

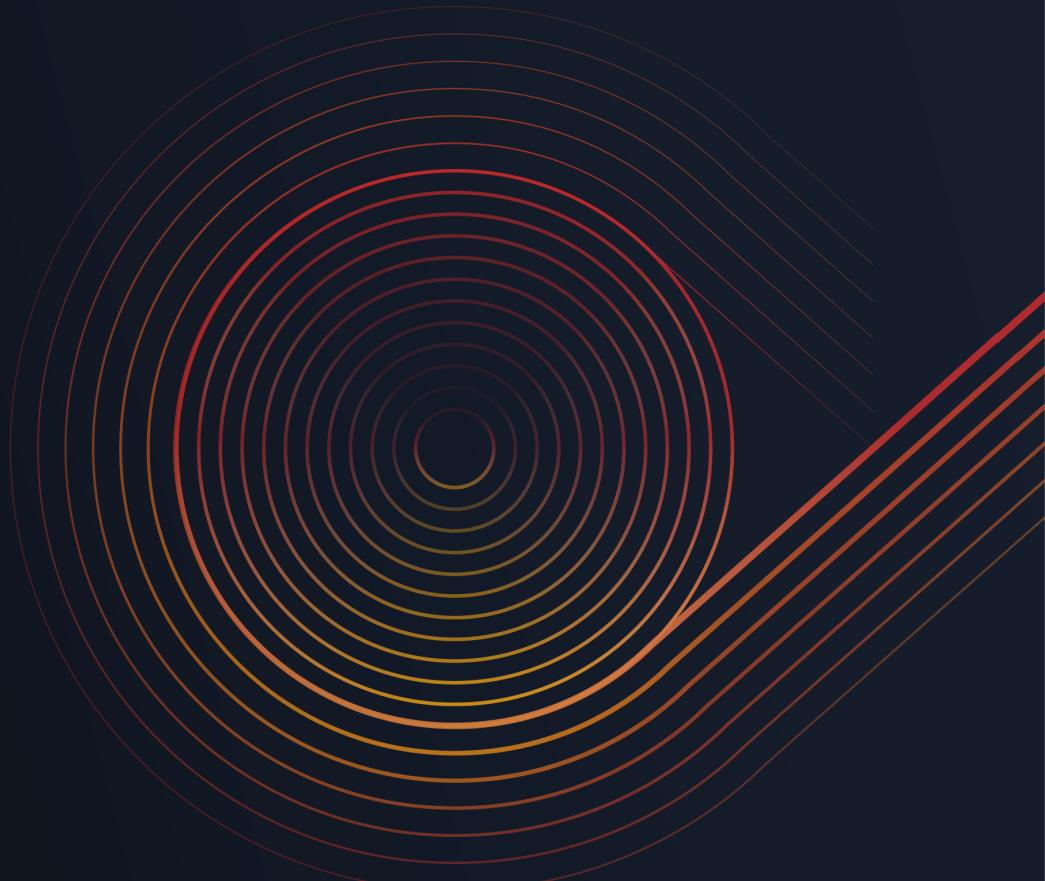


Serverless

O que preciso saber para começar ?

Fernando Sapata

LatAm Business Development Manager, **Serverless**



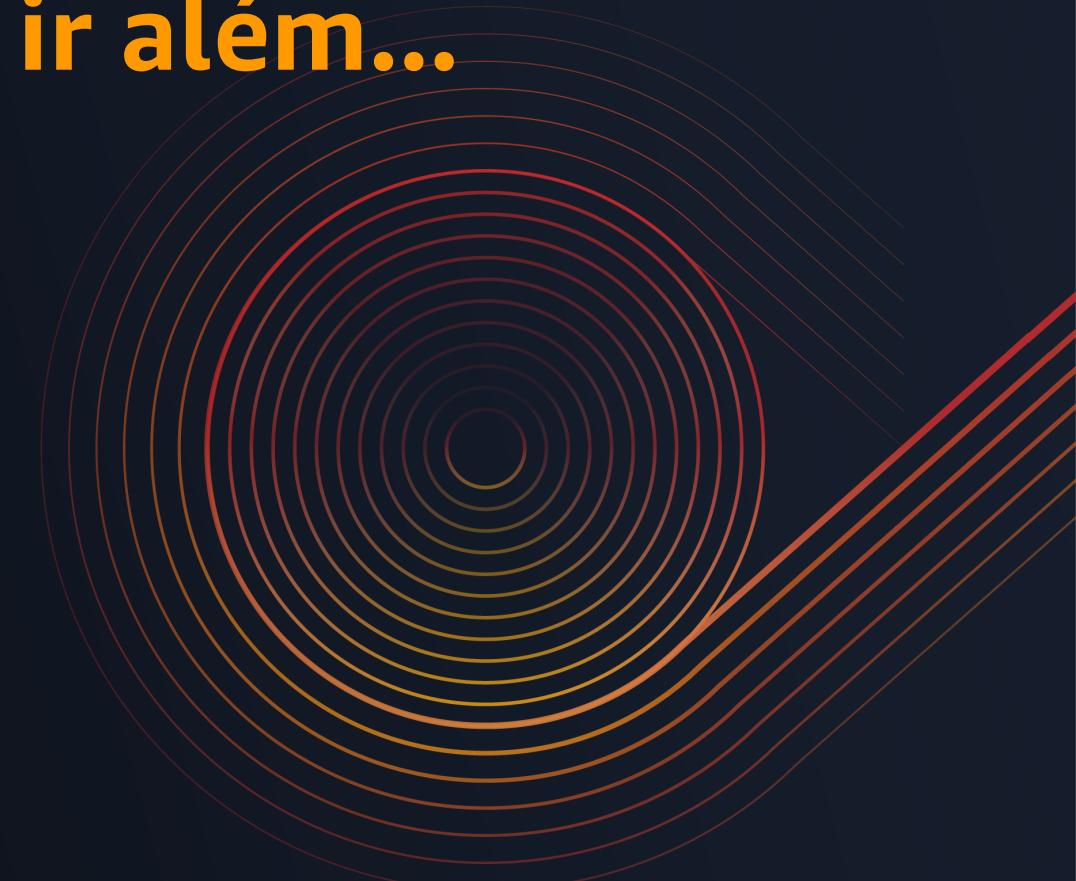


Serverless

O que preciso saber para começar ? e ir além...

