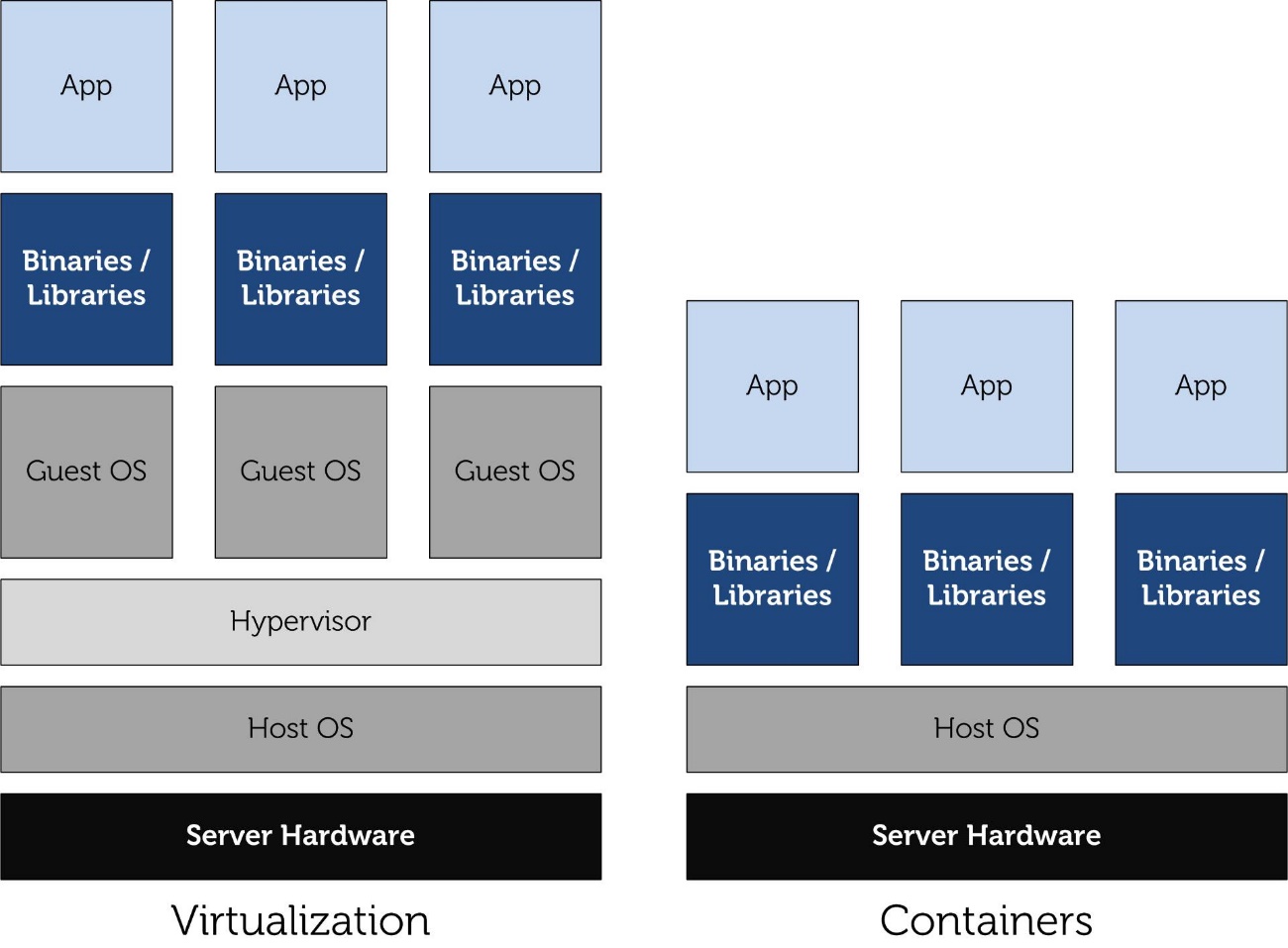
**Docker (Containers)**

Docker é como um conjunto de caixas de brinquedos. Imagine que você tem muitos brinquedos diferentes e quer organizá-los. Você pode colocar todos os seus carrinhos em uma caixa, todas as suas bonecas em outra caixa e assim por diante. Cada caixa tem seu próprio conjunto de brinquedos que não se misturam com os outros.

