Atividade-15

Pesquise e responda as perguntas abaixo.
1) Como definir um volume no Docker Compose para persistir os dados do banco de dados PostgreSQL entre as execuções dos containers?

> Para definir um volume no Docker Compose e persistir os dados do banco de dados PostgreSQL entre as execuções dos containers, você pode usar a seção "volumes" no arquivo docker-compose.yml. Por exemplo:

```
version: '3'
services:
  db:
    image: postgres
    volumes:
      - ./data:/var/lib/postgresql/data
```

2) Como configurar variáveis de ambiente para especificar a senha do banco de dados PostgreSQL e a porta do servidor Nginx no Docker Compose?

> Para configurar variáveis de ambiente no Docker Compose, você pode usar a seção "environment" em cada serviço no arquivo docker-compose.yml. Por exemplo:

```
version: '3'
services:
  db:
    image: postgres
    environment:
      - POSTGRES_PASSWORD=minhasenha
  nginx:
    image: nginx
```

```
    ports:
      - "80:${NGINX_PORT}"
    environment:
      - NGINX_PORT=8080
```

3) Como criar uma rede personalizada no Docker Compose para que os containers possam se comunicar entre si?

Para criar uma rede personalizada no Docker Compose, você pode usar a seção "networks" no arquivo docker-compose.yml. Por exemplo:

```
version: '3'
services:
  db:
    image: postgres
    networks:
      - mynetwork
  nginx:
    image: nginx
    networks:
      - mynetwork

networks:
  mynetwork:
```

4) Como configurar o container Nginx para atuar como um proxy reverso para redirecionar o tráfego para diferentes serviços dentro do Docker Compose?

Para configurar o container Nginx como um proxy reverso no Docker Compose, você precisa definir a configuração do Nginx em um arquivo de configuração personalizado e montá-lo como um volume no container. Por exemplo:

```
version: '3'
services:
  nginx:
    image: nginx
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
```

5) Como especificar dependências entre os serviços no Docker Compose para garantir que o banco de dados PostgreSQL esteja totalmente inicializado antes do Python iniciar?

Para especificar dependências entre os serviços no Docker Compose e garantir que o banco de dados PostgreSQL esteja totalmente inicializado antes do serviço Python iniciar, você pode usar a seção "depends_on" no arquivo docker-compose.yml. No entanto, é importante destacar que o Docker Compose não aguarda a total inicialização do serviço dependente antes de iniciar o próximo serviço. O "depends_on" apenas controla a ordem de inicialização dos serviços. Você pode usar ferramentas adicionais, como o wait-for-it.sh, para esperar explicitamente até que o serviço dependente esteja pronto antes de iniciar o próximo serviço.

6) Como definir um volume compartilhado entre os containers Python e Redis para armazenar os dados da fila de mensagens implementada em Redis?

Para definir um volume compartilhado entre os containers Python e Redis no Docker Compose, você pode usar a seção "volumes" para cada serviço no arquivo docker-compose.yml. Por exemplo:

```
version: '3'
services:
  python:
```

```
    image: python
    volumes:
      - ./shared:/app/shared
  redis:
    image: redis
    volumes:
      - ./shared:/data
```

7) Como configurar o Redis para aceitar conexões de outros containers apenas na rede interna do Docker Compose e não de fora?

Para configurar o Redis para aceitar conexões apenas da rede interna do Docker Compose, você pode definir o parâmetro "bind" no arquivo de configuração do Redis dentro do container. No entanto, no Docker Compose, todos os serviços dentro de uma rede têm acesso mútuo por padrão, portanto, não há necessidade de configurações adicionais para restringir as conexões apenas à rede interna do Docker Compose.

8) Como limitar os recursos de CPU e memória do container Nginx no Docker Compose?

Para limitar os recursos de CPU e memória do container Nginx no Docker Compose, você pode usar a seção "resources" no arquivo docker-compose.yml. Por exemplo:

```
version: '3'
services:
  nginx:
    image: nginx
    ports:
      - "80:80"
    resources:
      limits:
        cpus: '0.5'
```

```
        memory: 512M
```

9) Como configurar o container Python para se conectar ao Redis usando a variável de ambiente correta especificada no Docker Compose?

Para configurar o container Python para se conectar ao Redis usando a variável de ambiente correta no Docker Compose, você pode definir a variável de ambiente no serviço Python no arquivo docker-compose.yml. Por exemplo:

```
version: '3'
services:
  python:
    image: python
    environment:
      - REDIS_HOST=redis
    ...
  redis:
    image: redis
```

10) Como escalar o container Python no Docker Compose para lidar com um maior volume de mensagens na fila implementada em Redis?

Para escalar o container Python no Docker Compose para lidar com um maior volume de mensagens na fila implementada em Redis, você pode usar o comando `docker-compose up --scale` seguido pelo nome do serviço e o número de réplicas desejadas. Por exemplo:

```
docker-compose up --scale python=3
```