Docker Command

Every commands use in the course.

* docker container run hello-world

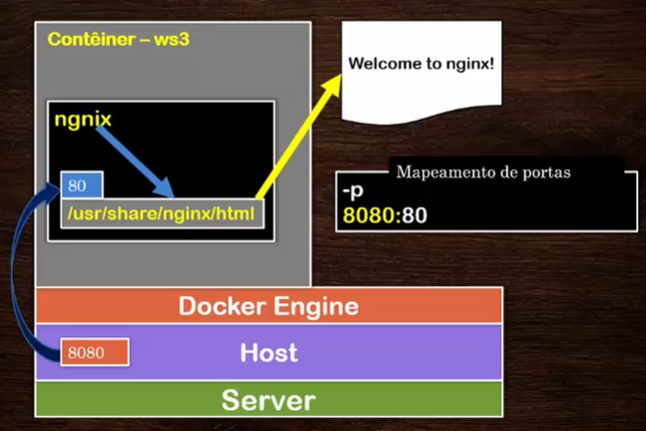
docker image pull

run docker container create

docker container start

docker container exec

* docker container run alpine ls -l
* docker container run -it alpine bin/sh
* docker container exec container\_id ls -l
* docker container stop container\_id
* docker container run –name novoNome imagem
* docker container rm container\_id
* docker container run --name ws1 -p 8080:80 -d(Segundo plano) nginx



* docker container run --name wsTeste -p 8080:80 -v c:/users/Jefferson.ricardo/downloads/html:/usr/share/nginx/html -d nginx
* docker container inspect container\_id

Criando container com image dotNet

* docker container run --name demoNet -it mcr.microsoft.com/dotnet/sdk

Criando appConsole

* - dotnet new console

Build appConsole

* - dotnet build

Executando appConsole

* - dotnet run

Entrar no container após criado e parado.

* docker container start -ia dotnet

Interface gráfica do usuário

Descrição gerada automaticamente

* docker container prune
* docker image prune

Criando Imagens

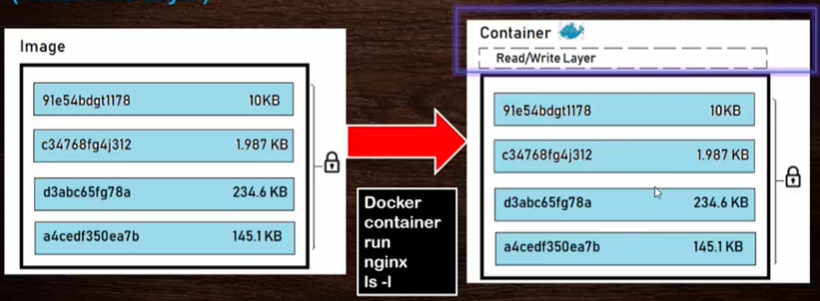
O dockerfile é um arquivo texto com instruções, comandos e passos que é executado através do comando build para gerar uma imagem.

Uma imagem é um binário que inclui todos os requisitos para a criação e execução de único contêiner do docker, bem como metadados que descrevem suas necessidades e capacidades, incluindo o código do aplicativo dentro do contêiner e suas configurações.

Docker build -t <imagem>



Ao criar o container, o Docker cria uma nova camada acima da última camada da imagem, e nessa camada podemos ler e escrever.



Instruções e Argumentos são interpretados linha a linha pelo engine do Docker para criação da imagem.



