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 garantir a persistência dos dados no banco de dados PostgreSQL entre as execuções dos containers, deve-se usar a diretiva **volumes** no arquivo **docker-compose.yml**
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 e especificar a senha do banco de dados PostgreSQL e a porta do servidor Nginx, usa-se a diretiva **environment** no arquivo **docker-compose.yml**
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 e permitir a comunicação entre containers, utiliza-se a diretiva **networks** no arquivo **docker-compose.yml**
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 para redirecionar o tráfego para diferentes serviços dentro do Docker Compose, cria-se um arquivo de configuração para o Nginx, montando-o como um volume no container
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 serviços no Docker Compose e garantir que o banco de dados PostgreSQL esteja totalmente inicializado antes do serviço Python iniciar, usa-se a diretiva **depends_on** no arquivos **docker-compose.yml**
   Vale ressaltar que a diretiva **depends_on** não aguarda até que o serviço dependente esteja realmente "pronto" ou totalmente inicializado. Ela apenas define a ordem de inicialização dos serviços
   Para solucionar esse problema, uma abordagem comum é adicionar um script de inicialização personalizado no serviço **python** que aguarde explicitamente até que o banco de dados esteja pronto antes de continuar
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 e armazenar os dados da fila de mensagem implementada no Redis, utiliza-se a diretiva **volumes** no arquivos **docker-compose.yml**
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 de outros containers na rede interna do Docker Compose e não de fora, usa-se a diretiva **bind** no arquivo de configuração do Redis

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, utiliza-se as diretivas **cpus** e **mem_limit** no arquivo **docker-compose.yml**
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 especificada no Docker Compose, usa-se o pacote Python **redis** para acessar a variável de ambiente definida no **docker-compose.yml**
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 e lidar com um maior volume de mensagens na fila implementada em Redis, utiliza-se a funcionalidade de dimensionamento do Docker Compose
    O dimensionamento do Docker Compose, por sua vez, refere-se a configuração e ajuste dos recurso necessários para executar e dimensionar aplicativos em containers usando o Docker Compose