SERVERLESS WEEK

SERVERLESS WEEK 2020

A escalabilidade ideal

Roberto Alves

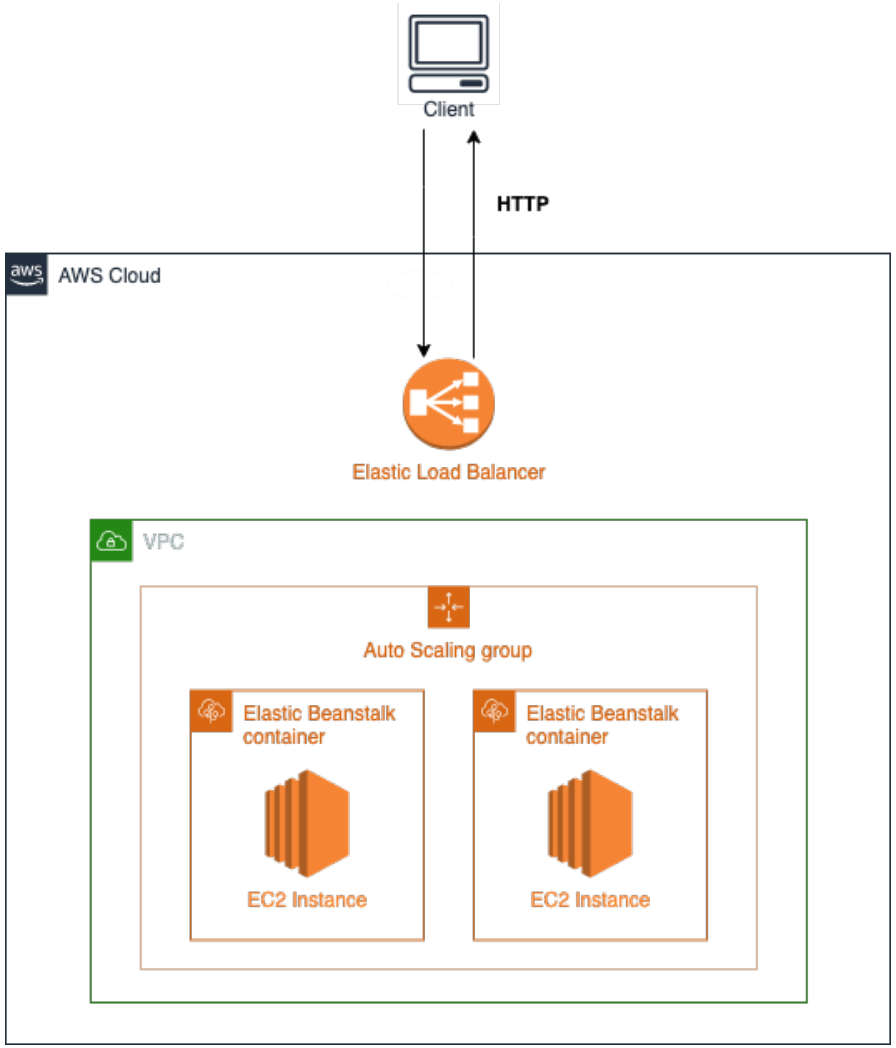
Lead Software Engineer @ Altran

Escalabilidade em software

Habilidade de atender novas demandas de processamento, sem que isso afete, de forma abrupta a saúde do seu sofware.

