Guia Rápido de Docker - Esquenta Imersão Full Stack && Full Cycle

Instagram: @devfullcycle Youtube: youtube.com/fullcycle

O que são Containers

"Um container é um padrão de unidade de software que empacota código e todas as dependências de uma aplicação fazendo que a mesma seja executada rapidamente de forma confiável de um ambiente computacional para o outro." docker.com

Evitar:

1. Na minha máquina funciona!
2. Garantir o mesmo ambiente para todos
3. Rápido e leve para trabalhamos, tanto em dev como em produção

O que é o Docker

Por que utilizar o Docker

É um software que utiliza um mecanismo de execução de containers através de processos.

Quais os pontos chave que o Docker utiliza para rodar de forma estável?

Porque diferentemente de máquinas virtuais, o Docker consegue de forma extremamente rápida e leve executar seus containers sem a necessidade de instalação completa de um sistema operaciona. O docker "empresta" todos os recursos, incluindo o Kernel para que seus containers possam ser utilizados.

Pontos chave para funcionar

1. Namespaces - Isolamento de processos
2. Cgroups - Controla os recursos do sistema operacional
3. Sistema de arquivos - OFS (Overlay File System)

Comandos básicos

Containers

Executar um container:

```
docker run -it ubuntu /bin/bash
```

Verificar containers em utilização:

```
docker ps
```

Verificar todos os containers:

```
docker ps -a
```

Compartilhamento / Exposição de portas

```
docker run -p 8080:80 nginx
```

Executar um container em segundo plano (modo detached)

```
docker run -d -p 8080:80 nginx
```

Remover um container

```
docker rm <id ou nome do container> (-f se quiser forçar)
```

Remover um container após sair

```
docker run --rm -p 8080:80 nginx
```

Remover todos os containers (linux/mac/bash,etc)

```
docker rm $(docker ps -a -q) -f
```

Imagens

O que são imagems? Imagens a grosso modo são uma espécie de template de container que será gerado. A

imagem é imutável. Todo container através de uma imagem possui uma camada de gravação de arquivos.

Remover uma imagem

```
docker rmi <nome da imagem>
```

Remover todas as imagens

```
docker rmi $(docker images -q) -f
```

Dockerfile

O que é o Dockerfile

É um arquivo declarativo que tem o objetivo de construir uma imagem através de outra imagem.

O Dockerfile possui toda a estrutura e comandos necessários que ações sejam executadas no processo de "build", ou seja, no processo de construção de uma imagem.

Gerando build de uma imagem

Exemplo de Dockerfile

```
FROM golang:1.14

WORKDIR /go/src/

COPY . .

RUN GOOS=linux go build main.go

EXPOSE 8081

ENTRYPOINT ["./main"]
```

Após a criação do arquivo Dockerfile, você poderá criar sua própria image utilizando o seguinte comando:

```
docker build -t <seu-user>/<nome-da-image>:<versao da imagem> .
```

1. O ponto final sinaliza em qual diretório encontra-se o Dockerfile.
2. Caso a versão da imagem não seja informada, ela será nomeada automaticamente como "latest".

Publicando imagem no Docker Hub

O Dockerhub é um repositório aonde você pode disponibilizar suas imagems, de forma pública ou privada. Para que a publicação seja possível, você primeiramente você terá que realizar o login em sua conta digitando:

```
docker login
```

Realizado o login basta realizar o push de sua imagem:

```
docker push <nome da imagem>
```

Docker-compose

O que é o docker-compose

O docker-compose é uma ferramenta cujo o objetivo é facilitar o processo de executar containers docker de forma declarativa. Cada container é executado como um serviço.

O arquivo utilizado para que o docker-compose seja executado com sucesso chama-se por padrão docker-compose.yaml.

Exemplo:

```
version: '3'
services:
  nginx:
    image: nginx
    volumes:
      - ./nginx:/usr/share/nginx/html/
    ports:
      - 8080:80

  redis:
    image: redis:alpine
    expose:
      - 6379
```

Se você verificar o exemplo acima, perceberá que teremos dois serviços a serem executados.

1. O primeiro chama-se nginx. Ele utilizará a imagem do nginx como base e fará um compartilhamento de

volume. Ou seja, a pasta local do computador será compartilhada com o container. Nesse caso, tudo que existir na pasta nginx do computador, será automaticamente replicado no endereço: /usr/share/nginx/html/ do container. Também a porta 8080 será redirecionada para a porta 80 do container; isso significa que quando acessarmos no computador: localhost:8080 automaticamente o docker fará o redirecionamento da requisição para a porta 80 do container.

2. O segundo serviço chama-se redis e nesse caso é extremamente simples. Ele utiliza o redis:alpine como imagem base e expõe a porta 6379 do container. Isso significa que o container do nginx poderá se comunicar na rede local criada pelo docker utilizando a porta 6379.

Comandos úteis para o docker-compose

Para iniciar os serviços declarados no docker-compose.yml, basta executar:

```
docker-compose up
```

Ao executar esse comando, os serviços serão inicializados, porém, você perceberá que seu terminal ficará bloqueado, uma vez que o processo está sendo executado. Para executar de forma desatachada, basta informar o parâmetro "-d" no final da instrução.

```
docker-compose up -d
```

Para encerrar os serviços, basta executar:

```
docker-compose down
```

Caso queira ver de forma mais "organizada" somente os containers dos serviços sendo executados, basta rodar:

```
docker-compose ps
```