

# Azure Kubernetes Service (AKS)



TFTEC CLOUD



# Introdução aos containers: conceitos e benefícios



# O que são containers?

Imagine um container como uma caixa compacta e autocontida, onde você pode empacotar uma aplicação e tudo o que ela precisa para funcionar – desde código e bibliotecas até configurações e arquivos de sistema. Essa 'caixa' garante que a aplicação funcione de maneira uniforme e eficiente em qualquer ambiente, seja em um servidor local ou na nuvem.

## **Isolamento e Eficiência:**

Cada container funciona isoladamente, o que significa que se um falhar ou for comprometido, os outros continuarão funcionando normalmente, assim como um problema em um apartamento não afeta os vizinhos. Essa característica traz uma camada extra de segurança e eficiência, especialmente importante para ambientes de desenvolvimento e produção em larga escala.

## **Flexibilidade e Portabilidade:**

Containers são incrivelmente portáteis. Você pode desenvolver uma aplicação em um container no seu laptop e movê-la sem problemas para um ambiente de produção na nuvem, tudo isso sem a necessidade de fazer ajustes ou modificações. Essa flexibilidade é um grande avanço em comparação com os métodos tradicionais de desenvolvimento e implantação de software.



**TFTEC CLOUD**

# Diferença entre Containers e Máquinas Virtuais (VMs)

A escolha entre containers e VMs depende das necessidades específicas do projeto.

Containers são ideais para aplicações que requerem eficiência, escalabilidade e portabilidade, enquanto VMs são mais adequadas para aplicações que necessitam de um isolamento completo e um ambiente operacional dedicado.

## Containers:

**Natureza Leve:** Containers são incrivelmente leves porque compartilham o mesmo sistema operacional do host. Eles virtualizam apenas o espaço de usuário, não o hardware, o que os torna muito mais eficientes em termos de uso de recursos.

**Rapidez de Implantação:** Devido ao seu tamanho reduzido, containers podem ser iniciados quase instantaneamente, proporcionando maior agilidade em ambientes de desenvolvimento e produção.

**Isolamento a Nível de Processo:** Containers isolam processos e aplicações, mas compartilham o mesmo kernel do sistema operacional, o que os torna ideais para microserviços e aplicações distribuídas.

## Máquinas Virtuais:

**Virtualização Total:** VMs virtualizam hardware inteiro, rodando um sistema operacional completo para cada instância. Isso oferece um isolamento mais completo, mas com um custo maior de recursos.

**Inicialização Mais Lenta:** Cada VM é um sistema operacional completo, o que significa que leva mais tempo para iniciar, em comparação com um container.

**Uso de Recursos:** VMs geralmente requerem mais recursos de memória e processamento, já que cada uma executa uma cópia completa do sistema operacional além da aplicação.