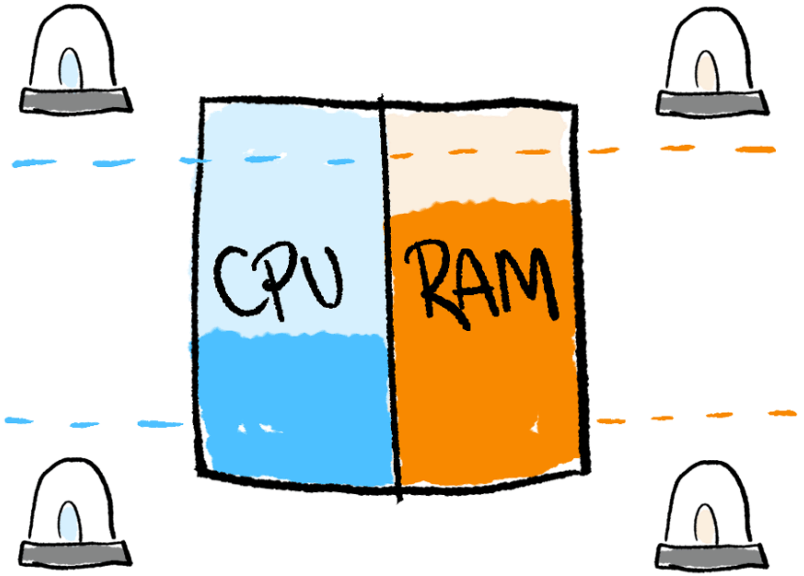
A arquitetura inicial

O nosso caso de uso



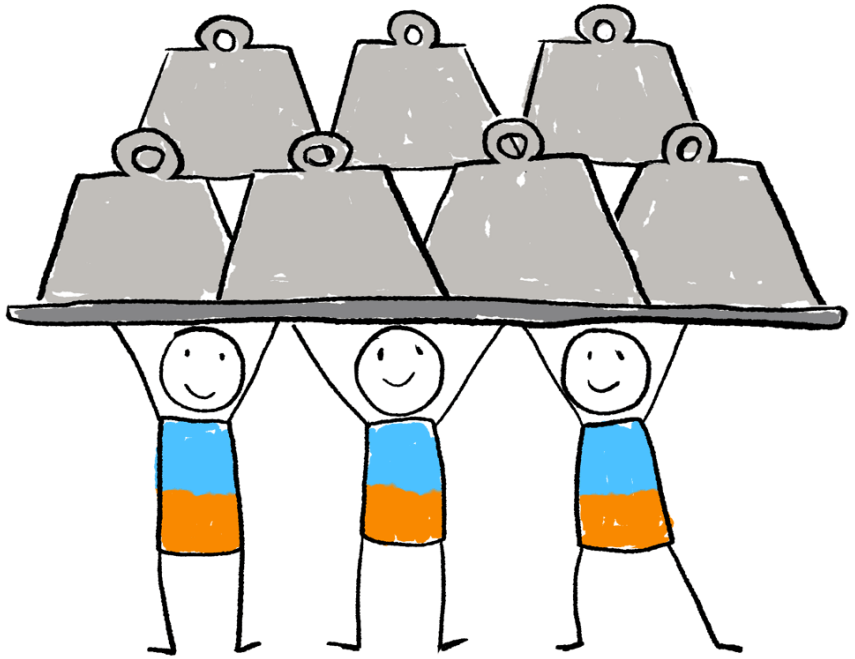
O nosso problema

O tempo de resposta para o AWS EC2 Auto Scaling subir um novo contêiner



Em media, 2.5 minutos

Na documentação oficial do AWS EC2 Auto Scaling diz “A maioria das substituições acontece em menos de 5 minutos”.



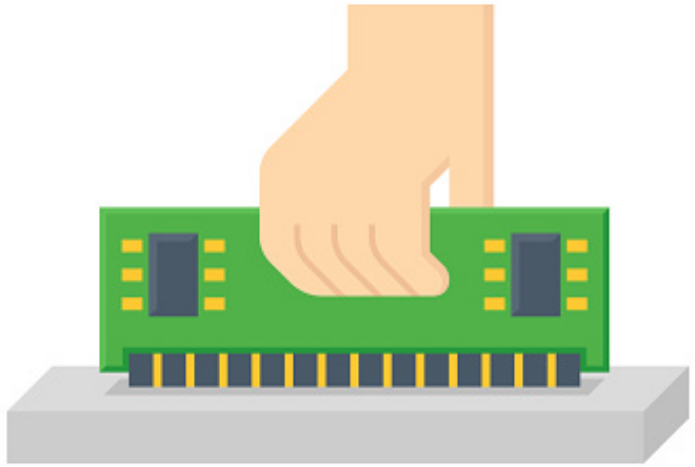
Precisa de tanto?

