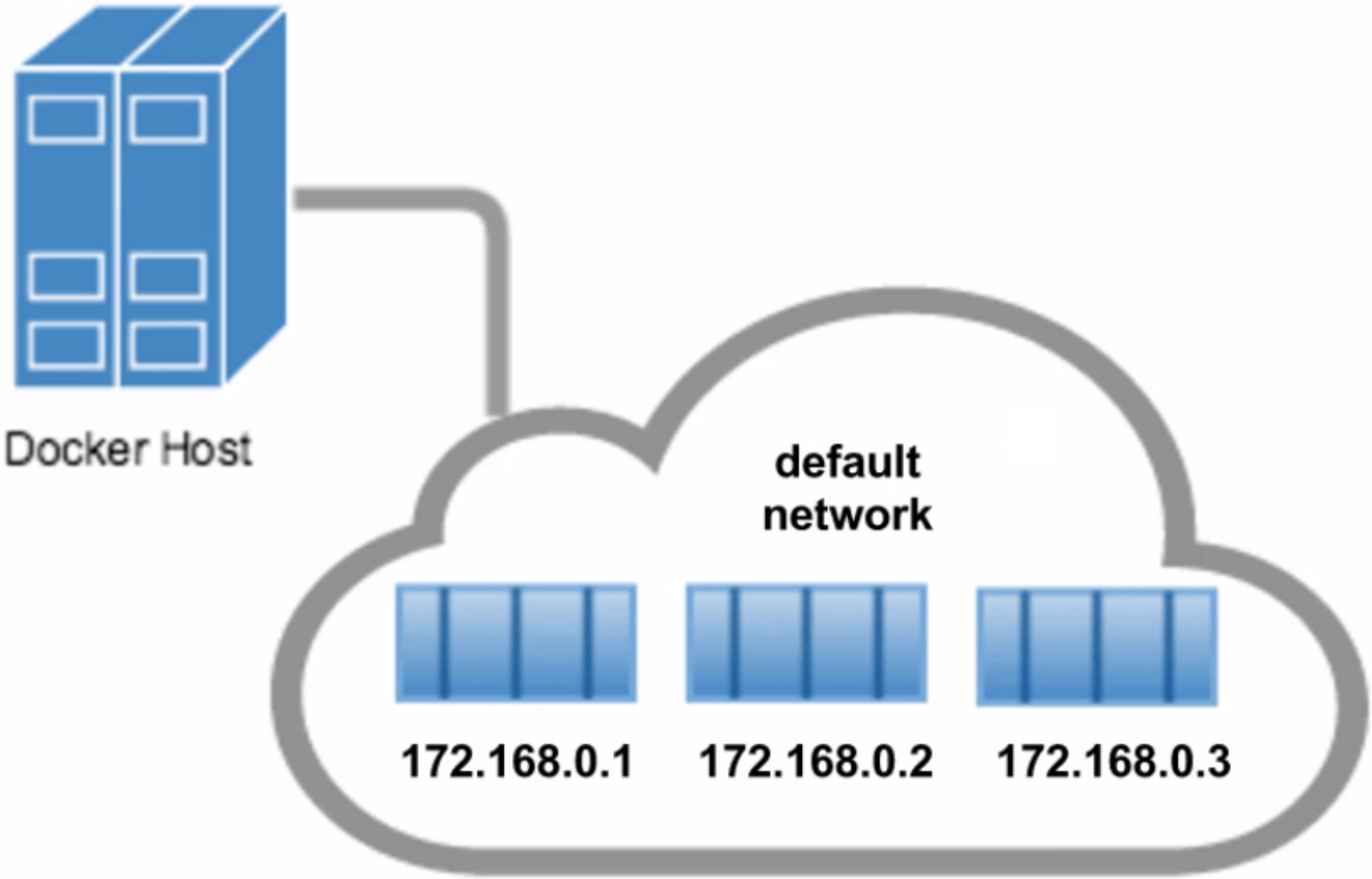
Transcrição

Neste capítulo, veremos como funciona a rede, e como fazemos para interligar diversos *containers* no Docker. Normalmente uma aplicação é composta por diversas partes, sejam elas o *load balancer*/proxy, a aplicação em si, um banco de dados, etc. Quando estamos trabalhando com *containers*, é bem comum separarmos cada uma dessas partes em um *container* específico, para cada *container* ficar com somente uma única responsabilidade.

Mas se temos uma parte da nossa aplicação em cada *container*, como podemos fazer para essas partes falarem entre elas? Pois para a nossa aplicação funcionar como um todo, os *containers* precisam trocar dados entre eles.