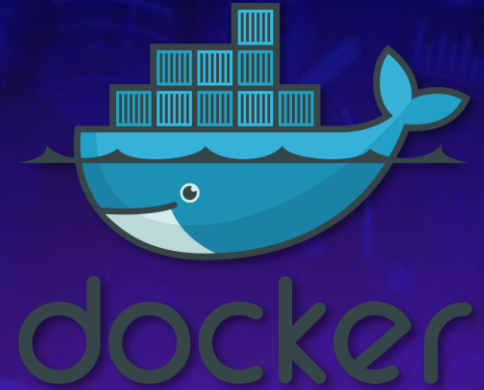


# O que é Docker, Dockerfile e Docker Hub

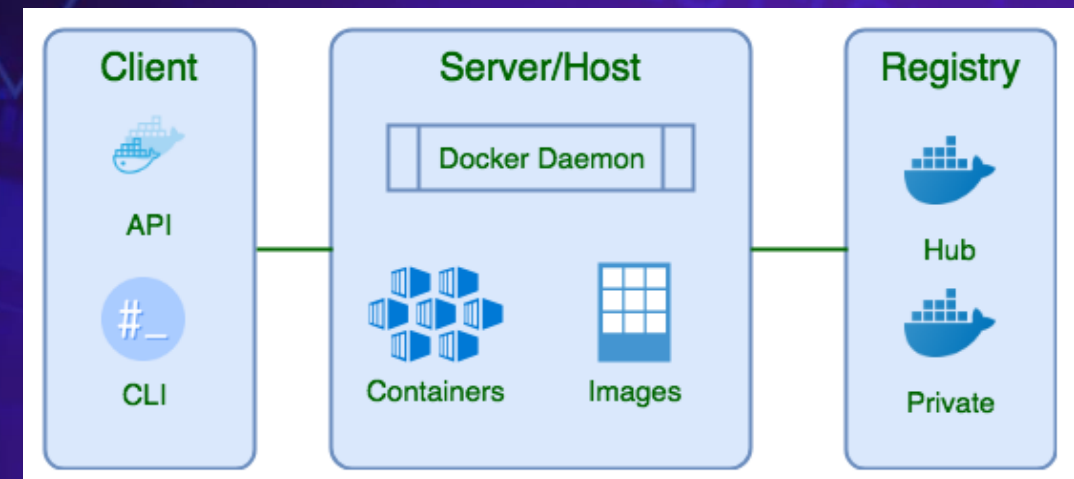


# Docker - Potencializando tecnologia de containers

Docker é uma plataforma inovadora que facilita a criação, execução e gerenciamento de containers. Ele usa a tecnologia de containerização para empacotar aplicações e suas dependências em um formato padronizado, tornando-as portáteis e eficientes.



- O Docker simplifica o processo de criação e gerenciamento de containers, tornando-os acessíveis até mesmo para aqueles sem experiência profunda em infraestrutura.
- Com Docker, os desenvolvedores podem focar na construção de aplicações sem se preocupar com o ambiente onde elas serão executadas. Isso é possível porque o Docker garante que os containers rodem de forma consistente em qualquer ambiente.
- Docker não é apenas uma ferramenta, mas um ecossistema amplo que inclui o Docker Hub, um repositório de imagens de containers, e uma série de outras ferramentas integradas que facilitam o ciclo de vida do desenvolvimento de software.



# Dockerfile - A fundação das imagens Docker

Dockerfile é um arquivo de texto que contém uma série de instruções para construir uma imagem Docker. É como uma receita ou um roteiro que o Docker segue para criar uma imagem de container.

```
1  # Usando a imagem oficial do Node.js como base
2  FROM node:14
3
4  # Definindo o diretório de trabalho no container
5  WORKDIR /usr/src/app
6
7  # Copiando os arquivos de definição de dependências
8  COPY package*.json ./
9
10 # Instalando as dependências do projeto
11 RUN npm install
12
13 # Copiando os arquivos do projeto para o diretório de trabalho do container
14 COPY . .
15
16 # Expondo a porta 3000 para ser acessada fora do container
17 EXPOSE 3000
18
19 # Comando para iniciar a aplicação
20 CMD ["node", "app.js"]
```

