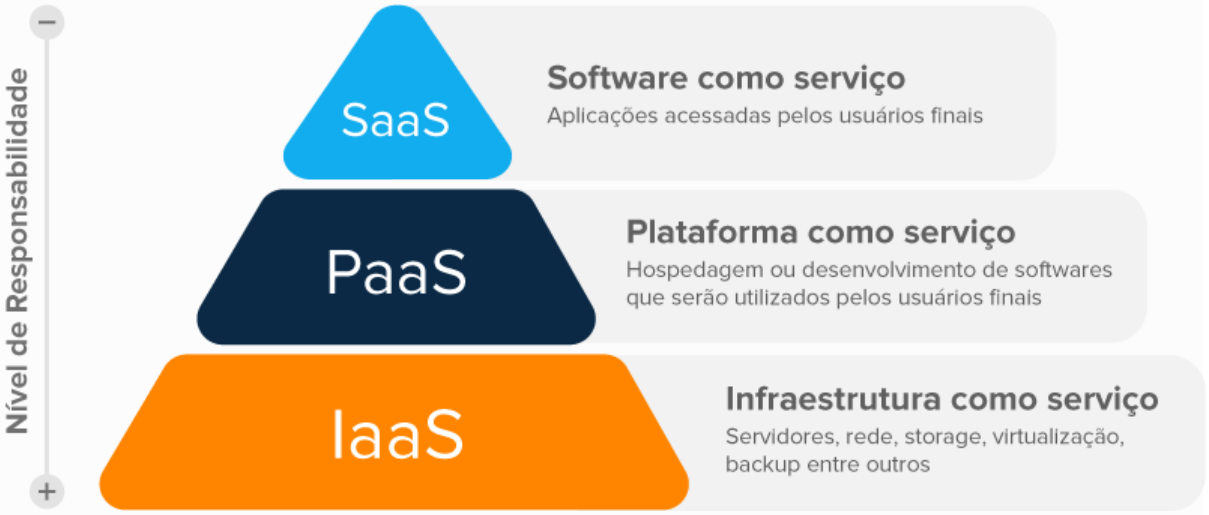
Camadas de Computação:

Na área de TI, é comum ouvir a palavra “camada” quando alguém quer se referir às tecnologias que compõem a infraestrutura de um ambiente computacional, por exemplo. Isso porque **o ambiente computacional depende de diferentes tecnologias** trabalhando juntas para, de fato, funcionar.

Existem camadas dentro de camadas, níveis de responsabilidade e gerenciamento que definem a forma como a empresa vai lidar com a tecnologia que chega até seus colaboradores (os clientes ou usuários finais). Até chegar nesse ponto, temos:

- Infraestrutura (servidores, storage, rede e virtualização)
- Plataformas (sistema operacional, middleware, runtime)
- Softwares (aplicações e dados)

Em cloud computing, essas camadas são popularmente divididas em três níveis:



Camadas da computação em nuvem e níveis de responsabilidade em IaaS, PaaS e SaaS.

Entender que é possível ter diferentes níveis de camadas é essencial para compreendermos o que é um cluster de container.

O que é um container na TI?

Resumindo -> é V (**virtualização**)
Uma máquina virtual permite a virtualização de toda a infraestrutura de TI (rede, servidores e storage) através de um sistema operacional próprio (hipervisor), criando uma nova camada de computação: a virtualização. Já o container permite a virtualização de aplicativos de software utilizando o sistema operacional do host em um kernel compartilhado.

Container é um processo de computação que utiliza uma pequena fração do recurso de hardware para viabilizar a execução de aplicativos de forma mais eficiente. Isolando, para entender melhor, vamos retomar o conceito de virtualização:
Pontos importantes: -> testar em diferentes tipos de ambientes

