- TRABALHANDO COM CONTAINERS

- Para listar todos os containers em execução:
$ docker ps

```
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$
```

- Para listar todos os containers em execução ou não:
$ docker ps -a

```
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS
PORTS     NAMES
62a19bf4dfff   alpine         "/bin/sh"     40 hours ago    Exited (0) 40 hours ago
webhost
581647ff3cff   ubuntu:14.04   "/bin/bash"   3 days ago      Exited (0) 40 hours ago
meu_ubuntu
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$
```

Os números em hexadecimal “`62a19bf4dfff` e “`581647ff3cff`”, são os “ids” dos containers, neste exemplo, possuem 2 containers. Um com a imagem do “alpine” com o nome “webhost” e o outro com a imagem do “ubuntu versão 14.04” com o nome “meu_ubuntu”.

- Baixando e executando o primeiro exemplo de container (hello-world):
$ docker run hello-world

```
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$
```

Após as seguintes mensagens, o container é “morto”

- Baixando e executando o container do ubuntu (última versão), com o comando "-ti" entraremos no modo terminal do container:

$ docker run -ti ubuntu /bin/bash

- Para sair finalizando o container: **"Ctrl+d"** ou digite **“exit”**

- Para sair do container sem finalizar: **"Ctrl+p+q"**

- Baixando e executando o container do alpine (última versão):
$ docker run -ti alpine sh

- Para pausar a execução do container:
$ docker pause “id ou nome do container”

- Para parar a execução do container:
$ docker stop “id ou nome do container”

- Para continuar a execução do container:
$ docker start “id ou nome do container”

- Para voltar para dentro do container:
$ docker attach "id ou nome do container"

- Para parar um container em execussão e exclui-lo:
$ docker rm -f “id ou nome do container”

- Para excluir um container sem estar em execussão:
$ docker rm “id ou nome do container”

Exemplo, vamos apagar o container hello-world:

```
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS
PORTS     NAMES
05c3959b7fc2   hello-world    "/hello"      2 minutes ago   Exited (0) 2 minutes ago
quirky_einstein
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$ docker rm 05c3959b7fc2
05c3959b7fc2
fabricio@notebook-lg:~/GitHub/Docker-Help/Apostila$
```

- Baixando e executando o container do ubuntu versão14.04, dando um nome específico ao container e entrando no modo terminal:
$ docker run -ti --name meu_ubuntu ubuntu:14.04 /bin/bash

- Para verificar os “ids” dos containers em execução:
$ docker ps -q

- Para verificar os “ids” de todos os containers em execução:
$ docker ps -aq

- Para parar todos os containers de uma vez só:
$ docker stop $(docker ps -q)

- Para executar todos os containers de uma vez só:
$ docker start $(docker ps -aq)

- TRABALHANDO COM REDES

- Para verificar quais opções posso trabalhar com o docker:

$ docker network --help

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network --help

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Podemos também, saber mais informações sobre o comando específico (exemplo “create”):

$ docker network create --help

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network create --help

Usage:  docker network create [OPTIONS] NETWORK

Create a network

Options:
      --attachable           Enable manual container attachment
      --aux-address map      Auxiliary IPv4 or IPv6 addresses used by
                             Network driver (default map[])
      --config-from string   The network from which to copy the configuration
      --config-only          Create a configuration only network
  -d, --driver string        Driver to manage the Network (default "bridge")
      --gateway strings      IPv4 or IPv6 Gateway for the master subnet
      --ingress              Create swarm routing-mesh network
      --internal             Restrict external access to the network
      --ip-range strings     Allocate container ip from a sub-range
      --ipam-driver string   IP Address Management Driver (default "default")
      --ipam-opt map         Set IPAM driver specific options (default map[])
      --ipv6                 Enable IPv6 networking
      --label list           Set metadata on a network
  -o, --opt map              Set driver specific options (default map[])
      --scope string         Control the network's scope
      --subnet strings       Subnet in CIDR format that represents a
                             network segment
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Para listar todas as redes do docker:

$ docker network ls

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network ls
NETWORK ID     NAME        DRIVER    SCOPE
e2aeae6ddeb3   bridge      bridge    local
1404cd9dfe9e   host        host      local
f2eb7bb60c00   none        null      local
2e0fc3474581   redelocal   bridge    local
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Para saber as informações de uma rede (exemplo a rede “bridge”):

$ docker network inspect bridge

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "e2aeae6ddeb3117d3da59ba14afa748934df88ac60b7ea4eb71d2590562f223a",
        "Created": "2021-10-14T21:01:59.460041387-03:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Para criar uma nova rede (exemplo “nova_rede”):

$ docker network create nova_rede

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network create nova_rede
af738829bc6a88832611b5b4a48518a88e00f72a55420ea62bfaed14ccd47e6e
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Para criar uma rede com um “range” diferente (exemplo “minha_rede2”):

$ docker network create minha_rede2 --subnet 192.168.134.0/24 --gateway 192.168.134.1

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network create minha_rede2 --subnet
192.168.134.0/24 --gateway 192.168.134.1
6ce1d7c42a6124898b6930987e12ab120229cc9de0a35057c8a13b0f0c5c2672
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Associando um container criado e executando a nova rede criada:

