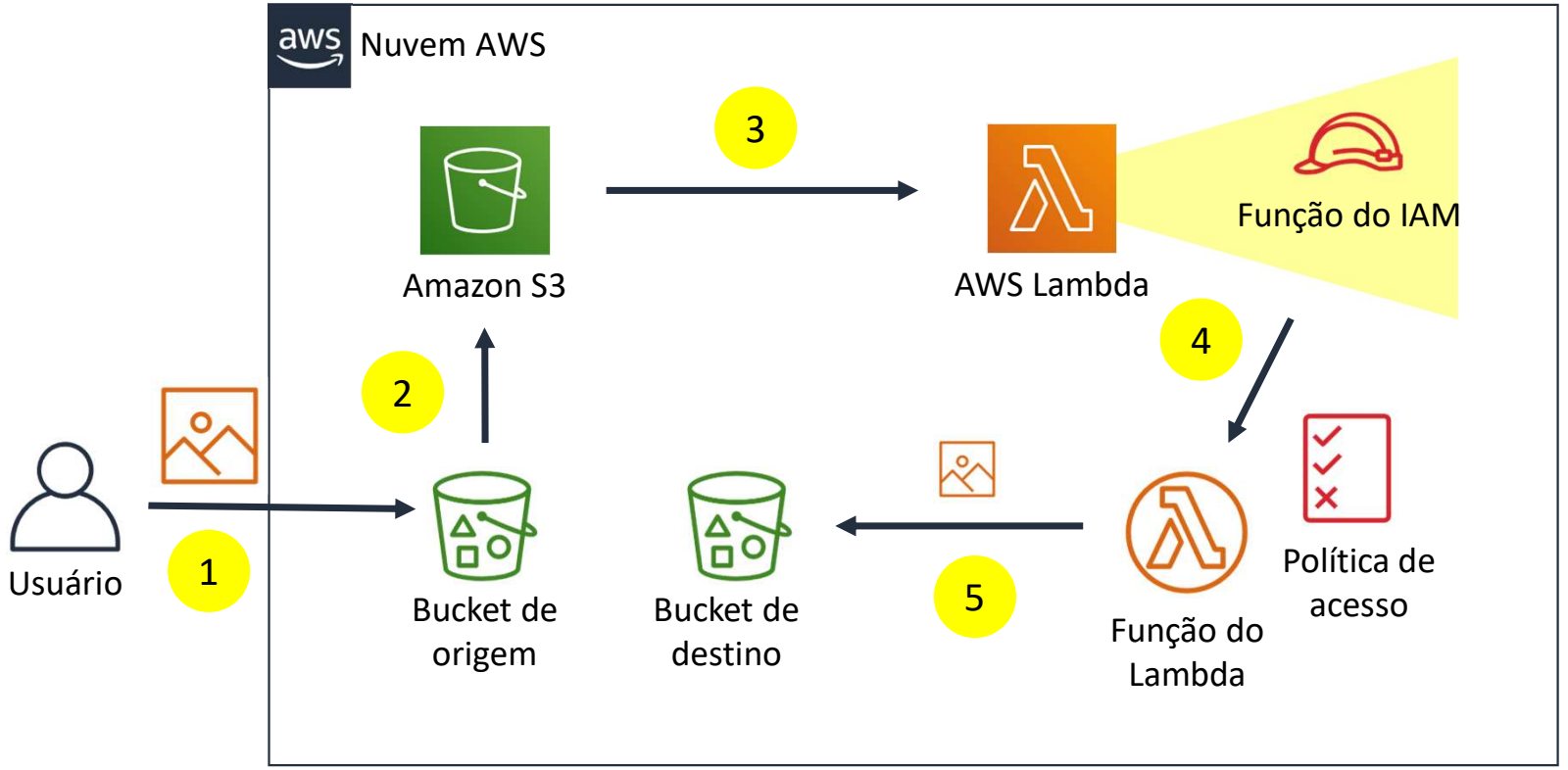
1. O usuário carrega a imagem para o bucket do S3 de origem.

2. O Amazon S3 detecta o evento criado por objeto.

3. O Amazon S3 publica o evento no Lambda.

4. O Lambda executa a função do Lambda.

5. A função do Lambda redimensiona a imagem original e salva a miniatura no bucket do S3 de destino.



```
import boto3
import os
import sys
import uuid
from PIL import Image
import PIL.Image

s3_client = boto3.client('s3')

def resize_image(image_path, resized_path):
    with Image.open(image_path) as image:
        image.thumbnail((128, 128))
        image.save(resized_path)

def handler(event, context):
    for record in event['Records']:
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']
        download_path = '/tmp/{}{}'.format(uuid.uuid4(), key)
        upload_path = '/tmp/resized-{}'.format(key)

        s3_client.download_file(bucket, key, download_path)
        resize_image(download_path, upload_path)
        s3_client.upload_file(upload_path, '{}-resized'.format(bucket), key)
```

