**Docker**

**Comandos:**

**Images:**

Para fazer o download de uma imagem do docker se usa o comando “pull”, exemplo:

docker pull portainer

Dessa forma será feito o download da imagem, caso o nome esteja errado será retornado um erro do docker. Para listar as imagens do docker baixadas se usa o comando “images” ou “image ls”, exemplo:

docker images

Ou

docker images ls

Dessa forma será exibido todas as imagens do docker baixadas localmente.

Para remover uma imagem do docker se usa o “rmi” ou “image rm” e o id da imagem ou o nome da imagem, exemplo:

docker image rm httpd:latest

Ou

docker image rm dabbfbe0c57b

Ou

docker rmi httpd:latest

Ou

docker rmi dabbfbe0c57b

Para ver as informações de uma imagem, se usa o comando “inspect”, exemplo:

docker image inspect postgres

Dessa forma será retornado um documento JSON, contendo todas as informações da imagem.

**Run:**

O comando “docker container run” serve para criar e executar um container, exemplo:

docker container run hello-world

ou

docker run hello-world

Dessa forma ele vai criar um container da imagem “hello-world”, e caso a imagem não exista no seu computador local, ele irá baixar a imagem, e após baixar executara o container, sempre que executar novamente o comando “run”, ele criará um novo container. Também é possível executar um container e já remove-lo da lista de containers, usando a flag “—rm”, exemplo:

docker container run --rm debian bash --version

Ou

docker run --rm debian bash --version

Dessa forma ele executara o container, e já o removera da lista. Também é possível executar o container e já acessar o terminal do container usando a flag “-it”, exemplo?

docker container run -it debian bash

Ou

docker run -it debian bash

Dessa forma ele executará um container da imagem do debian, e já acessará o terminal do container do debian.

Também é possível atribuir um nome para o container no momento de executa-lo usando a flag “—name”, exemplo:

docker container run -—name mydeb -it debian bash

Ou

docker run -—name mydeb -it debian bash

Dessa forma o container será criado com o nome “mydeb”, e não com um nome aleatório criado pelo docker, sendo que cada container deve ter um nome único.

Para expor um container para uma porta fora do container, basta passar a flag “-p” e passar a porta, sendo os parâmetros “portaExposta:portaDentroDoContainer”, exemplo:

docker container run -p 8080:80 nginx

Ou

docker run -p 8080:80 nginx

Dessa forma o container que será gerado da imagem “nginx” rodará dentro do container na porta “80”, e fora do container(no localhost do computador) rodará na porta “8080”.

Para mapear um volume, basta usar a flag “-v” e passar a pasta do host que vc quer mapear separado por dois pontos e a pasta do container que você deseja fazer esse mapeamento, exemplo:

docker container run -p 8080:80 -v $(pwd)/html:/usr/share/nginx/html nginx

Ou

docker run -p 8080:80 -v $(pwd)/html:/usr/share/nginx/html nginx

Neste caso ele vai pegar o diretório atual que você está no terminal através do método “$(pwd)” do unix, mais a pasta“html”, e vai fazer com que ele pare de apontar para a pasta padrão do nginx que seria “/usr/share/nginx/html”, e aponte para a pasta “html” do diretório atual da máquina host.

Para rodar um container em modo background, basta usar a flag “-d”, exemplo:

docker container run -d --name ex-daemon-basic -p 8080:80 -v $(pwd)/html:/usr/share/nginx/html nginx

Ou

docker run -p -d --name ex-daemon-basic 8080:80 -v $(pwd)/html:/usr/share/nginx/html nginx

Dessa forma o docker irá retorar o id do container, e o container ficará rodando em segundo plano como um processo normal do sistema.

