Labs unidade 3

Observação

**Lembre-se que o docker desktop possui o compose integrado ao comando docker, invocando-o com `docker compose`. Antigas instalações utilizarão o `docker-compose`

Lab 1

Objetivo: Aprender a usar Docker Compose para definir e gerenciar um ambiente multi-contêiner, incluindo a construção de imagens Docker personalizadas

1. **Criar um Diretório para o Projeto:**
    - Crie um diretório para o seu projeto.

```
mkdir lab1
cd lab1
```

2. **Criar um Arquivo Dockerfile:**
    - Dentro do diretório do projeto, crie um arquivo `Dockerfile` para o serviço `web`:

```
# Use a imagem base do Nginx
FROM nginx:latest

# Copie o conteúdo do diretório local 'html' para o diretório padrão do Nginx
COPY html /usr/share/nginx/html
```

3. **Criar Conteúdo HTML:**
    - Crie um diretório `html` dentro do diretório do projeto e adicione um arquivo `index.html` com algum conteúdo.

```
mkdir html
echo "<h1>Bem vindo ao  Docker Compose com Build!</h1>" > html/index.html
```

4. **Criar um Arquivo Docker Compose:**
    - Dentro do diretório do projeto, crie um arquivo `docker-compose.yml` com o seguinte conteúdo:

```
version: '3.8'

services:
  web:
    build: .
    ports:
      - "8080:80"
    networks:
      - webnet

  redis:
    image: redis:latest
    ports:
      - "6379:6379"
    networks:
      - webnet

  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: exemplo
    ports:
      - "3306:3306"
    networks:
      - webnet

networks:
  webnet:
```

5. **Construir e Executar os Serviços com Docker Compose:**
    - No diretório do projeto, use o Docker Compose para construir a imagem personalizada e iniciar os serviços definidos no arquivo `docker-`

`compose.yml`.

```
docker compose up --build -d
```

6. **Verificar os Serviços:**
   - Liste os contêineres em execução para verificar se os serviços `web`, `redis` e `db` estão em execução.

```
docker-compose ps
```

7. **Acessar o Serviço Web:**
   - Abra o navegador e acesse `http://localhost:8080` para ver o conteúdo do arquivo `index.html`.
8. **Interagir com o Serviço Redis:**
   - Conecte-se ao contêiner `redis` e execute alguns comandos Redis.

```
docker-compose exec redis redis-cli

127.0.0.1:6379> set chave "valor"
127.0.0.1:6379> get chave
127.0.0.1:6379> exit
```

9. **Adicionar um Banco de Dados (MySQL):**
   - Verifique a conexão ao banco de dados MySQL criado.

```
docker-compose exec db mysql -u root -p
# Use 'exemplo' como senha

mysql> SHOW DATABASES;
```

10. **Parar e Remover os Serviços:**
    - Pare e remova os serviços e os volumes criados.

```
docker-compose down --volumes
```

Lab 2

Objetivo: Usar dependências de serviços e escalar os containers

1. **Copie o Dockerfile, o compose e o diretório html do lab1 para o diretório do lab2**

```
cd lab2
cp -R ../lab1/{Dockerfile,html,docker-compose.yml} .
```

2. **Modifique o Dockerfile do serviço web para adicionar a configuração nginx provida no lab2**

```
# Use a imagem base do Nginx
FROM nginx:latest
# Copie o conteúdo do diretório local 'html' para o diretório padrão do Nginx
COPY html /usr/share/nginx/html
# config default modificada
COPY nginx.conf /etc/nginx/conf.d/default.conf
```

3. **Modifique o docker-compose.yml para adiconar o serviço contador provido pelo lab2 e adcionar a referencia no web**

```
...
 web:
     build: .
     ports:
        - "8080:80"
     networks:
       - webnet
     # Dependência do contador adicionada
     depends_on:
       - flask

 flask:
     build: contador # diretório ond está o serviço contador
     environment:
       - REDIS_HOST=redis # <- Note o uso de variáveis de ambiente a serem interpretadas pelo serviço,
       - REDIS_PORT=6379
     depends_on:
       - redis
     ports:
       - "5000"
     networks: # Adicionado o serviço na mesma rede dos outros
       - webnet
   ...
```

4. **Iniciar o novo compose:**

```
docker compose up --build
```

5. **Acesse os serviços pela porta 8080 no seu browser ou pelo curl**

```
curl http://localhost:8080
curl http://localhost:8080/contador #repita várias vezes
```

6. **Escalar o Serviço flask**:
    - Escale o serviço `flask` para ter 3 instâncias.

```
docker-compose up -d --scale flask=3
```

7. **Verificar as Instâncias:**
    - Liste os contêineres para verificar as 3 instâncias do serviço `flask`.

```
docker-compose ps
```

8. **Verifique se o serviço web está balanceando a carga entre os flask**

```
curl http://localhost:8080/contador #repita várias vezes
```

9. **Reinicie o serviço web e repita o teste**

```
docker compose restart web
curl http://localhost:8080/contador #repita várias vezes
```