Anotações Full Cycle

# Docker

#### Comandos básicos:

docker run [imagem]:[\*tag] [\*comando]

Roda um container da imagem que pode ser baixada do repositório e pode executar um comando no container baixado;

docker run -it ubuntu bash

Roda um container da imagem ubuntu (sem tag é baixada a imagem latest) que pode ser baixada em modo interativo (-i) e com o TTY ativado (-t) (terminal do container attached ao terminal do localhost) e executa o comando bash;

docker run -it --rm ubuntu bash

Roda um container da imagem ubuntu (sem tag é baixada a imagem latest) que pode ser baixada em modo interativo (-i) e com o TTY ativado (-t) (terminal do container attached ao terminal do localhost), executa o comando bash e remove o container logo depois de ele ter sido fechado;

docker run --rm [imagem] [\*comando]

Executa um container e logo depois sua execução já o remove;

docker ps

Lista os containers ativos;

docker ps -a

Lista todos os containers, ativos e inativos que foram armazenados na máquina local;

docker ps -a -q

Lista apenas os IDs de todos os containers ativos e inativos na máquina local;

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

Remove todos os containers ativos e inativos forçadamente;

docker start [nome do container]

Inicia um container armazenado localmente;

docker run -p 8080:80 nginx

Aponta a porta 8080 do localhost (máquina onde o docker engine está rodando) para a porta 80 do servidor nginx; -p serve para apontar uma porta local para uma porta de um container;

docker run -d -p 8080:80 nginx

Aponta a porta 8080 do localhost (máquina onde o docker engine está rodando) para a porta 80 do servidor nginx; -p serve para apontar uma porta local para uma porta de um container; -d indica que o terminal do container deve ser dettached do terminal do localhost, deixando o terminal do localhost livre para ser manipulado;

docker run -d -p 8080:80 --name fcnginx nginx

Aponta a porta 8080 do localhost (máquina onde o docker engine está rodando) para a porta 80 do servidor nginx; -p serve para apontar uma porta local para uma porta de um container; -d indica que o terminal do container deve ser dettached do terminal do localhost, deixando o terminal do localhost livre para ser manipulado e nomeia a container de fcnginx;

docker exec [nome\_do\_container] [comando] Ex: docker exec fcnginx ls

Comando exec executa um comando dentro de um determinado container. No exemplo acima estamos executando o comando ls dentro do container fcnginx;

docker exec -it fcnginx bash

Executará o comando bash em modo interativo e com o TTY ativado no container fcnginx;

docker run -d --name fcnginx -p 9876:80 -v /home/maranhao/Público/docker/html:/usr/share/nginx/html nginx

Executará o container espelhando/criando um volume local na pasta selecionada no container;

* DICA: echo $(pwd) no terminal linux mostra o caminho completo no qual o terminal está rodando.

docker run -d --name fcnginx -p 9876:80 --mount type=bind,source=”$(pwd)”/html,target=/usr/share/nginx/html nginx

Executará o container montando um volume espelhado da máquina local na imagem do container selecionado;

* DICA: -v cria diretório local e remoto caso não exista; --mount apenas monta no container um volume idêntico à pasta local;

docker attach [nome\_container]

Permite conectar o terminal da máquina local com um container rodando, pode-se interagir com o container como se os comandos estivessem rodando no seu terminal; Muito usado para interagir com containers na realização de soluções de problemas;

docker logs [container]

Exibe os logs/saídas do container informado;

Volumes

Volumes podem ser criados localmente através do comando docker volume create e pode ser inspecionados através do comando docker volume inspect.

Volumes são utilizados para facilitar o espelhamento das pastas locais em um container (através do -v ou do --mount type=volume). Os comandos anteriores podem inclusive criar uma pasta no container que vai espelhar a pasta local. Os volumes criados localmente podem ser compartilhados com vários containers ao mesmo tempo.

Ex:

docker volume create meovolume

docker run -d --name fcnginx -p 9876:80 --mount type=volume,source=meovolume,target=/app nginx

Executa a imagem do container nginx, mapeando a porta local 9876 à porta 80 do container e espelhando o volume local meovolume na pasta /app na da imagem do container (caso a pasta app não exista ela será criada remotamente).

