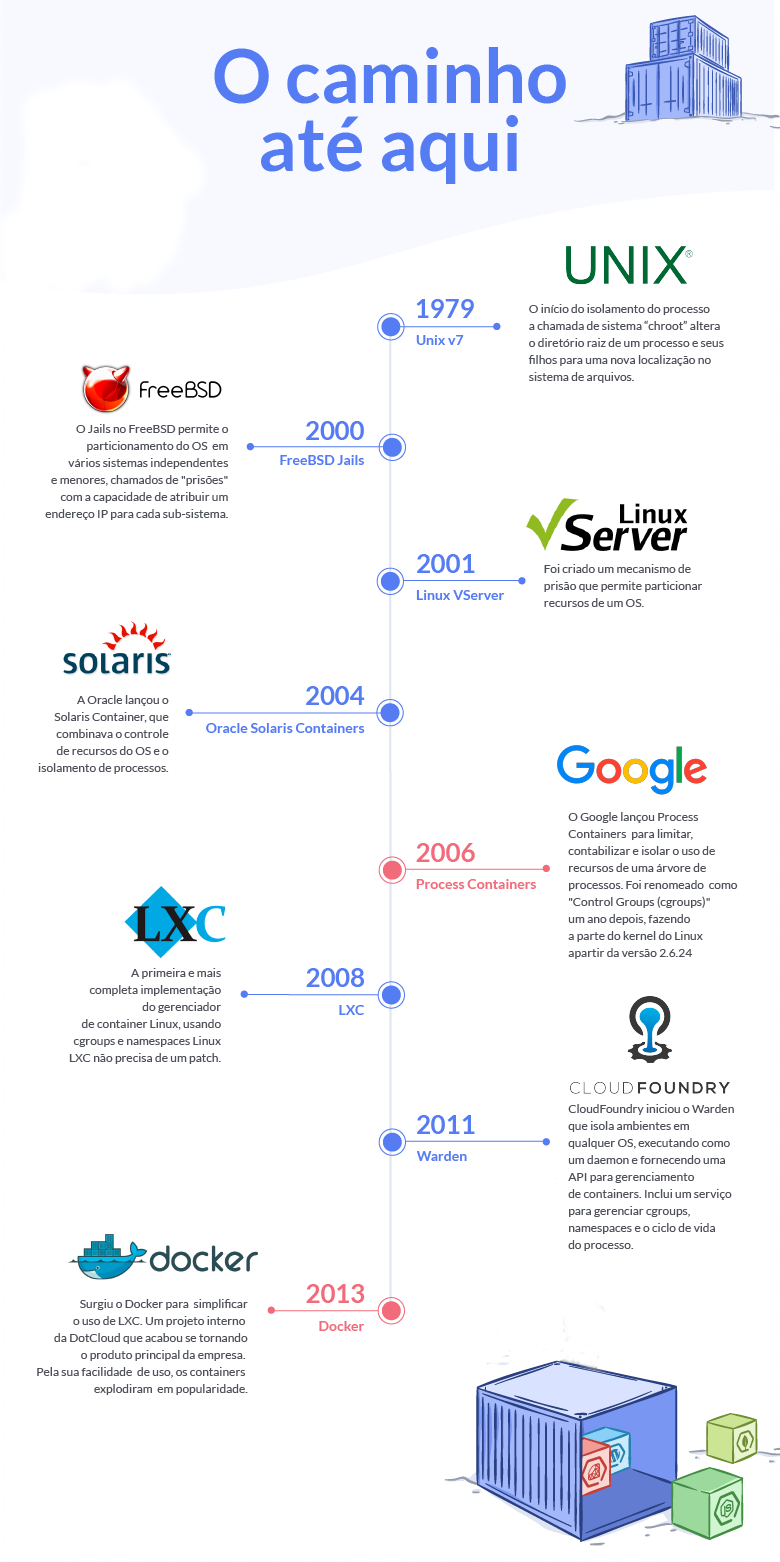
**GUIA DE BOLSO DOCKER**

**Por: Gleidson Duarte**

****

****

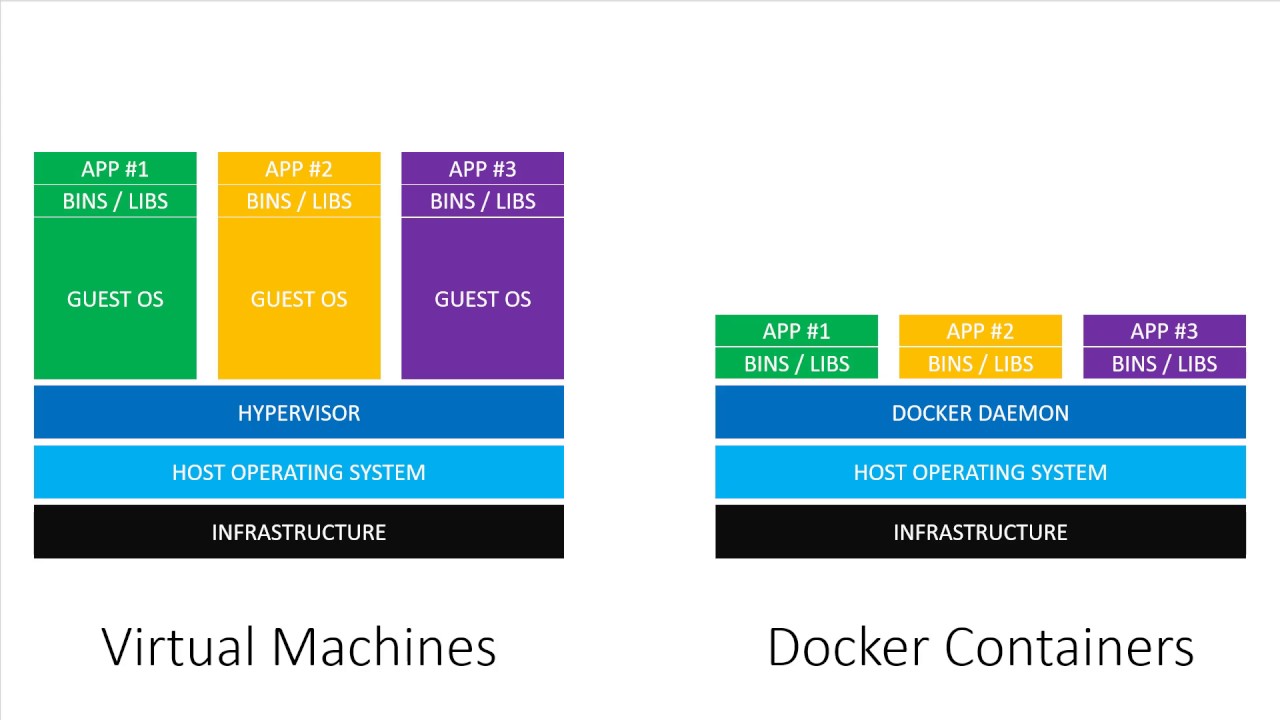
**MÁQUINA VIRTUAL x CONTÊINER**

**MÁQUINA VIRTUAL**

O objetivo desse modelo é compartilhar os recursos físicos entre vários ambientes isolados, sendo que cada um deles tem sob sua tutela uma máquina inteira, com memória, disco, processador, rede e outros periféricos, todos entregues via abstração de virtualização. É como se dentro de uma máquina física criasse máquinas menores e independentes entre si. Cada máquina dessa tem seu próprio sistema operacional completo, que por sua vez interage com todos os hardwares virtuais que lhe foi entregue pelo modelo de virtualização a nível de máquina. Com as novas funcionalidades de gerência de ambientes de máquinas virtuais é possível especificar quanto recurso físico cada ambiente virtual usará e até mesmo aumentar gradualmente em caso de necessidade pontual.

**CONTÊINER**

Esse modelo de virtualização está no nível de sistema operacional, ou seja, ao contrário da VM um contêiner não tem visão de uma máquina inteira, ele é apenas um processo em execução em um kernel compartilhado entre todos os outros contêineres. Como já foi dito nesse artigo, ele utiliza o namespace para prover o devido isolamento de memória RAM, processamento, disco e acesso à rede, ou seja, mesmo compartilhamento o mesmo kernel, esse processo em execução tem a visão de estar usando um sistema operacional dedicado. Outra novidade do Docker foi a criação do conceito de “imagens”, que grosseiramente podemos descrever que as imagens são definições estáticas de como os contêineres devem ser no momento da sua inicialização. São como fotografias de um ambiente. Uma vez instanciadas, colocadas em execução, elas assumem a função de contêineres, ou seja, saem da abstração de definição e se transformam em processos em execução, dentro de um contexto isolado, que enxergam um sistema operacional dedicado pra si, mas na verdade compartilham o mesmo kernel. Aos desenvolvedores, imaginem que a **imagem** está para uma **classe**, assim como o **contêiner** está para um **objeto**.