Agora, imagine que cada caixa é um “container” Docker. Cada container tem seu próprio software, bibliotecas e arquivos que ele precisa para funcionar. Assim como seus brinquedos não se misturam com os outros quando estão em suas próprias caixas, o software em um container Docker não se mistura com o software em outros containers. Isso ajuda a manter tudo organizado e funcionando sem problemas!

E a melhor parte é que você pode mover essas caixas (ou containers) para qualquer lugar! Se você for para a casa da vovó, pode levar suas caixas de brinquedos com você e brincar com elas lá. Da mesma forma, você pode mover os containers Docker de um computador para outro, e eles ainda funcionarão da mesma maneira!

[O que é Docker? | Mundo Docker](https://www.mundodocker.com.br/o-que-e-docker/)



O Docker possibilita o empacotamento de uma aplicação ou ambiente inteiro dentro de um container, e a partir desse momento o ambiente inteiro torna-se portável para qualquer outro Host que contenha o Docker instalado.

O Docker possibilita o empacotamento de uma aplicação ou ambiente inteiro dentro de um container, e a partir desse momento o ambiente inteiro torna-se portável para qualquer outro Host que contenha o Docker instalado.

[O tempo de deploy se refere ao período necessário para mover uma aplicação ou atualização de software do desenvolvimento para a produção, tornando-a acessível aos usuários finais1](https://kenzie.com.br/blog/o-que-e-deploy/)[2](https://www.hostgator.com.br/blog/o-que-e-deploy-e-como-realiza-lo/).

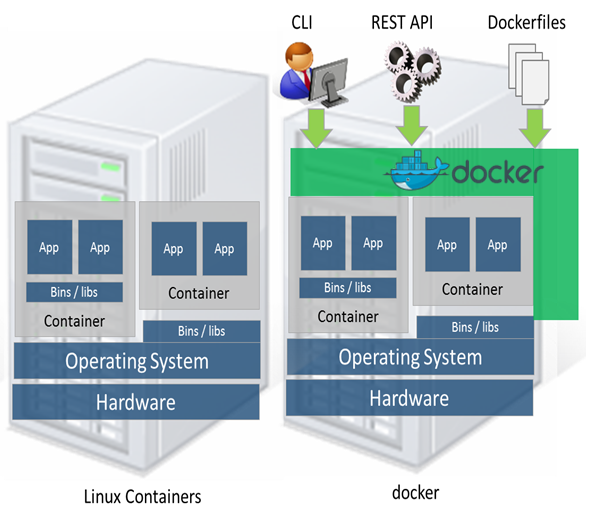
O processo de deploy geralmente envolve várias etapas, incluindo:

1. [Desenvolvimento: Esta é a fase inicial, onde o software é desenvolvido e testado na máquina do programador1](https://kenzie.com.br/blog/o-que-e-deploy/).
2. [Staging/Teste: Nesta fase, o software é movido para um ambiente de teste, onde é submetido a mais testes para garantir que está funcionando corretamente1](https://kenzie.com.br/blog/o-que-e-deploy/).
3. [Produção: Após todos os testes serem concluídos e quaisquer problemas serem corrigidos, o software é finalmente movido para o ambiente de produção, onde está disponível para os usuários finais1](https://kenzie.com.br/blog/o-que-e-deploy/).

O tempo de deploy pode variar dependendo da complexidade do software, da infraestrutura de TI da organização e das práticas de deploy utilizadas. [Por exemplo, algumas organizações utilizam práticas de Continuous Deployment (Implantação Contínua), que visam minimizar o tempo entre o desenvolvimento de uma nova linha de código e seu uso pelos usuários3](https://www.objective.com.br/insights/continuous-deployment/)[4](https://gaea.com.br/o-que-e-continuous-deployment/).

Outra facilidade do Docker é poder criar suas imagens (containers prontos para deploy) a partir de arquivos de definição chamados Dockerfiles (veremos isso melhor em posts futuros).