Dockerfile

FROM: Designa a imagem base que será manipulada pelo dockerfile;

WORKDIR: Imagem será carregada para manipulação dentro do diretório informado nesse comando, se o diretório não existir ele será criado no momento da execução da imagem;

USER: Usuário que será logado ao carregar a imagem (usuário root é o default);

RUN: Executa comandos durante a construção (build time) da imagem e o resultado dos comandos é comitado na imagem que será criada;

COPY: copia arquivos de um diretório da origem para o diretório da imagem do container;

CMD: Executa comandos após a imagem ter sido construída (container run time). Se houver mais de um comando, apenas o último é executado;

ENTRYPOINT: Especifica um comando default que é executado assim que o container é iniciado e tudo que vem depois dele são argumentos desse comando (como o CMD apenas a ultima entrada de ENTRYPOINT é levada em consideração na execução do container);

EXPOSE: Expõe uma determinada porta do container para conexões externas;

* DICA: Se ao fim de um arquivo shell script (.sh) houver a linha exec “$@” significa que o script executará comandos e argumentos que são passados pra ele.

Imagens e Docker Hub

docker images

Lista as imagens baixada no computador local;

docker pull [imagem]:[\*tag]

Baixa imagem selecionada do Docker Hub (Registry) para máquina local;

docker build -t [usuario\_dockerhub]/[nome\_da\_imagem]:[\*tag] .

Cria uma imagem no docker registry do usuário logado a partir de um dockerfile contido no diretório . ;

docker push [imagem]

Envia a imagem para o Docker Hub;

Network

Tipos de rede entre containers:

* BRIDGE: Tipo de rede default do docker, cria uma rede bridge via software entre o container e o host; Containers conectados
* HOST: Containers participam da mesma rede do host, compartilham a mesma rede do host sem restrições. Possível acessar portas sem a necessidade de expô-las e os container podem acessar a rede diretamente;
* OVERLAY: Rede distribuída para possibilitar o uso de vários hosts. Serve para conectar containers a vários hosts ao mesmo tempo na mesma rede;
* MACVLAN: Define um endereço MAC pro container. Permite aos container se conectar diretamente com a interface de rede do host, útil em cenários os quais o container deve aparecer como um dispositivo físico separado na rede;
* NONE: Container isolado.

docker network

* DICA 1: No MacOS e no Windows para que um container tenha acesso a uma porta da máquina host do docker ele pode fazer uma requisição para [host.docker.internal](https://host.docker.internal):[PORTA];
* DICA 2:
  + 127.0.0.1 é o **endereço de loopback**, também conhecido como localhost. Ele é usado para comunicação dentro do próprio computador. Quando um aplicativo está configurado para ouvir em 127.0.0.1, ele só aceitará conexões locais nesse endereço. Ou seja, apenas processos no mesmo host podem se comunicar com ele.
  + 0.0.0.0 tem diferentes significados, mas no contexto de configuração de servidores, quando um servidor é instruído a ouvir em 0.0.0.0, isso significa “**ouvir em todas as interfaces de rede disponíveis**”. O adaptador de loopback com o endereço IP 127.0.0.1, do ponto de vista do processo do servidor, parece igual a qualquer outro adaptador de rede na máquina. Portanto, um servidor configurado para ouvir em 0.0.0.0 aceitará conexões em todas as interfaces de rede, incluindo o loopback.

#### Docker Compose

Docker Compose é uma ferramenta para definição e execução de aplicações multi-container.

Compose simplifica o controle de todas as camadas da aplicação, tornando simples o gerencimento de services, networks, e volumes em um simples e compreensível arquivo de configuração YAML. Com apenas um comando, podem ser criados e executados todos os services do seu arquivo de configuração.

Seções

* version: Indica versão do compose que está sendo utilizada;
* services: Service é uma definição abstrata de um recurso computacional com uma aplicação o qual pode ser escalado e replicado independentemente de outro componentes;
  + Uma implementação compose que necessite rodar uma aplicação na máquina local precisa ter o suporte a *(re)buildar* a aplicação a partir do fonte. A *Compose Build Specification* deixa você definir o processo de build a partir de um arquivo Compose de forma portável. build pode ser especificado tanto como uma simples string definindo um context path quanto como uma descrição detalhada de como deverá ser executado o build da aplicação.
* networks: Docker networks que serão criadas;

Comandos

docker-compose up

Usa o arquivo docker-compose.yaml como guia e levanta os serviços e redes definidas;

docker-compose up -d --build

Levanta os serviços de forma dettached e caso um dockerfile de algum serviço tenha sido modificado ele refaz a build sem aproveitar o que já foi guardado em cache;

docker-compose down

Derruba todos os serviços, remove os containers responsáveis pelos serviços;