**Atividade 15**

**Sistemas Operacionais de Redes 2**

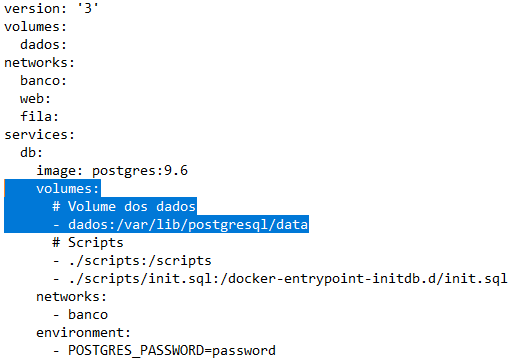
**Professor: Ricardo Duarte Taveira**

**Aluno: Erick Carneiro de Oliveira Rocha**

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

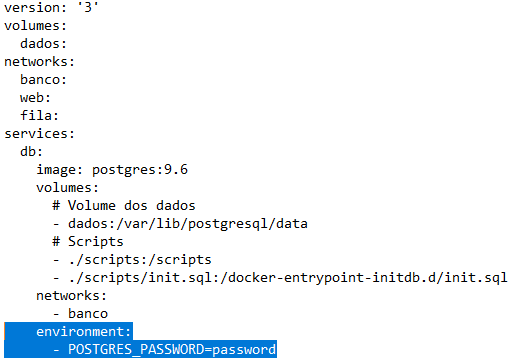
Resposta: Ao escrever o docker-compose.yml, especificar o caminho do repositório do volume no container Postgres.

O código é o seguinte:

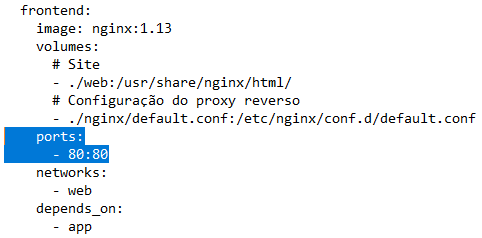


2) Como configurar variáveis de ambiente para especificar a senha do banco de dados PostgreSQL e a porta do servidor Nginx no Docker Compose?

Resposta: No arquivo docker-compose.yml, definir no container Postgres as seguintes linhas para denominar a senha a ser utilizada pelo serviço:

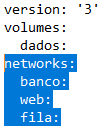


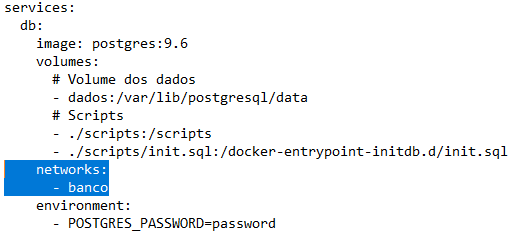
Em seguida, no container Nginx escrever as seguintes linhas para especificar a porta a ser utilizada:

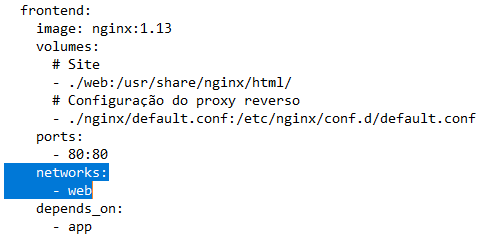


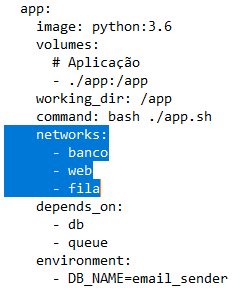
3) Como criar uma rede personalizada no Docker Compose para que os containers possam se comunicar entre si?

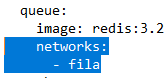
Resposta: No arquivo docker-compose.yml, definir as redes a serem utilizadas por cada serviço. Isso a feito por meio da declaração de networks.

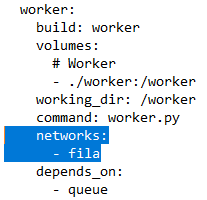






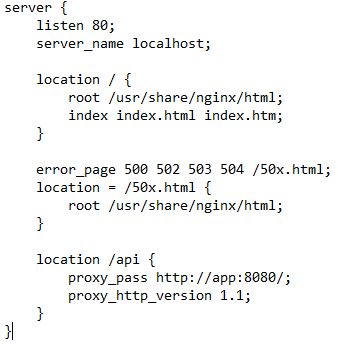






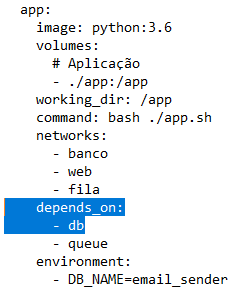
4) Como configurar o container Nginx para atuar como um proxy reverso para redirecionar o tráfego para diferentes serviços dentro do Docker Compose?

Resposta: Especificar os parâmetros de configuração próprios no arquivo nginx.conf dentro do diretório no qual o docker-compose utilizará para executar o container nginx. Tais parâmetros são os de roteamento de portas e endereços.



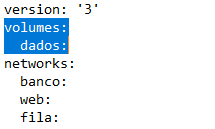
5) Como especificar dependências entre os serviços no Docker Compose para garantir que o banco de dados PostgreSQL esteja totalmente inicializado antes do Python iniciar?

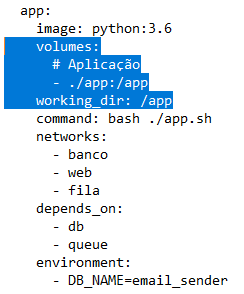
Resposta: Por meio da declaração de dependências escrita no arquivo docker-compose.yml, nas linhas especificadas como “depends\_on:”. Nesse caso específico do Python depender do Postgres já estar instanciado, é na seguinte linha:

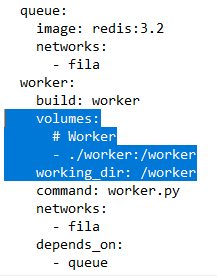


6) Como definir um volume compartilhado entre os containers Python e Redis para armazenar os dados da fila de mensagens implementada em Redis?

Resposta: Dentro do arquivo docker-compose.yml determinar um volume, e nos containers do Python e do Redis especificar o direcionamento para o volume definido.



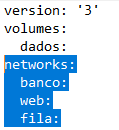


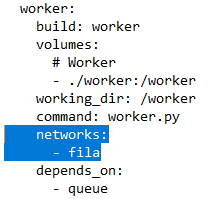


7) Como configurar o Redis para

aceitar conexões de outros containers apenas na rede interna do Docker Compose e não de fora?

Resposta: No arquivo docker-compose.yml especificar as redes para os containers por meio da definição networks, e na secção do container do Redis definir networks sendo essas as redes anteriormente especificadas.





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

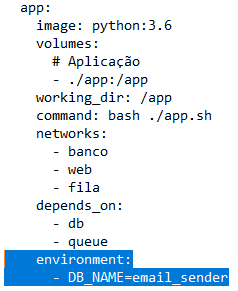
Resposta: No arquivo docker-compose.yml definir no container Nginx as seguintes seções



A opção “cpus:” determina o limite de recursos de CPU a serem utilizados, já mem\_limit determina o limite de recursos de memória.

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

Resposta: Dentro docker-compose.yml, no container Python definir a secção Environment.



Já no arquivo do script Python que esse container vai executar, as seguintes linhas são responsáveis pelos parâmetros de conexão com o Redis.



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

Resposta: No arquivo docker-compose.yml definir no container Python a secção deploy com a propriedade réplicas com o número desejado para escalamento.



Neste caso, o container estará escalado para 5 instâncias.