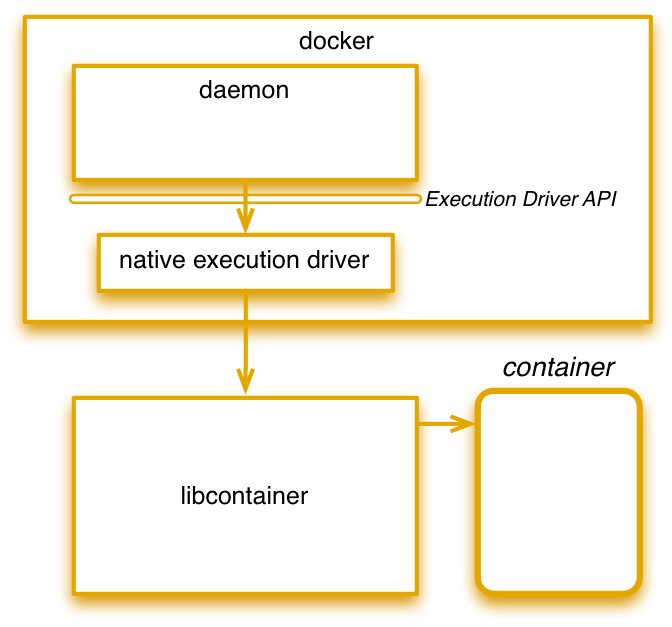
Não podemos nos esquecer também de que o Docker utiliza como backend default o LXC, com isso é possível definir limitações de recursos por container (memória, cpu, I/O, etc.)



**Como o Docker faz isso?**

Como ele trabalha utilizando cliente e servidor (toda a comunicação entre o Docker Daemon e Docker client é realizada através de API), basta apenas que você tenha instalado o serviço do Docker em um lugar, e aponte em seu Docker Client para esse servidor. A plataforma do Docker em si utilizada alguns conjuntos de recursos, seja para a criação ou administração dos containers, entre esses conjuntos podemos destacar a biblioteca libcontainer, que é responsável pela comunicação entre o Docker Daemon e o backend utilizado, é ela a responsável pela criação do container, e é através dela que podemos setar os limites de recursos por container.

Workflow dentro do Docker:



DIEDRICH, C. **O que é Docker? | Mundo Docker**. Disponível em: <https://www.mundodocker.com.br/o-que-e-docker/>. Acesso em: 28 set. 2023.

‌

Pesquisa 2

[O que é docker, como funciona e quais as vantagens (certificacaolinux.com.br)](https://www.certificacaolinux.com.br/o-que-e-docker/)

RIBEIRO, U. **O que é docker, como funciona e quais as vantagens**. Disponível em: <https://www.certificacaolinux.com.br/o-que-e-docker/>. Acesso em: 28 set. 2023.

‌

**O que é Docker?**

Plataforma open source, o Docker viabiliza a virtualização e criação de ambientes isolados, ou containers no Linux.

Com todas as ferramentas que o compõem, é possível executar esses ambientes de forma eficiente e otimizada.

Essa tecnologia também permite que qualquer equipe opere e tenha acesso às funcionalidades da infraestrutura do sistema e do servidor, sem precisar de qualquer permissão especial.

Afinal, podemos comparar sua aplicação com a das máquinas virtuais (do inglês, Virtual Machine ou VM). A diferença é que ele também é compatível com o armazenamento em nuvem.

Tais características facilitam muito o trabalho em conjunto dos profissionais e desenvolvedores, contribuindo para a flexibilidade e agilidade das entregas.

**O que faz o Docker?**

O Docker utiliza recursos padrão do [kernel do Linux](https://www.certificacaolinux.com.br/como-funciona-o-kernel-do-linux/) para sua execução. Isso faz com que seus processos sejam independentes, com a possibilidade de serem operados individual e simultaneamente.

Quaisquer arquivos do Docker podem ser “empacotados”, ou reunidos, em containers, e podem ser transportados de um ambiente a outro de modo simples e rápido. Assim, a portabilidade é uma das características mais marcantes do Docker.

