

**Objetivo do curso:**

O curso visa fornecer uma apresentação básica sobre o que é o docker, como funciona o dockerfile e o docker compose e mostrar os conceitos de ambiente em container.

O Docker foi desenvolvido em 2013 por uma empresa chamada dotCloud, Inc (que futuramente se chamará Docker, INC) com a necessidade de melhoramento da grande demanda de máquinas virtuais. A criação das máquinas virtuais dependiam de grandes espaços de discos e outros recursos e ainda havia a questão das aplicações, que poderiam ser corrompidas em algumas situações pelas próprias máquinas virtuais. Com isso, houve a necessidade desenvolver um sistema e assim surgiu o LXC.\*

O LXC foi criado em 2008 e é uma tecnologia que trabalha com a criação de instâncias isoladas de um sistema operacional dentro da máquina hospedeira. Ou seja, a base do docker começou a ser desenvolvida. Com sua facilidade de uso, o mercado obteve uma rápida adoção dessa tecnologia contanto houve um “boom” nas empresas após mostrar que o desenvolvedor conseguiria colocar suas aplicações diretamente do notebook para a produção.\*

O Docker é a terceira plataforma mais utilizada pelos desenvolvedores no mundo. Empresas como Microsoft, Google, Red Hat (IBM) começaram a utilizá-lo em meados de 2014. O investimento dessas grandes empresas na comunidade de desenvolvimento, ajudou com que a ferramenta melhorasse a cada a dia mais. Com isso, desenvolvedores, sysadmin e até mesmo os DBA’s usam docker em suas squads e projetos.\*

O Stack Overflow, na pesquisa sobre os interesses dos desenvolvedores, disse que Docker e Kubernetes eram as plataformas mais amadas e procuradas entre os entrevistados. \*

O Docker já é uma realidade no mercado brasileiro. Basta uma breve pesquisa no linkedin e você já terá resultados de vagas que incluem ao técnico experiência ou convívio com Docker e tecnologias que envolvem contêiners.\*

\*FONTE: (<https://4linux.com.br/o-que-e-docker/> - 15/08/2022)

**O que é Docker?**

É uma ferramenta que se apoia em recursos existentes no kernel, inicialmente Linux, para isolar  
a execução de processos. As ferramentas que o Docker traz são basicamente uma camada de  
administração de containers, baseado originalmente no LXC

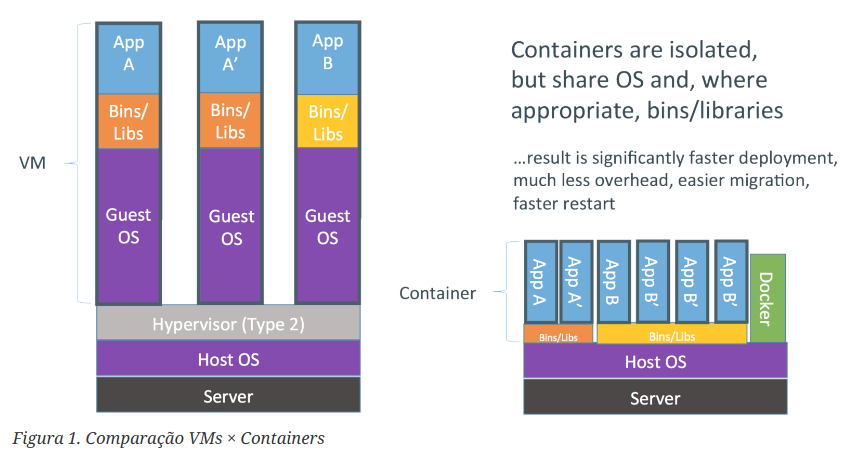
(KERNEL: Kernel é uma palavra inglesa usada na computação para designar o núcleo do sistema operacional, que é a parte principal de um computador. Uma simples alteração da versão do Kernel para uma mais antiga ou mais atual pode ser suficiente para resolver problemas de hardware e também de compatibilidade no computador. Com um controle total de tudo relacionado ao sistema, o Kernel é um dos primeiros programas a ser carregado durante a inicialização. Assim que começa a ser executado, o Kernel inicia um processo que detecta todo o hardware necessário para que ocorra um bom funcionamento do computador. Além disso, um Kernel opera solicitações de entrada e de saída de software, e gere, por exemplo, memória e aparelhos periféricos utilizados.)

LXC - Linux Containers é um método de virtualização em nível de sistema operacional para executar vários sistemas Linux isolados em um host de controle usando um único kernel Linux.

Alguns isolamentos possíveis  
• Limites de uso de memória  
• Limites de uso de CPU  
• Limites de uso de I/O  
• Limites de uso de rede  
• Isolamento da rede (que redes e portas são acessíveis)  
• Isolamento do file system  
• Permissões e Políticas  
• Capacidades do kernel

**Definição oficial**

Containers Docker empacotam componentes de software em um sistema de arquivos  
completo, que contêm tudo necessário para a execução: código, runtime, ferramentas de  
sistema - qualquer coisa que possa ser instalada em um servidor.  
Isto garante que o software sempre irá executar da mesma forma, independente do seu  
ambiente.