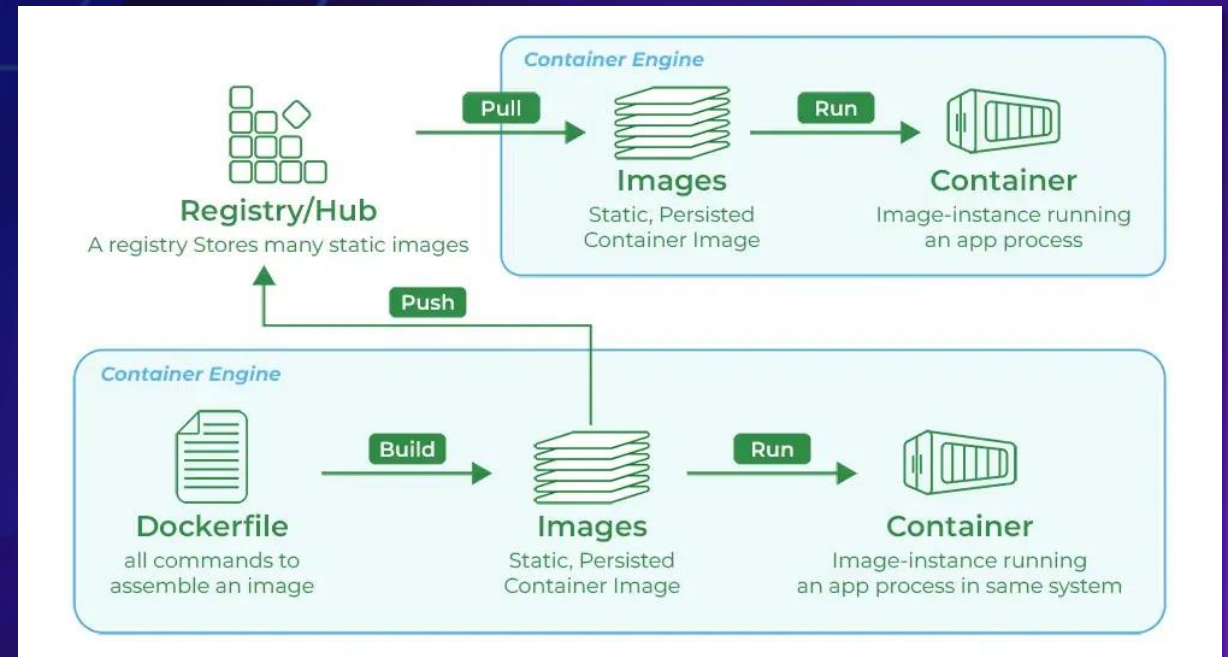
Cada linha no Dockerfile representa uma instrução, como instalar um software, definir variáveis de ambiente, copiar arquivos, etc.

Quando você executa o comando **docker build**, o Docker lê o Dockerfile, executa as instruções nele contidas, e assim, constrói a imagem do container.

# Docker Hub - O coração do ecossistema Docker

Docker Hub é um serviço de registro de imagens Docker na nuvem, que permite aos desenvolvedores compartilhar e acessar imagens de container publicamente ou privadamente.

- **Repositório de Imagens:** Hospeda uma vasta coleção de imagens Docker, tanto oficiais quanto contribuídas pela comunidade.
- **Integração com Docker:** Permite que desenvolvedores façam push (enviem) e pull (baixem) imagens de/para seus ambientes de desenvolvimento e produção, utilizando comandos Docker.