- Arquitetura distribuída entre microsserviços
- Aplicação com uso 24x7 com acessos internacional
- Streaming de vídeo
- Em média, possui 12 integrações no fluxo convencional
- Composto por aplicações hospedadas em cloud público e cloud privado

Uma solução paliativa

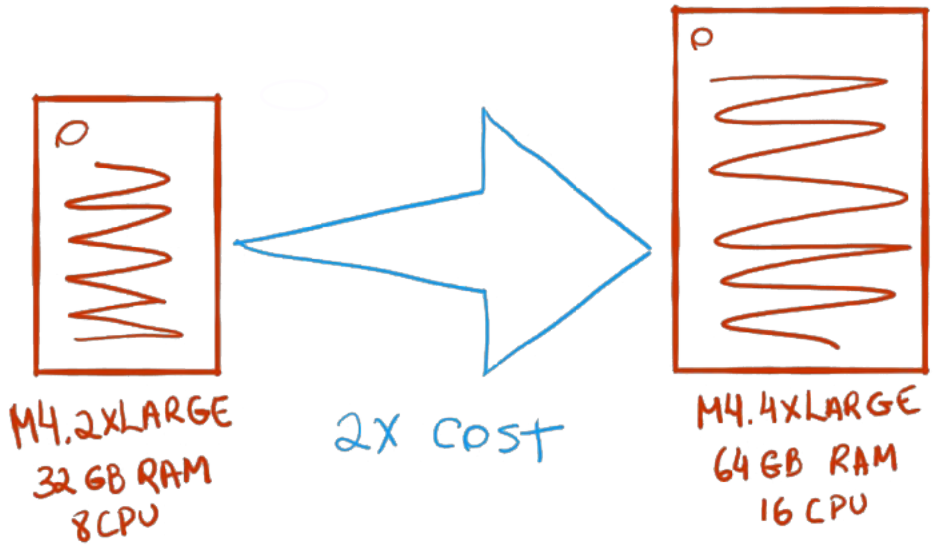
Uma adaptação rápido e simples,

escalabilidade vertical



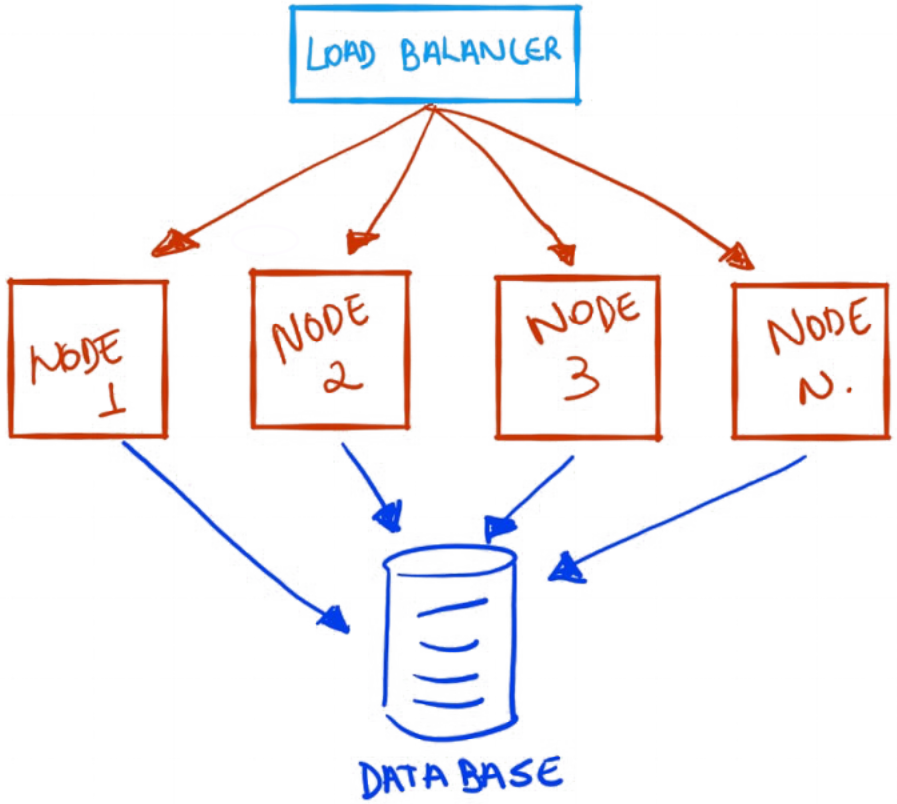