****

### CONCLUSÃO DO PONTO DE VISTA DA INFRAESTRUTURA

Com os dados apresentados, podemos perceber que o ponto de conflito entre as soluções é baixo. Elas podem e normalmente são adotadas em conjunto. Você pode provisionar uma máquina física com um servidor de máquinas virtuais, onde serão criadas máquinas virtuais hospedes, que por sua vez terão o Docker instalado em cada máquina. Nesse Docker serão disponibilizados os ambientes com seus respectivos serviços segregados, cada um em um contêiner.

### CONCLUSÃO DO PONTO DE VISTA DOS DESENVOLVEDORES(AS)

Você já ouviu aquela famosa frase "**na minha máquina funciona"**? Este é o problema que o Docker veio resolver. Em si o Docker é um **gerenciador de contêineres** que encapsula sua aplicação dentro dele. Não importa o ambiente em que você esteja rodando, o Docker garanteque a sua aplicação rodará sempre da mesma maneira pois o contêiner, além de conter a sua aplicação, também contém tudo o que você precisa para que o seu sistema funcione. Além disso você pode integrar o Docker ao seu ambiente de desenvolvimento (código) com o **docker-composer** a fim de garantir que qualquer um desenvolvedor(a) consiga dar continuidade no seu código sem ter que se preocupar com a instalação local de serviços como apache, php, mysql e etc dentro das versões corretas, pois tudo já estará devidamente configurado por você com o **dockerfile**, e uma vez feitas essas configurações, basta apenas subir os containers para levantar o ambiente e sair codando.

**DOCKER**

**Como instalar o Docker no Ubuntu?**

**Direto do repositório Ubuntu via terminal:**

sudo apt update

sudo apt install docker.io

**Como saber qual versão do Docker está instalada?**

**Simples:**

docker -v

**→ Com mais informações:**

docker version

**Como ver mais detalhes sobre o Docker instalado na máquina?**

docker info

**Como administrar o serviço do Docker no Ubuntu?**

**Verificar o status do serviço:** systemctl status docker.service

**Ativar o serviço:** sudo systemctl start docker.service

**Desativar o serviço:** sudo systemctl stop docker.service

**Reiniciar o serviço:** sudo systemctl restart docker.service

**Como permitir usuários não administradores (non-root) a utilizar o Docker?**

**Identificar usuários no ubuntu:**

passwd -S

**→** Caso queira você também pode utilizar o **finger**:

sudo apt install finger

finger

**→** Listar os **grupos** existentesno Ubuntu**:**

cat /etc/group

**Adicionando usuários ao grupo Docker:**

sudousermod -a -G docker **usuario\_maquina\_local**

**CONTAINERS**

**Como criar um novo contêiner (ubuntu)?**

**→** Cria o contêiner e logo depois já o **encerrar**:

docker run --name ubuntu-pravaler ubuntu

**Obs.:** O contêiner continua existindo, porém está inativo. Você pode usar o parâmetro **--name** para dar um apelido ao contêiner.

**→** Cria o contêiner e o torna **interativo**:

docker run -i -t (-it) ubuntu:18.04 /bin/bash

**Obs.:** Ao tornar um contêiner interativo **(-i)**, você também pode executar comando sem que o mesmo seja encerrado em primeiro plano, como por exemplo adicionar o parâmetro **-t** para executar o bash do Ubuntu. Também é possível definir a **versão** da imagem como no exemplo acima logo após os dois pontos **(:18.04)**.

**→** Cria o contêiner e o deixa rodando em **segundo plano**:

docker run -d ubuntu

**Obs.:** Se você listar os contêineres com **docker ps**, o contêiner rodando em segundo plano não será listado, porém ainda sim o mesmo estará rodando, caso tenha exposto uma porta você poderá acessar os serviços do contêiner por lá.