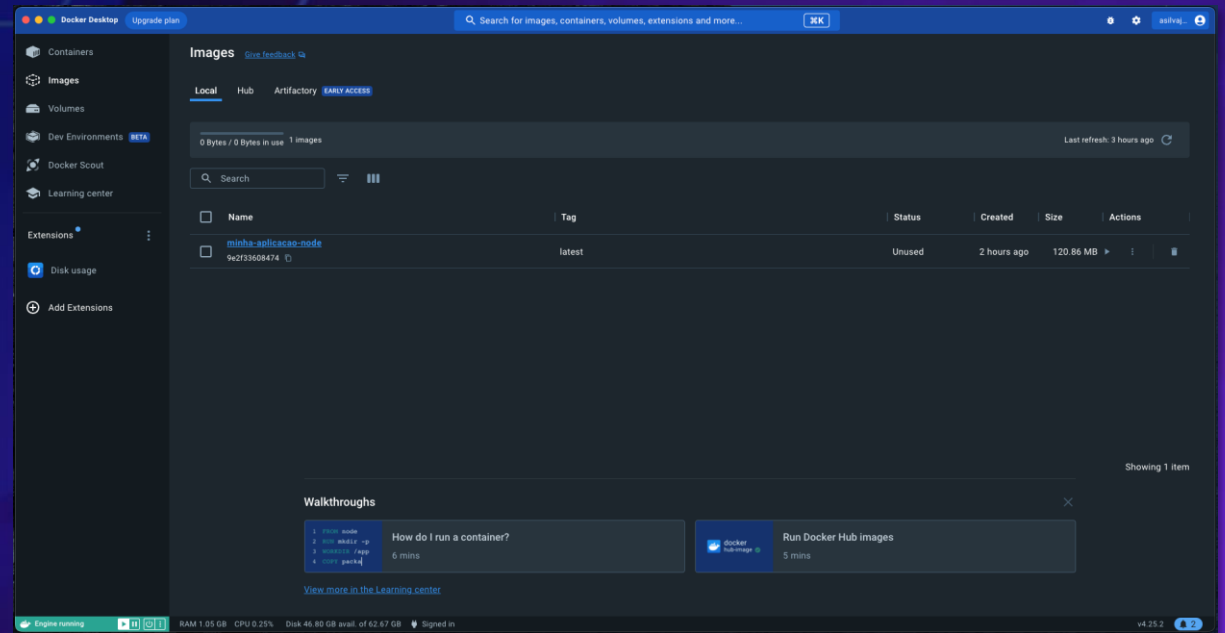




# Docker Desktop - A interface gráfica para Docker

Docker Desktop é uma aplicação GUI (Interface Gráfica do Usuário) que facilita a gestão de containers Docker em ambientes de desenvolvimento. Disponível para Windows e Mac, integra-se com os sistemas operacionais para oferecer uma experiência de desenvolvimento Docker suave e eficiente.

- **Fácil de Usar:** Interface intuitiva para gerenciar imagens, containers, volumes, redes e outros recursos Docker.
- **Ideal para Desenvolvedores:** Facilita o início e o gerenciamento do ambiente Docker, ideal para desenvolvedores que preferem interfaces visuais.