Escalabilidade vertical

Adiciona mais recurso computacional ao hardware do servidor



Escalabilidade horizontal

Adiciona novos servidores para dividir o processamento



Escalabilidade vertical

Por que não mantermos permanentemente essa solução?

- Na maior parte do tempo não usamos todo esse poder computacional

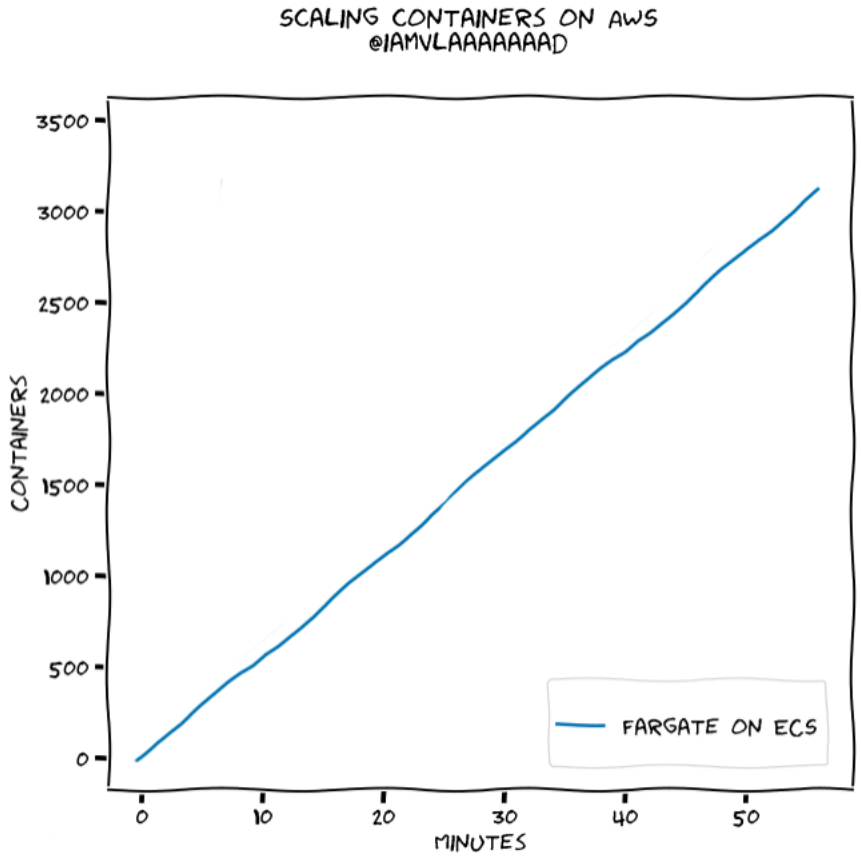
- Custo “elevado”, de forma desnecessária, no cloud

