

# Descomplicando a Cloud com o Delphi

{ Palestrante

Daniel Fernandes

Embarcadero Conference 2023



# Sobre a palestra

- Monolitos
- Microsserviços
- O que são as FaaS (Function as a Service)
- Arquitetura
- Google Cloud Platform (GCP)
- Docker
- Parte Pratica
- Contatos
- Duvidas

# Sobre mim

- Nome: Daniel Fernandes
- Desenvolvedor de Software Sênior na AquaSoft.
- Mais de 7 anos de experiência em Delphi
- Graduado em Gestão da Tecnologia da Informação (FATEC Bragança Paulista)
- Pós-graduado em Engenharia de Software (Estácio)
- Criador do @DinosDev no Instagram
- Desenvolvedor do componente DinosOffices
- Palestrante na DelphiCon2023 e Embarcadero Conference 2022/2023



# MONÓLITOS

- **O que é um monólito ?**

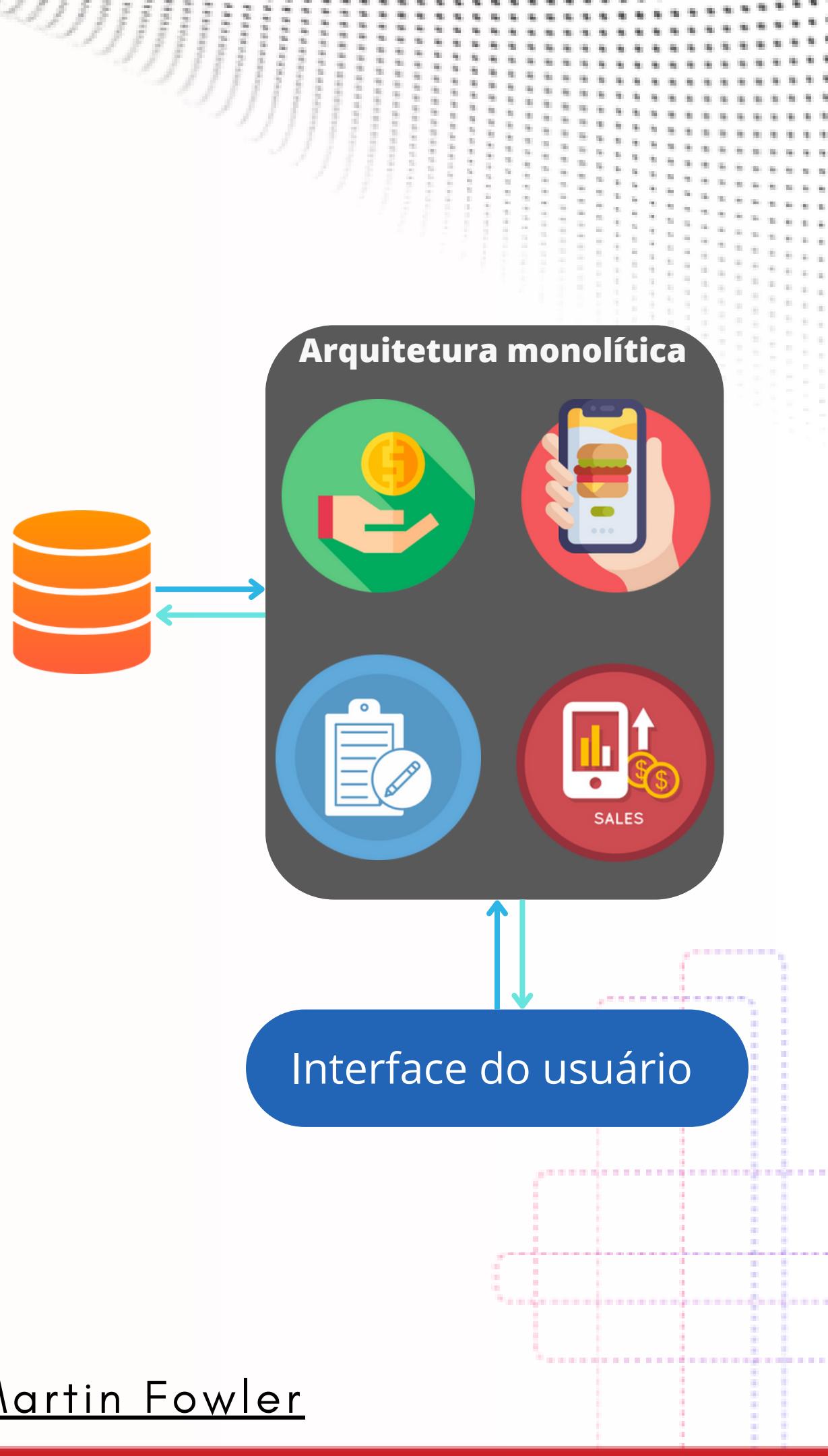
Um monólito é uma arquitetura de software em que todos os componentes e funcionalidades estão integrados em uma única aplicação.



# MONÓLITOS

## • Vantagens

- Simplicidade inicial na arquitetura
- Bom para criar um MVP's pois é fácil e rápido de desenvolver
- Coesão da equipe de Devs
- É simples o deploy para produção
- Fácil de debugar