Fernando Sapata

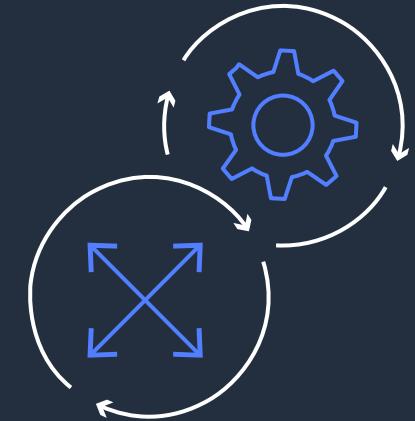
LatAm Business Development Manager, **Serverless**



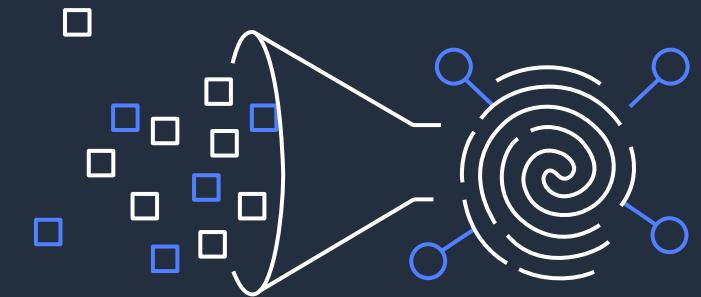
O que nossos clientes querem



Construir aplicações,
não infra-estrutura



Dimensionar rapidamente
e perfeitamente



Segurança e isolamento
por padrão

Complexidade essencial e acidental



Serverless significa...



Serverless significa...



Não é preciso gerenciar servidores



Escale conforme o uso

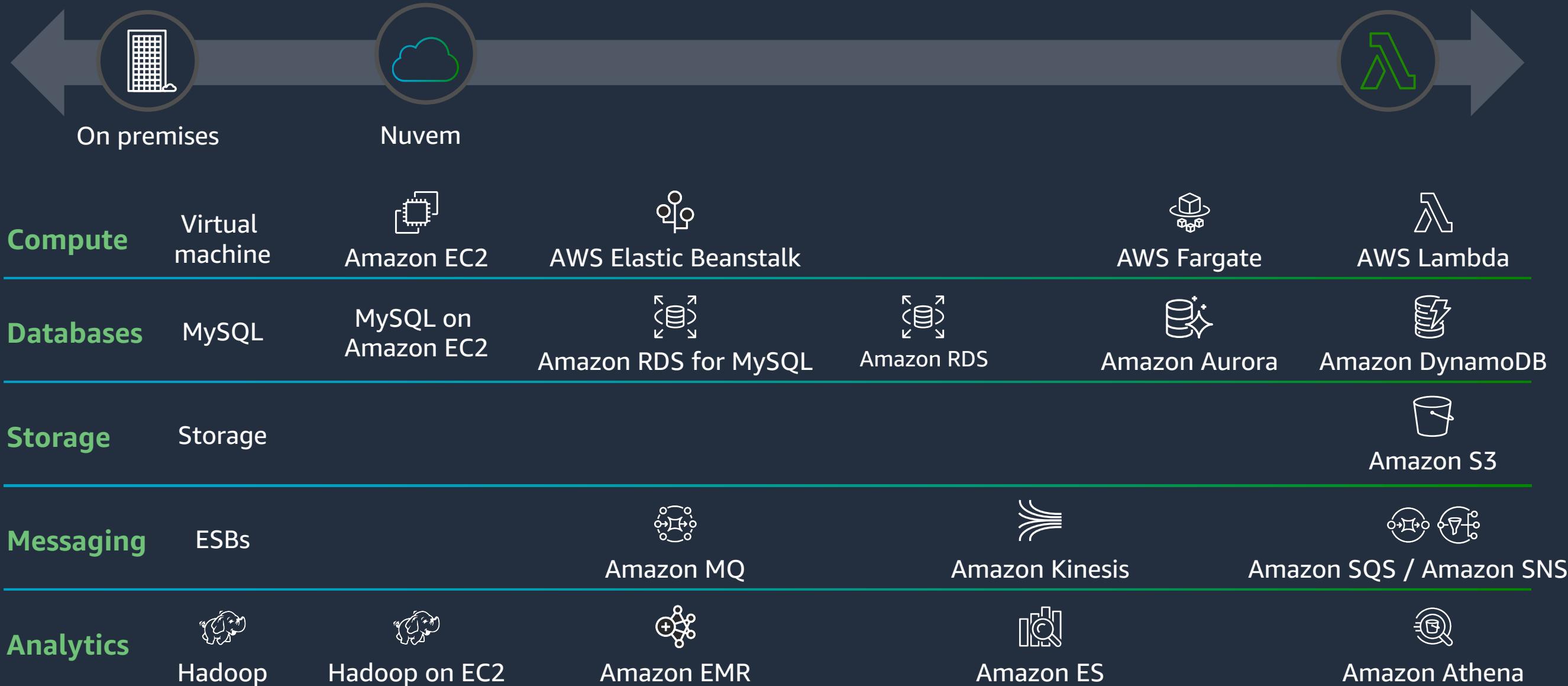


Não pague pela ociosidade

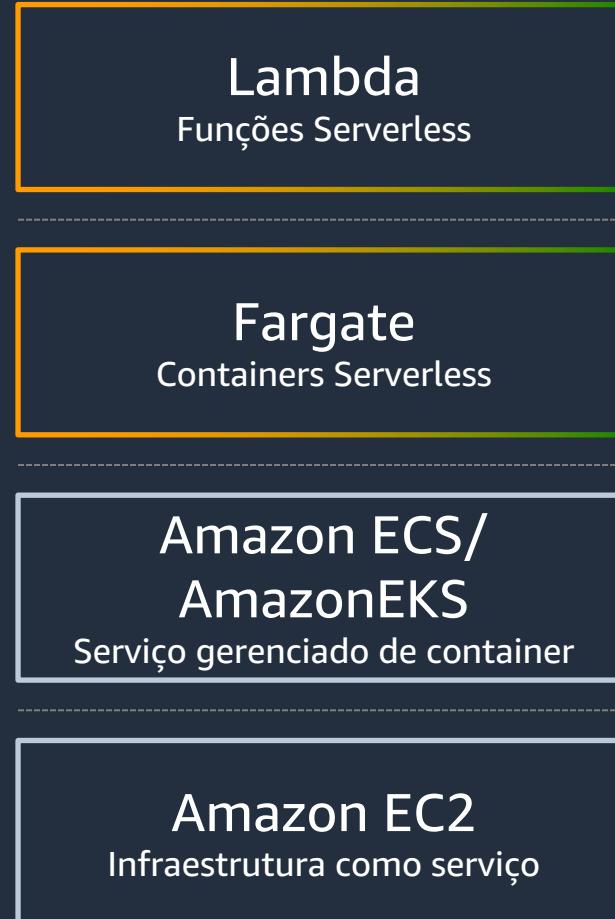


Altamente disponível

Modelos de responsabilidade operacional da AWS



Comparação de responsabilidade operacional



AWS gerencia

- Integração com fontes de dados
- Hardware, software, rede, e instalações físicas
- Provisionamento