Docker vs **VM:**

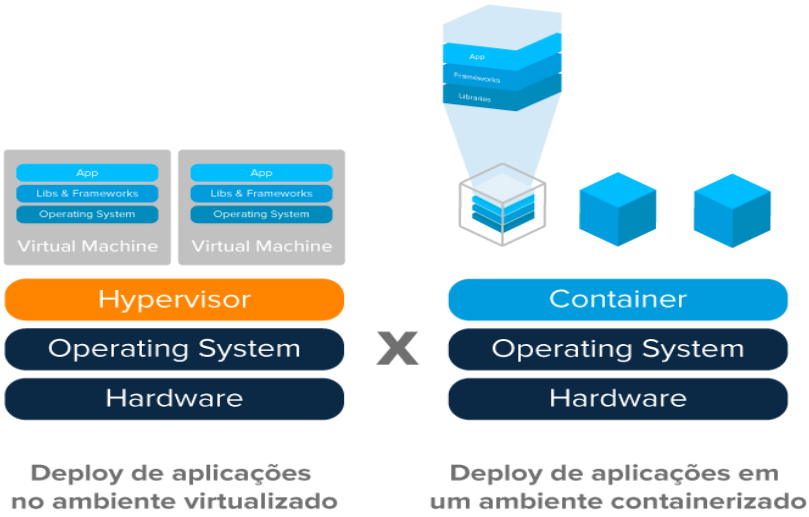
As **novas práticas de TI** (desenvolvimento nativo em nuvem, CI/CD e DevOps) existem graças à divisão das cargas de trabalho nas menores unidades úteis possíveis, que geralmente são uma função ou um microsserviço. Essas unidades são melhor empacotadas em containers. Assim, várias equipes podem trabalhar em partes separadas de uma aplicação ou serviço sem interromper ou pôr em risco o código empacotado em outros containers.

X

Nas **arquiteturas de TI tradicionais** (monolíticas e legadas), todos os elementos de uma carga de trabalho são mantidos em um arquivo grande que não pode ser dividido. Por isso, ele precisa ser empacotado como uma unidade completa em um ambiente maior, frequentemente uma máquina virtual. Era comum criar e executar uma aplicação inteira dentro de uma máquina virtual, mesmo sabendo que ao armazenar todo o código e dependências, ela ficava grande demais, e isso poderia gerar falhas em cascata e downtime durante as atualizações.

VM : As máquinas virtuais são similares aos servidores físicos e agem como eles, o que pode multiplicar as desvantagens de grandes infraestruturas de sistema operacional e das dependências da aplicação. Na maioria das vezes, essas infraestruturas não são necessárias para executar uma aplicação ou microsserviços.

Containers: Os containers armazenam um microsserviço ou aplicação, além de todos os elementos necessários para executá-los. Tudo que eles contêm é mantido em um recurso chamado de imagem: um arquivo baseado em código que inclui todas as bibliotecas e dependências.Pense nesses arquivos como uma instalação da distribuição Linux, já que a imagem inclui pacotes RPM e arquivos de configuração. Como os containers são muito pequenos, geralmente há centenas deles levemente acoplados.



Diferença entre o deploy de aplicações no ambiente virtualizado e o deploy de aplicações em um ambiente containerizado.

O que é Docker?

O Docker é uma plataforma de código aberto que possibilita o empacotamento de uma aplicação dentro de um container. Com isso, ele viabiliza a portabilidade da aplicação para qualquer outro host que tenha o Docker instalado, facilitando a criação e administração de ambientes isolados.

Pense em uma loja de aplicativos de smartphone, como a Play Store ou a App Store. Nela, é possível encontrar a imagem de diversos aplicativos, baixar e começar a utilizá-los instantaneamente. Essa tecnologia funciona de maneira semelhante: no Docker é possível baixar imagens de aplicações conteinerizadas e começar a usá-las rapidamente.

O que é um cluster de container?

Um cluster de container é um agrupamento de contêineres que compartilham os mesmos recursos computacionais, como armazenamento. Um cluster de container permite a execução de centenas de containers de aplicações, de forma eficiente e sem concorrência de recursos.

Porém, criar um cluster de container requer um trabalho de orquestração de containers, ou seja: a organização, gestão e monitoramento dos containers em execução dentro do cluster. Para facilitar esse trabalho, o mercado possui ferramentas de orquestração de clusters de container. O próprio Docker possui o Docker Swarm, porém é mais comum que as empresas utilizem uma ferramenta mais conhecida: Kubernetes.

Docker Swarm:

O Docker Swarm é uma ferramenta nativa do Docker que permite a criação de clusters de Docker, ou seja, podemos fazer com que diversos hosts de Docker estejam dentro do mesmo pool de recursos, facilitando assim o deploy de containers. É possível por exemplo criar um container sem necessariamente saber em qual host ele está, pois o Swarm disponibiliza uma API de integração, onde é possível realizar grande parte das atividades administrativas de um container.

Exemplo: Curso na Dio Innovation One | DOC - Docker Swarm

O que é Kubernetes?

Resumindo: Parecido **(Docker Swarm)**
Kubernetes é um plataforma open source que permite a orquestração de containers distribuídos em clusters. O objetivo é cuidar do ciclo de vida dos contêineres dentro do cluster, distribuindo-os conforme suas especificações ou as demandas da sua operação.