Recebe o evento do S3 e faz download da imagem para o armazenamento local

Redimensiona a imagem

Carrega a imagem redimensionada para o – bucket *redimensionado*

Limites do AWS Lambda

- Limites flexíveis por região:
  - Execuções simultâneas = 1.000
  - Função e armazenamento de camadas = 75 GB
- Limites rígidos para funções individuais:
  - Alocação máxima de memória da função = 3.008 MB
  - Tempo limite da função = 15 minutos
  - Tamanho do pacote de implantação = 250 MB descompactados, incluindo camadas
- Limites adicionais também existem. Os detalhes estão na documentação de limites do AWS Lambda

Principais lições da Seção 5

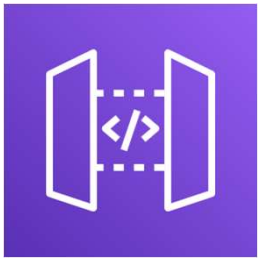
- O Lambda é um serviço de computação sem servidor que oferece tolerância a falhas e escalabilidade automática integradas
- Uma função do Lambda é um código personalizado escrito por você e que processa eventos
- Uma função do Lambda é chamada por um manipulador, que pega um objeto de evento e um objeto de contexto como parâmetros
- Uma fonte de evento é um serviço da AWS ou aplicação criada por desenvolvedor que aciona a execução de uma função do Lambda
- As camadas do Lambda permitem que as funções compartilhem código e mantenham os pacotes de implantação pequenos

Módulo 13: Criar microsserviços e arquiteturas sem servidor

Seção 6: Estender arquiteturas sem servidor com o Amazon API Gateway

aws academy

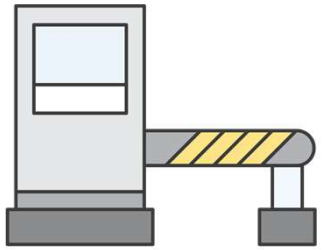
Amazon API Gateway



Amazon API Gateway

- Permite criar, publicar, manter, monitorar e proteger APIs que funcionam como pontos de entrada para recursos de back-end para as aplicações
- Lida com até centenas de milhares de chamadas à API simultâneas
- Pode lidar com cargas de trabalho que são executadas em:
  - Amazon EC2
  - Lambda
  - Qualquer aplicação Web
  - Aplicações de comunicação em tempo real
- Pode hospedar e usar várias versões e fases das APIs

