Exercícios Docker

- Autor: Thiago Geremias de Oliveira
- Data de revisão: 17/04/2025

Crie um repositório no github público e coloque todos os comandos e prints se necessários para evidenciar a execução da lista de exercícios, ao final envie o link do repositório para conferência.

Fácil

1. **Rodando um container básico**
   Execute um container usando a imagem do **Nginx** e acesse a página padrão no navegador. Use a landing page do TailwindCSS como site estático dentro do container.

2. **Criando e rodando um container interativo**
   Inicie um container **Ubuntu** e interaja com o terminal dele. Teste um script Bash que imprime logs do sistema ou instala pacotes de forma interativa.

3. **Listando e removendo containers**
   Liste todos os containers em execução e parados, pare um container em execução e remova um container específico.

4. **Criando um Dockerfile para uma aplicação simples em Python**
   Crie um Dockerfile para uma aplicação **Flask** que retorna uma mensagem ao acessar um endpoint, para isso utilize o projeto Docker Flask

Médio

5. **Criando e utilizando volumes para persistência de dados**

Execute um container **MySQL** e configure um volume para armazenar os dados do banco de forma persistente. Para aplicar esse conceito você pode utilizar o react-express-mysql

6. **Criando e rodando um container multi-stage**
   Utilize um **multi-stage build** para otimizar uma aplicação **Go**, reduzindo o tamanho da imagem final. Utilize para praticar o projeto GS PING desenvolvido em Golang.

7. **Construindo uma rede Docker para comunicação entre containers**
   Crie uma rede Docker personalizada e faça dois containers, um **Node.js** e um **MongoDB**, se comunicarem, sugestão, utilize o projeto React Express + Mongo

8. **Criando um compose file para rodar uma aplicação com banco de dados**
   a. Utilize **Docker Compose** para configurar uma aplicação com um banco de dados **PostgreSQL**, use para isso o projeto pgadmin.

Difícil

9. **Criando uma imagem personalizada com um servidor web e arquivos estáticos**
   Construa uma imagem baseada no **Nginx** ou **Apache**, adicionando um site HTML/CSS estático. Utilize a landing page do Creative Tim para criar uma página moderna hospedada no container.

10. **Evitar execução como root**
    Ao rodar containers com o usuário root, você expõe seu sistema a riscos maiores em caso de comprometimento. Neste exercício, você deverá criar um Dockerfile para uma aplicação simples (como um script Python ou um servidor Node.js) e configurar a imagem para rodar com um usuário não-root.
    Você precisará:

a. Criar um usuário com useradd ou adduser no Dockerfile.
b. Definir esse usuário como o padrão com a instrução USER.
c. Construir a imagem e iniciar o container.
d. Verificar se o processo está rodando com o novo usuário usando docker exec <container> whoami.

11. Analisar imagem Docker com Trivy

Trivy é uma ferramenta open source para análise de vulnerabilidades em imagens Docker. Neste exercício, você irá analisar uma imagem pública, como python:3.9 ou node:16, em busca de vulnerabilidades conhecidas. Você deverá:

a. Instalar o Trivy na sua máquina (via script ou pacote).
b. Rodar trivy image <nome-da-imagem> para analisar.
c. Identificar vulnerabilidades com severidade HIGH ou CRITICAL.
d. Anotar os pacotes ou bibliotecas afetadas e sugerir possíveis ações (como atualização da imagem base ou substituição de dependências).

12. Corrigir vulnerabilidades encontradas

Após identificar vulnerabilidades com ferramentas como o Trivy, o próximo passo é corrigi-las. Imagens grandes e genéricas frequentemente trazem bibliotecas desnecessárias e vulneráveis, além de usarem o usuário root por padrão. Neste exercício, você irá trabalhar com um exemplo de Dockerfile com más práticas e aplicar melhorias para construir uma imagem mais segura e enxuta. Identifique as melhorias e gere uma nova versão de Dockerfile

Dockerfile

```
Dockerfile vulnerável
FROM python:3.9
WORKDIR /app COPY requirements.txt . RUN pip install -r
requirements.txt
COPY . .
CMD ["python", "app.py"]
```

requirements.txt

```
flask==1.1.1
requests==2.22.0
```