A ferramenta possui diversos comandos que garantem **mais facilidade no gerenciamento das cargas de trabalho**, de forma automatizada e escalável, como:

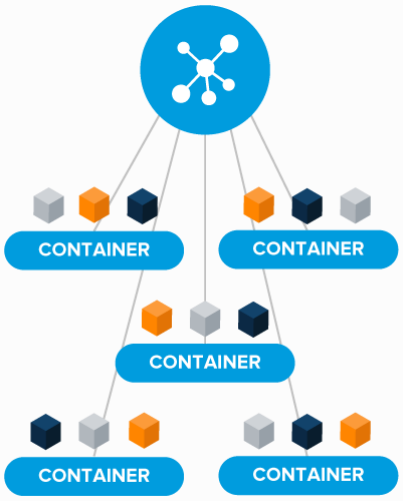
- **kube-apiserver:** front end para o gerenciamento de Kubernetes que permite o escalonamento horizontal de instâncias com balanceamento de carga.
- **etcd:** repositório de apoio do cluster de container que utiliza armazenamento do tipo chave-valor, garantindo consistência e alta disponibilidade.
- **kube-scheduler:** automação do gerenciamento de componentes das aplicações que atribui um conjunto de servidores de processamento para aplicações que foram criadas sem um nó.
- **kube-controller-manager:** controlador automatizado, que cuida de diversos aspectos do cluster de containers, como a identificação de indisponibilidades e garantia da execução de processos.

Esses foram apenas alguns exemplos de serviços da camada de gerenciamento do Kubernetes. Dentro da ferramenta, é possível automatizar e gerenciar o provisionamento e implantação de containers, configurar e alocar recursos, monitorar a disponibilidade de containers, escalar automaticamente, rotear tráfego e remover containers, **de acordo com os parâmetros definidos pela sua operação**.

Então, qual é a diferença entre Kubernetes e Docker?

Docker é uma tecnologia que permite o isolamento de aplicações dentro de um container, portanto, uma ferramenta de conteinerização, capaz de construir, distribuir e rodar aplicações como container. Já Kubernetes é uma ferramenta que permite o gerenciamento de vários containers que compartilham recursos, trabalhando em forma de cluster.

Na prática, **Kubernetes e Docker são tecnologias que trabalham juntas** para criar e rodar aplicações conteinerizadas dentro de um cluster de container. Enquanto Docker viabiliza a criação, isolamento e execução de aplicações distintas, Kubernetes gerencia diversos containers de aplicações dentro do cluster.



Orquestração de um cluster de containers

Demonstração sobre como funciona a orquestração de um cluster de containers através de uma plataforma de orquestração, como Kubernetes.

Loader.io - Teste de carga e escalabilidade.

Loader.io é um serviço SaaS de teste de carga e escalabilidade baseado em nuvem que permite que os desenvolvedores testem seus aplicativos da Web e API com milhares de conexões simultâneas.

Site: https://loader.io/
Aplicando: Estressando um container

NFS (Linux)

NFS, ou Network File System, é um protocolo de sistema distribuído de arquivos que permite a montagem de diretórios remotos no seu servidor. Isso permite que você gerencie o espaço de armazenamento em um local diferente e grave nesse espaço a partir de vários clientes. O NFS fornece uma maneira relativamente padronizada e de bom desempenho para acessar sistemas remotos através de uma rede e funciona bem em situações onde os recursos compartilhados precisam ser acessados regularmente.

- No servidor host, instale o pacote nfs-server:

```
1. sudo apt install nfs-server
```

- No servidor cliente, precisamos instalar um pacote chamado nfs-common:

```
2. sudo apt install nfs-common
```

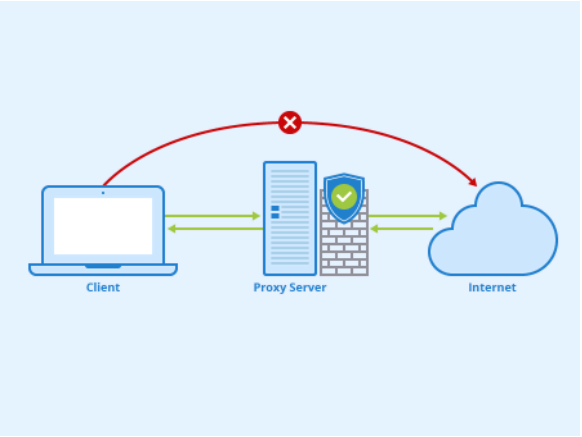
Explicação no Curso:
[replicando o volume dentro do cluster](#) | [tutorials to set up an NFS on ubuntu](#)

Proxy

O **proxy** – ou servidor **proxy** – é uma solução de TI que funciona como um intermediário entre o dispositivo do usuário e os serviços de internet que ele acessa.

