

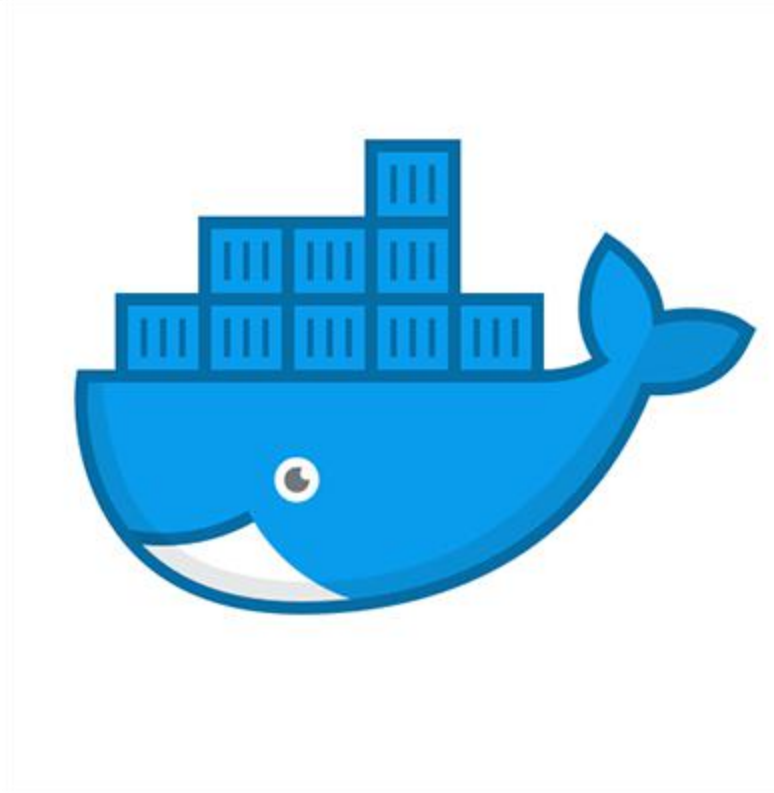


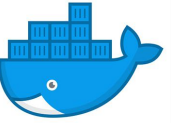
# Geek University

**Evolua seu lado geek!**

[www.geekuniversity.com.br](http://www.geekuniversity.com.br)

# Como funciona a comunicação em uma rede no Docker





# Como funciona a comunicação em uma rede no Docker

Computadores se comunicam em uma rede através de endereços IP (Internet Protocol)

Os endereços IP servem de endereçamento do dispositivo e de identificador do dispositivo na rede.

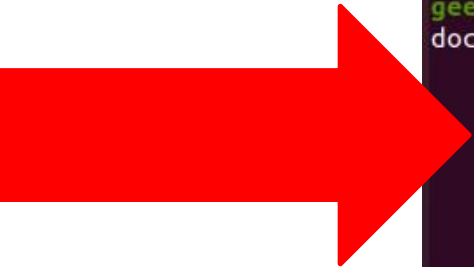
Não pode haver repetição de IP em uma rede.





# Como funciona a comunicação em uma rede no Docker

Quando trabalhamos com Docker este cria uma rede interna, em outra classe, e distribui IPs aos containers criados.



```
geek@university:~$ docker ps
CONTAINER ID          IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
geek@university:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
    ether 02:42:f0:7e:3a:86  txqueuelen 0  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.100.7  netmask 255.255.255.0  broadcast 192.168.100.255
    inet6 fe80::b2aa:6648:b9c9:1384  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:3b:51:d8  txqueuelen 1000  (Ethernet)
    RX packets 2716  bytes 3671047 (3.6 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 766  bytes 92284 (92.2 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 188  bytes 16412 (16.4 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 188  bytes 16412 (16.4 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

geek@university:~$
```



# Como funciona a comunicação em uma rede no Docker

Quando trabalhamos com Docker este cria uma rede interna, em outra classe, e distribui IPs aos containers criados.

```
geek@university:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
d9337458149f        guniversity/servidor_web:v1  "/docker-entrypoint..."  7 seconds ago      Up 5 seconds        0.0.0.0:8080->80/tcp  servidor_web

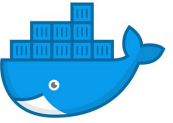
geek@university:~$ ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:f0ff:fe7e:3a86 prefixlen 64 scopeid 0x20<link>
    ether 02:42:f0:7e:3a:86 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 3619 (3.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.7 netmask 255.255.255.0 broadcast 192.168.100.255
    inet6 fe80::b2aa:6648:b9c9:1384 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:3b:51:d8 txqueuelen 1000 (Ethernet)
    RX packets 9786 bytes 13763785 (13.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1965 bytes 203980 (203.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 247 bytes 22713 (22.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 247 bytes 22713 (22.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vethe8a84eb: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::4c93:21ff:fe82:7006 prefixlen 64 scopeid 0x20<link>
    ether 4e:93:21:82:70:06 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 42 bytes 5986 (5.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

geek@university:~$
```

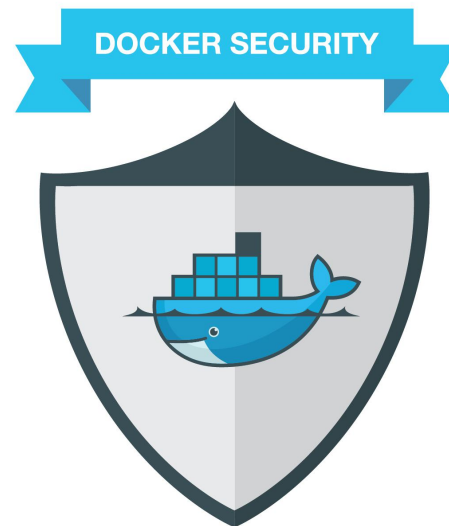


# Como funciona a comunicação em uma rede no Docker

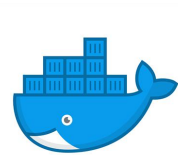
Quando estamos trabalhando com vários containers em um projeto, devemos levar em conta, por motivos de segurança, quais containers precisam/podem ter seus serviços/IPs/portas expostas e quais não.