**Por que Docker não é uma VM?** O Docker tende a utilizar menos recursos que uma VM tradicional, um dos motivos é não precisar de uma pilha completa como vemos em Comparação VMs × Containers. O Docker utiliza o mesmo kernel do host, e ainda pode compartilhar bibliotecas.  
Mesmo utilizando o mesmo kernel é possível utilizar outra distribuição com versões diferentes das  
bibliotecas e aplicativos.VM - Virtual Machine (máquina virtual), recurso extremamente usado atualmente para isolamento deserviços, replicação e melhor aproveitamento do poder de processamente de uma máquina  
física.

**Devo trocar então minha VM** **por um container?**

Nem sempre, os containers Docker possuem  
algumas limitações em relação as VMs:  
• Todas as imagens são linux, apesar do host poder ser qualquer SO que use ou emule um kernel  
linux, as imagens em si serão baseadas em linux.  
• Não é possível usar um kernel diferente do host, o Docker Engine estará executando sob  
uma determinada versão (ou emulação) do kernel linux, e não é possível executar uma versão  
diferente, pois as imagens não possuem kernel.

**O que são containers?**

Container é o nome dado para a segregação de processos no mesmo kernel, de forma que o processo  
seja isolado o máximo possível de todo o resto do ambiente.  
Em termos práticos são File Systems, criados a partir de uma "imagem" e que podem possuir  
também algumas características próprias.

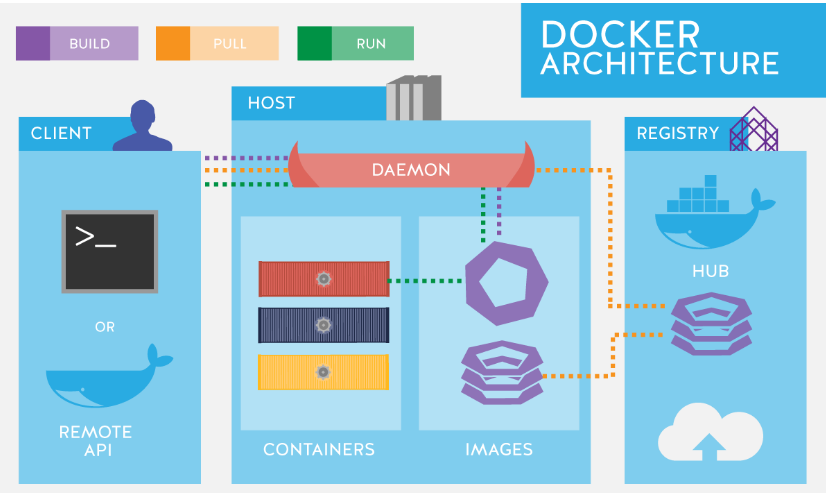
**O que são imagens Docker?**

Uma imagem Docker é a materialização de um modelo de um sistema de arquivos, modelo este  
produzido através de um processo chamado build.  
Esta imagem é representada por um ou mais arquivos e pode ser armazenada em um repositório.

**Arquitetura Docker.**

[De maneira simplificada podemos dizer que o uso mais básico do Docker consiste em:  
• Ter o serviço](https://hub.docker.com/) Docker Engine rodando  
• Ter acesso a API Rest do Docker Engine, normalmente através do Docker Client  
• Baixar uma imagem do Docker Registry, normalmente do registry público oficial:  
4

https://hub.docker.com  
• Instanciar um container a partir da imagem baixada.

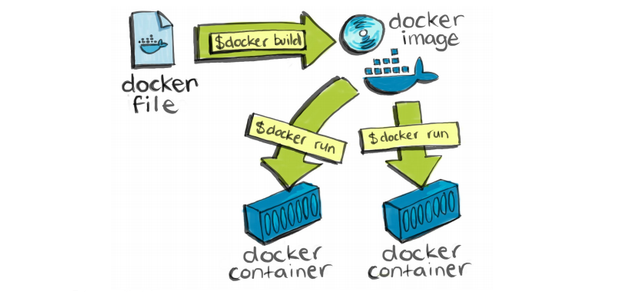


Diferenças entre container e imagem.

A imagem é um arquivo e o contêiner é um processo. Da perspectiva do kernel Linux, um contêiner é um processo com restrições. No entanto, ao invés de executar um único arquivo binário, um contêiner executa uma imagem. Uma imagem é um pacote de sistema de arquivos que contém todas as dependências necessárias para executar um processo: arquivos de biblioteca, arquivos no sistema de arquivos, pacotes instalados, recursos disponíveis, processos em execução e módulos do kernel.\*

Como os arquivos executáveis são a base para os processos em execução, as imagens são a base para contêineres em execução. Os contêineres em execução usam uma visão imutável da imagem, permitindo vários recipientes para reutilizar a mesma imagem simultaneamente. Como as imagens são arquivos, elas podem ser gerenciadas por sistemas de controle de versão, melhorando a automação do contêiner e o provisionamento.\*