- Orquestração de container, provisionamento
- Escalabilidade do cluster
- Hardware, SO, rede, e instalações físicas

- Orquestração de container
- Plano de controle
- Hardware, software, rede, e instalações físicas

- Hardware, software, rede, e instalações físicas

Cliente gerencia

- Código da aplicação

- Código da aplicação
- Integração com fontes de dados
- Configurações de segurança, configurações de rede, gerenciamento de tarefas

- Código da aplicação
- Integração com fontes de dados
- Clusters
- Configurações de segurança, configurações de rede, gerenciamento de tarefas

- Código da aplicação
- Integração com fontes de dados
- Escalabilidade
- Configurações de segurança, configurações de rede, gerenciamento de tarefas
- Provisionamento, gerenciamento de escalabilidade e patching de servidores

Serverless é um modelo operacional que está presente em diferentes categorias de serviços

PROCESSAMENTO



Lambda



Fargate

ARMAZENAMENTO DE DADOS



Amazon
S3



Amazon Aurora
Serverless



Amazon
DynamoDB

INTEGRAÇÃO



Amazon
API Gateway



Amazon
SQS



Amazon
SNS

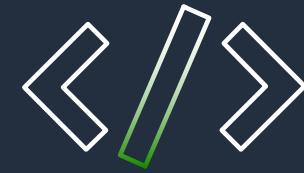


AWS
Step Functions



AWS
AppSync

Tornando o desenvolvimento fácil com Lambda



Acessível para qualquer desenvolvedor

Suporte para todos runtimes com Lambda Layers e Runtime API

ISO, PCI, HIPAA, SOC, GDPR, e FedRamp compliances



Melhor produtividade

Ferramentas para IDEs populares:

VSCode, IntelliJ e PyCharm

Deploy simplificado



Permite novos padrões de aplicações

Funções de 15 minutos

SQS para Lambda

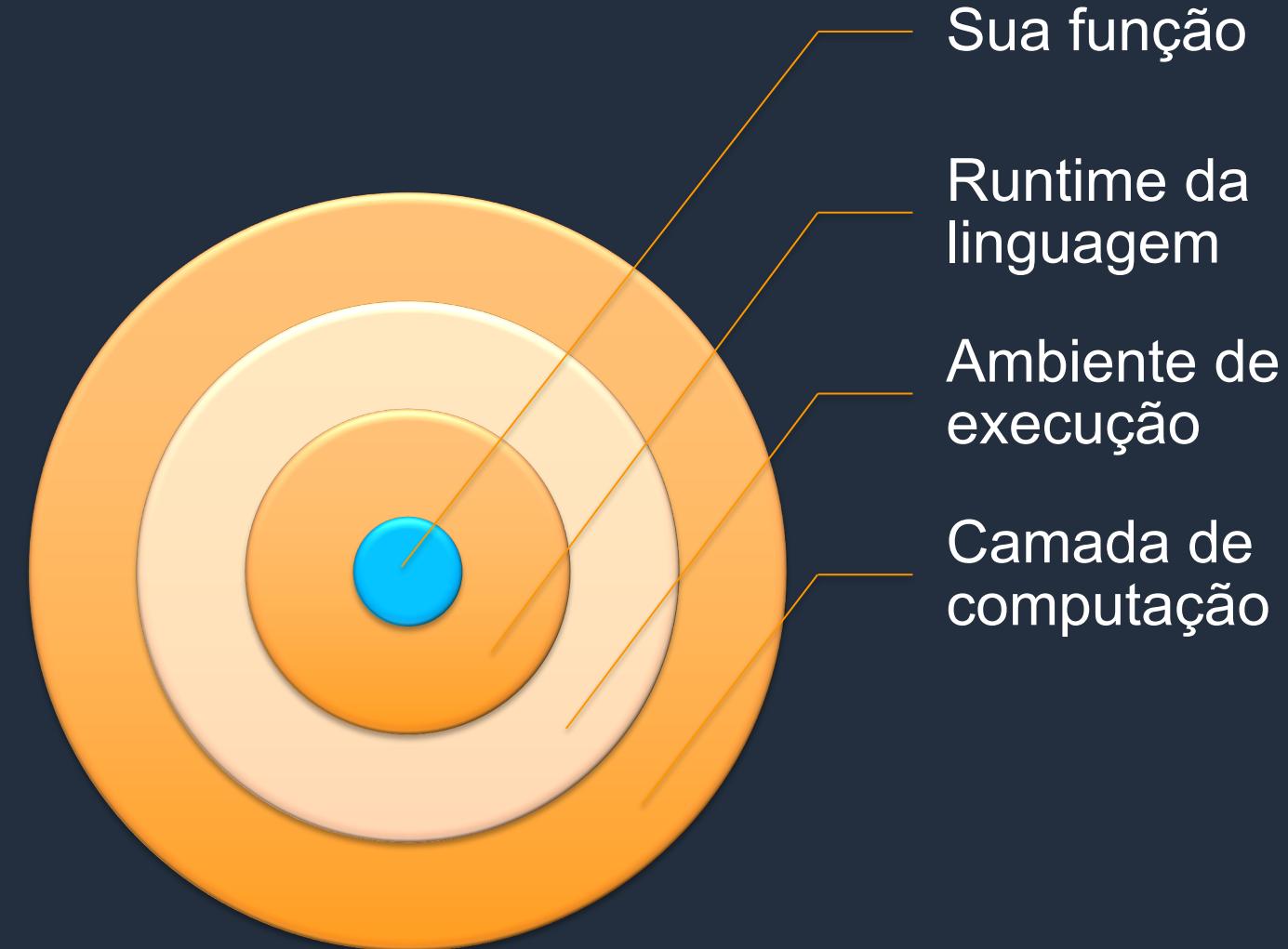
Balanceamento de carga automático para Lambda