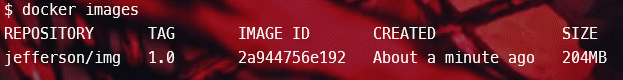
Figura 1: Processo de builder

Definindo as etapas que vamos usar para criar a imagem

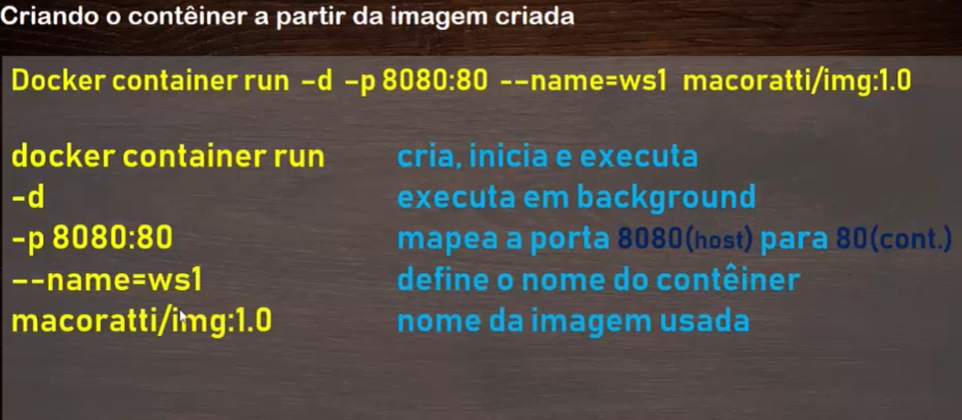
1. Definir uma imagem base.
2. Definir informações para a imagem.
3. Executar comandos para instalar e iniciar o nginx.
4. Expor qual porta o servidor vai atender.
5. Definir o ponto de entrada a aplicação
6. Definir a execução de um comando para inicializar o servidor nginx.

docker build -t jefferson/img:1.0 .

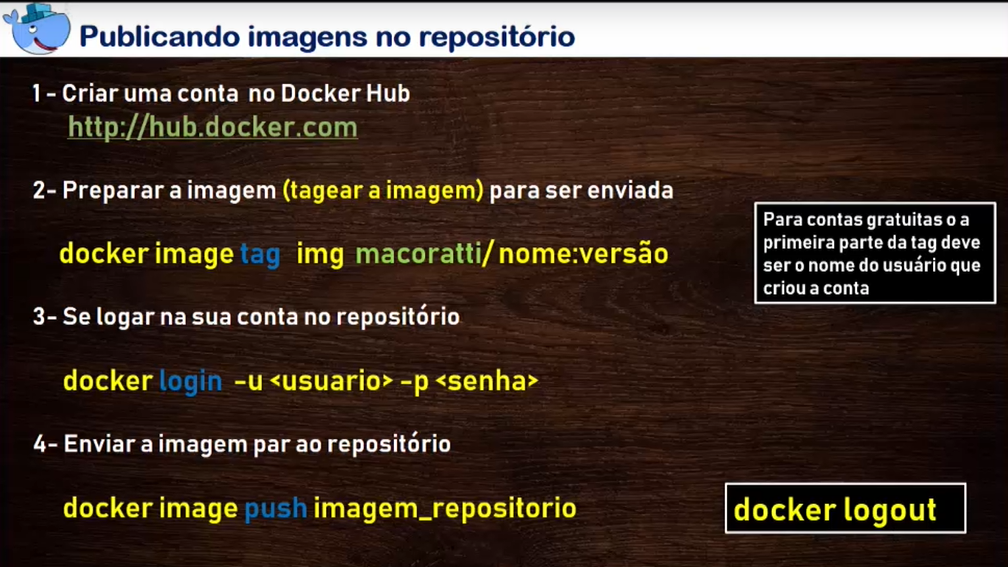
|  |  |
| --- | --- |
| Docker build | O commando que constroi a imagem |
| -t | Usado para informar que a imagem pertence ao meu usuário |
| jefferson/img:1.0 | O nome da imagem e a tag atribuída a imagem |
| . | Significa o diretório atual, onde está o dockerfile. |
|  |  |



* docker container run -d -p 8080:80 --name=ws1 jefferson/img:1.0

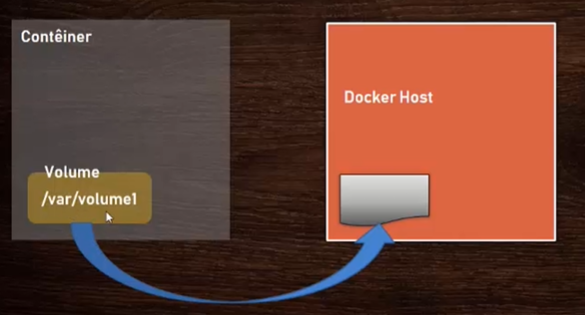


* docker login
* docker tag 2a944756e192(da img) 40509477(use dockerhub)/debian:1.0(img name and version)
* docker push 40509477/debian:1.0



Criar Volumes

Os volumes separam os arquivos de dados que são gerados por um aplicativo ou banco de dados do restante do armazenamento do contêiner.