$ docker network connect minha_rede2 meu_ubuntu

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network connect minha_rede2 meu_ubuntu
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Para saber os containers (em execução) associados a rede:

$ docker network inspect minha_rede2

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network inspect minha_rede2
[
    {
        "Name": "minha_rede2",
        "Id": "c4a068d87bbf2759b7218882f9b6cb4b067aa65cec3a1a30d2aabcbf924cc350",
        "Created": "2021-10-14T22:02:18.468161564-03:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "192.168.134.0/24",
                    "Gateway": "192.168.134.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "581647ff3cffdabaea19af49fc7b801e77f04afe1dc3015cc76eea9a7c16f8f1": {
                "Name": "meu_ubuntu",
                "EndpointID":
"07fab6120081095dcdcf2fd64fc82e0487fea138049f0716d146b8cd23f59fb5",
```

```
                "MacAddress": "02:42:c0:a8:86:02",
                "IPv4Address": "192.168.134.2/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Para remover uma rede criada (exemplo “nova_rede”):

$ docker network rm nova_rede

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network rm nova_rede
nova_rede
fabricio@notebook-lg:~/GitHub/Docker-Help$
```

- Para remover todas as redes que não estão sendo utilizadas:

$ docker network prune

```
fabricio@notebook-lg:~/GitHub/Docker-Help$ docker network prune
WARNING! This will remove all custom networks not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Networks:
redelocal
minha_rede2

fabricio@notebook-lg:~/GitHub/Docker-Help$
```

Alguns parâmetros são importantes na criação de containers utilizando redes:

- Configuração de dns

Ex:
docker run -ti **--dns 8.8.8.8** debian

- Configuração de hostname

Ex:
docker run -ti **--hostname catota** debian

- COMUNICAÇÃO ENTRE CONTAINERS UTILIZANDO DNS

-Crie o Container1:
$ docker run -ti --name container1 ubuntu /bin/bash

-Instale o comando ping:
 # apt-get update && apt-get install -y iputils-ping && apt-get clean

saia do container com "**Ctrl+P+Q**"

-Crie o Container2:
$ docker run -ti --name container2 ubuntu /bin/bash

-Instale o comando ping:
 # apt-get update && apt-get install -y iputils-ping && apt-get clean
saia do container com **"Ctrl+P+Q"**

-Verifique se os containers foram criados e estão sendo executados:
$ docker ps

-Crie a rede:
$ docker network create rede

-Verifique se a rede foi criada:
$ docker network ls

- Caso queira apagar as redes que não estão sendo utilizadas:
$ docker network pune

- Conecte o container1 na rede:
$ docker network connect rede container1

- Conecte o container2 na rede:
$ docker network connect rede container2

- Verifique se os containers foram conectados a rede:
$ docker network inspect rede

ou

$ docker network inspect rede | grep Name

- Entre dentro do container1:
$ docker container attach container1

- Execute um ping para o container2
 # ping container2
Para sair, **Ctrl+c**
Saia do container1 **Ctrl+P+Q**

- Entre dentro do container2:
$ docker container attach container2

- Execute um ping para o container1

 # ping container1
Para sair, **Ctrl+c**
Saia do container2 **Ctrl+P+Q**

-Caso queira desativar os containers 1 e 2:
$ docker container stop container1
$ docker container stop container2

-Caso queira ativar os containers 1 e 2:
$ docker container start container1
$ docker container start container2

O mesmo exemplo pode ser executado através do **Dockerfile** juntamente com o **Docker-compose**:

Antes de criar os arquivos dockerfile e docker-compose.yml, crie uma pasta com o nome **"two-ubuntu"**, dentro da pasta, crie outra pasta com o nome **"pasta"**.

A pasta "pasta"é um volume lincado com a pasta /home dos dois containers ubuntu criados.

Dockerfile

```
FROM ubuntu:14.04
RUN apt-get update && apt-get install -y iputils-ping && apt-get clean
```

docker-compose.yml

```
version: "3.7"

services:

   container1:
        build: .
        image: teste-ubuntu:14.04
        container_name: container1
        image: teste-ubuntu:14.04
        networks:
           ubuntu_rede:
             aliases:
               - rede
        volumes:
               - ./pasta:/home/
        entrypoint: /bin/sh
        stdin_open: true
        tty: true

   container2:
        container_name: container2
        image: teste-ubuntu:14.04
        networks:
           ubuntu_rede:
             aliases:
               - rede
        volumes:
               - ./pasta:/home/
        entrypoint: /bin/sh
        stdin_open: true
        tty: true
```

```
networks:
   ubuntu_rede:
      name: rede
      driver: bridge
      ipam:
         driver: default
```

Para executar:

- Processar o arquivo de composição e verificar a sintaxe:
$ docker-compose build

- Processar o arquivo de composição e iniciar a aplicação:
$ docker-compose up -d

- Para remover os containers, redes e volumes descritos no arquivo de composição:
$ docker-compose down -v

- Para mostrar os containers criados:
$ docker-compose ps