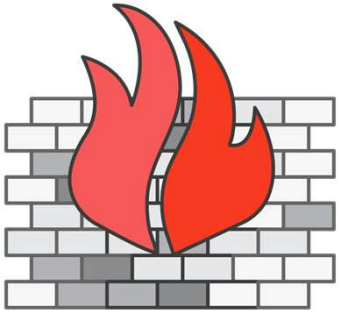
Segurança do Amazon API Gateway



Solicitar autorização

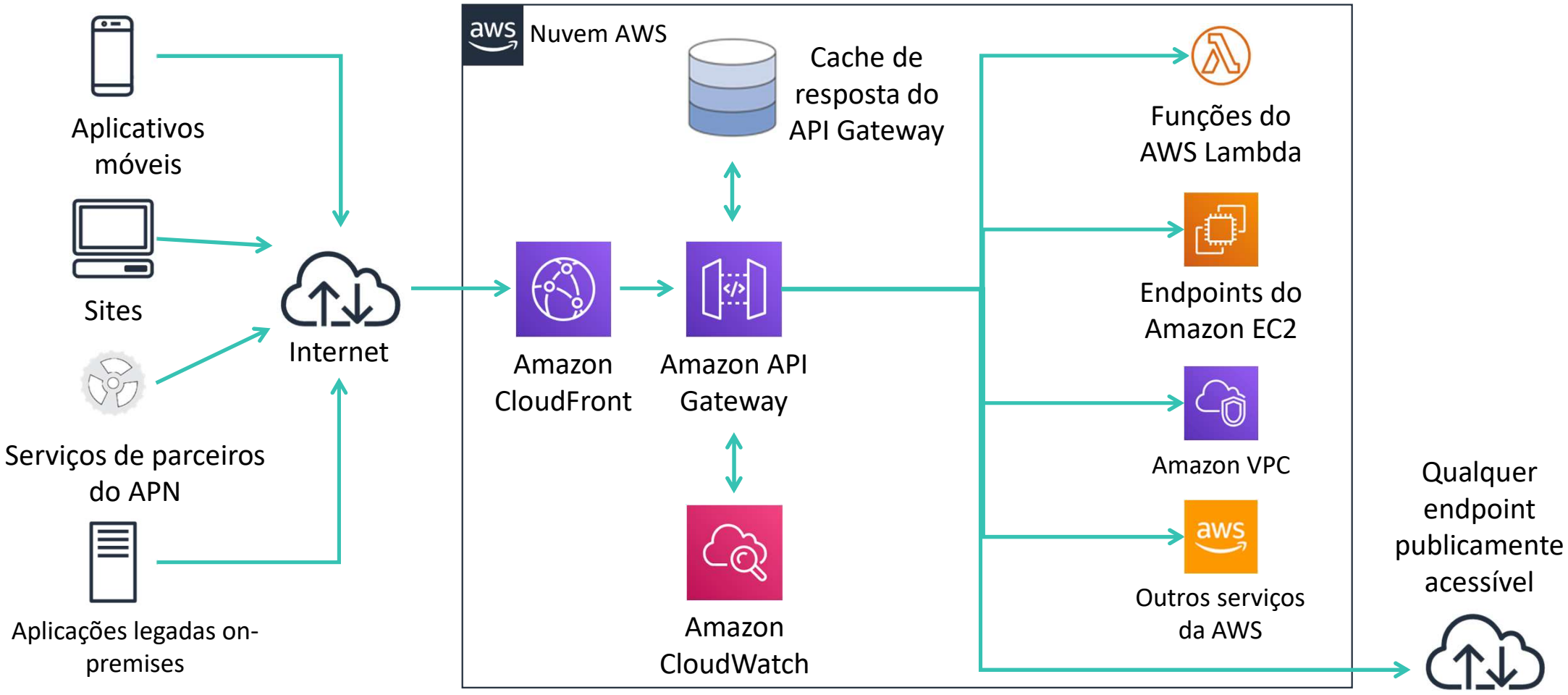
Aplicar políticas de recursos

Configurações de controle de utilização

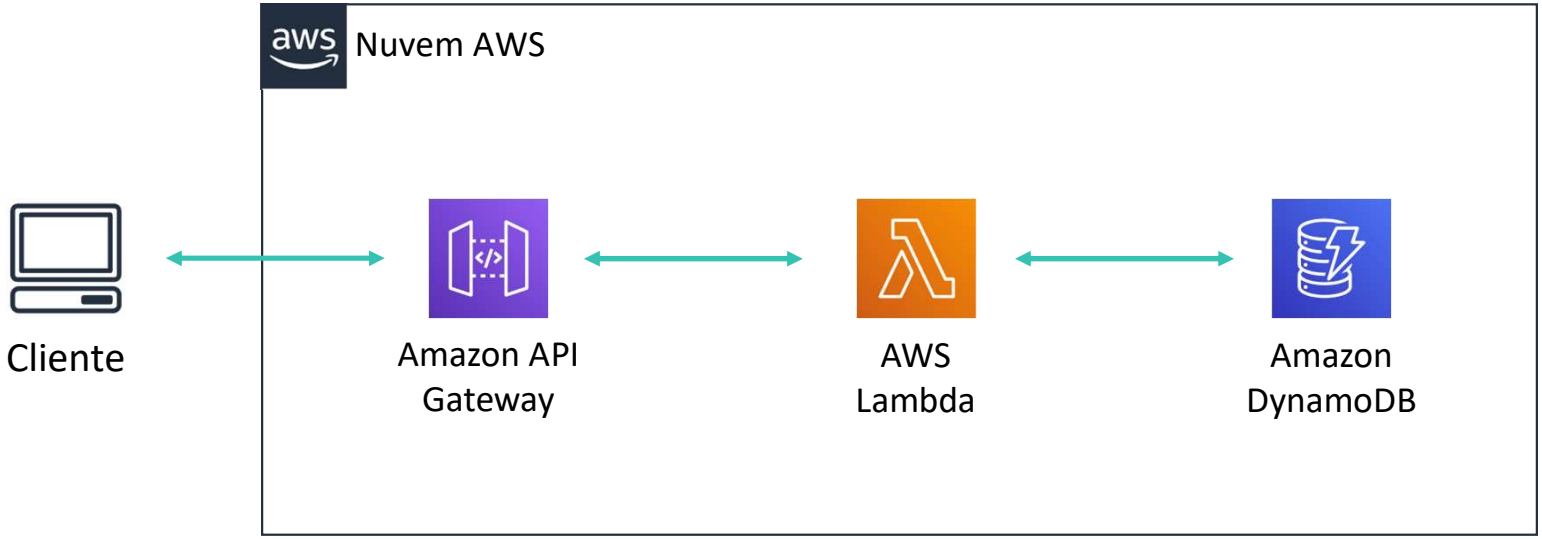


Proteção contra ataques de Distributed Denial of Service (DDoS – Negação distribuída de serviço) e de injeção

