



**INSTITUTO FEDERAL**

Ceará  
Campus Fortaleza

### Avaliação 15

Aluno: Matheus Germano Cirino de Souza

Professor: Ricardo Duarte Taveira

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

R: Para definir um volume no Docker Compose e persistir os dados do PostgreSQL, adicione a seção volumes no arquivo docker-compose.yml, mapeando um diretório do container para um volume. Ao executar o comando docker-compose up, o volume será criado e os dados do banco de dados serão mantidos no volume, permitindo o acesso aos mesmos dados entre as execuções dos containers.

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

R: Adicione a seção environment em cada serviço desejado no arquivo docker-compose.yml. Defina as variáveis de ambiente usando o formato NOME\_VARIAVEL=valor. Por exemplo, para a senha do PostgreSQL, use POSTGRES\_PASSWORD=senha e para a porta do Nginx, use PORT=porta. Assim, você pode personalizar facilmente esses valores e controlar a configuração do seu ambiente de contêineres.

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

R: Para criar uma rede personalizada no Docker Compose para permitir a comunicação entre containers, adicione a seção networks no arquivo docker-compose.yml. Defina uma rede personalizada e associe os serviços desejados a essa rede. Ao executar o comando docker-compose up, a rede será criada e os containers poderão se comunicar entre si usando os nomes dos serviços como endereço de host.

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

R: Adicione o serviço Nginx no arquivo docker-compose.yml com as configurações necessárias. Crie um arquivo de configuração nginx.conf para definir as regras de redirecionamento. Ao executar o docker-compose up, o Nginx funcionará como um proxy reverso, redirecionando o tráfego para diferentes serviços com base nas configurações definidas no arquivo nginx.conf. Isso permite gerenciar vários serviços em um único ponto de entrada.

- 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?

R: Para especificar dependências entre serviços no Docker Compose e garantir que o banco de dados PostgreSQL esteja totalmente inicializado antes do Python iniciar, você pode usar a opção `depends_on`. No entanto, isso não garante a disponibilidade completa do banco de dados. Para isso, você pode adicionar um mecanismo de espera no código Python, utilizando bibliotecas como `psycopg2`, para aguardar a disponibilidade do banco de dados antes de continuar a execução do código Python. Combinando o uso de `depends_on` no Docker Compose e um mecanismo de espera no código Python, é possível garantir que o banco de dados PostgreSQL esteja pronto antes do serviço Python iniciar.

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

R: Para definir um volume compartilhado entre os containers Python e Redis no Docker Compose e armazenar os dados da fila de mensagens do Redis, adicione a seção `volumes` no arquivo `docker-compose.yml`. Crie um volume chamado `fila-de-mensagens` e mapeie-o para os diretórios `/app/fila` no container Python e `/data` no container Redis. Isso permitirá que os dados da fila sejam persistidos e compartilhados entre os containers, garantindo a disponibilidade e consistência dos dados, mesmo com reinicializações ou recriações dos containers.

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

R: Defina a configuração do serviço Redis com a opção `ports` vazia no arquivo `docker-compose.yml`. Dessa forma, o Redis estará acessível apenas dentro da rede interna do Docker Compose, permitindo que outros serviços se conectem a ele usando o nome do serviço como `hostname`. Conexões de fora da rede interna serão bloqueadas, garantindo que o Redis seja acessível apenas pelos serviços dentro do ambiente controlado do Docker Compose.

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

R: Para limitar os recursos de CPU e memória do container Nginx no Docker Compose, adicione as opções `cpus` e `mem_limit` no arquivo `docker-compose.yml`. Defina o valor desejado para `cpus` para limitar a porcentagem de uso de CPU e para `mem_limit` para limitar o uso de memória. Isso permite controlar e limitar os recursos consumidos pelo Nginx durante a execução.

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

R: Defina as variáveis de ambiente no arquivo `docker-compose.yml` para o serviço Python. No código Python, acesse essas variáveis de ambiente usando `os.environ.get()` para obter as informações de conexão do Redis. Isso permite que você se conecte ao Redis usando as informações de conexão especificadas no arquivo `docker-compose.yml` e evite codificar informações específicas do ambiente no código Python.

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

R: Para escalar o container Python no Docker Compose e lidar com um maior volume de mensagens na fila do Redis, adicione a opção `scale` no arquivo `docker-compose.yml` para o serviço Python, seguido pelo número de réplicas desejado. Em seguida, execute o comando `docker-compose up --scale python=N`, substituindo N pelo número desejado de réplicas. Isso criará múltiplas instâncias do container Python que se conectam ao mesmo Redis, permitindo processar um maior volume de mensagens. Certifique-se de projetar seu código para lidar com múltiplas instâncias de forma adequada e garantir a consistência dos dados no Redis.