Suporte para Kinesis Data Streams Enhanced Fan-Out e HTTP/2

Anatomia de uma função Lambda



Anatomia de uma função Lambda



Anatomia de uma função Lambda



Sua função

Runtime da
linguagem

Ambiente de
execução

Camada de
computação

**Camadas que
podem impactar a
performance**

Anatomia de uma função Lambda



Sua função

Runtime da
linguagem

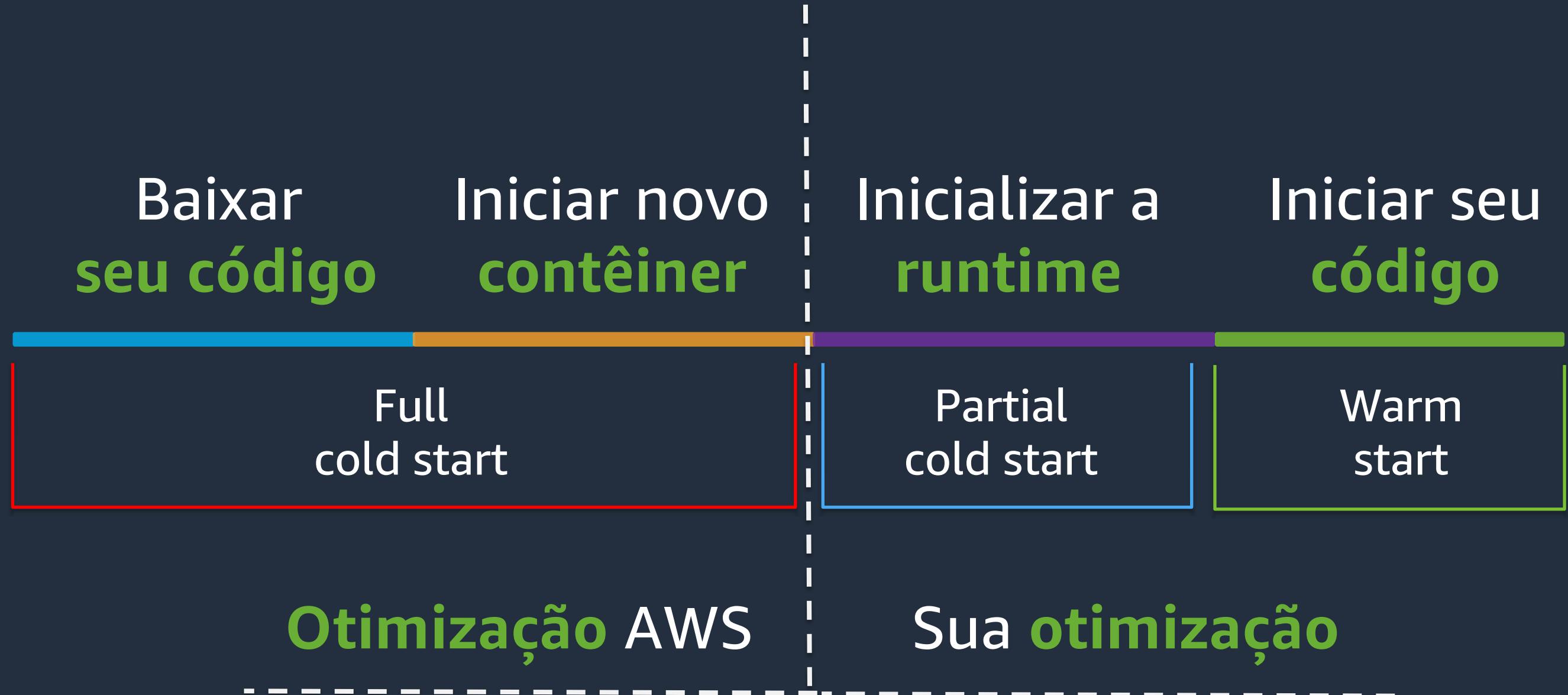
Ambiente de
execução

Camada de
computação



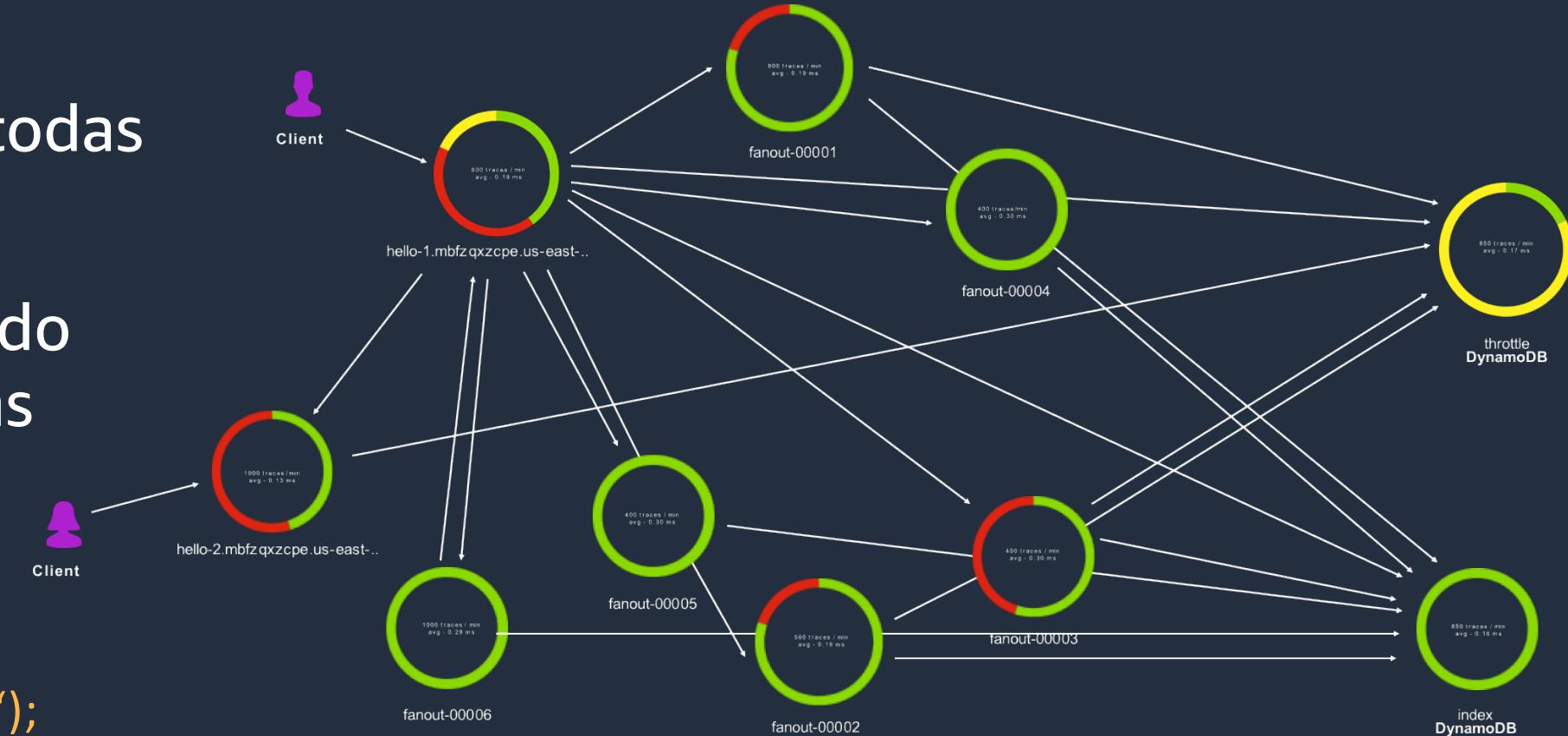
**Camadas que
podem impactar a
performance**

