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

É preciso utilizar a seção ‘volumes’ no arquivo baixado ‘docker-compose.yml’. Dentro do arquivo, usamos o serviço criado baseado na imagem do banco de dados, e podemos definir o volume dentro da seção ‘volumes’ que mapeia o diretório do sistema de arquivos local, então quando um dado entra no branco de dados, ele é armazenado localmente no host e com isso persiste 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?

Para especificar a senha do banco de dados, dentro do serviço ‘db’ na seção ‘environment’ é possível guardar a senha dentro da variável de ambiente “**POSTGRES_PASSWORD**”, podendo substituir a senha existente pela desejada. E para definir a porta do servidor, modifica-se o valor dentro da variável de ambiente “**NGINX_PORT**” no serviço ‘nginx’, quando ele é modificado, também deve-se alterar o valor da porta interna do container do Nginx na seção “ports”.

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

Por meio da criação de serviços que servirão como os aplicativos e criar uma rede própria, define-se a imagem que cada serviço usa e definindo a seção ‘networks’ dentro dos serviços atribui-se a mesma rede para ambas. Depois, na seção ‘networks’ fora dos serviços, define-se a rede criada com o driver ‘bridge’ que permite a comunicação entre containers no mesmo host.

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

É preciso criar o diretório ‘nginx’ no mesmo diretório onde está o arquivo ‘docker-compose.yml’, e dentro do diretório criado é preciso criar o arquivo ‘default.conf’ e adicionar a configuração do proxy reverso. Agora, no ‘docker-compose.yml’ inclui-se o serviço Nginx e monta-se o diretório ‘nginx’ como um volume para persistir os dados que serão salvos no host local, definindo as seções de imagem, rede e portas.

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?

Apesar do Docker Compose disponibilizar a definição de dependências entre serviços com a chave ‘depends_on’, essa seção não garante que um serviço esteja completamente inicializado antes do outro, então deve-se implementar um meio de espera no script de inicialização do Python. Com o pacote ‘psycopg2’ é possível criar uma função que tenta estabelecer uma conexão com o serviço do PostgreSQL e caso não consiga, tenta novamente depois de esperar o tempo especificado. A

função criada poderá então ser chamada no script de inicialização do Python e fará com que o serviço Python só se inicie após o PostgreSQL.

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

Dentro do arquivo ‘docker-compose.yml’ você pode definir um volume nomeado por exemplo, ‘redis_data’, que será compartilhado com o serviço do Python e do Redis. Dentro do serviço Python criado, o volume é montado no diretório ‘app/data’ para que o Python acesse e armazene os dados da fila de mensagens em tal diretório. E no serviço do Redis, o volume é montado no diretório ‘/data’ dentro do container Redis, o que fará com que os dados sejam armazenados neste local e compartilhados com o serviço do Python. Quando for feito o comando ‘docker compose up’, o Docker Compose criará o volume e o associará aos containers dos dois serviços, então os dados da fila de mensagens em Redis persistirão no volume compartilhado.

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

Deve ser criado um arquivo de configuração do Redis chamado ‘redis.conf’ no mesmo diretório do ‘docker-compose.yml’ e adicionar a linha de código “**bind 0.0.0.0**” para que o Redis aceite conexões de qualquer endereço IP, dentro ou fora da rede do Docker Compose. Então, no arquivo do docker compose, monta-se o arquivo de configuração ‘redis.conf’ nos serviços desejados e no do Redis, adiciona-se a seção ‘command’ para iniciar o Redis usando a configuração definida. Dessa forma, quando for feito o comando docker-compose up o Redis apenas aceitará conexões de containers dentro da mesma rede do Docker Compose.

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

Dentro do arquivo ‘docker-compose.yml’ é possível limitar os recursos usando as opções ‘cpus’, definindo de forma decimal a quantidade de núcleos que o container Nginx pode usar e a opção ‘memory’ para definir o limite máximo de memória RAM que o container pode consumir, usando valores de megabytes, gigabytes, etc.

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

Dentro do arquivo ‘docker-compose.yml’ na seção ‘environment’ é preciso adicionar a variável de ambiente ‘REDIS_HOST’ no serviço Python e atribuir a ela o valor ‘redis’, que deve ser o serviço no qual está sendo executado o Redis. No código Python é possível acessar a variável ‘REDIS_HOST’ por meio da biblioteca “os” e se conectar ao Redis.

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

No ‘docker-compose.yml’ adiciona-se a opção ‘scale’ no serviço do Python que será utilizado, deve-se atribuir então o valor de instâncias que devem ser criadas do mesmo serviço, dessa forma, com um número maior de instâncias, um volume maior de mensagens vai poder ser processada. Ao executar o comando ‘docker-compose up --scale **nome do serviço python=número de instâncias**’