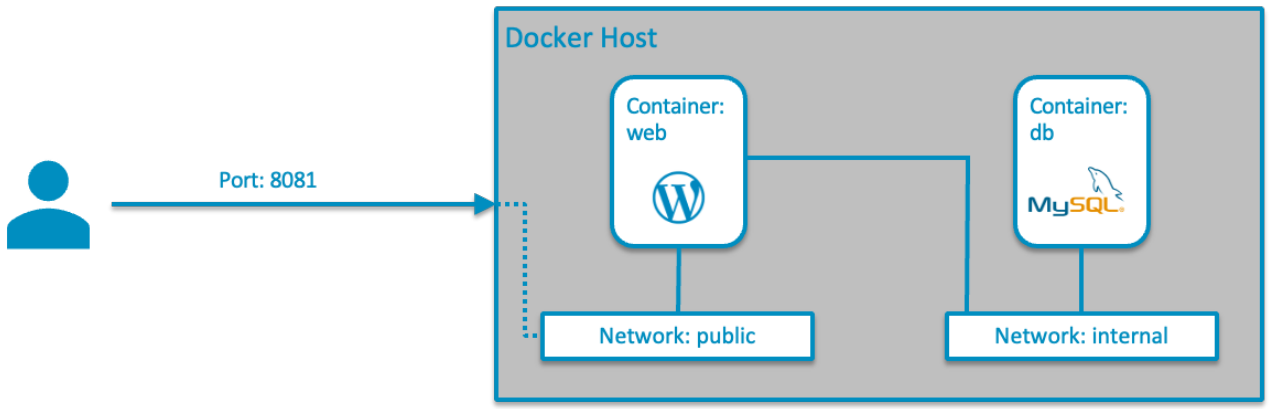
Certified Tech Developer
The Ultimate Degree

Objetivos

Construir um ambiente WordPress com Docker nos permitirá ver a tecnologia em ação e entender seu potencial.

O que vamos construir?

Um ambiente semelhante a este:



Para eles vamos:

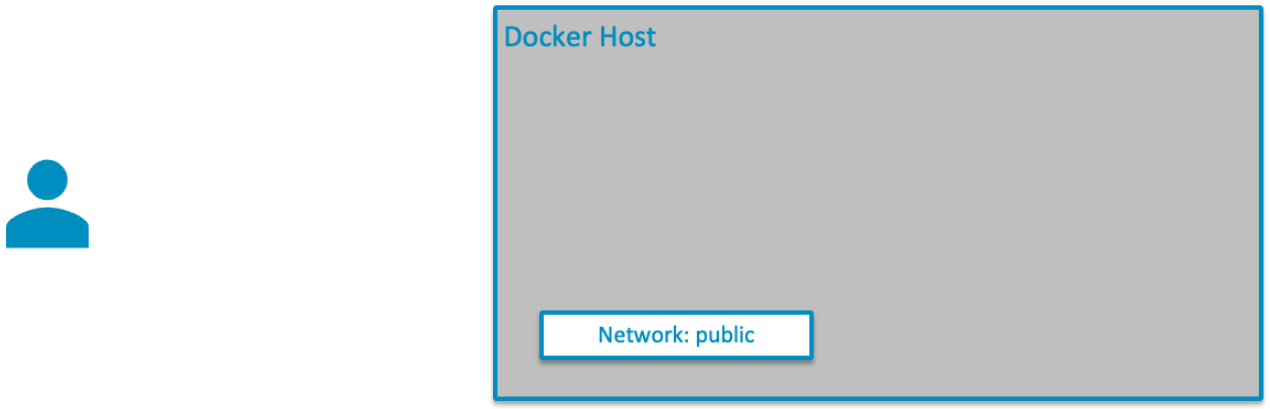
- Construir 2 redes do tipo 'bridge', a primeira chamada 'public', e a segunda chamada 'internal'.
    - A rede 'internal' deve permitir apenas tráfego entre os containers conectados a ela e não permitir tráfego externo.
- Crie um container chamado 'db' com a imagem do MySQL que deve estar conectada à rede 'internal'.
- Crie um container chamado 'web' com a imagem do WordPress que deve estar conectada às duas redes.
    - Este container deve estar exposto na porta 8081 do host para poder acessar o site WordPress.
- Tanto o container 'db' quanto o container 'web' devem ser configurados para que possam conversar entre si (veremos que ambos os serviços podem ser configurados usando variáveis de ambiente).