O ciclo de vida da função Lambda



Integrando o AWS X-Ray com aplicações serverless

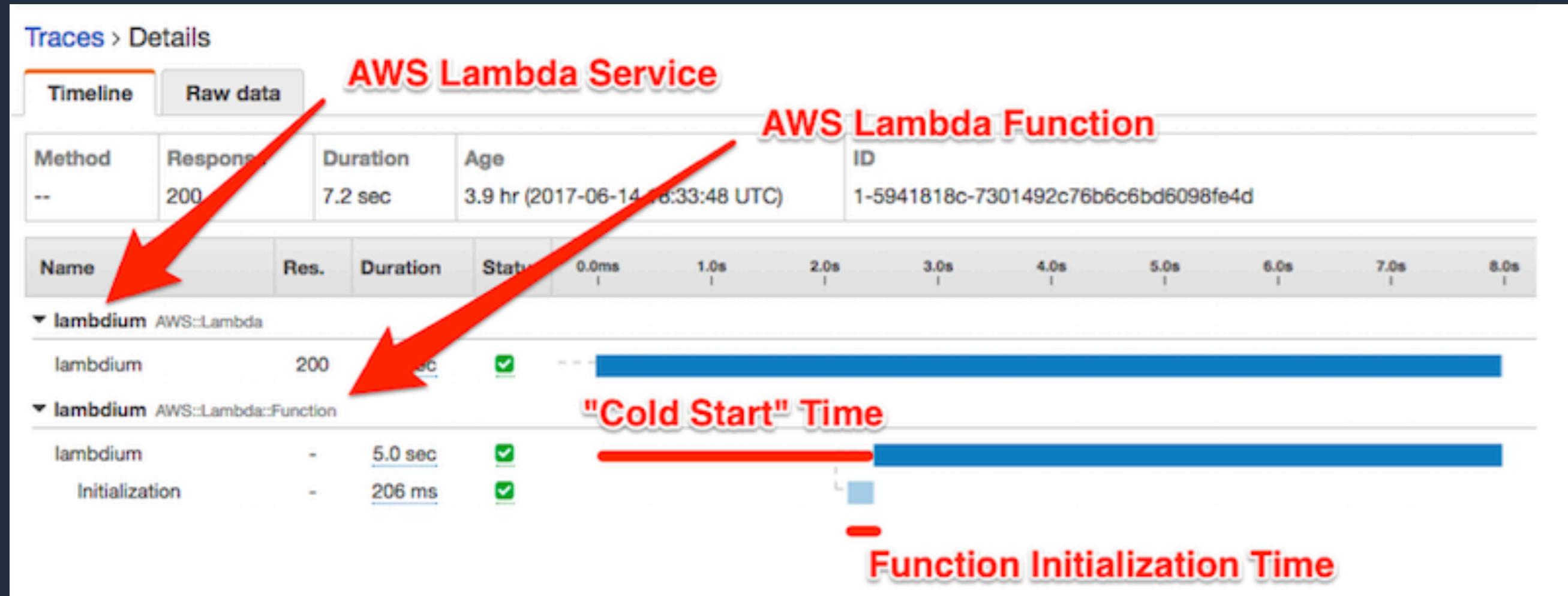
- O Lambda instrumenta as requisições de entrada para todas as linguagens suportadas
- O Lambda executa o serviço do X-Ray em todas as linguagens utilizando um SDK



```
var AWSXRay = require('aws-xray-sdk-core');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));
S3Client = AWS.S3();
```



Usando o AWS X-Ray para visualizer o “Cold Start”



Ajuste o poder de computação da sua função



O Lambda permite controlar o total de memória alocada para a função, sendo o **% de CPU e capacidade de rede** alocadas proporcionalmente.

Alocação de recursos inteligente

Estatísticas de uma função Lambda que calcula **1.000 vezes** todos os números primos **<= 1.000.000**

128 MB	11.722965sec	\$0.024628
256 MB	6.678945sec	\$0.028035
512 MB	3.194954sec	\$0.026830
1024 MB	1.465984sec	\$0.024638