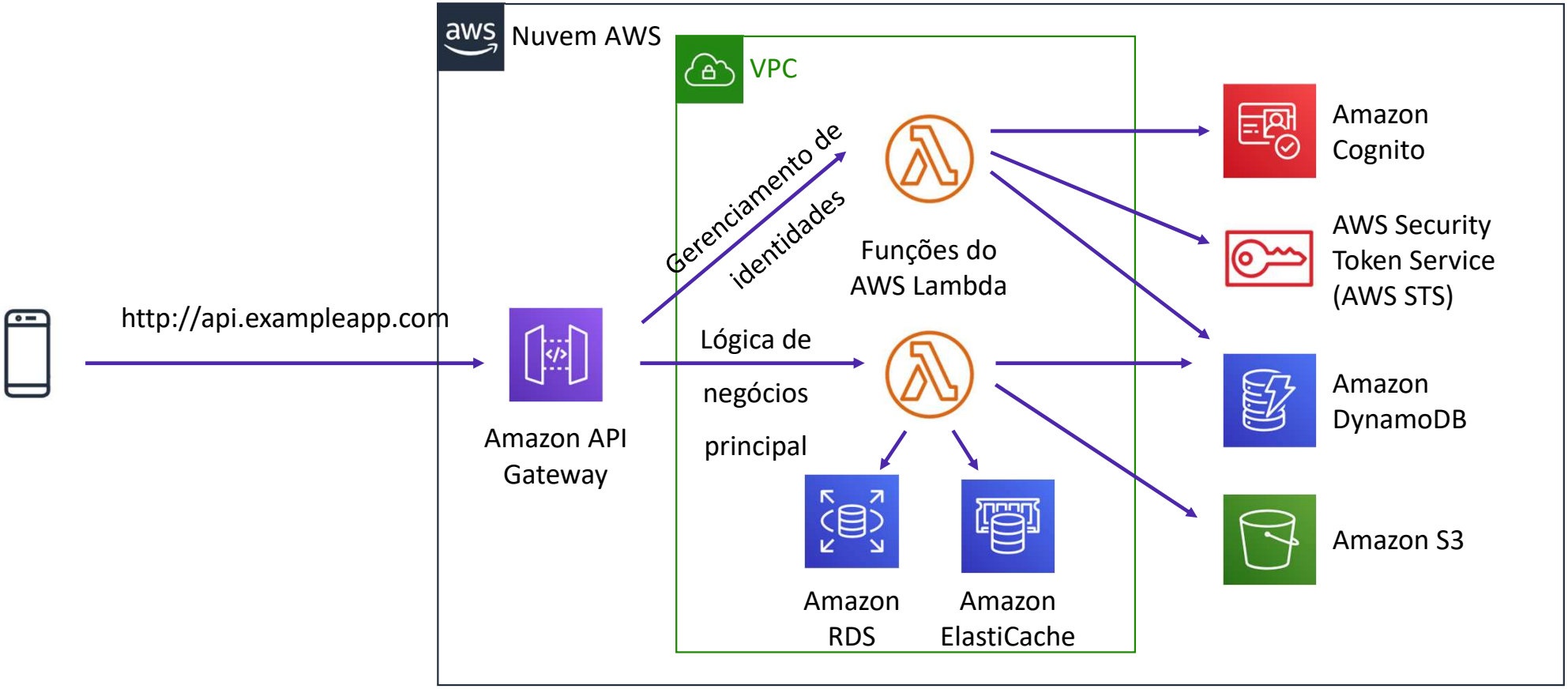
Amazon API Gateway: exemplo de arquitetura comum