Instruções

A primeira coisa que vamos fazer é criar uma rede chamada 'public'. Para isso, vamos usar o seguinte comando:

```
docker network create public
```

Executando este comando chegamos a este estado:



Dica: Se analisarmos o comando que acabamos de executar, vamos ver que é composto por quatro termos:

- docker: para invocar o já famoso cliente Docker ou Docker CLI (Command Line Interface)
- network: este termo engloba todas as operações relacionadas às redes docker.
- create: ao usar este termo, no contexto de 'network', estamos indicando que queremos criar uma rede.
- public: finalmente, este termo é um parâmetro do comando 'create' e indica o nome que queremos atribuir a essa rede que estamos criando.

Para saber mais sobre os comandos que estão no grupo 'network' podemos executar:

```
docker network --help
```

Para saber mais sobre os parâmetros que podemos usar para o comando 'create' podemos executar:

```
docker network create --help
```

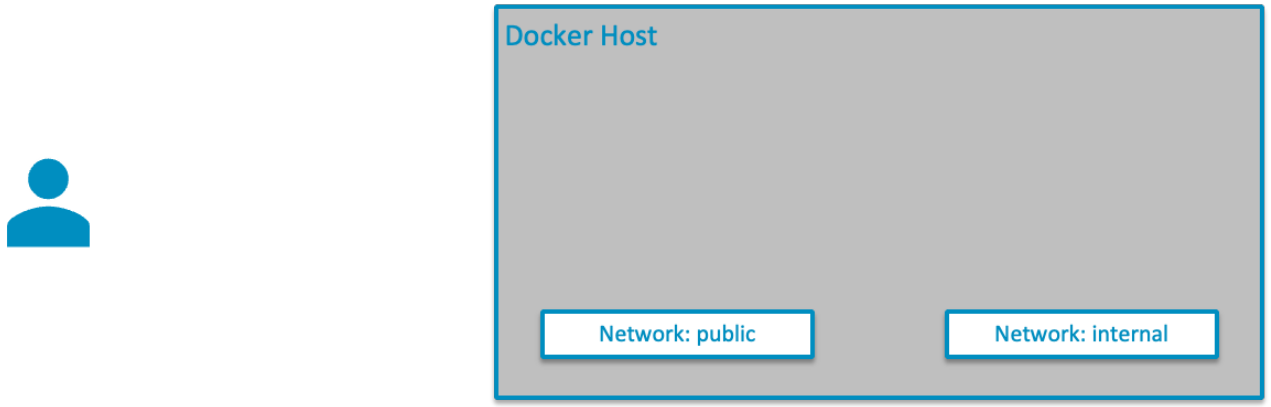
Pergunta: Dissemos que as redes que iríamos criar seriam do tipo 'bridge'. Como o comando '`docker network create`' sabe qual o tipo de rede deve ser?

Resposta: A menos que indique o contrário na criação, todas as redes criadas no Docker são definidas como 'bridge' por padrão.

Em seguida, vamos prosseguir com a criação da segunda rede, mas desta vez o comando será um pouco diferente:

```
docker network create internal --internal
```

Tendo executado este comando, estamos agora no seguinte estado:



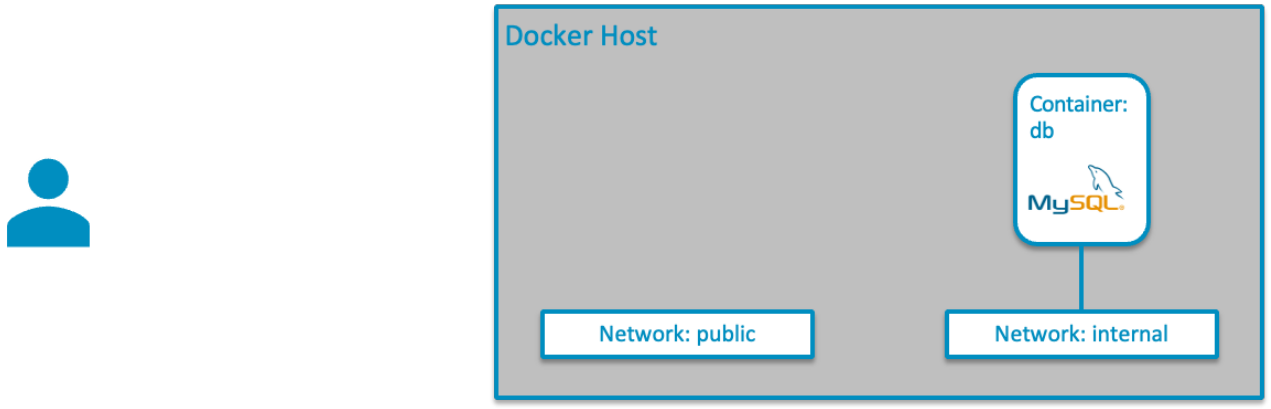
Dica: Se analisarmos este segundo comando que acabamos de executar, veremos que é composto por cinco termos, sendo os quatro primeiros iguais ao comando anterior (o quarto termo varia, mas é porque vamos atribuir um nome diferente para a rede), mais um novo termo:

- `--internal`: este termo também é um parâmetro do comando 'create' e indica que a rede que vamos criar não permitirá que os containers que conectamos a ela seja acessível de fora da rede. Eles só podem ser acessados por outros containers conectados à mesma rede.

Neste ponto, já deveríamos ter criado nossas redes. Vamos instanciar nosso primeiro container, o banco de dados.

```
docker run -d --name db --network internal -e
MYSQL_ROOT_PASSWORD=my-secret-pw -e MYSQL_USER=wpuser -e
MYSQL_PASSWORD=my-secret-pw -e MYSQL_DATABASE=wordpressdb -e
WORDPRESS_TABLE_PREFIX=hj mysql:5.7
```

Neste ponto, a implementação ficará assim:



Dica: Este comando tem muitos termos. Vamos analisá-los:

- docker: novamente, para invocar o cliente Docker ou Docker CLI (Command Line Interface)
- run: este termo indica que vamos executar um novo container.
- -d: usando este termo estamos indicando ao Docker que vamos rodar o novo container no modo 'de-attached'(desanexado), o que significa que o container vai rodar em segundo plano.
- --name: com este parâmetro, atribuímos um nome ao nosso container, neste caso 'db'.
- --network: usamos este parâmetro para indicar a qual rede vamos conectar nosso novo container.
- -e: usamos para definir variáveis de ambiente dentro do container. O que essas variáveis fazem, quais variáveis precisamos definir ou como elas são consumidas pelos aplicativos executados dentro do container dependerá inteiramente do aplicativo. Para saber mais sobre isso, o que podemos fazer é visitar a página do Docker Hub da imagem que vamos executar e todas as instruções associadas às variáveis disponíveis estarão documentadas lá.
- mysql:5.7: é o nome da imagem (mysql) e tag (5.7) da imagem que vamos executar.

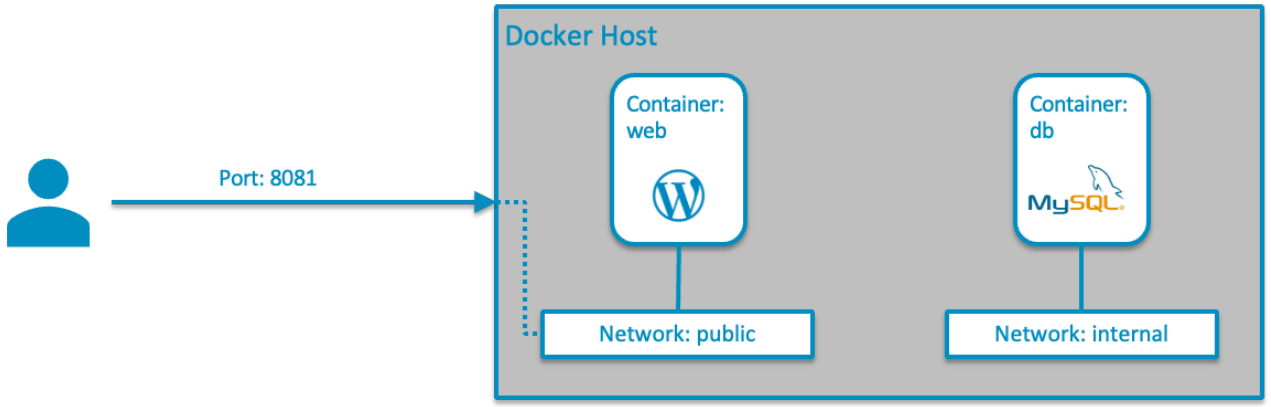
Para saber mais sobre os parâmetros que podemos usar para o comando 'run', podemos executar:

```
docker run --help
```