**Como visualizar quais contêineres estão em execução (ativos)?**

docker ps

**→** Listar todos os contêineres, até mesmo os inativos:

docker ps -a

**Como acessar um contêiner?**

docker attach **conteiner\_id**

**Como executar algo em um contêiner?**

docker **exec** -it **conteiner\_id** /bin/bash

**Como sair de um contêiner?**

Estando dento do contêiner...

1º - Digite **exit** no terminal (para o contêiner).

2º - Aperte as teclas **Ctrl + P** **+ Q** no teclado (o contêiner ainda continua rodando).

**Como remover um contêiner existente?**

docker rm **conteiner\_id**

**Como administrar os contêineres no Docker?**

**Ativar o serviço:** docker start **conteiner\_id**

**Desativar o serviço:** docker stop **conteiner\_id**

**Reiniciar o serviço:** docker restart **conteiner\_id**

**Como acessar o terminal dentro de um contêiner?**

docker exec -it **conteiner\_id** /bin/bash

**Como ver os logs do contêiner?**

docker logs **conteiner\_id**

**Como visualizar os processos que estão sendo executados dentro dos contêineres?**

docker top **conteiner\_id**

**Como obter informações detalhadas do contêiner?**

docker inspect **conteiner\_id**

**IMAGES**

**Como baixar imagem e armazenar localmente a partir do Docker Hub?**

**1.** Ele fará o download de uma imagem marcada como **latest**

docker pull **nome\_imagem**

**2.** Com a opção -a, você fará o download de todas as versões da imagem:

docker pull -a **nome\_imagem**

**Como visualizar as imagens baixadas localmente?**

docker images

**Como criar uma nova imagem de um contêiner que foi alterado?**

docker commit **conteiner\_id** **nome\_nova\_imagem:nome\_tag**

**Obs.:** Se você não colocar um nome de tag a imagem ficará como **latest** (versão mais recente).

**Como salvar uma imagem em uma pasta?**

docker save **nome\_imagem:tag\_imagem > nome\_imagem.tar**

**Como importar uma imagem?**

docker load **< nome\_imagem.tar**

**Como ver o histórico dos comandos usados para criar a imagem?**

docker history **nome\_imagem**

**Como marcar (renomear) uma imagem ou tag?**

docker tag **imagem\_id** **nome\_repositorio/nome\_nova\_imagem**:**tag\_imagem**

**→** Para todas as ações abaixo, é necessário ter uma conta criada no [**Docker Hub**](https://hub.docker.com/).

**Como publicar uma imagem no Docker Hub?**

**1. Marcar a imagem:** docker tag **imagem\_id** **usuario\_docker\_hub/nome\_imagem:tag\_imagem**

**2. Faça o login no Docker Hub:** docker login (insira seu login e senha).

**3. Publicando a imagem**: docker push **usuario\_docker\_hub/nome\_imagem:tag\_imagem**

**4.** Estando tudo certo a imagem foi enviada para o seu diretório público WEB do Docker Hub.

**Como publicar uma imagem privada no Docker Hub?**

**1.** Va ao Docker Hub e crie um repositório privado (guarde o nome dele).

**2. Marcar a imagem:**

docker tag **imagem\_id usuario\_docker\_hub/nome\_repositorio\_privado:tag\_imagem**

**3. Faça o login no Docker Hub:** docker login (insira seu login e senha).

**4. Publicando a imagem**:

docker push **usuario\_docker\_hub/nome\_repositorio\_privado:tag\_imagem**

**5.** Estando tudo certo a imagem foi enviada para o seu repositório público do Docker Hub.

**Como usar um registro público ou privado no Docker Hub?**

docker pull **usuario\_docker\_hub/nome\_repositorio**

**VOLUMES**

**Volumes** compartilham dados em comum para serem acessados de quaisquer contêineres, mesmo quando o contêiner não está em execução.

**Como criar um volume?**

docker volume create --name **nome\_volume**

**Como criar um volume junto com um contêiner?**

docker run -it -v /**nome\_volume** ubuntu /bin/bash

**→** Para acessar o volume de um contêiner através da sua máquina local (Ubuntu) é necessário realizar os seguintes passos abaixo (todos os passos são realizados no terminal do seu S.O).