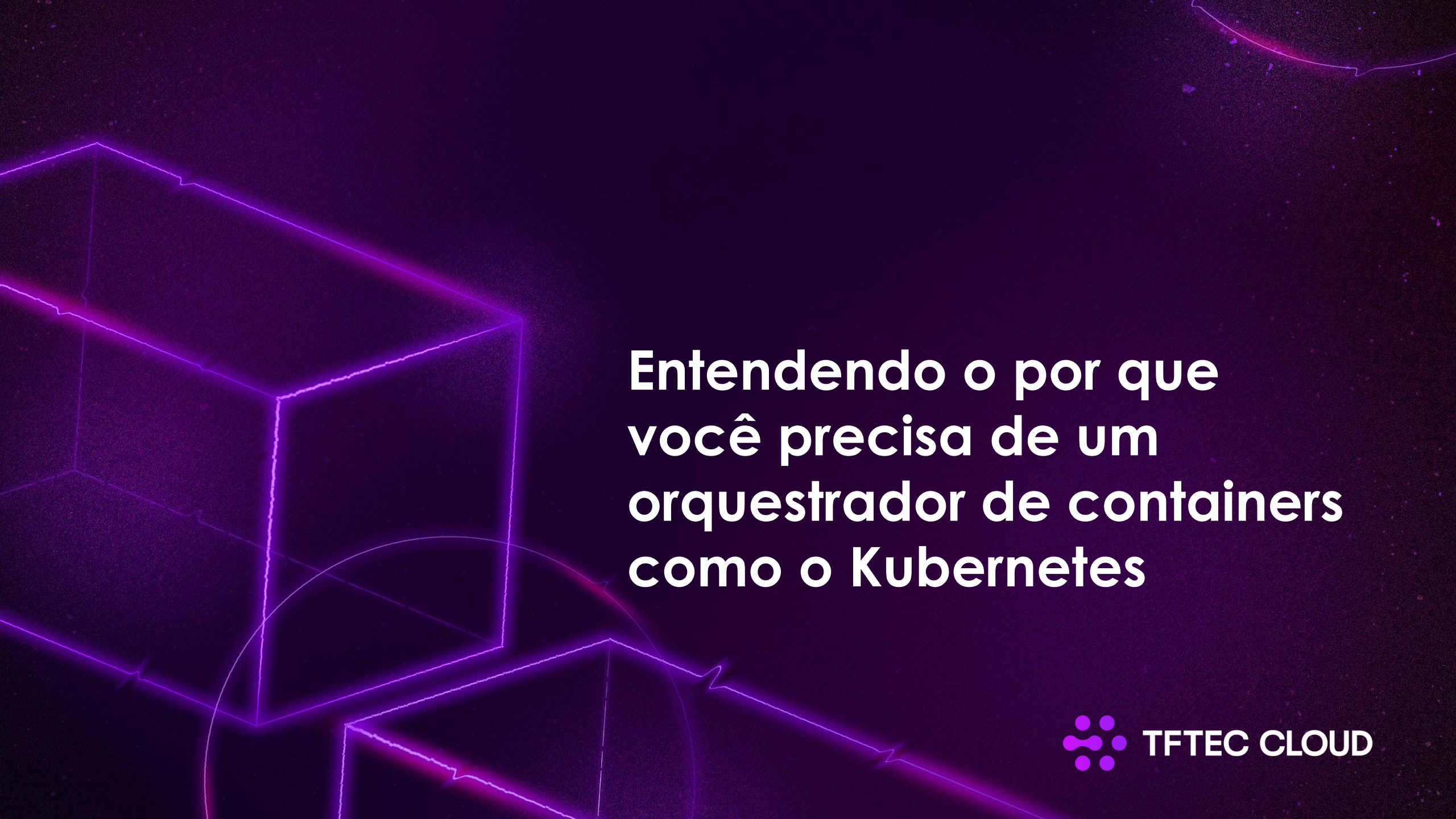


# Docker Demo



TFTEC CLOUD

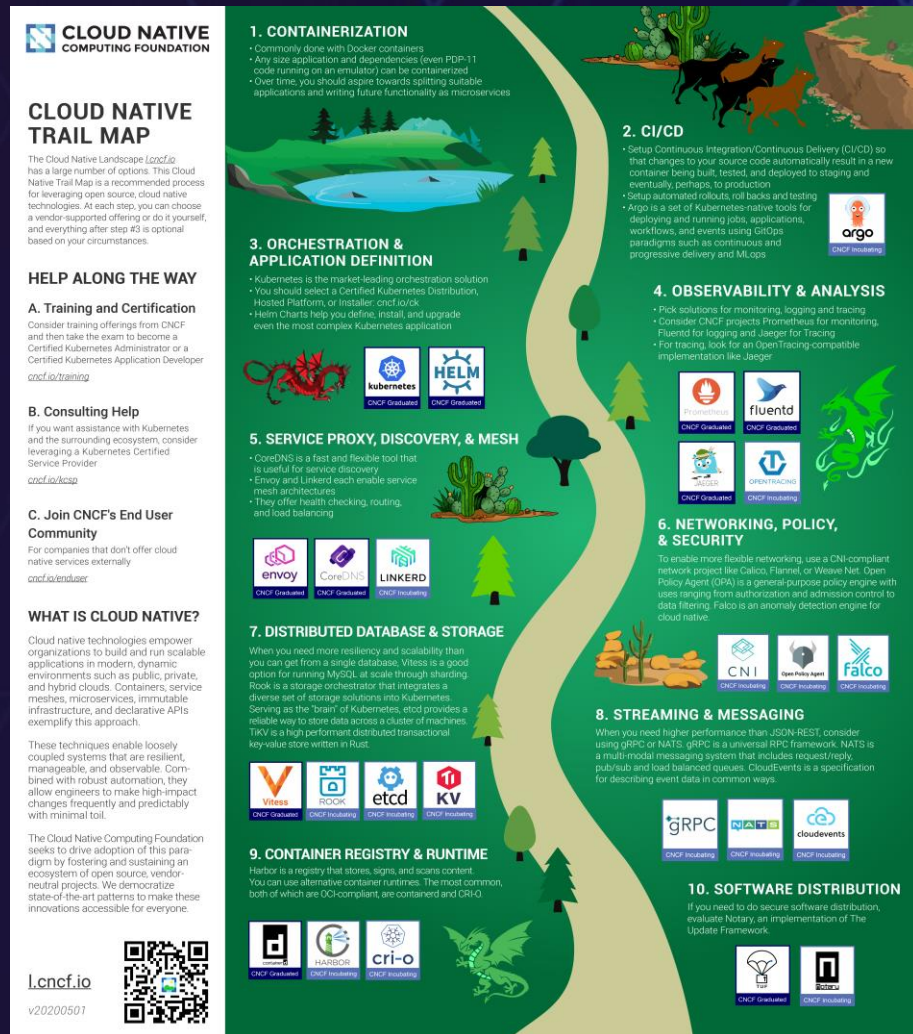


The background features several glowing purple geometric shapes, including a large cube on the left and various lines and arcs extending across the frame, creating a sense of depth and complexity.