A solução em definitivo

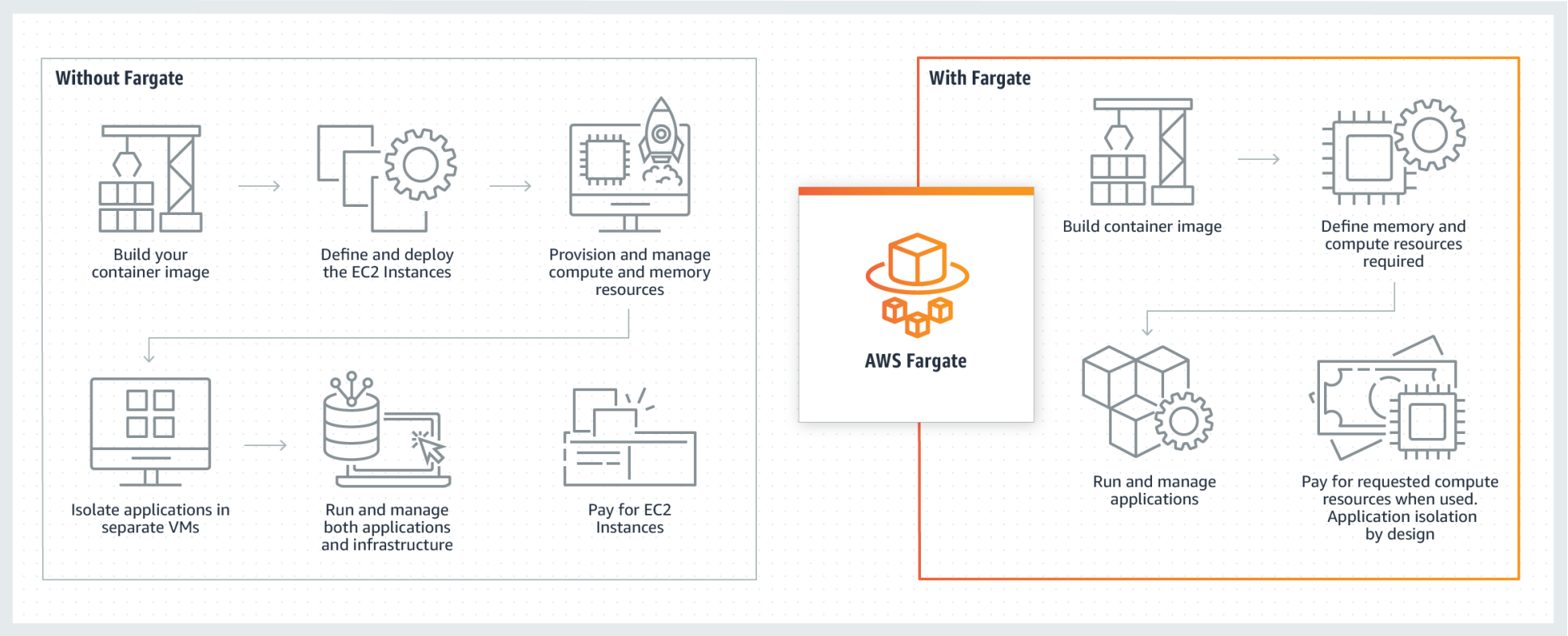
A melhor forma que encontramos foi mudar o modelo da aplicação para *serverless containers*