Verde==Melhor

Vermelho==Pior

Alocação de recursos inteligente

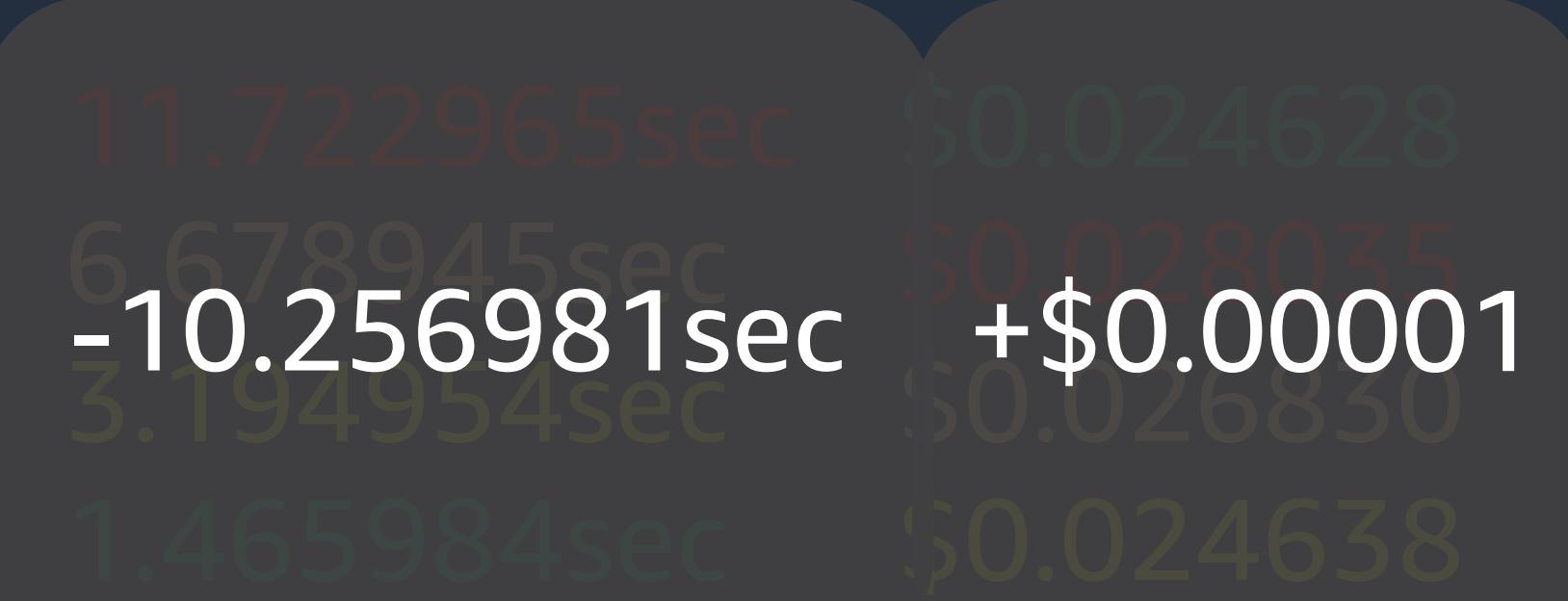
Estatísticas de uma função Lambda que calcula **1000 vezes** todos os números primos **<= 1000000**

128 MB

256 MB

512 MB

1024 MB



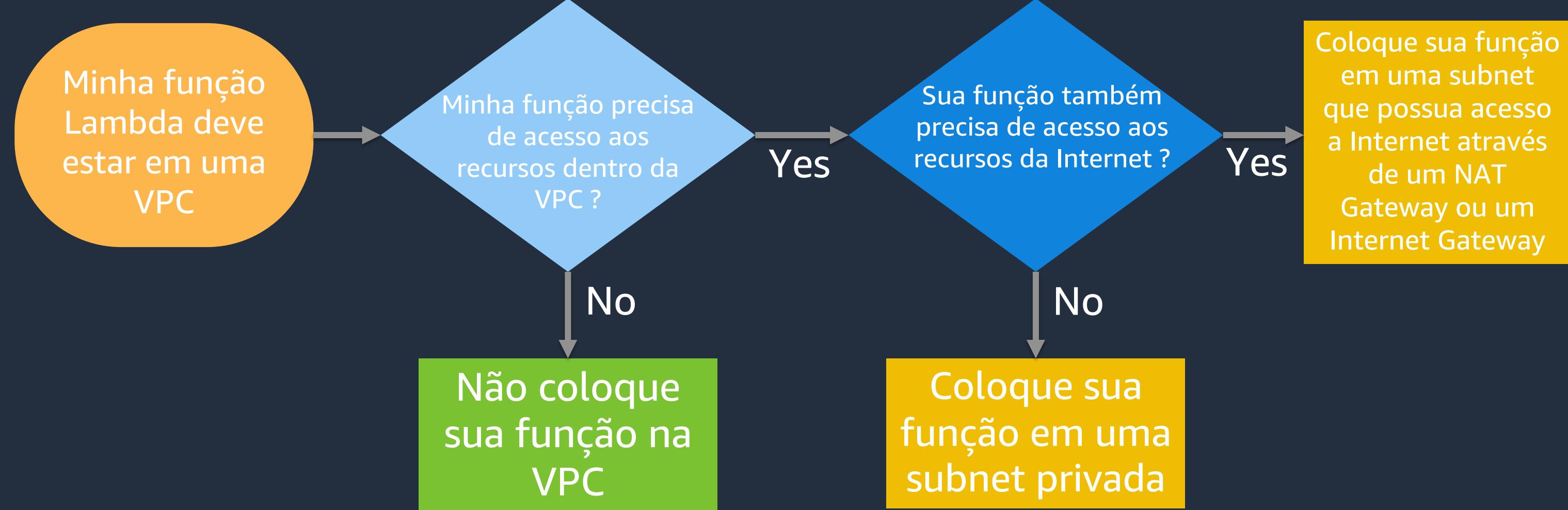
Verde==Melhor

Vermelho==Pior

Multithreading? Talvez!

- <1.8GB = single core
 - Funções orientadas a CPU não verão ganhos – os processos compartilham os mesmos recursos
- >1.8GB = multi core
 - Funções orientadas a CPU verão ganhos

Preciso de uma VPC ?



VPC x Resiliência

- SEMPRE configure no mínimo 2 Zonas de Disponibilidade
- Crie subnets específicas para seus Lambdas
- Crie subnets com um range de Ips suficientes para seus Lambdas
- Se a sua função precisar acessar recursos na Internet, configure um NAT Gateway!



Anatomia de uma função Lambda



Sua função

Runtime da
linguagem

Ambiente de
execução

Camada de
computação



**Camadas que
podem impactar a
performance**



Anatomia de uma função Lambda

Handler() da função

Função que será executada na invocação do Lambda

Objeto event

Dados enviados para o Lambda no momento da invocação

Object context

Métodos disponíveis para interagir com informações de tempo de execução (request ID, log group, etc.)

```
public String handleRequest(Book book, Context context) {  
    saveBook(book);  
  
    return book.getName() + " saved!";  
}
```

Ambiente de execução efêmero

- O Lambda processa um evento por contêiner
- **LEMBRE-SE** – os contêineres são reutilizados
- Variáveis com escopo global devem ser carregadas sob demanda (**lazy loading**)
- **Não** carregue informações que você não irá utilizar, pode impactar na iniciação da função (cold start)

```
import boto3
client = None

def my_handler(event, context):
    global client
    if not client:
        client = boto3.client("s3")

    # process
```

```
const aws = require('aws-sdk');
const gm = require('gm').subClass({imageMagick: true});
const path = require('path');
const s3 = new aws.S3();

const destBucket = process.env.DEST_BUCKET;

exports.handler = function main(event, context) {
...
    for (let i = 0; i < event.Records.length; i++) {
        tasks.push(conversionPromise(event.Records[i], destBucket));
    }

    Promise.all(tasks)
        .then(() => { context.succeed(); })
        .catch((err) => { context.fail(err); });
};

function conversionPromise(record, destBucket) {
...
}

function get(srcBucket, srcKey) {
...
}

function put(destBucket, destKey, data) {
...
}
```

