Fundamentos do Docker

1. Introdução ao Docker - Hello from Docker!

Docker é uma plataforma que permite a criação, implantação e execução de aplicações em contêineres. Um contêiner é uma unidade leve e autossuficiente que inclui tudo o que a aplicação precisa para funcionar: código, bibliotecas, dependências e até mesmo o sistema operacional.

Para começar a explorar o Docker, você pode executar um contêiner simples que exibe a mensagem “Hello from Docker!”.

Passos para Executar o Hello from Docker!:

Instalação do Docker: Primeiro, é necessário instalar o Docker em sua máquina. No Google Cloud, você pode fazer isso em uma VM ou usar o Google Cloud Shell, que já vem com o Docker pré-instalado.

Executando o Contêiner:

No terminal, execute o comando docker run hello-world.

O Docker buscará a imagem hello-world no Docker Hub (se não estiver disponível localmente) e a executará, exibindo a mensagem de boas-vindas.

Exemplo Contextual: Este exemplo serve como uma introdução ao funcionamento básico do Docker, mostrando como é simples executar uma aplicação contida em um ambiente isolado.

2. Construção de uma Imagem Docker com um Aplicativo Simples

Depois de entender o básico, o próximo passo é construir sua própria imagem Docker a partir de um aplicativo simples.

Passos para Construir uma Imagem Docker:

Criação do Dockerfile:

O Dockerfile é um arquivo de texto que contém instruções passo a passo para construir uma imagem Docker.

Exemplo de Dockerfile para um aplicativo Python simples:

```
# Use uma imagem base do Python
FROM python:3.9-slim

# Defina o diretório de trabalho no contêiner
WORKDIR /app

# Copie os arquivos necessários para o contêiner
COPY . /app

# Instale as dependências do aplicativo
RUN pip install -r requirements.txt

# Comando para executar a aplicação
CMD ["python", "app.py"]
```

Construção da Imagem:

Navegue até o diretório que contém o Dockerfile e execute o comando:

docker build -t my-simple-app .

Suponha que você tenha um aplicativo Python que responde “Hello, World!” a cada solicitação HTTP. Com o Docker, você pode empacotar todo o ambiente do aplicativo em uma imagem, garantindo que ele funcione de maneira consistente em qualquer lugar.

3. Execução de Contêineres a Partir da Imagem Criada

Depois de construir a imagem Docker, você pode executar contêineres a partir dela.

Passos para Executar um Contêiner:

- **Executando o Contêiner:**
  - Use o comando docker run -d -p 5000:5000 my-simple-app para executar o contêiner.
  - A flag -d executa o contêiner em segundo plano (modo daemon).
  - A flag -p mapeia a porta 5000 do contêiner para a porta 5000 da máquina host.
- **Verificação:**
  - Acesse http://localhost:5000 no navegador para ver a aplicação em execução.

Se você implementou uma API simples, ela agora estará rodando em um contêiner Docker, acessível através do navegador ou de ferramentas como o curl.

4. Práticas de Depuração com Docker

Depurar contêineres Docker é uma habilidade essencial para identificar e resolver problemas que possam surgir durante a execução de aplicações.

Ferramentas e Comandos de Depuração:

- **Visualização de Logs:**
  - Use docker logs <container_id> para visualizar os logs gerados por um contêiner em execução.

- **Acessando o Terminal do Contêiner:**
  - Com o comando docker exec -it <container_id> /bin/bash, você pode acessar o terminal dentro do contêiner e executar comandos para inspecionar o ambiente.

- **Inspecionar Contêiner:**
  - docker inspect <container_id> fornece detalhes extensivos sobre a configuração e o estado do contêiner.

Exemplo Contextual: Se sua aplicação não está respondendo corretamente, acessar o terminal do contêiner e inspecionar os logs pode ajudar a identificar problemas de dependências, erros de código ou configurações incorretas.