As imagens do contêiner precisam estar disponíveis localmente ou armazenadas e mantidas em um repositório de imagens. Um repositório de imagens é apenas um serviço público ou privado onde as imagens podem ser armazenadas, pesquisadas e recuperadas.\*



\*FONTE: (<http://www.tecnisys.com.br/noticias/2021/qual-e-a-diferenca-entre-uma-imagem-docker-e-um-container> – 15/08/2022)

### O que é DockerFile, e como funciona?

Dockerfile é a forma de criar nossas próprias imagens, com as características necessárias para a aplicação funcionar. Lembre-se que uma IMAGEM é imutável, seria como se fosse um template para subir um container, ou seja, você sempre vai precisar de uma imagem para realizar essa ação.  
Ele funciona basicamente como um ‘livro de receitas’ onde é executada cada instrução em top-down, criando para cada instrução um step. Depois do arquivo criado, você executa o build, que criará uma nova imagem, e assim poderá utilizar a imagem já pronta com sua aplicação.

**O que é Docker-compose e como funciona?**

O Docker Compose é uma ferramenta para a criação e execução de múltiplos containers de aplicação. Com o Compose, você usar um arquivo do tipo yaml para definir como será o ambiente de sua aplicação e usando um único comando você criará e iniciará todos os serviços definidos.

**Redes padrões Docker**

O docker é disponibilizado com três redes por padrão. Essas redes oferecem configurações específicas para gerenciamento do tráfego de dados.

Cada container iniciado no docker é associado a uma rede específica. Essa é a rede padrão para qualquer container, a menos que associemos, explicitamente, outra rede a ele. A rede confere ao container uma interface que faz bridge com a interface docker0 do docker host. Essa interface recebe, automaticamente, o próximo endereço disponível na rede IP 172.17.0.0/16.

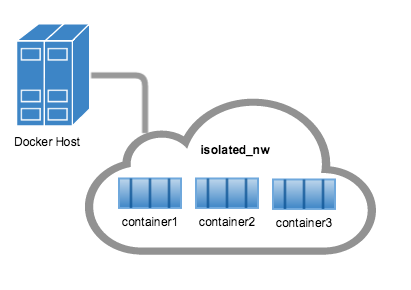
Todos os containers que estão nessa rede poderão se comunicar via protocolo TCP/IP. Se você souber qual endereço IP do container deseja conectar, é possível enviar tráfego para ele. Afinal, estão todos na mesma rede IP (172.17.0.0/16).

Um detalhe a se observar: como os IPs são cedidos automaticamente, não é tarefa trivial descobrir qual IP do container de destino. Para ajudar nessa localização, o docker disponibiliza, na inicialização de um container, a opção “–link“ que é responsável por associar o IP do container de destino ao seu nome.

As redes criadas pelo usuário com o driver bridge tem todas as funcionalidades descritas na rede padrão, chamada bridge, porém, com funcionalidades adicionais.

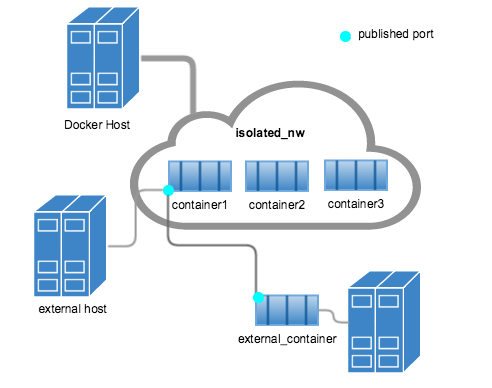
Dentre uma das funcionalidades: a rede criada pelo usuário não precisa mais utilizar a opção antiga “–link”. Pois, toda rede criada pelo usuário com o driver bridge poderá utilizar o DNS interno do Docker que, associa, automaticamente, todos os nomes de containers dessa rede para seus respectivos IPs da rede IP correspondente.

Para deixar mais claro: todos os containers que estiverem utilizando a rede padrão bridge não poderão usufruir da funcionalidade de DNS interno do Docker. Caso utilize essa rede, é preciso especificar a opção legada “–link” para tradução dos nomes em endereços IPs dinamicamente alocados no docker.



Vale salientar: um container que está em determinada rede não acessa outro container que está em outra rede. Mesmo que você conheça o IP de destino. Para que um container acesse outro container de outra rede, é necessário que a origem esteja presente nas duas redes que deseja alcançar.

Os containers que estão na rede isolated\_nw podem expor suas portas no docker host e essas portas podem ser acessadas tanto por containers externos a rede, chamada isolated\_nw, como máquinas externas com acesso ao docker host.



Criando um ambiente de desenvolvimento com Docker.

Para este modelo estaremos utilizando os seguintes serviços:

* NGINX
* PHP

**Estrutura do projeto.**

docker images

docker ps

docker run hello-world

docker ps -a

## docker status

docker exec

docker start

docker stop

docker-compose

docker container ls -a

docker container rm <container ID>

“ Eu acredito que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar.”

Alan Turing