Ao criar um volume e definir o seu mapeamento toda a informação que for armazenada no contêiner será armazenada na pasta mapeada docker host.

1. Para manter os dados quando um contêiner for removido.
2. Para compartilhar dados entre o sistema de arquivos do host e o contêiner do docker.
3. Para **compartilhar dados com outros contêineres do docker**.

**Criando volume e salvando container mysql**

1. Criando volume

docker volume create dadosdb

1. Baixando imagem mysql

docker image pull mysql:5.7

1. Inspecionar imagem mysql

docker image inspect mysql:5.7

/var/lib/mysql

1. Mapeando o armazenamento do container no volume criado.

docker container run -d --name mysql -v dadosdb:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=123456 mysql:5.7

1. Entrando dentro de container e criando o banco.

docker container exec -it mysql bash

1. Acessando o mysql

mysql -u root -p

**No cenário do Sql Server**

docker volume create volume\_data\_sqlserver

Texto

Descrição gerada automaticamente com confiança baixa

**Criando container apontando para o volume criado.**

docker container run -d --name sqlServer -v volume\_data\_sqlserver:/opt/mssql -e "ACCEPT\_EULA=Y" -e "SA\_PASSWORD=Chocolate2022" -p 1250:1433 --name sqlserverdb -d mcr.microsoft.com/mssql/server:2019-latest

**Publicando a aplicação em dist**

Dotnet publish –configuration Release –output dist

**Criar a imagem da aplicação usando dockerfile e depois criar container**

docker build -t app

**Não funcionou muito bem, acho que devido as restrições da emrpesa**

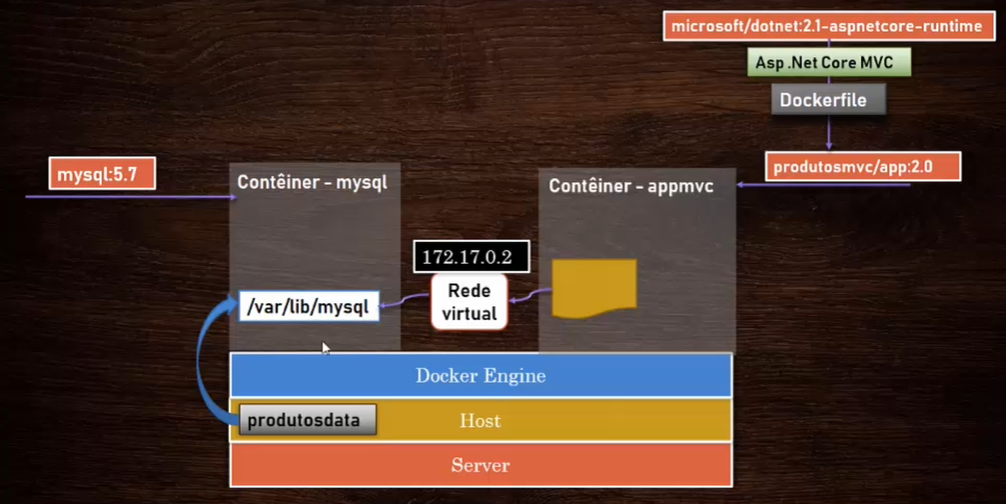
Docker container run -d –name appmvc -p 8000:80 -e DBHOST=172.17.0.2 imagem

Ao criar um banco, apagar o container e criar um novo mapeando para o volume, vamos ver que os dados permanecem lá. PRESERVA A INFORMAÇÃO DE UM CONTAINER