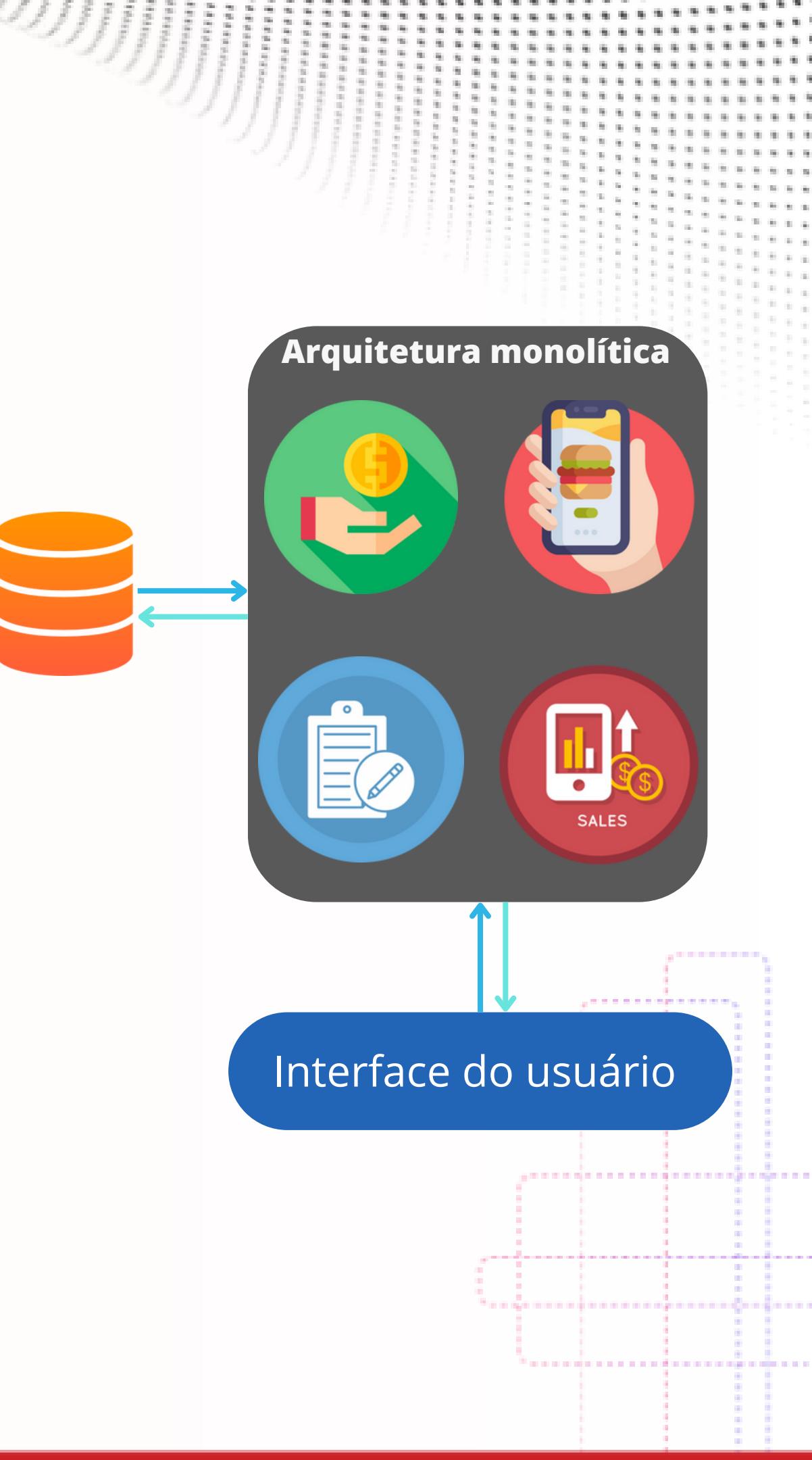


Monolith First - Martin Fowler

# MONÓLITOS

## • Desafios

- Tudo esta em um único lugar
- Alta dependência
- Demorar para um novo Dev se habituar
- O Problema em uma parte do software, afeta ele como todo
- Baixa Escalabilidade
- Base de dados gigante
- Uma única tecnologia



# MICROSERVIÇOS

- **O que é um Microsserviços ?**

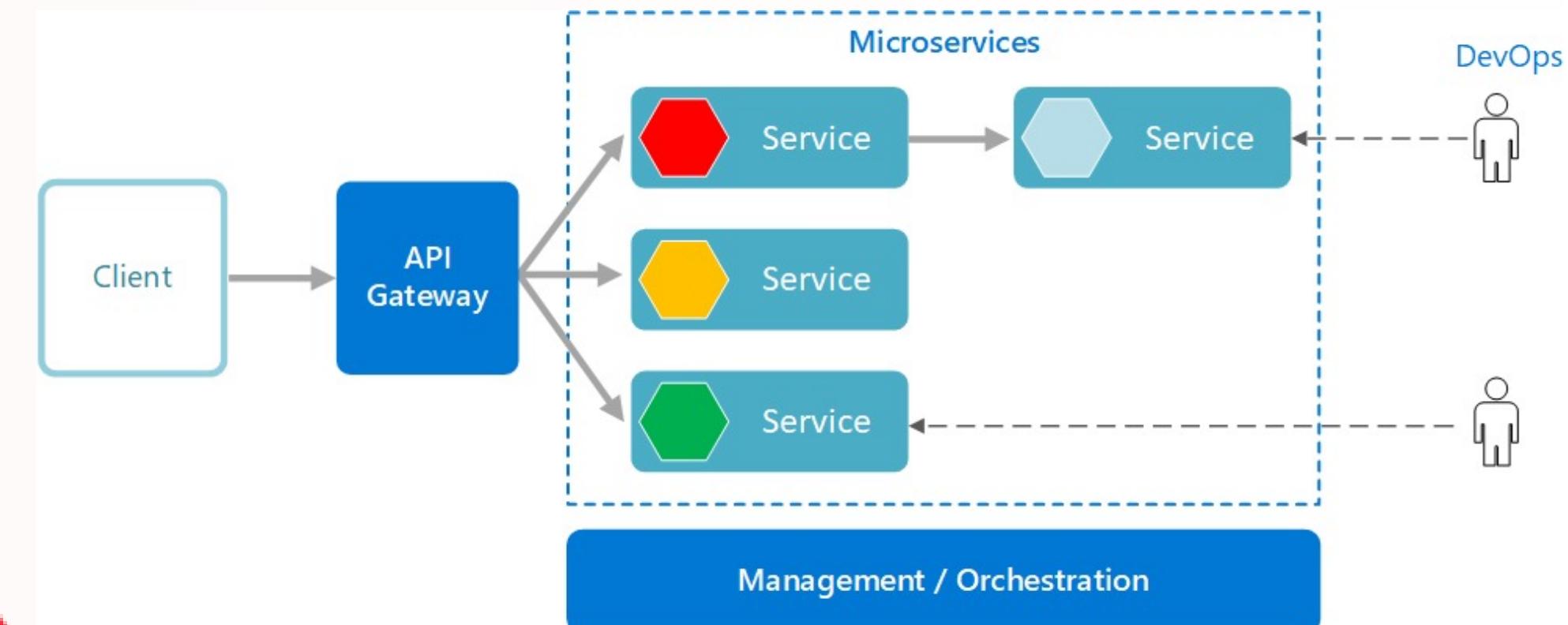
Microserviços são uma abordagem arquitetônica que desenha um software como um conjunto de pequenos serviços independentes, cada um executando uma única funcionalidade específica. Cada serviço opera como uma entidade autônoma e se comunica com os outros por meio de APIs, criando uma aplicação modular e escalável.



