Avaliação 16

Aluno: João Daniel Ferreira da Silva

1) Como definir um volume no Docker Compose para persistir os dados do banco de dados PostgreSQL entre as execuções dos containers?

O volume deve ser inserido no arquivo docker-compose.yml e criado fora da seção services do arquivo e fazer referência no serviço do banco de dados PostgreSQL usando o formato volume:/var/lib/postgresql/data. Após isso, rodar o comando docker-compose up.

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

Na seção YAML do arquivo docker-compose.ym, localizar o serviço relacionado ao banco de dados PostgreSQL. Incluir postgres_password=senha e nginx_port=porta, logo após, rodar o comando docker-compose down e docker-compose up -d para reiniciar o docker compose.

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

No docker-compose.yml adicionar a seção network, inserir o nome da rede e o driver, inserir também o nome da rede na seção services, o serviço webapp será conectado a rede.

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

Adicionar um serviço `nginx`, mapeia-se a porta 80 do host para a porta 80 do Nginx dentro do contêiner e montar um arquivo de configuração personalizado `nginx.conf` para o diretório `/etc/nginx/nginx.conf` dentro do contêiner.

Criar um arquivo `nginx.conf` no mesmo diretório do arquivo `docker-compose.yml` com a configuração do proxy reverso.

Definir o upstream `backend` com os endereços dos serviços de destino `service1` e `service2`. Em seguida, no bloco `server`, configurar a rota padrão para encaminhar o tráfego para o upstream `backend`.

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?

Definir `db` para o banco de dados PostgreSQL e `python` para o serviço Python.

O serviço `python` possui a opção `depends_on` configurada com `db`. Isso indica que o serviço `python` depende do serviço `db` e, portanto, o serviço `db` será iniciado antes do serviço `python`. No entanto, o `depends_on` não garante que o banco de dados PostgreSQL esteja totalmente pronto para aceitar conexões antes do serviço Python iniciar.

É necessário implementar uma biblioteca como `wait-for-it.sh` ou o uso de bibliotecas específicas da linguagem para esperar a disponibilidade do banco de dados.

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

Define-se python para o serviço Python e redis para o serviço Redis. Criar um volume chamado `shared-data f`ora dos serviços para compartilhar dados entre eles. O Docker

Compose criará um volume chamado `shared-data` que será compartilhado entre os containers Python e Redis. Dentro dos containers, esse volume será montado nos caminhos especificados nos serviços.

No serviço Python, especificaro volume `shared-data` para ser montado no diretório `/app/data` dentro do contêiner. Isso permitirá que o código Python acesse e armazene dados nesse diretório.

No serviço Redis, especificar o volume `shared-data` para ser montado no diretório `/data` dentro do contêiner Redis. Isso permitirá que o Redis armazene os dados da fila de mensagens nesse diretório, que é compartilhado com o serviço Python.

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

Adicionar a opção `ports` para mapear a porta do Redis apenas para a rede interna. Remover qualquer mapeamento de porta para o host. Por exemplo:

especificar ip e portas em ports

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

Adicionar as opções `cpus` e `mem_limit` ao serviço Nginx para definir os limites de recursos. A opção `cpus` especifica a quantidade de recursos de CPU que o container Nginx pode usar.

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

Configurar o serviço Python com as variáveis de ambiente `REDIS_HOST` e `REDIS_PORT`. Definimos `REDIS_HOST` como `redis`, que é o nome do serviço Redis no Docker Compose, e `REDIS_PORT` como `6379`, a porta em que o Redis está ouvindo. No código Python, pode-se usar as variáveis de ambiente para se conectar ao Redis. As variáveis de ambiente `REDIS_HOST` e `REDIS_PORT` são lidas usando `os.getenv()` e passadas como argumentos para a função `redis.Redis()`.

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

Adicionar a opção `scale` ao serviço Python e especificar o número desejado de instâncias. Por exemplo, para escalar para 3 instâncias, O comando `--scale python=3` escala o serviço Python para 3 instâncias. O Docker Compose irá criar e executar várias instâncias do serviço Python, cada uma com seu próprio contêiner. O balanceamento de carga é realizado automaticamente pelo Docker Compose, distribuindo as solicitações entre as instâncias.