Taken from: Sepia App in AWS Serverless Application Repository
<https://serverlessrepo.aws.amazon.com/applications/arn:aws:serverlessrepo:us-east-1:233054207705:applications~sepio>

```
const aws = require('aws-sdk');
const gm = require('gm').subClass({imageMagick: true});
const path = require('path');
const s3 = new aws.S3();
const destBucket = process.env.DEST_BUCKET;
```

Carregar dependências e inicializar as conexões com o DB

```
exports.handler = function main(event, context) {
...
for (let i = 0; i < event.Records.length; i++) {
  tasks.push(conversionPromise(event.Records[i], destBucket));
}
Promise.all(tasks)
  .then(() => { context.succeed(); })
  .catch((err) => { context.fail(err); });
};
```

A função Handler possui pouca lógica

```
function conversionPromise(record, destBucket) {
...
}
function get(srcBucket, srcKey) {
...
}
function put(destBucket, destKey, data) {
...
}
```

Regras de negócio desacopladas

Boas práticas de uma função Lambda

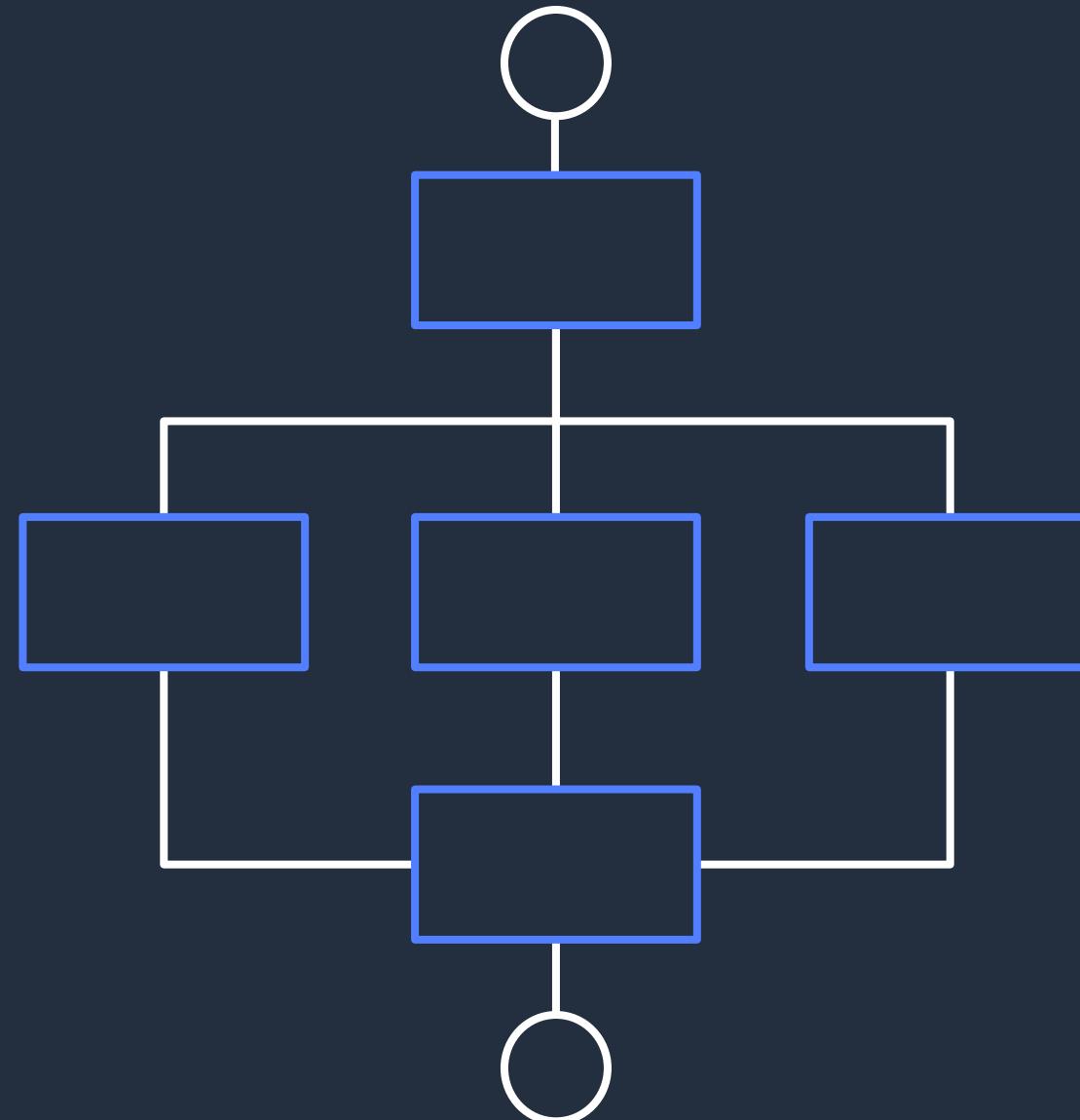
- Separe o handler (entry point) da lógica do negócio
- Separe as configurações da função utilizando;
 - Por função – Variáveis de ambiente
 - Entre funções – Amazon Parameter Store / Secrets Manager
- Leia somente o que for preciso

Não faça orquestrações no seu código



Coordenar a execução de funções

Guarda o status
dos dados e
execução



Remove código
redundante

+ Boas práticas!

