# Containers

Container nada mais é do que um ambiente isolado contido em um servidor que, diferentemente das máquinas virtuais, divide um único host de controle. Para contextualizar melhor, vamos imaginar um navio cargueiro com vários containers dentro. Se um dos recipientes se danificar, não afetará os outros ou o navio, pois cada um está isolado e protegido.

Esse isolamento de container, trazendo de volta para o mundo do desenvolvimento, possibilita uma utilização limitada do HD, memória RAM e processador. Ao utilizar um tipo de compartilhamento de kernel, os containers **apresentam uma capacidade de economia de recursos maior do que as máquinas virtuais**.

## O que é Docker?

Provavelmente você já ouviu o termo container junto de docker, como se fosse uma coisa só, não é mesmo? O Docker é uma plataforma Open source que foi desenvolvida na linguagem de programação do Google, o Go. O Funcionamento do docker é possibilitado pelo Linux container — LXC — que nada mais é do que um sistema do kernel do Linux.

É exatamente isso que possibilita a utilização de recursos isolados, diferentemente do que seria em uma máquina virtual, que depende da criação de um sistema operacional completo para cada módulo.

Em suma, podemos dizer que o docker junta as partes de softwares de um sistema de arquivo completo e reúne os recursos essenciais para a sua boa execução. Isso significa que os recursos são instalados no servidor e armazenados nos containers, dessa forma, os softwares poderão ser executados facilmente em diferentes ambientes de desenvolvimento.

Podemos dizer, então, que o docker separa os recursos e utiliza as bibliotecas de kernel em comum com outros containers. Essa facilidade em trabalhar sempre com as bibliotecas do sistema operacional do servidor, torna o container docker portável e **possibilita a implantação em ambientes heterogêneos e o trabalho em conjunto.**

Toda essa portabilidade e agilidade de implantação, sem a necessidade de um sistema operacional por módulo, faz com que o container docker ganhe cada vez mais mercado mundo afora, principalmente com o crescimento da computação em nuvem.

## Para que serve o container docker?

Boa parte das grandes e médias empresas ainda apostam em softwares como ERPs e CRMs, que iniciam como projetos básicos e se tornam ineficientes com o passar do tempo, graças aos seus códigos fontes estáticos e monolíticos.

O container surge como uma alternativa a isso, pois, esse modelo de desenvolvimentos é feito em diferentes etapas. A aplicação é dividida em componentes menores, também conhecidos como micros serviços. Com essa desagregação, **os desenvolvedores conseguem, então, adotar um tipo de arquitetura que melhora a eficiência operacional**.

Nesse cenário, há um código fonte para cada componente do software, permitindo assim a divisão em estágios, como por exemplo, ambiente de testes, virtual e de produção.

O container encapsula todos esses componentes e garante uma performance consistente, com um pacote único, leve, que permite a execução de aplicativos com consistência em ambientes físicos ou virtualizados. O sistema responsável pela comunicação entre cliente e o servidor é o docker , por meio de uma API.

## Como funciona um container docker ?

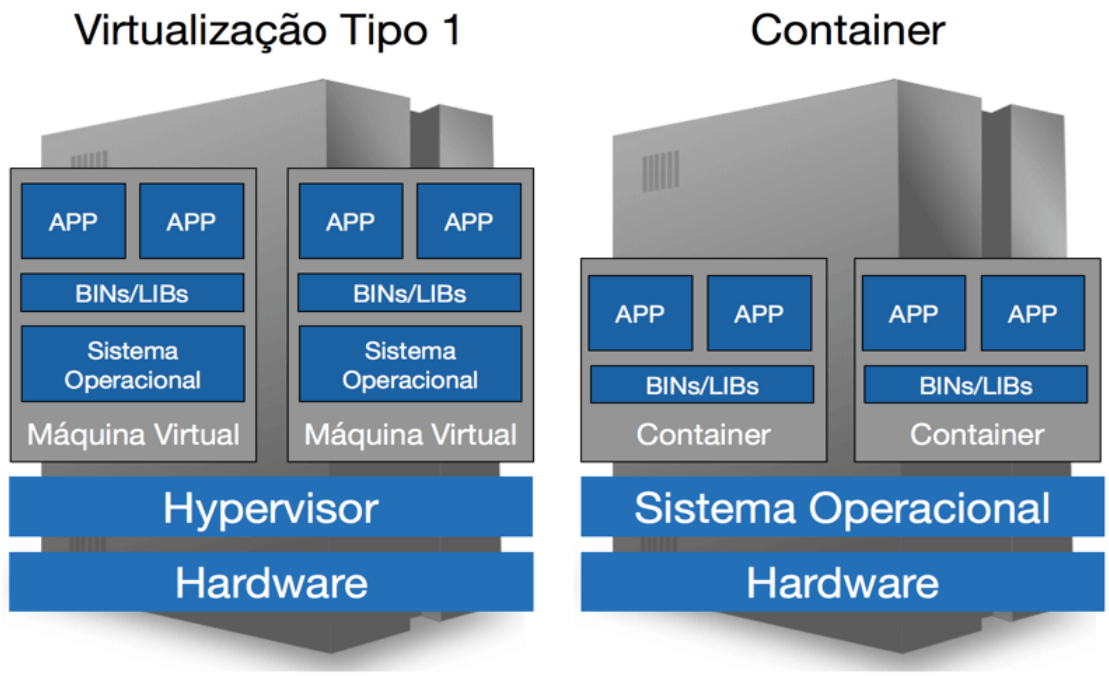
A plataforma utiliza um conjunto de recursos que possibilita a criação e administração dos containers, até mesmo a limitação de recursos é feita no docker. Entre esses conjuntos de recursos estão, a biblioteca *lib container*, que realiza a comunicação entre o Docker Daemon e o *backend*.

Outro fator importante a ser colocado aqui é que o docker funciona em camadas, ou seja, cada container é montado por meio de *chroot*, *namespaces*, *cgroups* e outras funcionalidades do Kernel. São essas layers que garantem o isolamento da área de sua aplicação.

Isso significa que, no docker, o kernel monta o *rootfs* apenas no modo de leitura e depois um arquivo de sistema é criado, como *read-write* sobre o *rootf*. Em seguida, o kernel substitui a partição raíz, como *read-only* e insere um novo file system sobre o *rootfs*.

O container fica pronto para executar a imagem depois do carregamento do *rootfs*, pois ele também é uma das camadas *read-write* que foi criada a partir de uma imagem de leitura.

**É isso tudo que faz com que o container docker seja uma excelente ferramenta** para DevOps. Para os administradores de sistema, um dos maiores benefícios é a flexibilidade, baixo custo e redução da área ocupada. Já os desenvolvedores ganham em liberdade para focar na sua atividade principal.




Elis Cristine Correa Silva (3 DES – 2021).

Pesquisa Professor: Wellington