**container:**

O comando “container ls” serve para listar todos os containers do docker que estão em com o status “running”, exemplo:

docker container ls

Ou

docker container list

Dessa forma será exibido todos os containers que estão em execução no momento, caso deseja visualizar também os containers que não estão mais em execução, basta usar a flag “-a”, que irá exibir todos os containers independente do status, exemplo:

docker container ls -a

Para iniciar um container já criado, basta usar o “start” e o nome do container, exemplo:

docker container start -ai mydeb

Ou

docker start -ai mydeb

Dessa forma ele vai iniciar o container nomeado como “mydeb” e a flag “-ai” ele vai anexar o terminal e o “i” é de modo interativo, ou seja, serve para acessar o terminal do container.

Para para um container em execução, basta usar o “stop”, exemplo:

docker container stop ex-daemon-basic

Ou

docker stop ex-daemon-basic

Dessa forma o container com o nome “ex-daemon-basic” será parado. Para reiniciar um container se usa o “restart”, exemplo:

docker container restart ex-daemon-basic

Ou

docker restart ex-daemon-basic

Para exibir os logs de um container, se usa o comando “logs”, exemplo:

docker container logs ex-daemon-basic

Ou

docker logs ex-daemon-basic

Dessa forma será impresso no terminal todos os logs do container executados até o momento.

Para exibir as informações do container se usa o “inspect”, exemplo:

docker container inspect ex-daemon-basic

Ou

docker inspect ex-daemon-basic

Dessa forma será impresso no terminal todas as informações de configuração do container no formato JSON, tendo a rede, local que é salvo os logs e etc.

Para executar um comando dentro do container, se usa o comando “exec”, exemplo:

docker container exec ex-daemon-basic uname -or

Ou

docker exec ex-daemon-basic uname -or

Dessa forma ele executara o comando “uname -or“ para verificar qual sistema está rodando o container. Outra forma também é acessar o terminal do container, exemplo:

docker container exec -it mariadb mysql -u root -p

Ou

docker exec -it mariadb mysql -u root -p

Dessa forma ele acessará o terminal do “mariaDB” dentro do container.

Para remover um cotainer basta usar o “rm” e o id do container ou o nome do container, exemplo:

docker container rm 299c8fd7f726

Ou

docker container rm priceless\_napier

Ou

docker rm 299c8fd7f726

Ou

docker rm priceless\_napier

**ps:**

O comando “docker container ps” serve para listar todos os containers que estão em execução(processos) no docker no momento, exemplo:

docker container ps

Ou

docker ps

Dessa forma ele exibira no terminal todos os containers em execução, caso também queira ver os containers que foram executados e não estão mais em execução, basta passar a flag “-a”, exemplo:

docker container ps -a

Ou

docker ps -a

Dessa forma ele exibirá não apenas os containers em execução, mas também aqueles que já foram executados e não estão mais ativos no momento.

**Network:**

O comando “network ls” serve para listar todos os networks/redes do docker que estão disponíveis, exemplo:

docker network ls

O comando “network create” serve para criar uma nova rede no docker, sendo o parâmetro “—driver” seguido pelo nome do driver de rede que será utilizado(bridge, host ou outra), exemplo:

docker network create --driver bridge rede\_nova

**Executando um container**

|  |  |
| --- | --- |
| **Parâmetro** | **Explicação** |
| -d | Execução do container em background |
| -i | Modo interativo. Mantém o STDIN aberto mesmo sem console anexado |
| -t | Aloca uma pseudo TTY |
| --rm | Automaticamente remove o container após finalização (**Não funciona com -d**) |
| --name | Nomear o container |
| -v | Mapeamento de volume |
| -p | Mapeamento de porta |
| -m | Limitar o uso de memória RAM |
| -c | Balancear o uso de CPU |

**Verificando a lista de containers**

|  |  |
| --- | --- |
| **Parâmetro** | **Explicação** |
| -a | Lista todos os containers, inclusive os desligados |
| -l | Lista os últimos containers, inclusive os desligados |
| -n | Lista os últimos N containers, inclusive os desligados |
| -q | Lista apenas os ids dos containers, ótimo para utilização em scripts |