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

   Adicionar a configuração ao arquivo docker-compose.yml:

```
version: '3'
services:
  db:
    image: postgres
    volumes:
      - pgdata:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=seu_password_aqui
volumes:
  pgdata:
```

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

   Usar a seção "environment" dentro de cada serviço no arquivo docker-compose.yml:

```
version: '3'
services:
  db:
    image: postgres
    environment:
      - POSTGRES_PASSWORD=senha_do_banco
  nginx:
    image: nginx
    ports:
      - "8080:80"
    environment:
      - NGINX_PORT=80
```

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

   Adicionar a seção "networks" ao seu arquivo docker-compose.yml:

```
version: '3'
services:
  db:
    image: postgres
    networks:
      - mynetwork
  nginx:
    image: nginx
    networks:
      - mynetwork
networks:
  mynetwork:
```

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

   Criar um arquivo de configuração do Nginx personalizado, por exemplo, chamado "nginx.conf", com as configurações desejadas para o proxy reverso:

```
events {}

http {
    server {
        listen 80;

        location /servico1 {
            proxy_pass http://servico1:8000;
        }

        location /servico2 {
            proxy_pass http://servico2:9000;
        }
    }
}
```

   No arquivo docker-compose.yml, adicionar o serviço Nginx e montar o arquivo de configuração personalizado como um volume:

```
version: '3'
services:
  nginx:
    image: nginx
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - servico1
      - servico2

  servico1:
    # Configurações do serviço 1

  servico2:
    # Configurações do serviço 2
```

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?

   Usar a opção "depends_on" no arquivo docker-compose.yml:

```
version: '3'
services:
  db:
    image: postgres
    environment:
      - POSTGRES_PASSWORD=senha_do_banco
  python:
    build: .
    depends_on:
      - db
```

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

   Utilizar um volume nomeado:

```
version: '3'
services:
  python:
    build: .
    volumes:
      - data-volume:/app/data

  redis:
    image: redis
    volumes:
      - data-volume:/data

volumes:
  data-volume:
```

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

   Criar um arquivo de configuração do Redis personalizado, por exemplo, redis.conf, para definir as configurações desejadas:

```
bind 127.0.0.1
protected-mode yes
```

   No arquivo docker-compose.yml, adicionar o serviço do Redis e montar o arquivo de configuração personalizado como um volume:

```
version: '3'
services:
  redis:
    image: redis
    volumes:
      - ./redis.conf:/usr/local/etc/redis/redis.conf
    command: redis-server /usr/local/etc/redis/redis.conf
```

   Executar "docker-compose up" para iniciar o serviço do Redis com a configuração personalizada.

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

   Usar as opções "cpus" e "mem_limit" no arquivo docker-compose.yml:

```
version: '3'
services:
  nginx:
    image: nginx
    ports:
      - "80:80"
    cpus: 0.5
    mem_limit: 256m
```

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

   Utilizar as variáveis de ambiente definidas no arquivo docker-compose.yml e acessá-las no código Python.

   No arquivo docker-compose.yml, definie a variável de ambiente para a conexão do Redis no serviço Python:

```
version: '3'
services:
  python:
    build: .
    environment:
      - REDIS_HOST=redis
      - REDIS_PORT=6379
```

   No código Python, deve-se acessar essas variáveis de ambiente para configurar a conexão com o Redis:

```
import os
import redis

redis_host = os.getenv('REDIS_HOST', 'redis')
redis_port = os.getenv('REDIS_PORT', 6379)

# Configuração da conexão com o Redis
r = redis.Redis(host=redis_host, port=redis_port)

# Utilize a conexão com o Redis aqui
```

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

    Usar a opção "scale" do Docker Compose.

    No arquivo docker-compose.yml, definir o serviço Python como escalável adicionando a opção "scale":

```
version: '3'
services:
  python:
    build: .
    environment:
      - REDIS_HOST=redis
      - REDIS_PORT=6379
    deploy:
      replicas: 5
```

    Executar o comando "docker-compose up --scale python=5" para iniciar o serviço Python com o número de réplicas especificado.