# MICROSERVIÇOS

## • Quando utilizar ?

- Em aplicações grandes e complexas que precisa ser altamente escaláveis e dimensionáveis.
- Aplicações que possui muitos domínios e subdomínios.
- Necessidade de integração/implementação continuar.

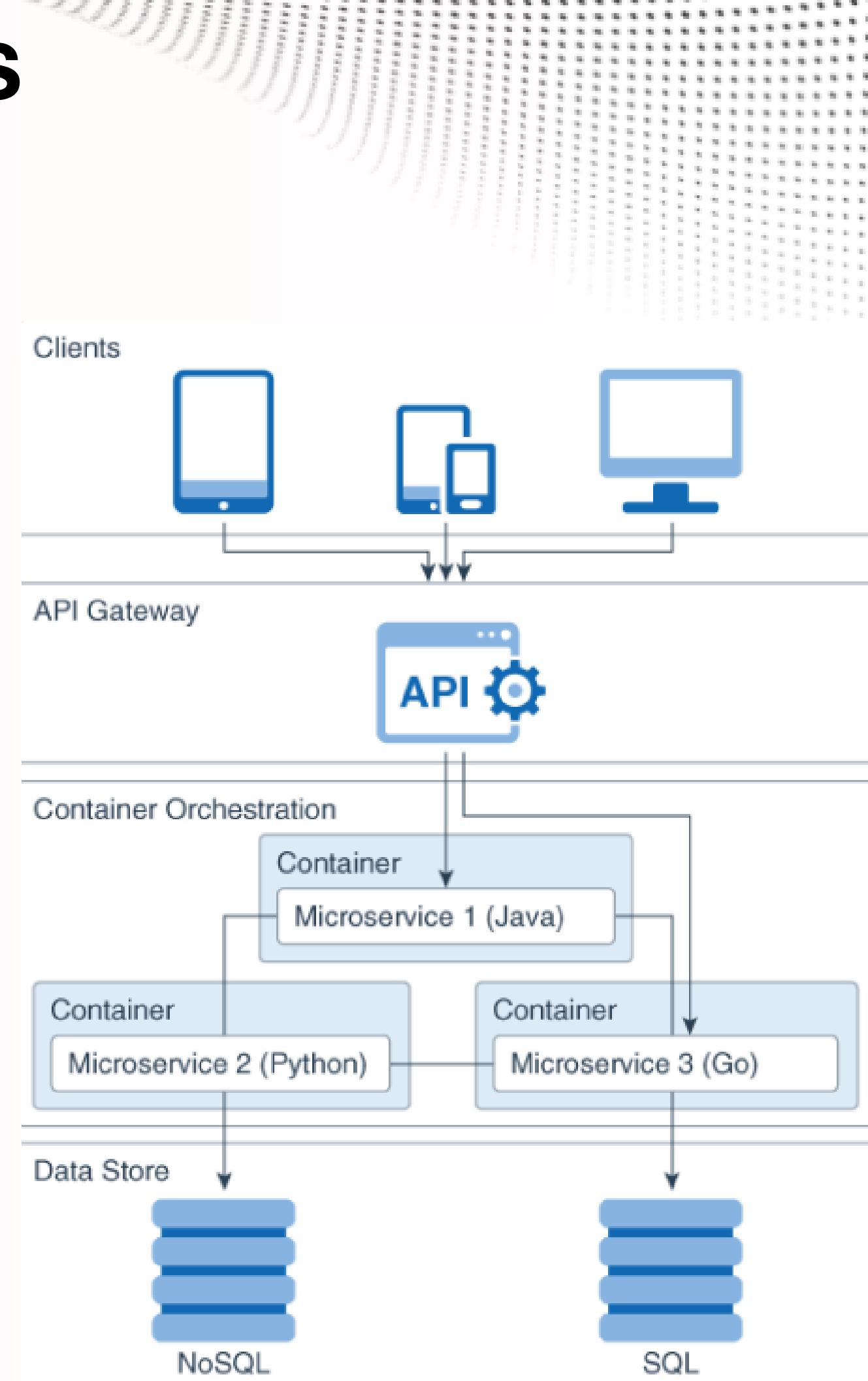


"Não use uma basuca para matar uma formiga"