# Entendendo o por que você precisa de um orquestrador de containers como o Kubernetes



# Cloud Native



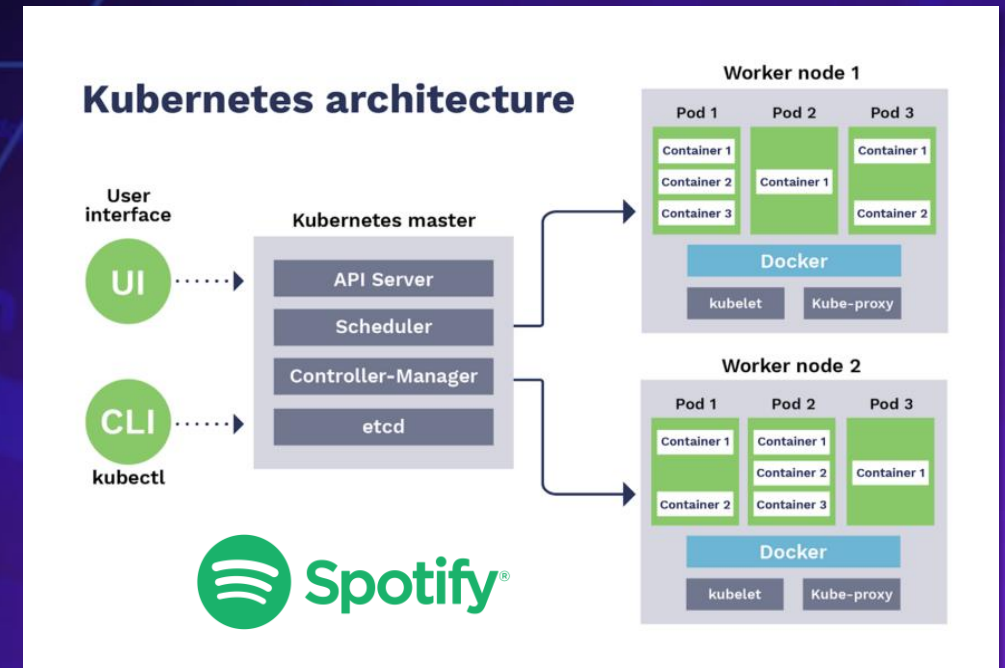
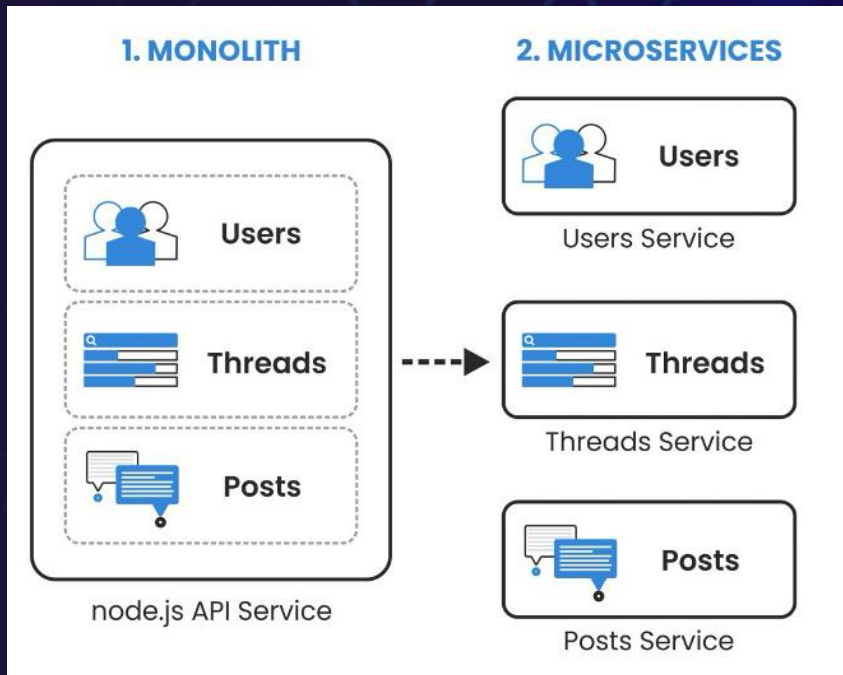
## Definição oficial Cloud Native Computing Foundation