5. Testando a Imagem Docker

Após construir e depurar a imagem Docker, é crucial testá-la para garantir que funcione conforme o esperado em diferentes ambientes.

Estratégias de Teste:

- **Testes Locais:**
  - Execute múltiplos contêineres localmente em diferentes portas para garantir que a aplicação escala corretamente e que as configurações

são consistentes.

- **Testes em Ambientes de Desenvolvimento/Produção:**
  - No Google Cloud, você pode usar o Google Kubernetes Engine (GKE) ou Cloud Run para implantar e testar sua imagem em um ambiente de produção simulado.

Imagine que você precisa testar se sua aplicação suporta conexões simultâneas e é capaz de lidar com falhas. Usando Docker, você pode simular esses cenários localmente antes de implantar em produção, garantindo que a aplicação esteja pronta para qualquer situação.

Tópicos Extras sobre Docker

1. Volumes em Docker

Volumes são a forma preferida de persistir dados gerados e usados por contêineres Docker. Ao usar volumes, você pode armazenar dados fora do contêiner, permitindo que eles sobrevivam a reinicializações, exclusões e recriações de contêineres.

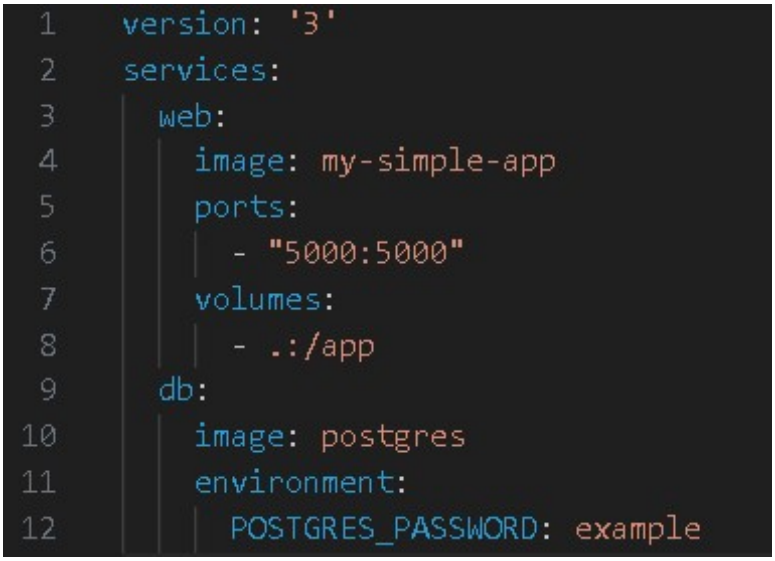
- **Criação e Montagem de Volumes:**
  - Comando para criar um volume: docker volume create my-volume.
  - Montagem de volume em um contêiner: docker run -d -v my-volume:/app/data my-simple-app.

Imagine um contêiner que executa um banco de dados. Usar volumes permite que os dados do banco persistam mesmo que o contêiner seja removido ou atualizado.

2. Docker Compose

Docker Compose é uma ferramenta que permite definir e executar aplicativos Docker com múltiplos contêineres. Com um único arquivo YAML (docker-compose.yml), você pode configurar a rede, volumes e dependências entre serviços, facilitando o gerenciamento de ambientes complexos.

- **Exemplo de Arquivo docker-compose.yml:**



```yaml
version: '3'
services:
  web:
    image: my-simple-app
    ports:
      - "5000:5000"
    volumes:
      - .:/app
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: example
```

- **Comandos Básicos:**
  - docker-compose up - Inicia todos os serviços definidos no arquivo.
  - docker-compose down - Para e remove contêineres, redes e volumes criados.

Em uma aplicação web que depende de um banco de dados, o Docker Compose facilita o desenvolvimento e testes ao permitir que ambos os serviços sejam iniciados e configurados com um único comando.

3. Redes em Docker

Redes permitem que os contêineres se comuniquem entre si. Docker fornece diferentes tipos de redes para diferentes casos de uso, como bridge, host, e overlay.