# MICROSERVIÇOS

## • Características

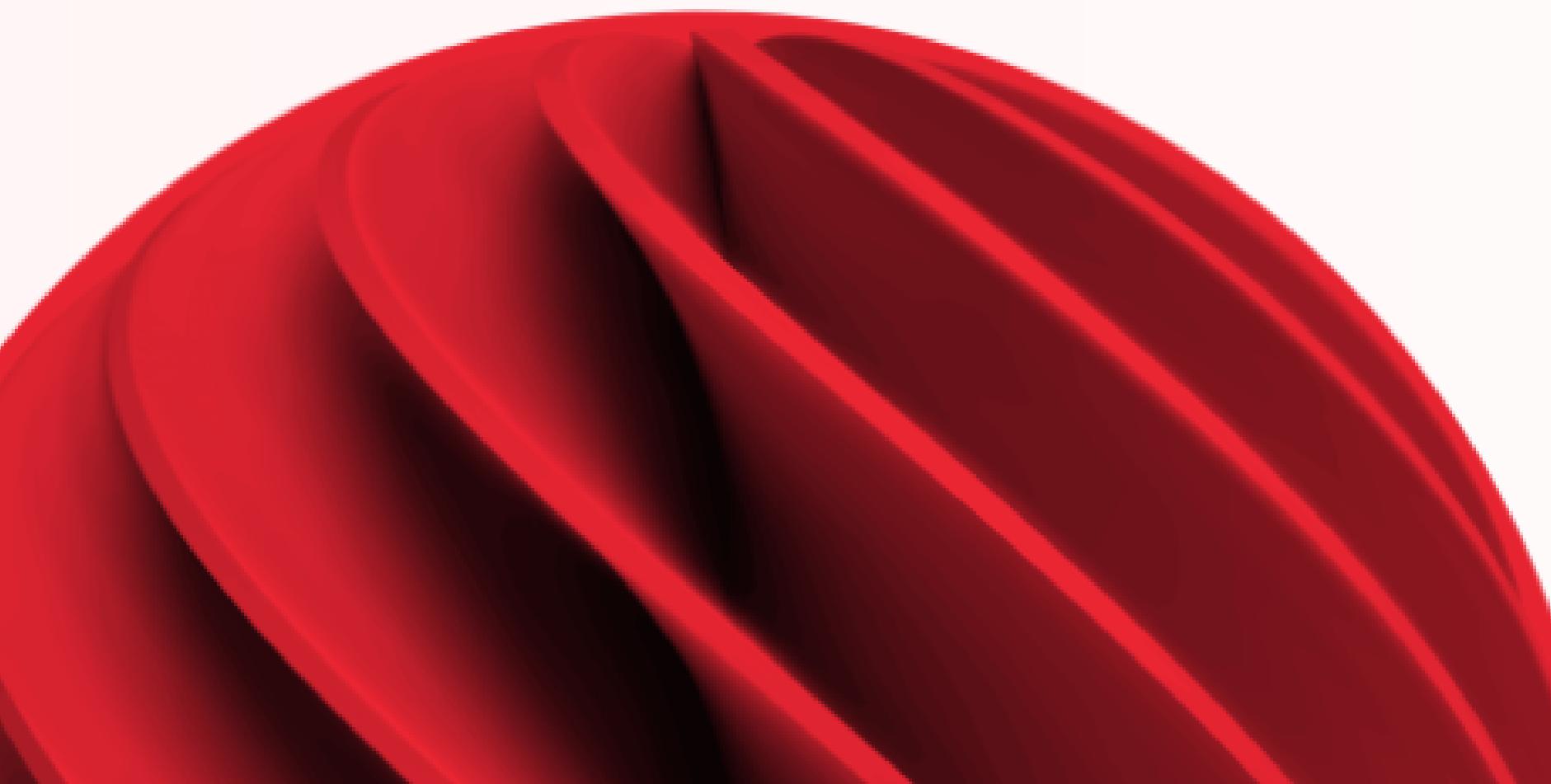
- Dividir as tarefas de formas independentes.
- Um serviço deve implementar apenas uma função.
- Troca de serviços de maneira ágil.
- A comunicação é feita por APIs HTTP bem modelados.
- É construído através de pequenas responsabilidades
- Deploys automatizados.
- Pode-se usar varias linguagens diferentes.



# MICROSERVIÇOS

## • Vantagens

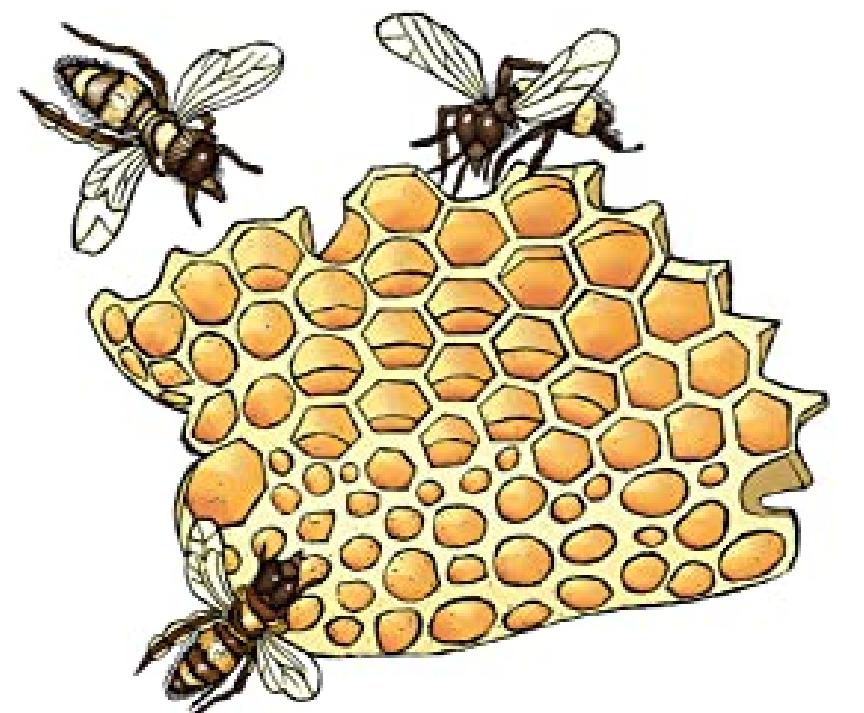
- Escalabilidade Eficiente:
- Facilidade de Manutenção
- Tecnologia Diversificada
- Resiliência e Tolerância a Falhas
- Desenvolvimento Paralelo
- Facilita a Escalabilidade Horizontal



O'REILLY®

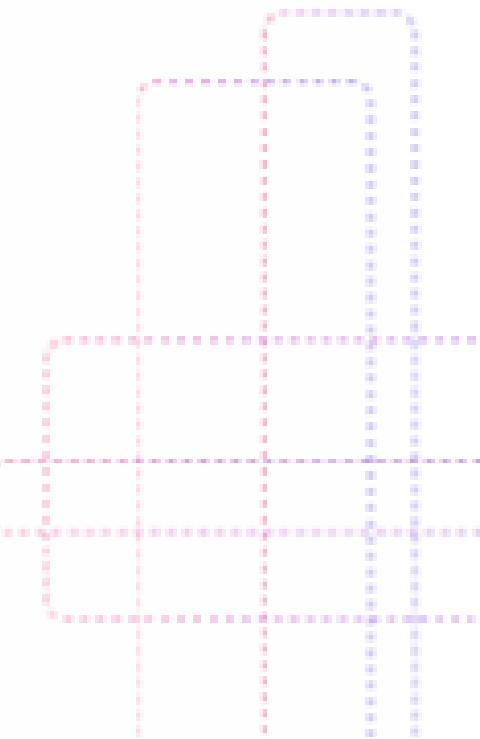
## Criando Microsserviços

Projetando sistemas com componentes  
menores e mais especializados



novatec

Sam Newman



# MICROSERVIÇOS

## • Desafios

- Complexidade
- Testes e Refatoração são mais difíceis.
- Aumento do custo para resolver bugs.
- Latência, pois a comunicação é feita por rede.
- Gerenciar dados armazenados ao longo prazo, pois são tratados separadamente.
- Nem todos os aplicativos são grandes o suficiente para serem divididos em microserviços