Exemplo: microsserviços RESTful



Exemplo: back-end móvel sem servidor



Principais lições da Seção 6

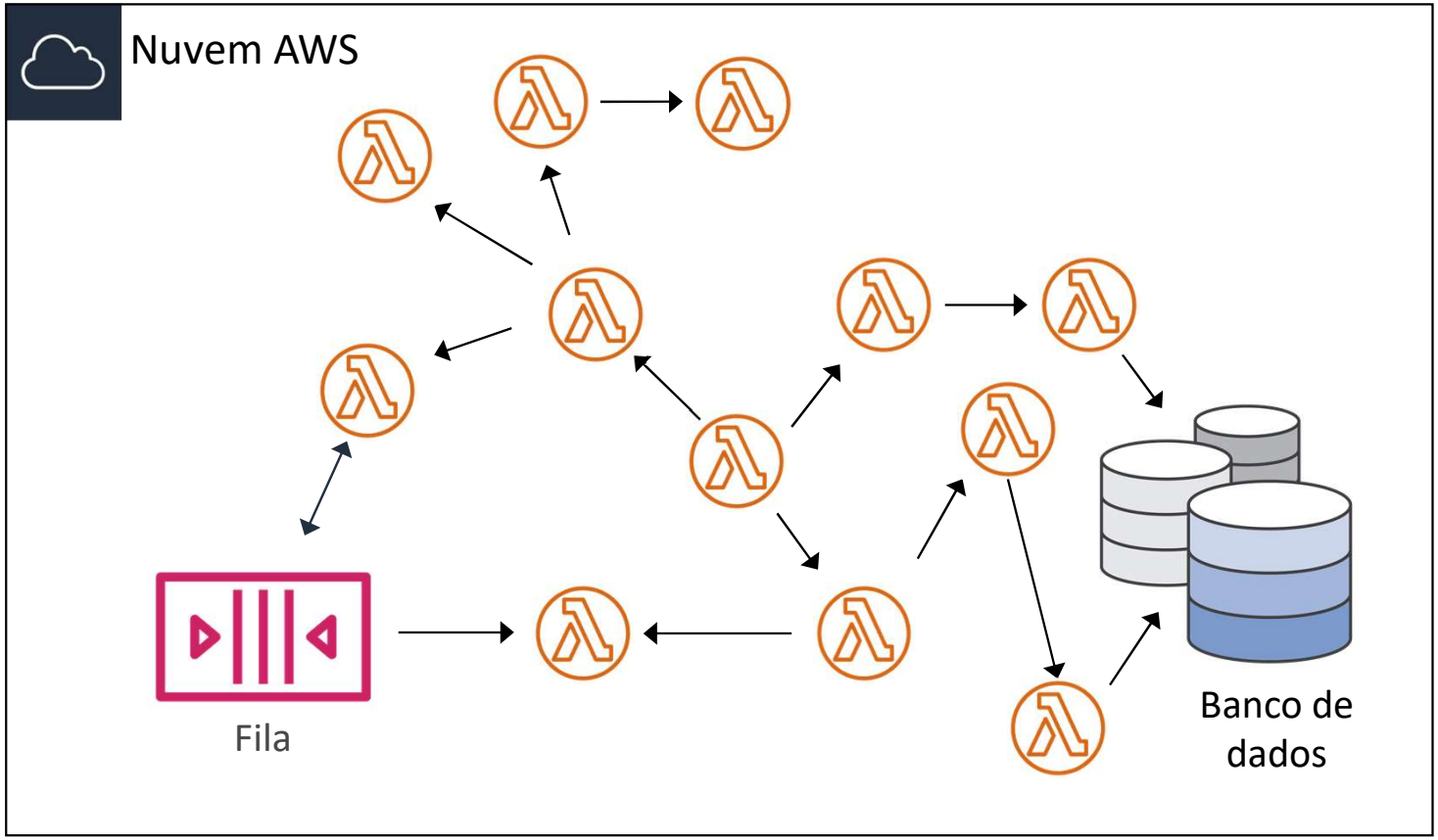
- O Amazon API Gateway é um serviço totalmente gerenciado que permite criar, publicar, manter, monitorar e proteger APIs em qualquer escala.
- O Amazon API Gateway atua como um ponto de entrada para recursos de back-end para as aplicações. Ele abstrai e expõe APIs que podem chamar várias aplicações de back-end. Essas aplicações incluem funções do Lambda, contêineres do Docker executados em instâncias do EC2, VPCs ou qualquer endpoint acessível publicamente.
- O Amazon API Gateway está profundamente integrado ao Lambda.

Módulo 13: Criar microsserviços e arquiteturas sem servidor

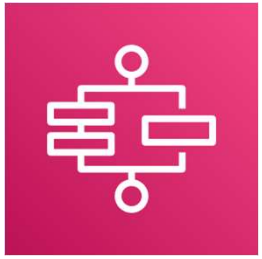
Seção 7: Orquestrar microsserviços com o AWS Step Functions

aws academy

Desafios com aplicações de microsserviços



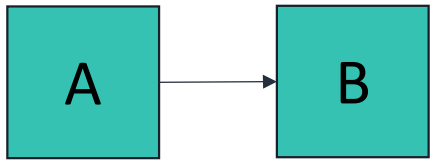
AWS Step Functions



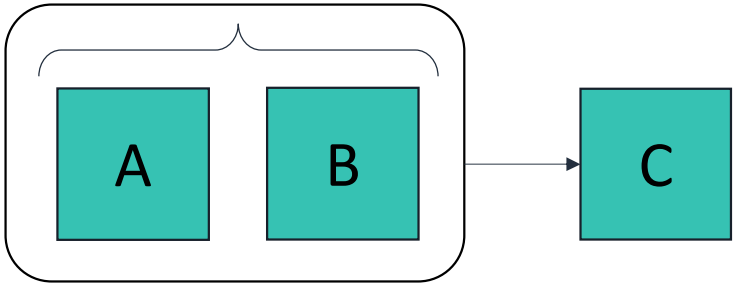
AWS Step Functions

- Coordena microsserviços usando fluxos de trabalho visuais
- Permite percorrer as funções da sua aplicação
- Dispara e rastreia automaticamente cada etapa
- Fornece captura de erros simples e registro em log se uma etapa falhar