60 contêineres por minuto

Em nossos testes, obtivemos uma média de 40 segundos para levantar 2 contêineres

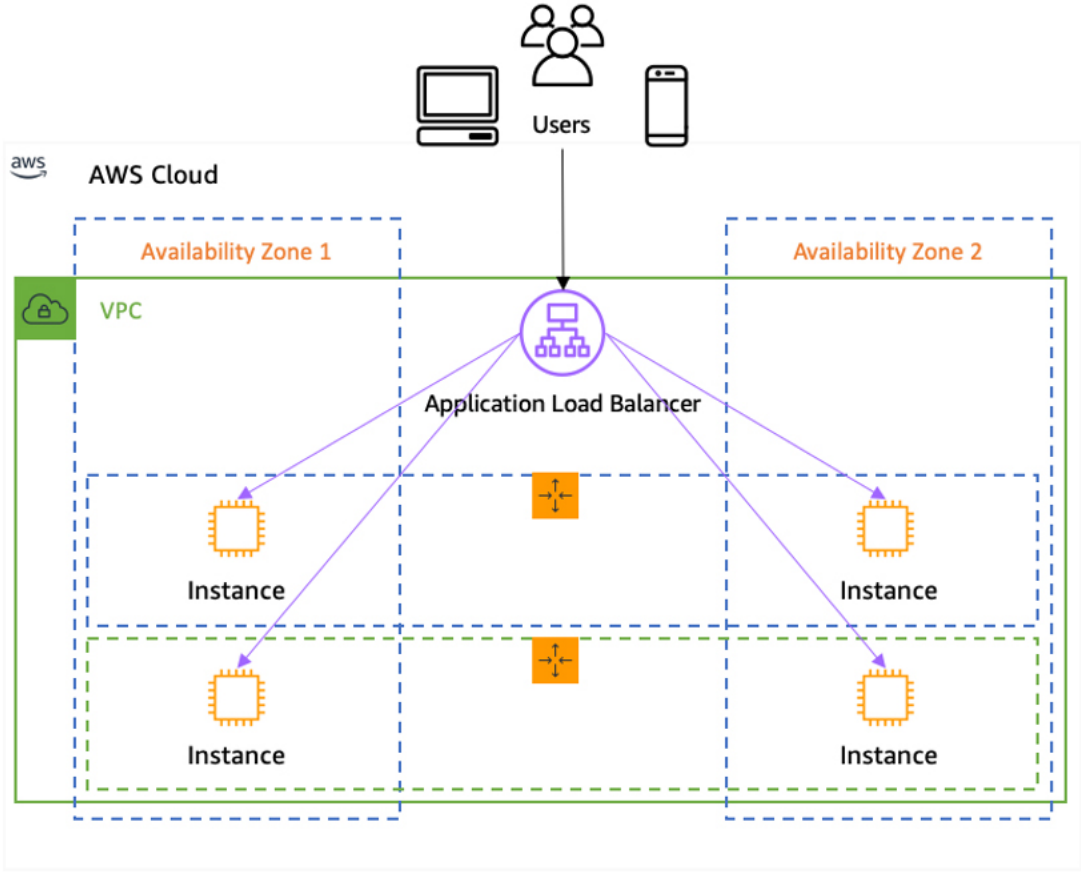


AWS Fargate


Without Fargate
Build your container image
Define and deploy the EC2 Instances
Provision and manage compute and memory resources
Isolate applications in separate VMs
Run and manage both applications and infrastructure
Pay for EC2 Instances
With Fargate
AWS Fargate
Build container image
Define memory and compute resources required
Run and manage applications
Pay for requested compute resources when used. Application isolation by design