Redes com Docker

A boa notícia é que no Docker, por padrão, já existe uma ***default network***. Isso significa que, quando criamos os nossos *containers*, por padrão eles funcionam na mesma rede:



Para verificar isso, vamos subir um *container* com Ubuntu:

```
docker run -it ubuntu
```

COPIAR CÓDIGO

Em outro terminal, vamos verificar o **id** desse *container* através do comando `docker ps` , e com ele em mãos, vamos passá-lo para o comando `docker inspect` . Na saída desse comando, em ***NetworkSettings***, vemos que o *container* está na rede padrão ***bridge***, rede em que ficam todos os *containers* que criamos.

Voltando ao terminal do *container*, se executarmos o comando `hostname -i` vemos o IP atribuído a ele pela rede local do Docker:

```
root@973feeeeb1df:/# hostname -i
172.17.0.2
```

COPIAR CÓDIGO

Então, dentro dessa rede local, os *containers* podem se comunicar através desses IPs. Para comprovar isso, vamos deixar esse *container* rodando e criar um novo:

```
docker run -it ubuntu
```

COPIAR CÓDIGO

E vamos verificar o seu IP:

```
root@dd316a9f585f:/# hostname -i
172.17.0.3
```

COPIAR CÓDIGO

Agora, no primeiro *container*, vamos instalar o pacote **iputils-ping** para podermos executar o comando `ping` para verificar a comunicação entre os *containers*:

```
root@973feeeeb1df:/# apt-get update && apt-get install iputils-ping
```

COPIAR CÓDIGO

Após o término da instalação, executamos o comando `ping`, passando para ele o IP do segundo *container*. Para interromper o comando, utilizamos o atalho **CTRL + C**:

```
root@973feeeeb1df:/# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.180 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.133 ms
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.148 ms
^C
--- 172.17.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.133/0.153/0.180/0.024 ms
```

COPIAR CÓDIGO

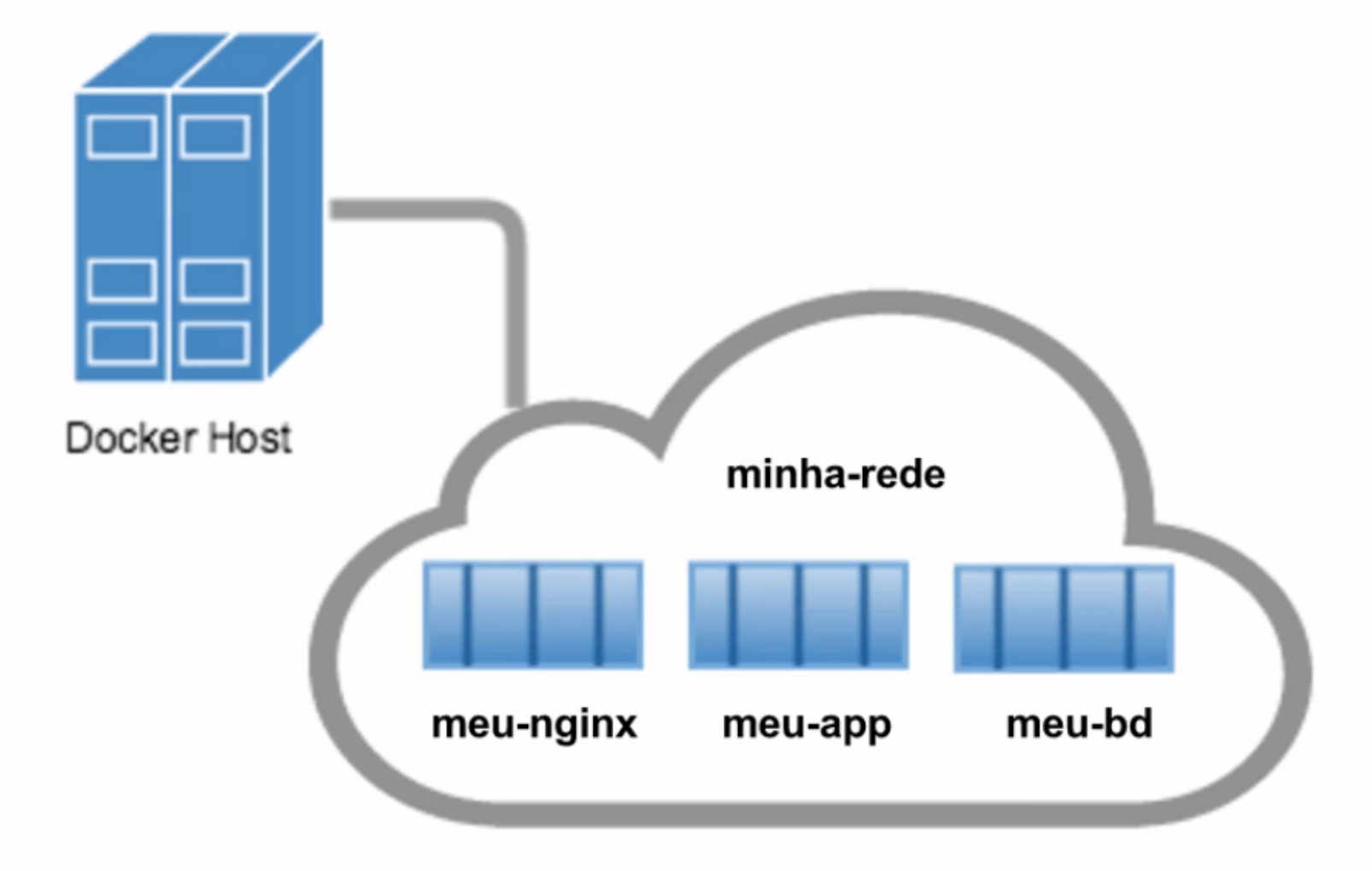
Assim, podemos ver que os *containers* estão conseguindo se comunicar entre eles.