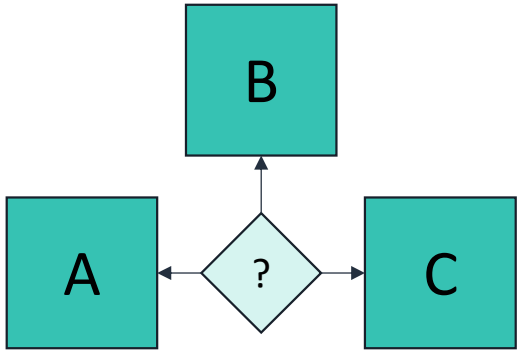
Coordenação de fluxo de trabalho



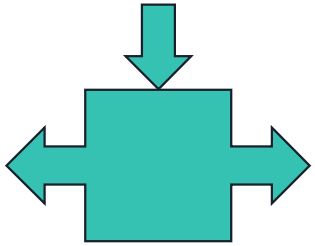
Executar tarefas em sequência



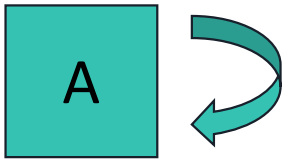
Executar tarefas em paralelo



Selecionar tarefa com base nos dados

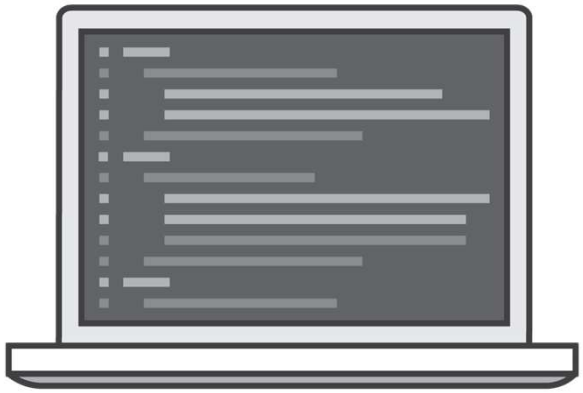


Gerenciar o comportamento do try-catch-finally

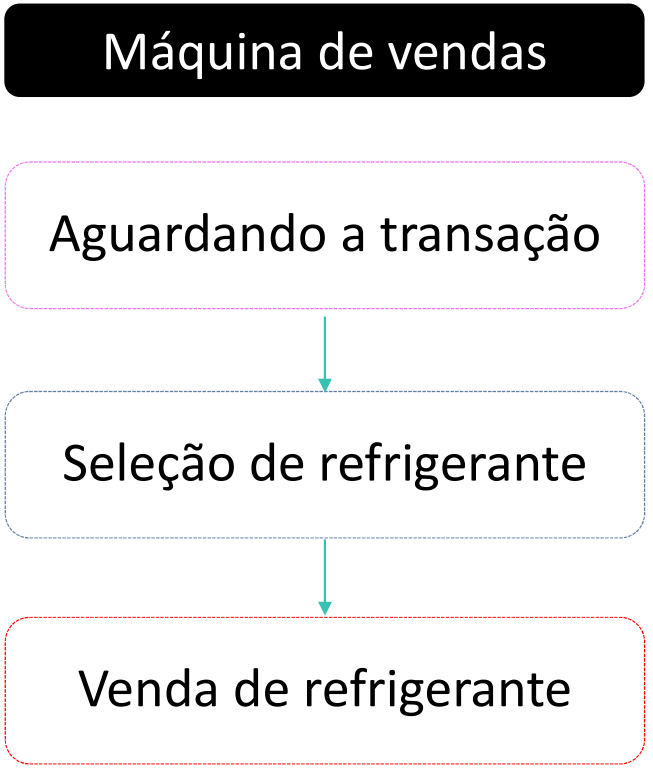


Tentar novamente tarefas com falha

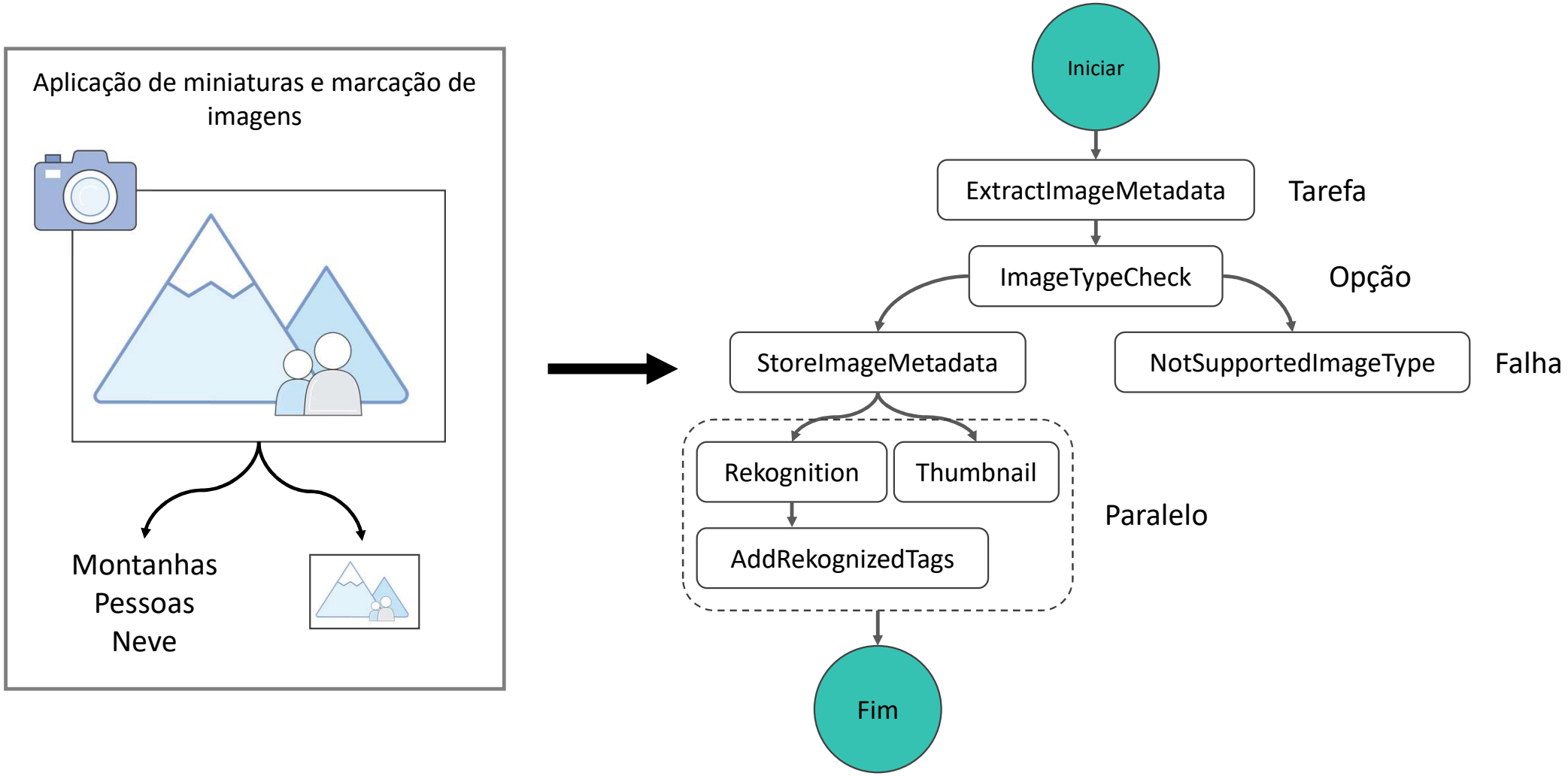
Máquinas de estado



Uma máquina de estado é uma coleção de estados que podem realizar um trabalho.



Estados


Aplicação de miniaturas e marcação de imagens
Montanhas
Pessoas
Neve
Iniciar
ExtractImageMetadata
Tarefa