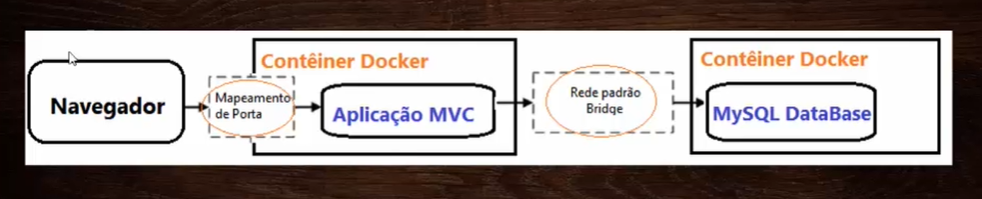
A resposta desse comando mostrará como o docker configurou a rede virtual e incluirá uma seção containers que mostra os contêineres conectados á rede e os endereços IPs que são atribuídos a eles.

1. docker network inspect bridge
2. docker container logs appmvc (mostrar algumas informações importantes.)

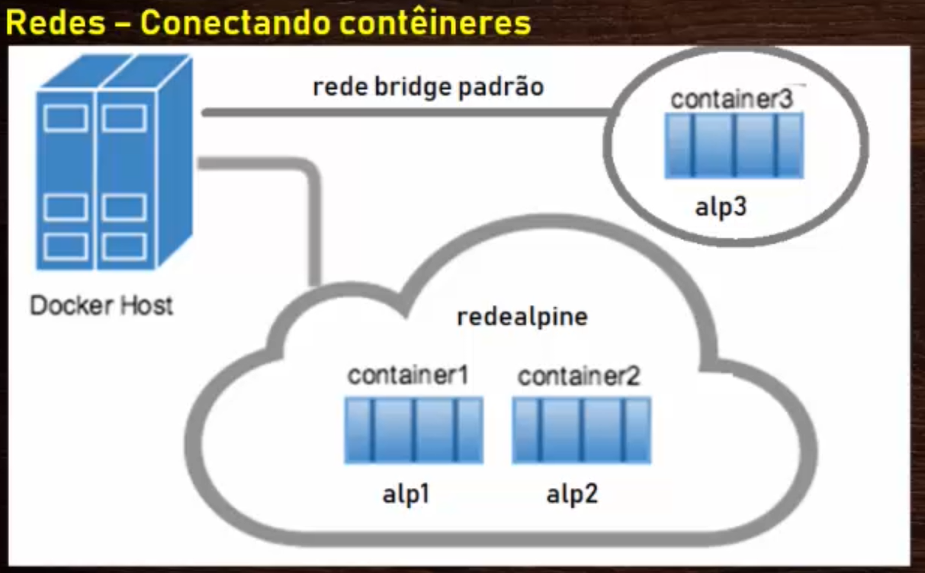
**Rede virtual**



1. docker network ls
2. docker network inspect bridge



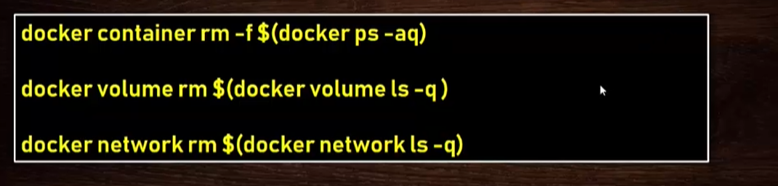
**Criando rede customizada**

****

docker network create --driver <nome> <nome\_da\_rede>

1. docker network create --driver bridge redealpine
2. docker container run -it --name alp1 --network redealpine alpine
3. docker container run -it --name alp2 --network redealpine alpine
4. docker network connect redealpine alp3