- **Criando e Usando Redes:**
  - Criação de uma rede: docker network create my-network.
  - Conectando contêineres à rede: docker run -d --network my-network --name my-app my-simple-app.
- **Tipos de Redes:**
  - Bridge: Rede padrão usada quando você inicia contêineres sem especificar uma rede.
  - Host: Permite que o contêiner use a interface de rede do host, útil para desempenho, mas com menos isolamento.
  - Overlay: Usado em Swarm, conecta contêineres em diferentes hosts.

Em um ambiente de microserviços, onde diferentes partes da aplicação são executadas em contêineres separados, o uso de redes Docker permite uma comunicação eficiente e segura entre eles.

4. Docker Swarm

Docker Swarm é uma ferramenta de orquestração de contêineres nativa do Docker que permite a implantação e gerenciamento de clusters de contêineres em múltiplos nós. Ele transforma um conjunto de máquinas Docker em um cluster, permitindo a escalabilidade e resiliência de aplicações.

- **Comandos Básicos:**

- docker swarm init - Inicializa um novo cluster Swarm.

- docker service create - Cria e gerencia serviços em um cluster Swarm.

Para uma aplicação que precisa escalar horizontalmente em múltiplos servidores, Docker Swarm facilita a distribuição de contêineres entre nós, garantindo alta disponibilidade.

5. Docker Registry

Docker Registry é um serviço para hospedar e distribuir imagens Docker. O Docker Hub é um exemplo de um registro público, mas você também pode configurar um registro privado para armazenar e distribuir suas próprias imagens.

- **Configuração de um Registro Privado:**
  - Comando para iniciar um registro privado: docker run -d -p 5000:5000 --name registry registry:2.

  - Push de uma imagem para o registro: docker tag my-simple-app localhost:5000/my-simple-app e docker push localhost:5000/my-simple-app.

Em uma empresa onde a segurança e controle sobre as imagens são cruciais, um Docker Registry privado permite armazenar e compartilhar imagens Docker de forma segura dentro da organização.

6. Segurança em Docker

Segurança é uma consideração crucial ao trabalhar com Docker, especialmente em ambientes de produção. Existem práticas recomendadas para minimizar riscos e proteger suas aplicações em contêineres.

- **Práticas de Segurança:**
  - Use imagens oficiais e verificadas sempre que possível.
  - Minimize as permissões dos contêineres usando a flag --cap-drop.
  - Execute contêineres como usuários não root.
  - Habilite a política de segurança do host Docker.

Ao implementar um contêiner para uma aplicação sensível, como um sistema de pagamento, é fundamental seguir as práticas de segurança para proteger dados do usuário e manter a integridade do sistema.

Conteúdo Bônus

Curso: "Fundamentos Docker | Curso Online Grátis"

Este curso introdutório da Alison aborda os conceitos básicos do Docker, incluindo instalação, criação de contêineres e gerenciamento de imagens.

Referências Bibliográficas

BASSO, D. E. **Administração de Redes de Computadores**. Contentus, 2020.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. 8. ed. Pearson, 2021.

MARINHO, A. L.; CRUZ, J. L. da. (Orgs.). **Desenvolvimento de Aplicações para Internet**. 2. ed. Pearson, 2020.

PUGA, S.; RISSETTI, G. **Lógica de Programação e Estruturas de Dados, com Aplicações em Java**. 3. ed. Pearson, 2016.

ROHLING, L. J. **Segurança de Redes de Computadores**. Contentus, 2020.

SILVA, C. F. da. **Projeto Estruturado e Gerência de Redes**. Contentus, 2020.

TANENBAUM, A. S.; FEAMSTER, N.; WETHERALL, D. J. **Redes de**

Computadores. 6. ed. Pearson, 2021.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais: Princípios e Aplicações**. 12. ed. Pearson, 2018.

Ir para exercício