A arquitetura atualmente

O nosso caso de uso



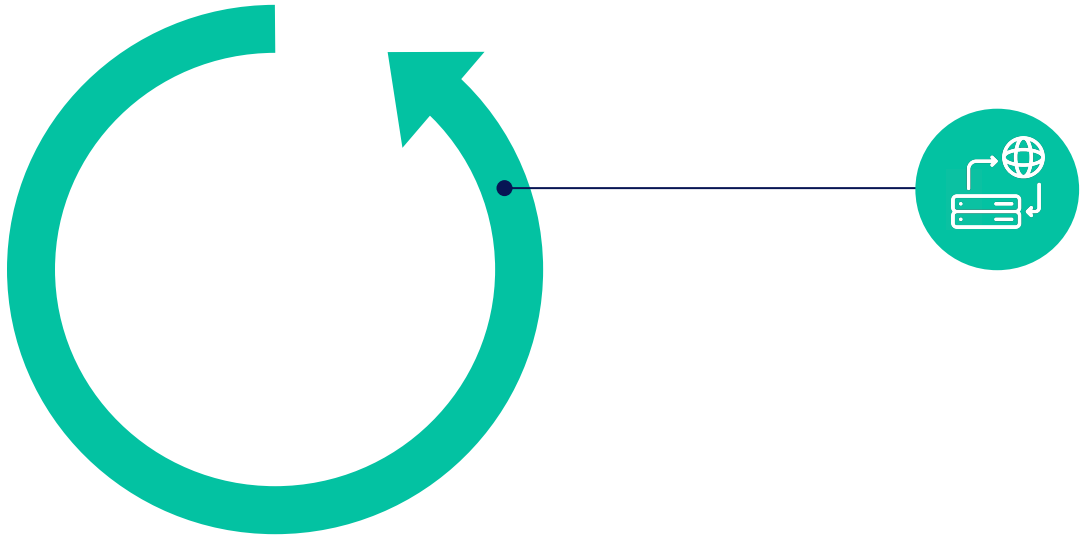
Métricas

Estado original

50 usuários concorrentes:
274 execuções
0% de falha
5s de latência média

100 usuários concorrentes:
593 execuções
2% de erro
9s de latência média

150 usuários concorrentes:
923 execuções
12% de erro
12s de latência média



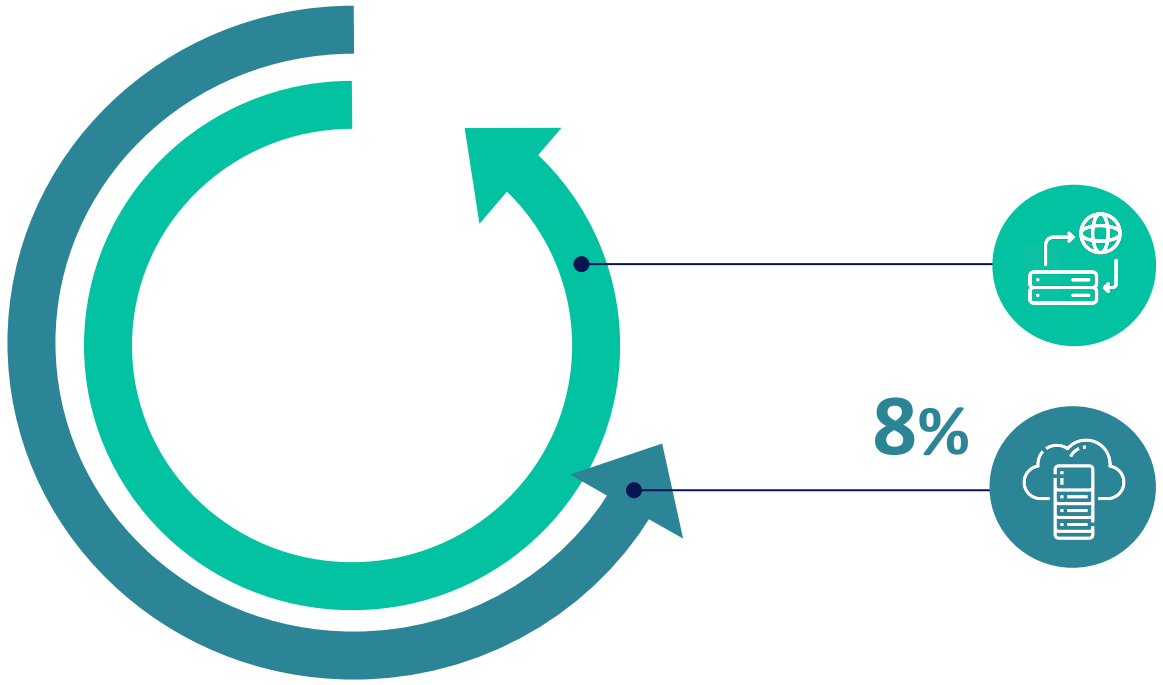
Métricas

Upgrade vertical

50 usuários concorrentes:
311 execuções
0% de falha
4s de latência média

100 usuários concorrentes:
668 execuções
1% de falha
8s de latência média

150 usuários concorrentes:
981 execuções
7% de falha
11s de latência média



