CONTAINER

* Container vs VM
  + CONTAINER vem da ideia de rodar aplicações específicas dentro de um processo isolado de seu computador, sendo que esse processo está rodando dentro do seu sistema operacional. Ele não afeta o sistema operacional, ele roda através de instruções do próprio sistema operacional. Não precisa mais sequestrar memória e cpu para rodar. Eles são processos que podem nascer e morrer rapidamente (muito mais velocidade). Docker é um container PAI, pois executa um ou uma junção de containers. Pode-se compartilhar informações entre o PAI, computador e o filho container.
  + VM é um sistema operacional instalado dentro de um sistema operacional (virtualização foi e é muito importante). Toda vez que se instala uma máquina virtual você tem que falar quanto de memoria vai usar? quanto de cpu? Quanto de ram vai usar? (recursos as vezes são desperdiçados em uma VM). Precisa-se realizar updates e afins;
* DOCKER DESKTOP é uma aplicação que faz com que rode o docker (foi feito para se rodar em linux). Para se rodar em Windows e Mac, ocorre uma simulação de Linux dentro desses sistemas.
* Todo CONTAINER contém uma IMAGEM. Quando se roda o Docker run ele cria uma imagem para a gente.
* O comando “docker ps” mostra quais containers estão rodando (fácil de verificar pela presença do CONTAINER ID)
* O comando “docker run -it ubuntu bash”. Ubuntu se refere para se rodar um container com a imagem do Ubuntu. Mas qual comando vai ser executado? No caso ele irá ao bash para ter acesso ao terminal do Ubuntu. -i se refere para ele rodar no modo interativo (para digitar e receber). Já para o modo de enviar e receber mensagens bidirecionais (código -it).
* Enquanto o bash estiver sendo executado, o container estará ativo.
* IMPORTANTE: Quando se executa o Docker run “IMAGEM” ele resgata do Docker Hub uma imagem com o referido nome.
* Comando “docker run -p 8080:80 nginx” indica que -p é o comando para redirecionar a porta 80 do container nginx vá para a porta 8080.
* O comando “docker exec” executa um comando em um CONTAINER.
  + O comando “docker exec IMAGEM ls” executa o comando ls (de listar) dentro do CONTAINER.
* O comando “docker exec -it IMAGEM bash” faz com que entremos dentro do CONTAINER.
* Se eu der um run em uma IMAGEM existência após ser criada, a atual será apagada e uma nova tomará o lugar, apagando todas as configurações feitas dentro do CONTAINER.
* Conseguimos, então, especificar um VOLUME que é uma pasta que contém em meu computador e uma pasta que contém no CONTAINER.
  + Comando “docker run -p 8080:80 -v CAMINHODAPASTA:CAMINHONOCONTAINER ngnix”
  + Tal comando, ao fazer modificações na PASTA do computador, no CONTAINER também será alterado.
* CRIAÇÃO DE IMAGEM
  + FROM nginx:lastest
  + COPY PASTA/ARQUIVO CAMINHODOCONTAINER
* Comando “docker build -t rodolphofranzsao/nginx-ime:latest .”
  + rodolphofranzsao/nginx-ime:latest é o nome da IMAGEM
  + . é onde o docker file esta.
* Ou seja, a partir da imagem do nginx, criamos outra com algumas alterações.
* SUBIR PARA O DOCKER HUB
  + Docker “push rodolphofranzsao/nginx-ime:latest” eu lanço IMAGEM criada para o Docker Hub.
* DOCKER COMPOSE
  + É um arquivo manifesta com extensão “yaml” onde falamos todos os CONTAINERES que desejamos rodar.
  + “docker composse up -d”. O up sobre os containeres. O -d significa para rodar depached (em background).
  + A pasta mysql começa a ser criada dentro do computador
  + “docker composse down” mata o container.
  + Ao subir novamente, se desejar, ele subirá normalmente.
* Quando programamos dentro do docker, podemos utilizar nossa máquina e ainda verificar o que está acontecendo dentro do container.