Arquitetura monolítica

# MICROSERVIÇOS

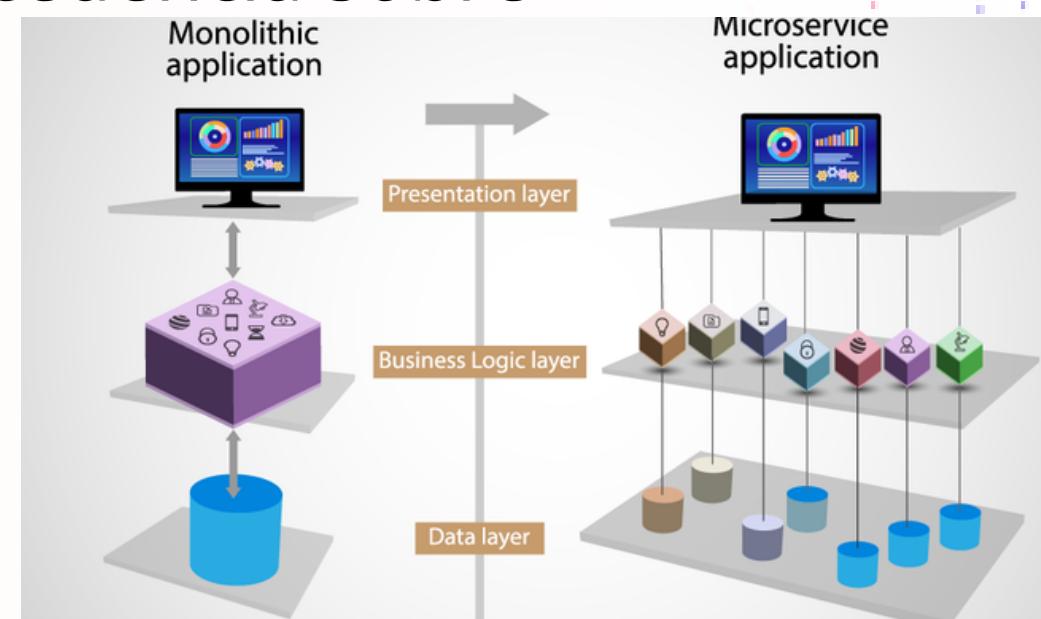
- **Boas práticas**

- Cada serviço em uma arquitetura de microserviço tem seu próprio banco de dados.
- Utilizar de preferência comunicação assíncrona.
- Comunicação entre microserviços deverá ser realizada sempre via exposição de API REST
- Os serviços que cuidam da base de dados devem ser privado.
- Evite acoplamento.
- APIGateway cuida da centralização da autenticação, rotas, balanceamentos de carga, limitação de taxas e até Logs.
- Evite falhas em cascatas.

# MICROSERVIÇOS

## • Como migrar seu monolito para microserviço?

- Comece extraindo um serviço do aplicativo monolítico e depois desenvolva, teste e implante-o na produção.
- Evite refatorar tudo de uma só vez, para priorizar o desacoplamento de serviços, avalie o custo-benefício.
- Os serviços em uma arquitetura de microsserviços são organizados em torno de preocupações de negócios.
- Ao migrar serviços gradualmente, configure a comunicação entre serviços e monolíticos para passar por contratos de API bem definidos. Pode-se utilizar neste momento o patern share database. e depois desacoplar os bancos.
- Os microserviços exigem muito mais automação: pense com antecedência sobre integração contínua



# FAAS

- **O que é FaaS? (Funções como serviço)**

- A FaaS é um serviço de back-end sem servidor que permite aos desenvolvedores escrever peças modulares de código em tempo real que podem ser executadas em resposta a determinados eventos.

- **Vantagens**

- **Melhoria da velocidade do desenvolvedor:** Com a FaaS, os desenvolvedores gastam mais tempo escrevendo a lógica do que se preocupando com servidores e implantações.
- **Escalabilidade embutida:** Como o código FaaS, os desenvolvedores não precisam se preocupar em criar contingências para alto tráfego e uso pesado. O provedor sem servidor lidará com todas as preocupações de dimensionamento.
- **Eficiência de custo:** Os provedores de FaaS sem servidor não cobram de seus clientes pelo tempo de computação ocioso.



- **Desafios**

- **Menos controle do sistema:** Ter um terceiro para gerenciar parte da infraestrutura dificulta a compreensão de todo o sistema e adiciona desafios de depuração.
- **Mais complexidade necessária para os testes:** Pode ser muito difícil incorporar o código FaaS em um ambiente de teste local, tornando o teste completo de um aplicativo uma tarefa mais intensiva.

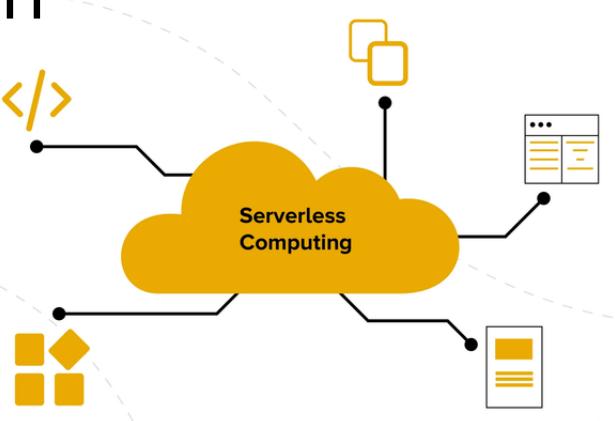
- **Conceito de serverless**

As aplicações serverless são implantadas em containers que são iniciados sob demanda e automaticamente quando chamados. São serviços sem servidores.

