UniSENAI

Trilha 05

Criando e Parametrizando Dockerfile. docker-compose.yml e nginx.conf

Instruções para a melhor prática de Estudo

1. **Leia atentamente todo o conteúdo:** Antes de iniciar qualquer atividade, faça uma leitura detalhada do material fornecido na trilha, compreendendo os conceitos e os exemplos apresentados.

2. **Não se limite ao material da trilha:** Utilize o material da trilha como base, mas busque outros materiais de apoio, como livros, artigos acadêmicos, vídeos, e blogs especializados. Isso enriquecerá o entendimento sobre o tema.

3. **Explore a literatura:** Consulte livros e publicações reconhecidas na área, buscando expandir seu conhecimento além do que foi apresentado. A literatura acadêmica oferece uma base sólida para a compreensão de temas complexos.

4. **Realize todas as atividades propostas:** Conclua cada uma das atividades práticas e teóricas, garantindo que você esteja aplicando o conhecimento adquirido de maneira ativa.

5. **Evite o uso de Inteligência Artificial para resolução de atividades:** Utilize suas próprias habilidades e conhecimentos para resolver os exercícios. O aprendizado vem do esforço e da prática.

6. **Participe de debates:** Discuta os conteúdos estudados com professores, colegas e profissionais da área. O debate enriquece o entendimento e permite a troca de diferentes pontos de vista.

7. **Pratique regularmente:** Não deixe as atividades para a última hora. Pratique diariamente e revise o conteúdo com frequência para consolidar o aprendizado.

8. **Peça feedback:** Solicite o retorno dos professores sobre suas atividades e participe de discussões sobre os erros e acertos, utilizando o feedback para aprimorar suas habilidades.

Essas instruções são fundamentais para garantir um aprendizado profundo e eficaz ao longo das trilhas.

Passo 5: Criando e Parametrizando Dockerfile, docker-compose.yml e nginx.conf

Conceito:

Agora vamos entender os três arquivos principais usados em projetos Docker: o **Dockerfile**, o **docker-compose.yml** e o **nginx.conf**.

Dockerfile:

O **Dockerfile** é usado para definir a imagem do ambiente onde o aplicativo será executado.

Estrutura básica de um Dockerfile:

```
FROM node:14

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

RUN npm run build
CMD ["npm", "start"]
```

- **FROM** especifica a imagem base.
- **WORKDIR** define o diretório de trabalho dentro do contêiner.
- **COPY** e **RUN** permitem copiar arquivos e executar comandos.
- **CMD** define o comando final a ser executado ao iniciar o contêiner.

docker-compose.yml:

O **docker-compose.yml** orquestra múltiplos contêineres.

Exemplo básico:

```
version: '3'
services:
  web:
    build: .
    ports:
      - "8080:80"
    volumes:
      - ./app:/app
    depends_on:
      - db
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: app_db
    volumes:
      - db_data:/var/lib/mysql
volumes:
  db_data:
```

- **services** define os contêineres.
- **ports** mapeia as portas (8080 no host para 80 no contêiner).
- **volumes** gerencia persistência de dados.
- **depends_on** define dependências entre serviços (ex: web depende do banco de dados).

nginx.conf:

O **nginx.conf** configura o servidor Nginx para servir sua aplicação e proxy para APIs.

Exemplo básico:

```
server {
    listen 80;

    server_name localhost;

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://backend:3000/;
    }
}
```

- **listen** define a porta (80 neste exemplo).
- **location** define como as rotas são tratadas.
- **proxy_pass** redireciona requisições para o backend (NodeJS, por exemplo).

Conceito sobre Portas:

As portas no Docker são configuradas com a sintaxe `host_port:container_port`. Isso mapeia uma porta do host para uma porta do contêiner.

Exemplo de Mapeamento:

```
ports:
  - "8080:80"
```

A porta 8080 no host é mapeada para a porta 80 no contêiner.

Exercícios de fixação:

1. Crie um **Dockerfile** para uma aplicação simples.
2. Crie um **docker-compose.yml** para orquestrar uma aplicação com MySQL.
3. Configure o **nginx.conf** para servir a aplicação e proxy para APIs.
4. Explique o mapeamento de portas no Docker.