- As tecnologias nativas de nuvem capacitam as organizações a criar e executar aplicativos escalonáveis em ambientes modernos e dinâmicos, como nuvens públicas, privadas e híbridas. Contêineres, malhas de serviço, microsserviços, infraestrutura imutável e APIs declarativas exemplificam essa abordagem.
- Essas técnicas permitem sistemas flexíveis que são resilientes, gerenciáveis e observáveis. Combinados com automação robusta, eles permitem que os engenheiros façam alterações de alto impacto de maneira frequente e previsível com o mínimo de trabalho.



# Arquitetura de microsserviços

**Microsserviços** referem-se à arquitetura de software que contém vários serviços responsáveis por operações específicas.





# Orquestradores - Por que eles são necessários?

A orquestração de containers é a automação que simplifica e otimiza a implantação, gestão, escalonamento e as redes de containers. É essencial em ambientes que precisam gerenciar um grande número de containers simultaneamente.

Em uma arquitetura de microserviços, a orquestração facilita a gestão de serviços complexos, abrangendo armazenamento, redes e segurança de forma mais integrada e menos propensa a erros.

## Principais objetivos da orquestração

- **Provisão e Deploy:** Implementação eficiente de containers conforme a necessidade.
- **Gestão de Recursos:** Alocação e otimização de recursos para maximizar a eficiência.
- **Alta Disponibilidade:** Assegura a disponibilidade contínua dos containers e seus serviços.
- **Balanceamento de Carga:** Distribuição inteligente da carga de trabalho e roteamento de tráfego.
- **Monitoramento de Integridade:** Vigilância contínua da saúde e do desempenho dos containers.
- **Configuração de Aplicativos:** Ajuste fino das configurações de aplicativos com base no ambiente de containerização.
- **Segurança:** Fortalecimento das interações entre containers para proteger contra vulnerabilidades.



# Por que escolher o Kubernetes?

O Kubernetes se tornou o padrão de facto para orquestração de containers, oferecendo uma solução robusta e flexível para ambientes complexos e de grande escala.

É considerado o sistema de orquestração de containers mais maduro e funcionalmente rico, adequado para uma variedade de ambientes e cargas de trabalho.