**1.** docker inspect -f {{.Mounts}} **conteiner\_id** (apresentará o hash e o diretório local do volume).

**2.** sudo su (Entre com a senha).

**3.** cd **diretorio\_volume**

**4.** ls (para listar os itens criados dentro do volume do contêiner em especifico).

**Obs.:** Também é possível especificar um diretório para criar o volume como por exemplo...

docker run -it -v **/root/nome\_diretorio:/nome\_volume** ubuntu /bin/bash

**Como fazer um contêiner acessa um volume já existente?**

docker run -it --volumes-from **nome\_volume** --name pravaler ubuntu /bin/bash

**Como visualizar os volumes?**

docker volume ls

**Como obter informações detalhadas do volume?**

docker volume inspect **nome\_volume**

**Como excluir volume?**

docker volume rm **nome\_volume**

**NETWORK**

**Como criar apenas uma rede?**

docker network create **nome\_rede**

**Como criar um contêiner e já fazê-lo se comunicar pela rede?**

docker run -d **-p 90:80** nginx

docker run -d -e PMA\_HOST=dbhost **-p 91:80** phpmyadmin/phpmyadmin

**Como visualizar as conexões de rede do docker?**

docker network ls

**Como excluir uma rede?**

docker network rm **rede\_id**

**Como ver a porta mapeada em um contêiner?**

docker port **nome\_conteiner**

**Como vincular contêineres?**

**1.** Executar contêiner chamado **nome\_conteiner1**.

docker run --name=**nome\_conteiner1** -d **conteiner\_id**

**2.** Executar um contêiner chamado **nome\_conteiner2** e vinculá-lo ao **nome\_conteiner1**.

docker run --name= **nome\_conteiner2** --link=**nome\_conteiner\_1:novo\_nome\_vinculo** -d **conteiner\_id**

**BUILDING IMAGES FROM A DOCKERFILE**

O nome do arquivo deve ser **Dockerfile**, sem extensão, com a primeira letra maiúscula e o restante em minúscula. Todos os arquivos na mesma pasta do Dockerfile serão incluídos na compilação.

|  |  |  |
| --- | --- | --- |
| **INSTRUÇÃO** | **DESCRIÇÃO** | **ADICIONAR NOVA CAMADA?** |
| # | É usado para comentários | Não |
| FROM | É usado para explicar em qual imagem é baseada. Deve ser a primeira instrução do Dockerfile. | Não |
| MAINTAINER | Informa o endereço de email. | Sim |
| RUN | Ele executa comandos no tempo de criação e adiciona camadas às imagens. É comum usá-las para instalar aplicativos. | Sim |
| CMD | Ele executa o comando dentro de um contêiner quando é lançado. Apenas um CMD por Dockerfile. Tem dois formulários a serem usados: Formulário Shell (Executado da mesma forma que os comandos shell) ou Formulário Exec (JSON array-style ["command", "arg1"]). Pode ser substituído se especificarmos os argumentos após o docker run **IMAGE\_NAME** | Sim |
| ADD | Adiciona arquivos a imagem. | - |
| EXPOSE | Ele irá expor as portas a serem usadas. | - |
| ENTRYPOINT | Não pode ser sobrescrita. As palavras após docker run **IMAGE\_NAME** serão interpretadas como argumentos. | - |
| ENV | Passar variáveis de ambiente para contêineres. | Sim |
| VOLUME | É usado para compartilhar dados comuns. | - |

**Exemplo de Dockerfile**

**1.** Crie uma pasta chamada “Dockerfile PRAVALER” na área de trabalho e adicione os seguinte aquivos dentro desse repositório.

**1º** Dockerfile

**2º** entrypoint.sh

**3º** example\_php.conf

**4º** www.conf

**Obs.:** O Dockerfile não tem extensão, apenas o seu nome mesmo.

**2.** Estando dentro da pasta “Dockerfile PRAVALER” execute o comando abaixo para **“buildar”** uma nova imagem conforme as especificações do Dockerfile.

docker build -t **nome\_imagem:tag\_imagem .**

**Obs.:** O **‘.’** representa o diretório atual onde está o arquivo Dockerfile.

**3.** Veja se a imagem foi gerada.

docker images

**4.** Crie a pasta **“teste”** na **área de trabalho** e adicione o seguinte arquivo dentro.

teste.html

**5.** Crie um contêiner usando a nova imagem.

docker run -it --name **nome\_conteiner** -v **diretorio\_teste:**/var/www/html -p 8080:80 -d **nome\_imagem**

**6.** Agora acesse o **localhost:8080/teste.html** através do navegador e visualize a página.