Por exemplo, um serviço HTTP precisa ter a porta exposta para que as pessoas acessem a aplicação web.

Mas um banco de dados que será acessado apenas pela aplicação web e não pelas pessoas fora do contexto do Docker não deveria ter seu IP/porta exposta.

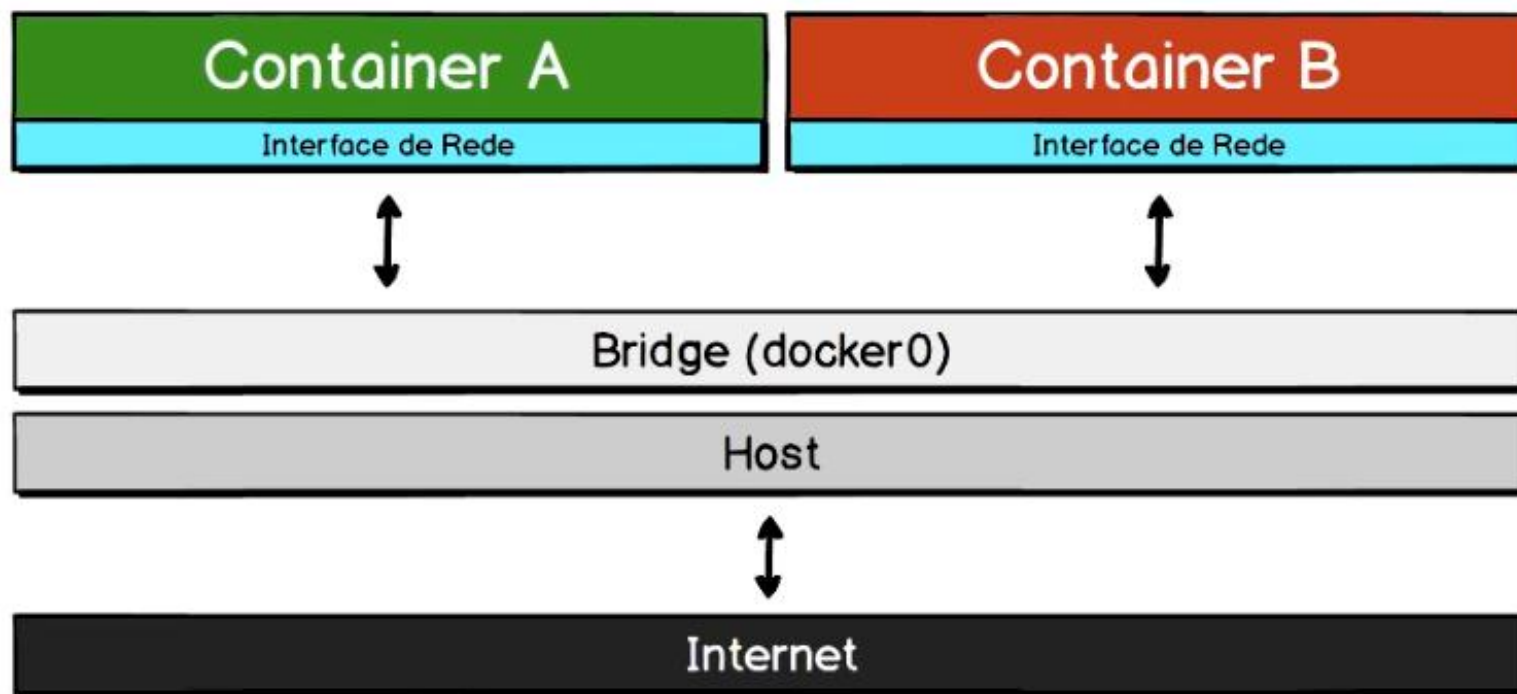






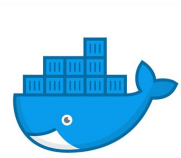
# Como funciona a comunicação em uma rede no Docker

O Docker por padrão utiliza um “driver” de rede conhecido como Bridge



Desta forma, ao criarmos um container, este irá receber uma interface de rede própria e um endereçamento IP.

O Docker dará então uma “ponte” entre o container e o host (nosso computador).



# Como funciona a comunicação em uma rede no Docker

## Outros tipos de rede no Docker

Além do modelo padrão, “bridge” podemos ter no Docker os seguintes tipos:

- None Network, no qual o próprio nome diz que não haverá nenhuma rede;
- Host Network, na qual ao invés de termos a “ponte” entre os containers e o Host (nosso computador) o acesso é direto entre o container e o host;
- Overlay Network, usado com o Docker Swarm que é um orquestrador\* de containers;

\* O Kubernetes é o orquestrador de containers padrão do mercado;





# Geek University

**Evolua seu lado geek!**

[www.geekuniversity.com.br](http://www.geekuniversity.com.br)