FaaS é um Servless

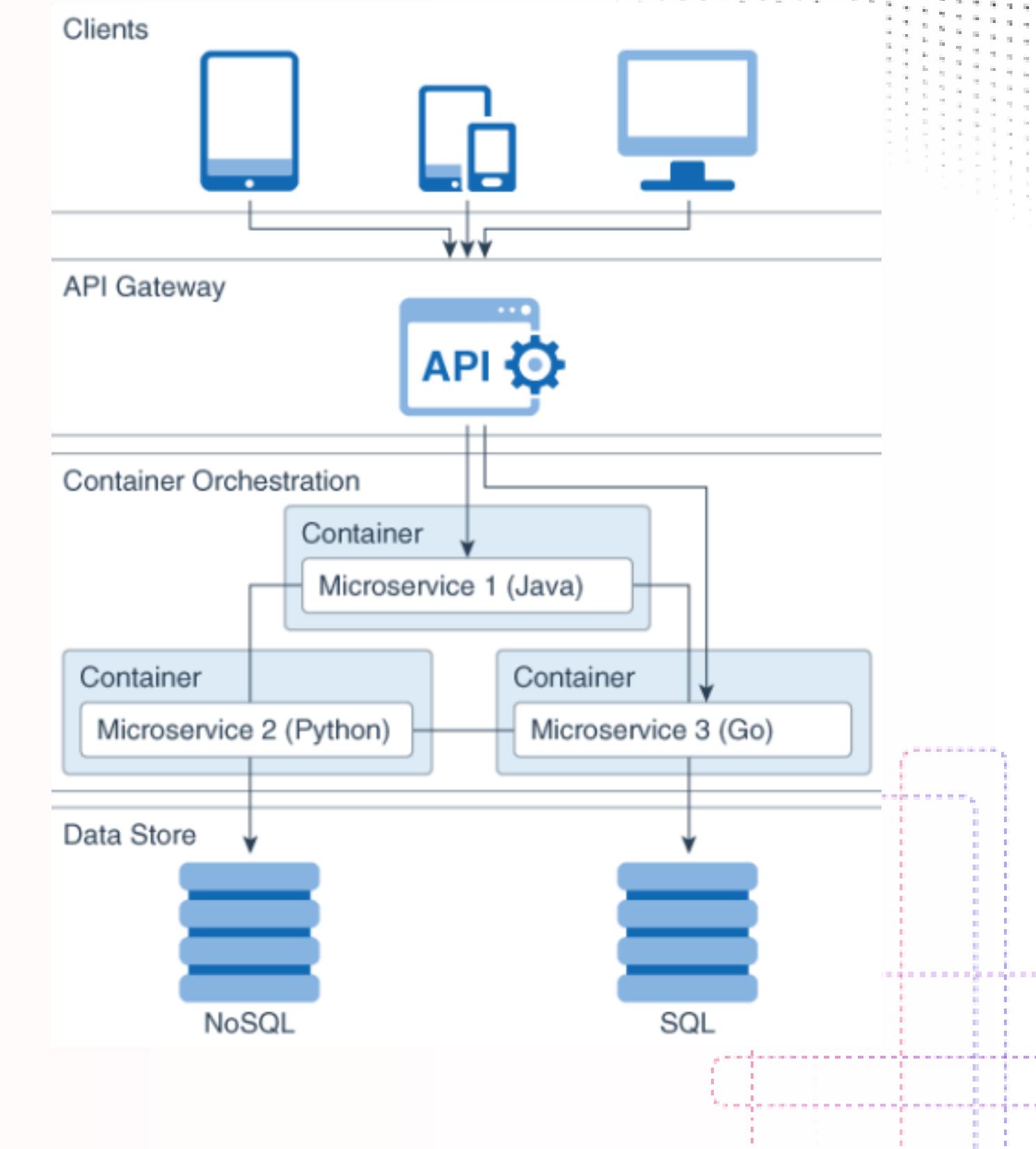
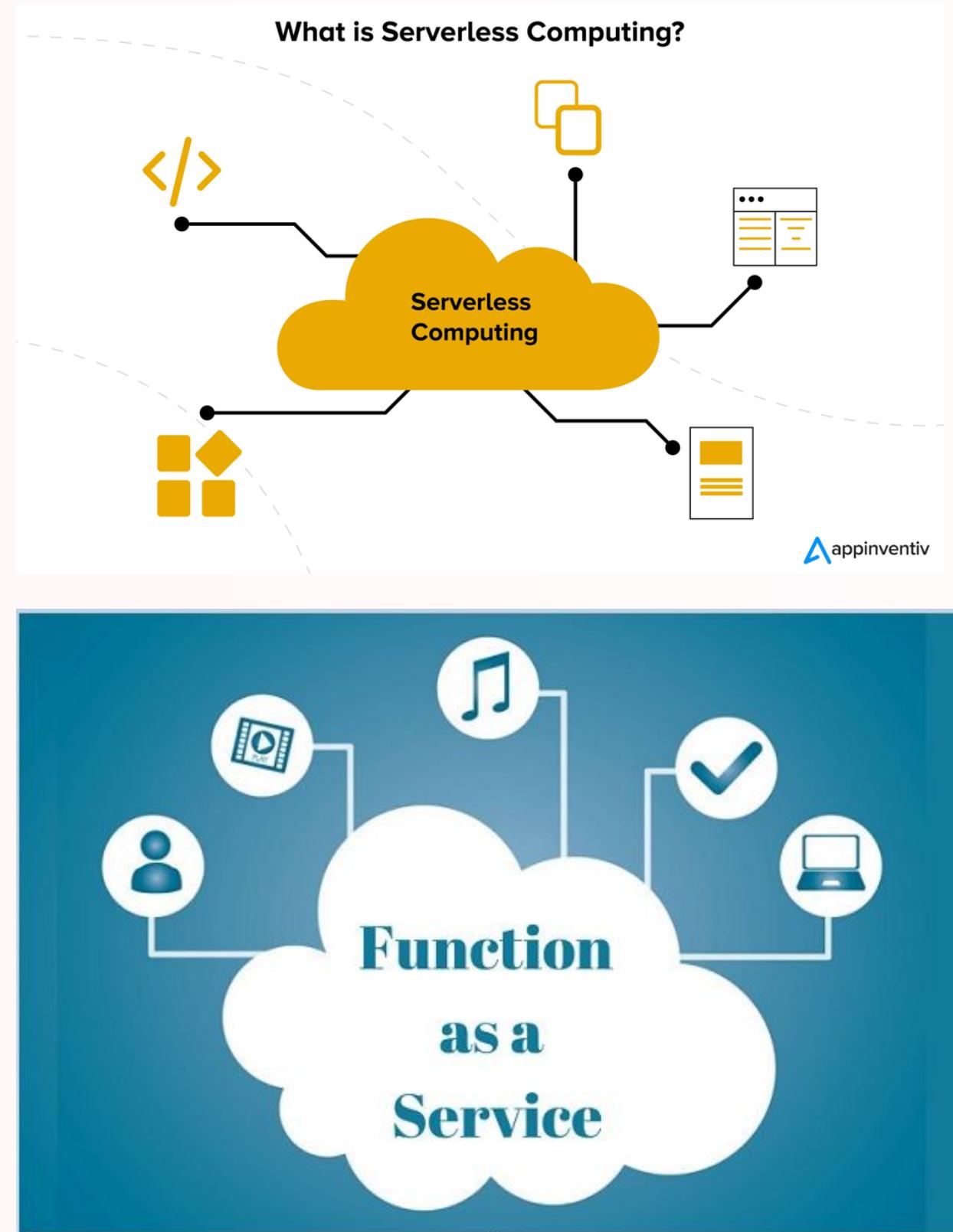
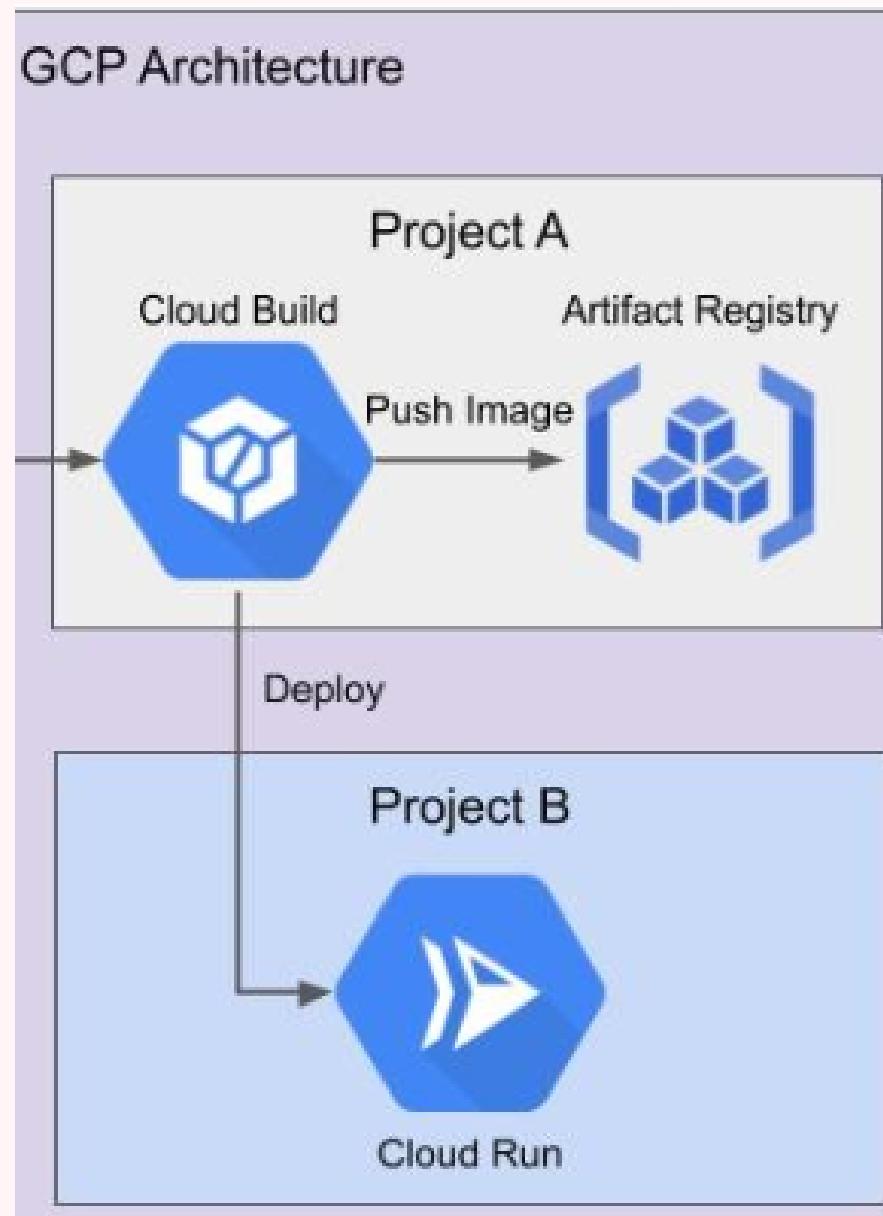
Arquitetura monolítica

