

DevOps

Aula 02 – Parte 1

Recapitulando

Git e GitHub

RECAPITULANDO



- Configuração Inicial
- Comandos Básicos
- Integração com GitHub
- Utilizando Branchs
- pull request
- Fluxo Colaborativo
- Resolvendo conflitos
- Repositório Colaborativo



Containers

Iniciando .1..2...3....

CONTAINERS




O que são máquinas
virtual e contêiners?

Qual a diferença?

Containers

Introdução

CONTAINERS

 **Containers** são uma tecnologia que empacota uma aplicação e todas as suas dependências (bibliotecas, configurações, etc.) em uma unidade leve e portátil.

 Compartilham o **mesmo kernel** do sistema operacional host, o que os torna mais eficientes em termos de recursos e desempenho



VIRTUALIZAÇÃO A NÍVEL DE SISTEMA OPERACIONAL




 Permite a criação de ambientes isolados (containers) que compartilham o mesmo kernel do host.



VIRTUALIZAÇÃO A NÍVEL DE SISTEMA OPERACIONAL



Diferença para a virtualização tradicional (hipervisores):

 **Máquinas virtuais:** Cada VM inclui seu próprio sistema operacional completo, o que exige mais recursos.

 **Containers:** Compartilham o kernel do host, sendo mais leves e iniciando mais rapidamente.



VIRTUALIZAÇÃO A NÍVEL DE SISTEMA OPERACIONAL



Benefícios

- ✓ **Eficiência:** Menor consumo de recursos e maior densidade de execução.
- ✓ **Velocidade:** Containers iniciam em segundos, ao contrário das VMs que podem demorar minutos.



Namespaces

Os containers utilizam **namespaces do kernel Linux** para isolar vários aspectos do sistema, como:

- ❖ **PID (Process ID):** Cada container tem seu próprio espaço de processos.
- ❖ **Network:** Isolamento de interfaces de rede e IPs.
- ❖ **IPC (Interprocess Communication):** Comunicação isolada entre processos.
- ❖ **Mount:** Cada container tem sua própria visão do sistema de arquivos.



Cgroups (Control Groups):

- ❖ Permitem **limitar e monitorar** o uso de **recursos** (CPU, memória, I/O, etc.) por cada container, garantindo que um container não consuma recursos de forma a prejudicar os demais.



ISOLAMENTO DE PROCESSOS E RECURSOS



Imagine que cada container é como um "**micro ambiente**" isolado que só pode ver e utilizar os recursos que lhe foram alocados, mesmo estando no mesmo host.



IMAGENS E CONTAINERS

O que são Imagens?

- **arquivo imutável** que contém tudo que a aplicação precisa para rodar
- **Camadas (Layers):**
 - imagens são construídas em camadas, permitindo a reutilização e a economia de espaço.
 - Cada comando no Dockerfile (por exemplo, ``RUN``, ``COPY``) cria uma nova camada.



IMAGENS E CONTAINERS

O que são Containers?

- **instância em execução** de uma imagem.
- quando uma imagem é executada, o container é criado e isolado
- pode ser iniciado, parado, reiniciado ou destruído sem afetar a imagem original.



REGISTRY E DISTRIBUIÇÃO DE IMAGENS

Registry de Containers:

- ❑ É um **repositório central** onde as imagens de containers são armazenadas, versionadas e distribuídas.

- ❑ Exemplos:
 - Docker Hub:
 - GitHub Container Registry:
 - Registries privados:



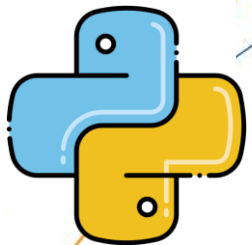
DOCKER: ATIVIDADE 1



Casos de uso práticos de containers em projetos de análise de dados, demonstrando como eles garantem reprodutibilidade e portabilidade dos ambientes de análise.

URL da Atividade

<https://github.com/luisrodrigonet/>



DESAFIO 01



DevOps - Docker

