**DOCKER: criando e gerenciando containers**

**Container:**

* **Processo/Aplicação isolada da infraestrutura de outros contêineres e de qualquer outro processo externo**
* **Possui um único objetivo de existência e possui dentro dele tudo que precisa pra executar seus objetivos**
* **É sempre criado a partir de uma imagem seja ela remota ou local**
* **Quando executamos um container precisamos definir de qual imagem iremos executar esse container**
* **Não iremos isolar mais a máquina servidor inteira e sim o processor/aplicação que queremos rodar;**
* **No windows os containers e as imagens são compartilhados por todos os usuários do computador**

**Parte 2: Instalação e criação de containers**

**docker run hello-wold: busca a última versão da imagem “hello-wold” localmente, caso não encontre busca remotamente e faz um pull da imagem no repositório de imagens do docker e em seguida executa a imagem**

**docker run -it ubuntu bash: cria imagem ubuntu via container e entra dentro do container de ubuntu**

**exit: sai de dentro do container e elimina o container da execução**

**Comando para trabalhar com container**

**docker ps: lista os containers em execução**

**docker ps -a: lista os containers que foram criados e estão parados (com status de exit)**

**docker start (id/nome do container): sobe um container que esta da listagem de container com status exited/parado (aparece em docker ps -a)**

**docker exec –it (id/nome do container) bash: entra no terminal do container**

**docker stop (id/nome do container): pausa a execução do container**

**docker rm (id/nome do container): remove o container**

**docker start –a –i (id/nome do container): inicia novamente o container e fica com acesso ao seu output (entra no terminal do ubuntu no caso de um container criado com o comando “bash”)**

**docker run --name Nome\_Container -it ubuntu bash: inicia e nomeia um contanier apartir da imagem ubuntu configurando com o comando bash**

**Parte 3: Criação de imagens**

**Imagens: são receitas/instruções de como executar determinado container para um determinado objetivo;**

**Existem inúmeras versões de imagens no docker hub**

**Cada imagem possui suas características, exemplos: porta que expõe, serviço que está executando, etc;**

[**https://hub.docker.com/**](https://hub.docker.com/) **(site onde iremos baixar a maioria das nossas imagens)**

**docker run –p 8080:80 –p 8443:443 -d dockersamples/static-site**

**-p: mapeia a porta do host na porta do container, permitindo o acesso pelo host (localhost:8080 acessa o site criado pela imagem dockersamples/static-site que está rodando no container)**

**-d: roda um container sem prender o terminal**

**docker images: mostra todas as imagens que temos localmente**

**docker rmi id\_da\_imagem: remove a imagem**

**Parte 4: Imagens personalizadas e gestão de volumes**

**Dockerfile: arquivo padrão para criação de imagem**

**FROM nginx:latest (define que nossa imagem vai usar como base a imagem do nginx na sua última versão, baixa do docker hub caso não ache localmente)**

**MAINTAINER Lilian Carvalho <lilian.tavares@dbserve.com.br> (define quem está criando a imagem)**

**COPY . /usr/share/nginx/html (instrução dada pra que na hora que formos construir essa imagem seja copiado tudo que tá no mesmo diretório do arquivo dockerfile para o diretório /usr/share/nginx/html**

**WORKDIR /usr/share/nginx/html (definimos o diretório padrão da nossa imagem)**

**EXPOSE 80 (expõe a porta 80)**

**docker build –t liliantavares/nginx:1.0.0 . (constrói uma imagem a partir de um arquivo de instruções de construção de imagem (Dockerfile)**

**-t (cria uma tag/marca a imagem que estamos construindo com esse nome e versão liliantavares/nginx:1.0.0)**

**. (informa que a imagem será criada a partir desse diretório)**

**Caso o nome no arquivo não seja Dockerfile é preciso usar –f para passar o nome do arquivo de instruçoes (-f nome\_do\_arquivo.dockerfile)**

**docker run –d –p 8080:80 liliantavares/nginx:1.0.0 (executa container da imagem)**