ImageTypeCheck
Opção
StoreImageMetadata
NotSupportedImageType
Falha
Rekognition
Thumbnail
AddRekognizedTags
Paralelo
Fim

Tipos de estado

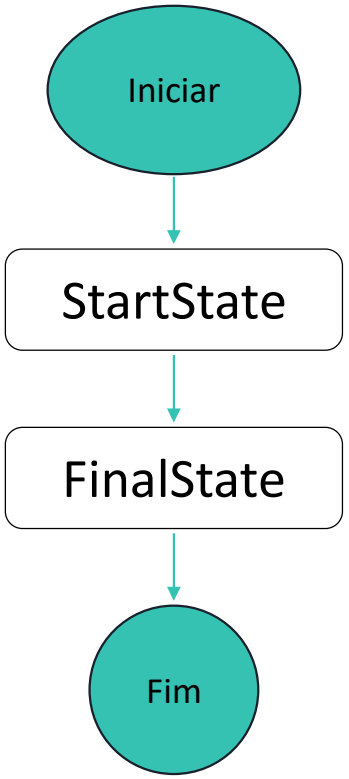
Tarefa	Uma unidade única de trabalho executada por uma máquina de estado
Opção	Adiciona a lógica de ramificação a uma máquina de estado
Falha	Interrompe uma máquina de estado em execução e a marca como uma falha
Êxito	Interrompe uma máquina de estado em execução com êxito
Transmitir	Transmite a entrada para a saída, sem executar trabalho
Aguardar	Atrasos de continuação por um período especificado
Paralelo	Cria ramificações paralelas para serem executadas na máquina de estado
Mapa	Itera dinamicamente as etapas