Um servidor proxy é como uma ponte entre você e a internet. Quando você tenta acessar um site pelo seu dispositivo, o navegador envia uma requisição ao servidor web deste site, o qual retorna com a página que você estava buscando. Assim, sem um proxy, você se comunica diretamente com a internet e com o servidor em que está hospedado o site que você deseja acessar.

Já quando você utiliza um servidor proxy, sua comunicação passa a ser intermediada por ele.



- É deste modo que um Proxy tem a capacidade de mascarar seu endereço de IP, ocultando essa informação da internet.

- Proteção e segurança são também motivos comuns que levam pessoas a utilizar um servidor proxy. Como ele oculta o endereço de IP e a localização, pode evitar roubos de informações e identidade.

- O administrador da rede pode restringir o acesso a certas páginas ou sites através de um proxy.

Um Proxy é um ótimo local para configurar um **firewall.**

Um **firewall** é um sistema de segurança de rede, que funciona como um filtro entre a internet e o navegador. Ele filtra informações, impedindo que malwares ou vírus infectem os dispositivos utilizados para navegação.

Lista de alguns tipos de servidores Proxy mais populares:

- **Proxy Reverso:**

Esse proxy faz é o **caminho inverso** do que faria um “forward proxy” — aqueles situados entre o computador do usuário e o servidor. Isto é, ele está no intermédio do servidor para com a internet.
Um Proxy Reverso filtra informações geradas por visitantes, encaminhando solicitações filtradas para o servidor. Por este motivo, um proxy reverso é mais comumente utilizado para lidar com requisições de servidores de hospedagem de sites, ou por grandes sites que precisam controlar o consumo de banda larga.

- **Proxy de Distorção:**

Esse tipo de servidor proxy se apresenta ao servidor de destino com um IP falso ou incorreto. Assim, mesmo não ocultando do servidor que um proxy está sendo utilizado, ele protege e esconde sua própria identidade. É normalmente utilizado para acessar sites específicos que só estejam disponíveis em um território, pois assim é possível ocultar e manipular a localização do servidor de origem que está tentando o acesso.

- **Proxy Elite:**

Proxy altamente anônimo ocultando seu IP e também a informação de que você está utilizando um proxy. Também altera o IP que utiliza com certa frequência, dificultando o monitoramento das origens do tráfego. Por isso, é o tipo de Proxy que oferece o maior nível de anonimato ao navegar na internet.

Proxy (Dio Innovation) - Criando um proxy utilizando o NGINX | Tutorial de realizar e conhecer NGINX

Load Balance -> Balanceamento de carga - Todo o hardware tem o seu limite, e muitas vezes o mesmo serviço tem que ser repartido por várias máquinas, sob pena de se tornar congestionado. Estas soluções podem-se especializar em pequenos grupos sobre os quais se faz um balanceamento de carga: utilização do CPU, de armazenamento, ou de rede.

Nginx

É um servidor web que também pode ser usado como proxy reverso e balanceador de carga, proxy de e-mail e servidor de cache.

O Nginx é conhecido por ser um servidor de alta performance, estável, de fácil configuração e baixo consumo de recursos. Escrito em C como vocês podem ver no github, é um servidor amplamente usado em toda a web e foi criado para solucionar o problema do **C10k** — o desafio de gerenciar dez mil conexões ao mesmo tempo.

Instalando no ubuntu:

```
sudo apt update .
sudo apt install nginx .
sudo service nginx start .
```

OBS: Tem no **docker** tbm.

O nginx possui muitas utilidades, mas normalmente é usado como proxy reverso e load balancer

- **NGINX como Proxy reverso:**

O Nginx também pode ser usado como um proxy reverso para suas aplicações, ele pode ser usado como um orquestrador de requisições onde receberia todas as requisições e as entregaria para seus "donos".

Pense que você poderia ter uma ou mais aplicações sendo executadas em portas ou domínios diferentes, você poderia configurar o Nginx para que ao receber uma requisição no domínio app1.meudominio.com ele a redirecionasse para 192.168.1.10:9000 e ao receber uma requisição no domínio app2.dominio.com ele a redirecionasse para 189.55.44.30:8001.

- **NGINX como Balanceador de carga:**

O balanceamento de carga é uma técnica para distribuir a carga de trabalho uniformemente entre dois ou mais computadores. É possível configurar de forma muito fácil um load balancer no Nginx.

- **NGINX como Servidor web:**

Com o Nginx é possível servir arquivos estáticos como imagens, css, javascript e HTML, além de conteúdos dinâmicos usando com manipuladores FastCGI, uwsgi e SCGI.
Por padrão o nginx já serve qualquer arquivo estático, então para um teste simples você pode navegar até o diretório /var/www/html ou /usr/share/nginx/html, a depender da distribuição linux, e altere o arquivo index.html.