Comunicação entre containers utilizando os seus nomes

Então, o Docker criar uma rede virtual, em que todos os *containers* fazem parte dela, com os IPs automaticamente atribuídos. Mas quando os IPs são atribuídos, cada hora em que subirmos um *container*, ele irá receber um IP novo, que será determinado pelo Docker. Logo, se não sabemos qual o IP que será atribuído, isso não é muito útil quando queremos fazer a comunicação entre os *containers*. Por exemplo, podemos querer colocar dentro do aplicativo o endereço exato do banco de dados, e para saber exatamente o endereço do banco de dados, devemos configurar um nome para aquele *container*.

Mas nomear um *container* nós já sabemos, basta adicionar o `--name`, passando o nome que queremos na hora da criação do *container*, certo? Apesar de conseguirmos dar um nome a um *container*, a rede do Docker não permite com que atribuamos um ***hostname*** a um *container*, diferentemente de quando criamos a nossa própria rede.

Na rede padrão do Docker, só podemos realizar a comunicação utilizando IPs, mas se criarmos a nossa própria rede, podemos "batizar" os nossos *containers*, e realizar a comunicação entre eles utilizando os seus nomes:



Isso não pode ser feito na rede padrão do Docker, somente quando criamos a nossa própria rede.

Criando a nossa própria rede do Docker

Então, vamos criar a nossa própria rede, através do comando `docker network create`, mas não é só isso, para esse comando também precisamos dizer qual *driver* vamos utilizar. Para o padrão que vimos, de ter uma nuvem e os *containers* compartilhando a rede, devemos utilizar o *driver* de ***bridge***.

Especificamos o driver através do `--driver` e após isso nós dizemos o nome da rede. Um exemplo do comando é o seguinte:

```
docker network create --driver bridge minha-rede
```

COPIAR CÓDIGO

Agora, quando criamos um *container*, ao invés de deixarmos ele ser associado à rede padrão do Docker, atrelamos à rede que acabamos de criar, através da *flag* `--network`. Vamos aproveitar e nomear o *container*:

```
docker run -it --name meu-container-de-ubuntu --network minha-rede ubuntu
```

COPIAR CÓDIGO

Agora, se executarmos o comando `docker inspect meu-container-de-ubuntu`, podemos ver em ***NetworkSettings*** o *container* está na rede **minha-rede**. E para testar a comunicação entre os *containers* na nossa rede, vamos abrir outro terminal e criar um segundo *container*:

```
docker run -it --name segundo-ubuntu --network minha-rede ubuntu
```

COPIAR CÓDIGO

Agora, no **segundo-ubuntu**, instalamos o **ping** e testamos a comunicação com o **meu-container-de-ubuntu**:

```
root@00f93075d079:/# ping meu-container-de-ubuntu
PING meu-container-de-ubuntu (172.18.0.2) 56(84) bytes of data.
```

```
64 bytes from meu-container-de-ubuntu.minha-rede (172.18.0.2): icmp_seq=1 ttl=64 time=0.210 ms
64 bytes from meu-container-de-ubuntu.minha-rede (172.18.0.2): icmp_seq=2 ttl=64 time=0.148 ms
64 bytes from meu-container-de-ubuntu.minha-rede (172.18.0.2): icmp_seq=3 ttl=64 time=0.138 ms
^C
--- meu-container-de-ubuntu ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.138/0.165/0.210/0.033 ms
```

COPIAR CÓDIGO

Conseguimos realizar a comunicação entre os *containers* utilizando somente os seus nomes. É como se o **Docker Host**, o ambiente que está rodando os *containers*, criasse uma rede local chamada **minha-rede**, e o nome do *container* será utilizado como se fosse um *hostname*.

Mas lembrando que só conseguimos fazer isso em redes próprias, redes que criamos, isso não é possível na rede padrão dos *containers*.