What is Serverless Computing?



Appinventiv

# ARQUITETURA GCP



# GOOGLE CLOUD PLATAFORM (GCP)

- **Mas o que é a Google Cloud ?**

- O Google Cloud consiste em um conjunto de recursos físicos (computadores e unidades de disco rígido) e recursos virtuais, como as (VMs), localizados nos data centers do Google por todo o mundo. Cada local do data center está em *regiões* isoladas entre si.

- **Google Cloud Platform**

- O Google Cloud Platform(GCP), é um provedor de recursos de computação em nuvem, utilizado para desenvolver, implantar e operar aplicativos na web.
- A plataforma do Google é uma infraestrutura completa, com diversas funcionalidades e ferramentas.

## Comandos basicos do shell

- **Criar o repositório Docker no Artifacts**

```
gcloud artifacts repositories create docker-repo --repository-format=docker \
--location=us-west2 --description="Docker repository"
```

- **Construindo o projeto**

```
gcloud builds submit --config cloudbuild.yaml
```

# (GCP) - BIBLIOTECAS

- **App Engine** 

- O App Engine serve para a criação de aplicativos com alto poder de escala em uma plataforma totalmente gerenciada e sem servidor.

- **Cloud Build API** 

- O Cloud Build API é responsável por criar e gerenciar os builds da aplicação, sendo homologação ou produção.

- **Artifact Registry API** 

- Armazenamento e gerencia de artefatos de build em um serviço escalonável e integrado baseado na infraestrutura do Google.

- **Cloud Run Admin API** 

- O Cloud Run é uma plataforma para a execução de contêineres diretamente na infraestrutura escalonável do Google.