Amazon States Language

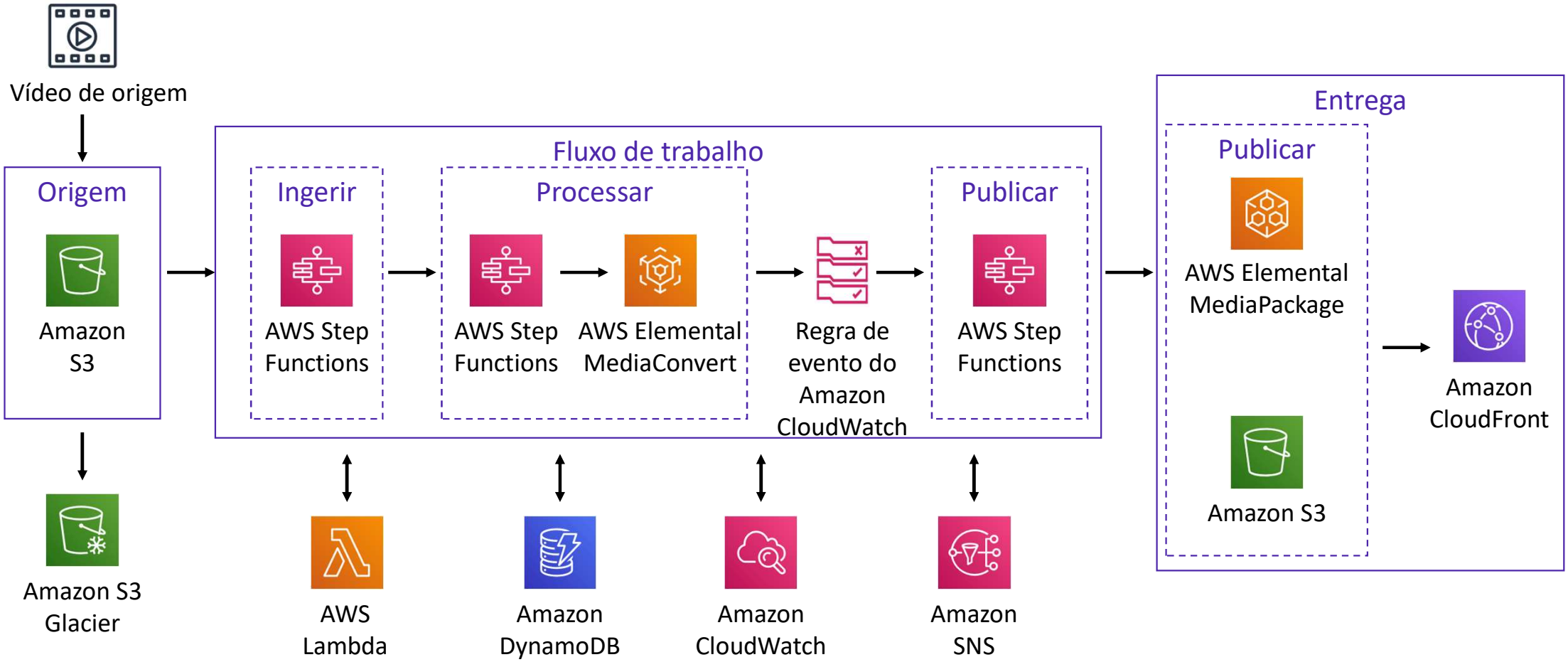
Definir o fluxo de trabalho em JSON usando a Amazon States Language

```
{
 "Comment": "An example of the ASL.",
 "StartAt": "StartState",
 "States": {
 "StartState": {
 "Type": "Task",
 "Resource": "arn:aws:lambda:us-east…,
 "Next": "FinalState"
 }
 "FinalState": {
 "Type": "Task",
 "Resource": "arn:aws:lambda:us-east…,
 "End": true
 }
 }
}
```

Visualizar o fluxo de trabalho no console do Step Functions



Exemplo do AWS Step Functions: arquitetura de Video-on-demand (VOD – vídeo sob demanda)



Principais lições da Seção 7

- O AWS Step Functions é um serviço web que permite coordenar os componentes de aplicações e microsserviços distribuídos usando fluxos de trabalho visuais
- O AWS Step Functions permite que você crie e automatize suas próprias máquinas de estado dentro do ambiente da AWS
- O AWS Step Functions gerencia a lógica da aplicação para você e implementa primitivas básicas, como ramificações sequenciais ou paralelas e tempos limite
- Você define máquinas de estado usando a Amazon States Language

Dúvidas?