Neste ponto, já temos as duas redes criadas e um dos containers. Agora, vamos criar um container adicional para WordPress e para isso, vamos executar o seguinte comando:

```
docker create -p 8081:80 --name web --network public -e
WORDPRESS_DB_HOST=db:3306 -e WORDPRESS_DB_USER=wpuser -e
WORDPRESS_DB_PASSWORD=my-secret-pw -e WORDPRESS_DB_NAME=wordpressdb -e
WORDPRESS_TABLE_PREFIX=hj wordpress:latest
```

Criando este novo container, levamos a implementação para o seguinte estado:



Dica: Este novo comando tem tantos termos como o anterior, mas com algumas mudanças sutis. Vamos dividi-los:

- create: ao invés de chamar o comando 'run', estamos chamando o comando 'create'. Fazemos isso porque queremos criar o container sem iniciá-lo. Por quê? Porque antes de iniciá-lo, queremos conectá-lo a outra rede (veremos isso no próximo comando que executarmos).
- -p: Este parâmetro nos permite mapear uma porta no container, neste caso a porta 80, para uma porta em nossa estação de trabalho, neste caso a porta 8081. Assim, uma vez que este container é iniciado, podemos visitar a porta 8081 (ex : https: //127.0.0.1:8081) e fazer login no WordPress.

Para saber mais sobre os parâmetros que podemos usar para o comando 'create'

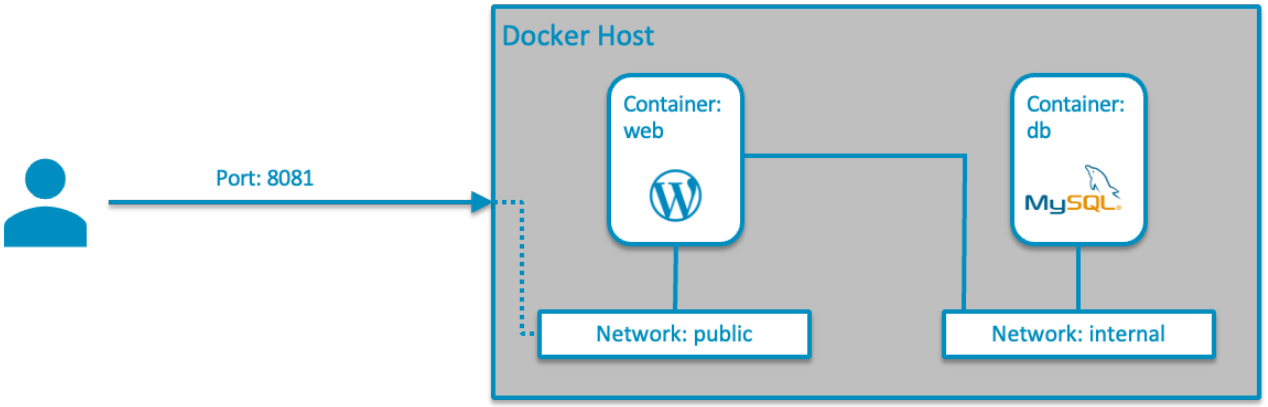
podemos executar:

```
docker create --help
```

Agora, vamos conectar nosso novo container à rede 'internal', e podemos fazer isso usando o seguinte comando:

```
docker network connect internal web
```

O que nos leva ao estado desejado:



Dica: Novamente invocamos um comando do espaço 'docker network', porém neste caso, não foi 'create', mas sim 'connect'. Vamos analisar o que conseguimos com este comando e os argumentos que passamos para ele:

- connect: usando este comando, dentro do contexto de 'network', estamos indicando que queremos conectar um container existente a uma rede existente.
- internal: é o nome da rede à qual vamos conectar o container.
- web: é o nome do container que vamos conectar à rede.

Para saber mais sobre os parâmetros que podemos usar para o comando 'connect' podemos executar:

```
docker network connect --help
```

Agora só temos que iniciar o container 'web', o que podemos fazer com o seguinte comando:

```
docker start web
```

Tendo concluído todos esses passos com sucesso, podemos visitar o URL http://127.0.0.1:8081 e devemos encontrar o portal de configuração inicial do WordPress.

Por fim, para remover tudo o que acabamos de criar, podemos executar os seguintes comandos:

```
docker stop web
docker stop db
docker rm web
docker rm db
docker network rm public
docker network rm internal
docker rmi mysql:5.7
docker rmi wordpress:latest
```