**5 vantagens de usar Docker:**

**Comunidade com repositório**

Existe uma grande comunidade com um imenso repositório de imagens do Docker, com modelos prontos para adaptação.

Desde pilares de infraestrutura até sistemas de integração, é possível encontrar no repositório, que tem imagens originais, já configuradas e devidamente regularizadas.

Basta, então, dar o seu “toque final” nas imagens e adicionais quaisquer configurações e adaptações que seu sistema precisar.

**Infraestrutura e Desenvolvimento com a mesma linguagem**

Ter uma linguagem só para a infraestrutura e o desenvolvimento em Docker era, até então, impossível.

Muitos problemas e impasses surgiram devido à divergência de linguagens. Mas foram amenizados com um idioma comum.

**Ambientes semelhantes**

Basta transformar um arquivo em imagem Docker e ele pode ser armazenado e transportado para qualquer container. São vários ambientes possíveis para manter o seu arquivo.

É possível, ainda, integrar banco de dados com o container e criar uma estrutura ainda mais completa e funcional. Essa opção é altamente útil para aqueles que fazem testes em diferentes ambientes ou servidores.

**Toda a aplicação e dependências em um só pacote**

Chega de sofrer com cada arquivo em um lugar! Com a metodologia Docker, você pode ter todas as aplicações e dependências em um só pacote.

Você precisa apenas se certificar de que as imagens são facilmente acessíveis, e disponibilizar a configuração de execução para os usuários que baixarem o arquivo.

Fases como armazenamento e atualização também são viabilizadas com o Docker. Afinal, é possível manter diversas versões de uma mesma imagem no container.

Quando um arquivo é modificado, basta executar alguns poucos [comandos Docker](https://www.certificacaolinux.com.br/comandos-docker/) para que a atualização aconteça também nos arquivos baixados.

**Ganho de tempo com replicação do ambiente docker**

Com tantas funcionalidades, é nítido que o usuário ou profissional ganhará tempo. Ainda outra utilidade é a padronização e possibilidade de replicação que o sistema oferece.

Para reproduzi-las fielmente, as imagens precisam seguir todas as práticas de construção durante seu processo de estruturação.

**O que são containers?**

Containers representam a virtualização de um ambiente isolado no kernel do sistema operacional Linux.

Isso, por sua vez, permite que cada nível seja executado separadamente, otimizando tempo e configurações do sistema.

Quando é necessário executar camadas de desenvolvimento simultaneamente, a tecnologia Docker em containers é a mais indicada.

**O que são imagens?**

Uma imagem Docker é como uma receita de bolo. Ela contém todas as instruções para criar um container Docker, assim como uma receita de bolo contém todas as instruções para fazer um bolo.

Quando você segue uma receita de bolo, você junta todos os ingredientes e segue as etapas para criar o bolo. Da mesma forma, quando o Docker usa uma imagem, ele junta todos os arquivos e configurações necessários e segue as instruções para criar um container.

Assim como você pode usar a mesma receita de bolo para fazer muitos bolos, você pode usar a mesma imagem Docker para criar muitos containers. E assim como diferentes receitas de bolo produzem diferentes tipos de bolos, diferentes imagens Docker produzem diferentes tipos de containers.

As imagens docker são os arquivos que constroem a aplicação, que geralmente é feita em camadas.

São as imagens as responsáveis por armazenar todas as informações que um processo precisa para ser executado, como:

* bibliotecas;
* sistemas de arquivos;
* pacotes e recursos;
* módulos do kernel e muito mais.

Assim, é como se o processo sempre dependesse de uma imagem para ser executado corretamente, trazendo resultados expressivos para o usuário.

**A diferença entre container e imagem:**

Ao passo que a imagem é como um arquivo responsável por armazenar todas as informações necessárias, o container corresponde ao processo que é executado por ela.

Cada imagem pode criar mais de um container e eles só podem ser virtualizados, como é a proposta, se tiverem acesso a imagens completas, com pacotes de arquivos exatos.

Os containers, quando estão em execução, são capazes de manter uma tarefa por vez, com uma responsabilidade por container.