- **Automação Completa:** Gerenciamento automático de implantação, escalonamento e operações de containers.
- **Auto-recuperação:** Capacidade de auto-cura, com reinício automático de containers que falham e substituição e recolocação de instâncias problemáticas.
- **Infraestrutura Agnóstica:** Funciona em ambientes on-premise, na nuvem e híbridos, oferecendo portabilidade e flexibilidade.
- **Customização:** Suporta uma variedade de padrões de workload, desde aplicações stateless até stateful e microserviços.
- **Suporte Robusto:** Amplo suporte da comunidade e de empresas, garantindo uma base sólida de conhecimento e recursos.
- **Ecossistema:** Uma gama abrangente de serviços integrados, ferramentas e extensões disponíveis.



**TFTEC CLOUD**



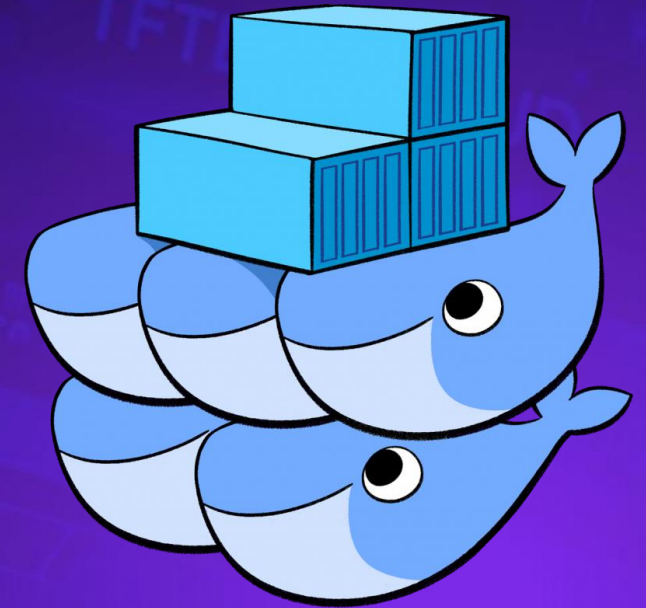
# Por que não o Docker Swarm?

O Docker Swarm é uma plataforma de orquestração de containers de código aberto, criada para funcionar de maneira integrada com o ecossistema Docker.

Projetado pela equipe do Docker, o Swarm é integrado ao Docker Engine, o que o torna fácil de configurar e começar a usar, especialmente para quem já está familiarizado com o Docker.

Docker Swarm é a escolha perfeita para equipes que já estão investidas no ecossistema Docker e que buscam uma solução de orquestração que mantém simplicidade sem comprometer a funcionalidade fundamental.

Mas em ambientes que exigem alta disponibilidade, complexidade de gerenciamento de serviços e escalabilidade automática, o Docker Swarm pode não atender a todas as necessidades empresariais.



**TFTEC CLOUD**



# Um simples lado a lado

## Docker Swarm:

- **Escalonamento Automático:** Não possui
- **Comunidade:** Boa
- **Iniciar um Cluster:** Fácil
- **Capacidades da API:** Limitado às capacidades da API do Docker
- **Experiência em Produção:** Menos experiência com implantações em produção e em escala

## Kubernetes:

- **Escalonamento Automático:** Possui
- **Comunidade:** Grande e ativa
- **Iniciar um Cluster:** Mais complexo
- **Capacidades da API:** Supera as limitações da API do Docker
- **Experiência em Produção:** Frequentemente utilizado em implantações em escala por organizações

Embora o Docker tenha dado passos significativos ao aprimorar suas ferramentas de orquestração, o Kubernetes emerge como líder na padronização e otimização da criação e operação de clusters de containers. Com sua abordagem abrangente e comunidade ativa, o Kubernetes não apenas atende mas antecipa as exigências do DevOps moderno, solidificando-se como a espinha dorsal da estratégia de cloud nativa e infraestrutura escalável.



# Obrigado