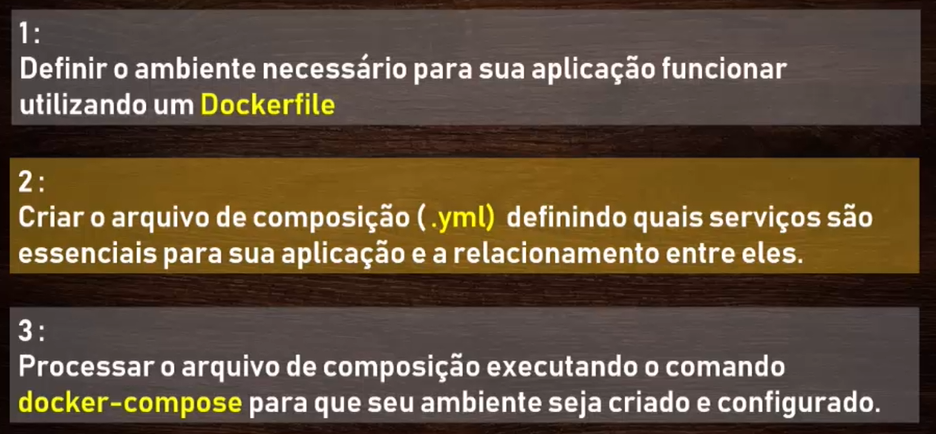
Docker compose é responsavel pelo gerenciamento e criação de redes, volume e comunicação.



**Docker Compose**



**Roteiro para as aplicações**

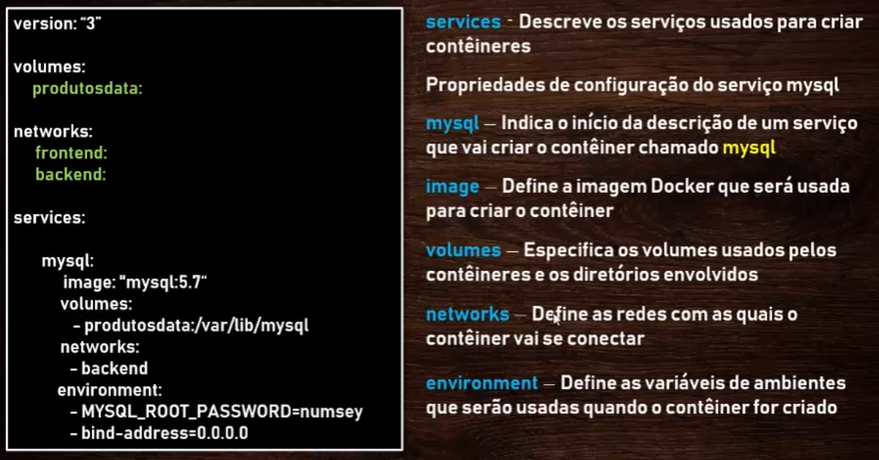


Processando o arquivo de composição

1. docker-compose -f docker-compose.yml build
2. -f nome do arquivo de composição
3. Build informa o docker para processar o arquivo.

Criação do contêiner a partir da imagem do mysql

1. Docker container run -d --name mysql -v produtosdata:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=numsey -e bind-address=0.0.0.0 mysql:5.7



Cenário de teste

Essas configurações foram o suficiente para realizar um teste e ver se está criando a imagem, o container e o volume. Mediante os comandos acima.

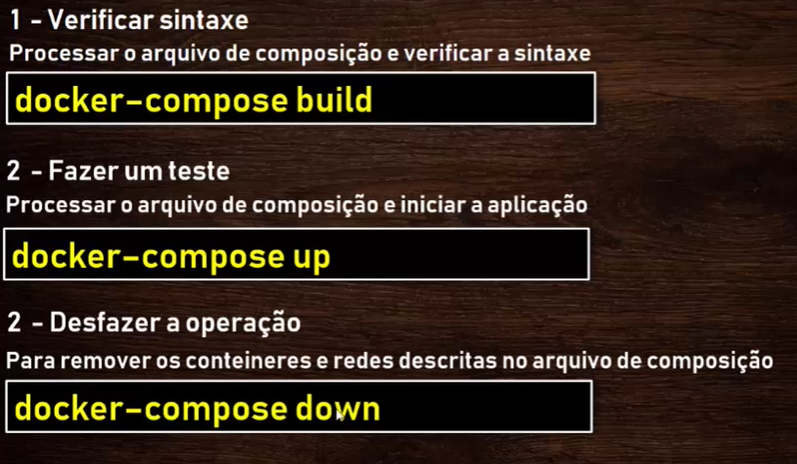


Figura : Comandos usados para testar.