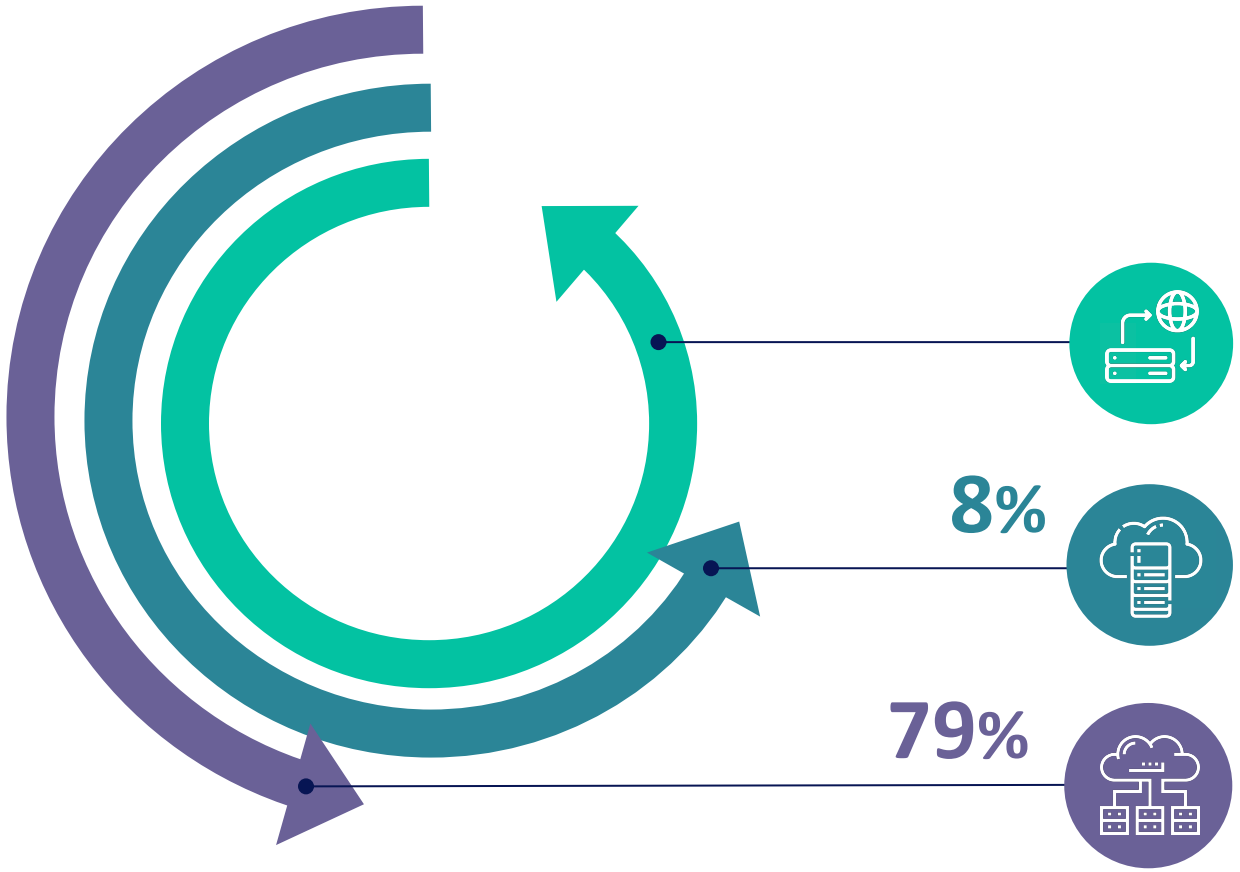
Métricas

Modelo Fargate

50 usuários concorrentes:
344 execuções
0% de falha
2s de latência média

100 usuários concorrentes:
712 execuções
0% de falha
2s de latência média

150 usuários concorrentes:
1253 execuções
0% de falha
2.5s de latência média



Precificação

Upgrade vertical: $4.900,00

Modelo Fargate: $3.500,00

Problemas nessa trajetória

- Modelo Fargate com contêineres com link

- Alta escalabilidade na aplicação com modelo serverless porém o banco de dados continua tradicional

Contato

- robertosousa1@uol.com.br
- https://www.linkedin.com/in/robertosousa01
- https://github.com/robertosousa1


SERVERLESS
WEEK