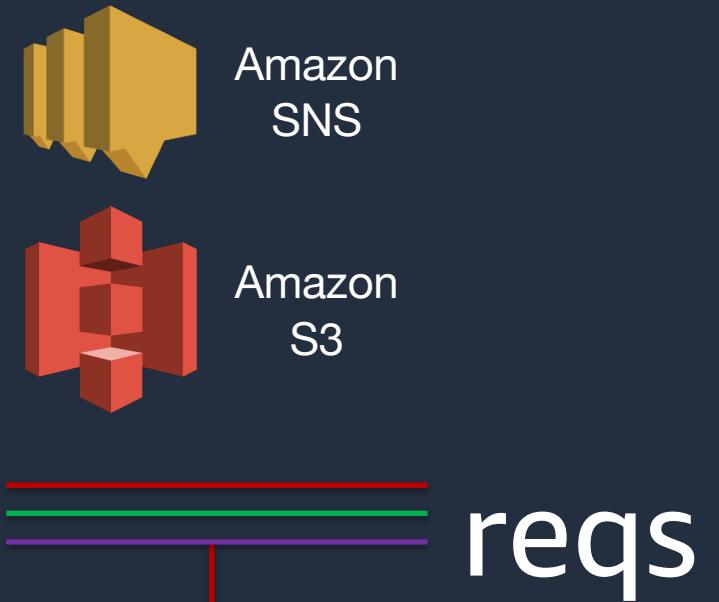
- Evite funções “fat”/monolíticas
- Controle as dependências no seu pacote de deploy
- Otimize a função para cada linguagem
 - Node – Browserfy, Minify

Modelos de execução Lambda

Síncrona (push)



Assíncrona (event)

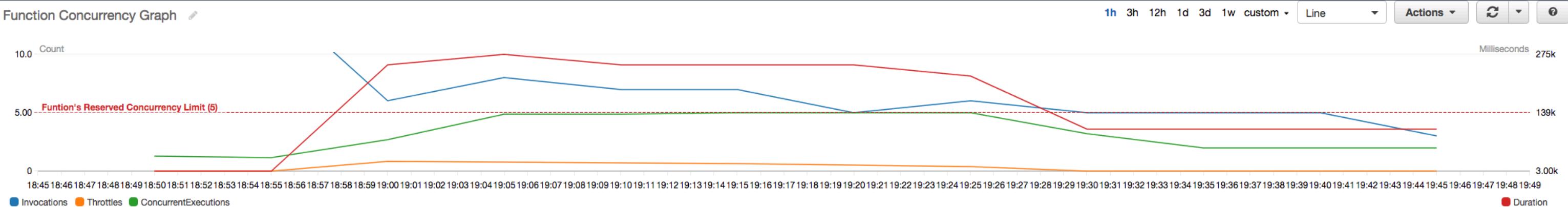


Poll-based



Controles de concorrência

- Limita a concorrência máxima de uma função
 - É importante quando a função acessa recursos como o RDS
- “Kill switch” – seta a concorrência da função para 0
- Sempre configure o timeout da função corretamente



Como descubro o que há de errado ?

Estas ferramentas já estão disponíveis

1. Ative o X-Ray
 1. Melhore a rastreabilidade com o SDK do X-Ray
2. Não subestime o poder de log do Lambda
 1. Simples “debug: in functionX” declarações funcionam bem e são fáceis de filtrar no CloudWatch Logs
3. As métricas mais importantes são as que tem relação com o seu cliente ou o seu negócio

Lambda Dead Letter Queues

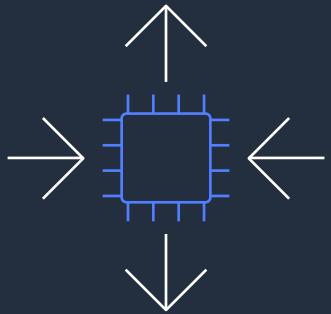
“Qualquer função do Lambda invocada de forma **assíncrona** é **repetida duas vezes** antes que o evento seja descartado.

Se as tentativas falharem e não souber o motivo, use Dead Letter Queues (DLQ) para direcionar os eventos não processados para uma fila do Amazon SQS ou para um tópico do Amazon SNS para analisar a falha.”



- https://docs.aws.amazon.com/pt_br/lambda/latest/dg/dlq.html

Entregue inovação



Agilidade

“Confiamos no AWS Lambda para colocar nossa plataforma no mercado em menos de quatro semanas. Dentro de seis meses, escalamos para 40.000 usuários sem executar um único servidor.”

— **A Cloud Guru**

“Podemos ter um commit em produção em literalmente minutos — bem como fornecer um monte de opções de roteamento flexíveis dinamicamente.”

— **Pinpoint**

Elasticidade

“Usando aplicações serverless baseados no Lambda, Resnap pode executar modelos de machine learning em uma média de 600 fotos, o que resulta em milhares de invocações e ainda gera um livro de fotos em um minuto.”

— **Resnap**

“Nossas abordagens serverless nos permitem veicular anúncios para públicos-alvo 60% mais rápido do que com abordagens baseadas em instâncias.”

— **Infinia Mobile**

Eficiência de custo

“Nossos custos caíram em mais de 25% e nosso tempo médio mensal para concluir o processamento de dados caiu para 7 segundos, tornando o processo mais de 99% mais rápido.”

— **Speed Shift Media**

“Usando o AWS Lambda e AWS Step Functions, reduzimos os tempos de integração do cliente de 20 minutos para 30 segundos e seus custos esperados são de 20 USD por 10.000 pedidos.”

— **Mercury**

AWS Serverless Application Repository

Discover, deploy, and publish serverless applications



alex

< 1 2 3 4

alex-a-anagram

Alexa responds with the count and anagrams for a requested word

evanchiu

3 deployments

anagram alexa nodejs

alex-a-random-restaurant

A basic python based back-end for an Alexa skill that randomly gives you an open restaurant in a specified city using the Yelp API.

Harsha Warrdhan Shar... 3 deployme...

alex-a-smart-home-skill-adapter

Provides the basic framework for a skill adapter for a smart home skill.

AWS

6 deployments

alex-a-skills-kit-color-expert

Demonstrates a basic skill built with the Amazon Alexa Skills Kit.

AWS

26 deployments

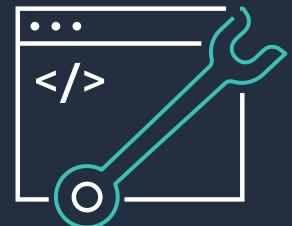
alex-a-skill-kit-sdk-factskill

alex-a-skill-kit-sdk-howtostill

alex-a-skill-kit-sdk-triviaskill

alex-a-skill-kit-color-expert

Aprenda a criar aplicações modernas na AWS



Habilite a rápida inovação desenvolvendo suas habilidades em projetar, criar e gerenciar aplicações modernas



Aprenda a modernizar suas aplicações com ofertas gratuitas de treinamento digital, incluindo Architecting on AWS, Developing on AWS e DevOps Engineering on AWS



Valide sua experiência com as certificações AWS Certified DevOps – Professional ou AWS Certified Developer – Associate exams

Visite a trilha de aprendizado para desenvolvedores em aws.amazon.com/training/path-developing

Obrigado!

Fernando Sapata

LatAm Business Development Manager, **Serverless**