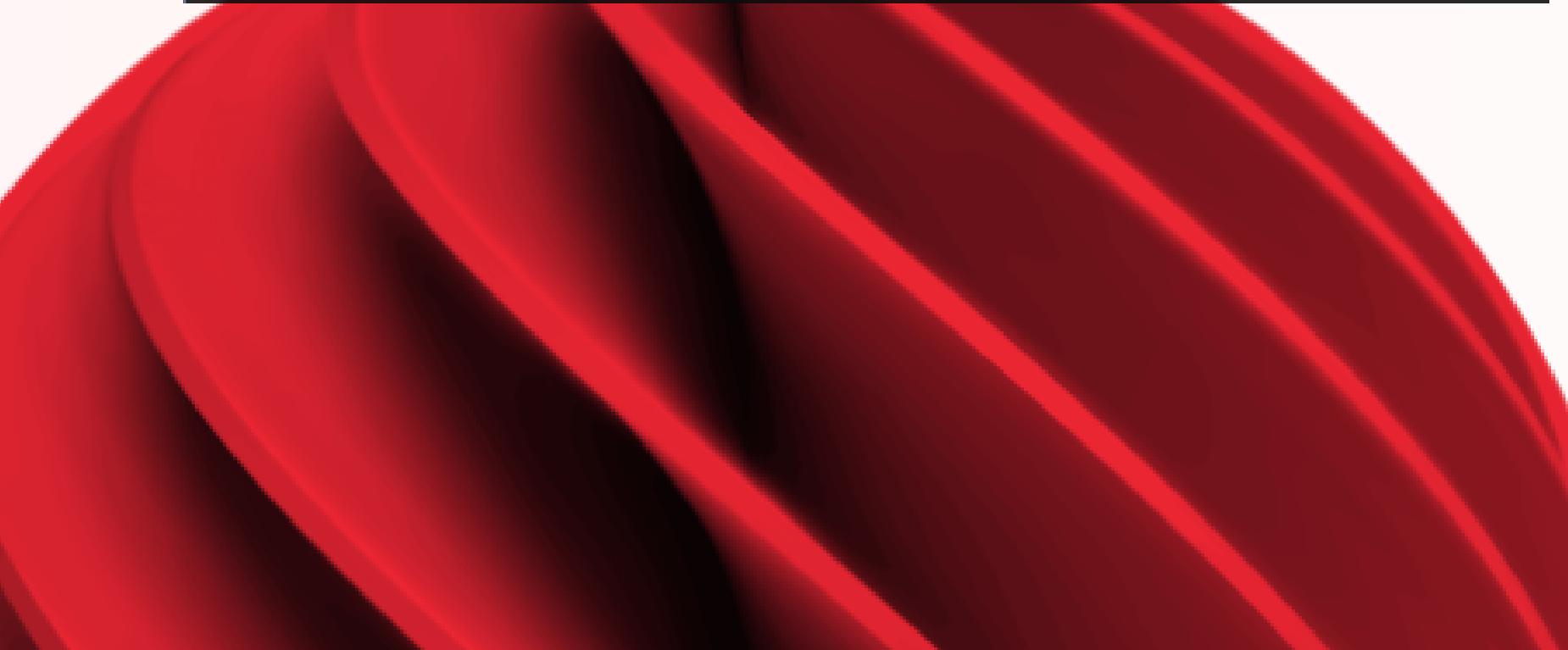
- **IAM & ADMIN** 

- O IAM permite conceder acesso granular a recursos específicos GCP, também permite que você adote o princípio de segurança onde ninguém precisa ter mais permissões do que o realmente necessário.

# DOCKER

## • O que é o Docker

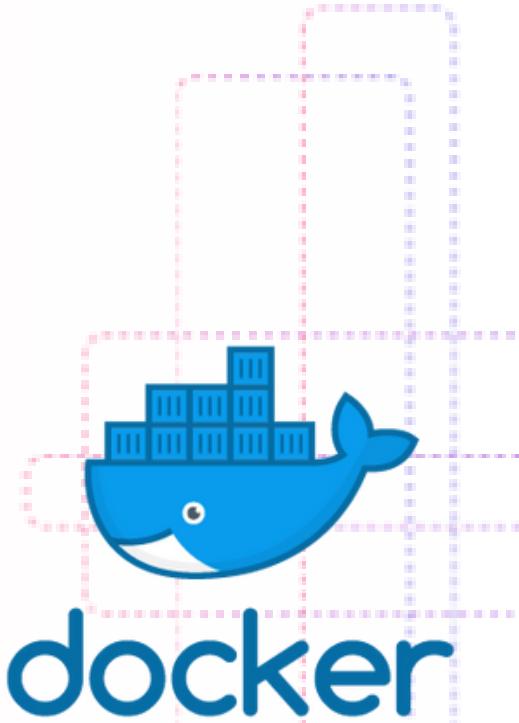
- O Docker é um software de código aberto usado para implantar aplicativos dentro de containers virtuais. A containerização permite que vários aplicativos funcionem em diferentes ambientes complexos.
- Ele roda as imagens dos sistemas operacionais Windows, Linux e macOS



```
daniel_94@AquaSoftDev:/mnt/c/Users/t_daniel.rodrigues$ sudo docker images -a
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
delphi-websocket-image  latest   a0566f885095  11 days ago  279MB
daniel_94@AquaSoftDev:/mnt/c/Users/t_daniel.rodrigues$ |
```

### Comandos básicos docker

- **sudo docker build -t delphi-websocket-image .**
- **sudo docker run -p 8080:8080 delphi-websocket-image**
- **sudo docker stop delphi-websocket-image**
- **sudo docker start delphi-websocket-image**
- **sudo docker container list -a**
- **sudo docker images -a**
- **sudo docker rmi d40dfe174c77 -f**
- **sudo docker rm d09a37314461 -f**

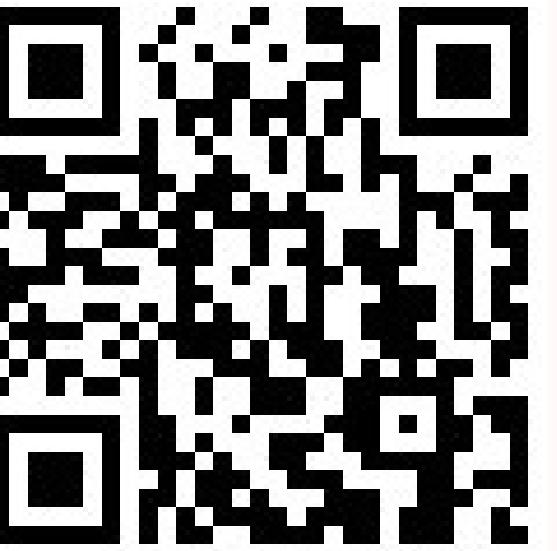


# VAMOS VER TUDO ISSO NA PRATICA?

Talk is cheap. Show me the code.

— Linus Torvalds





## Daniel Fernandes

Desenvolvedor Sênior na Aquasoft. Criador do componente DinosOffice – LibreOffice for Delphi, Criador de conteúdo para a comunidade no instagram @dinosdev

[daniel.rodrigues@aquasoft.com.br](mailto:daniel.rodrigues@aquasoft.com.br)



# Embarcadero Conference 2023

O que você achou da palestra?

