A tecnologia de container é um método utilizado na implementação e na execução de aplicativos distribuídos sem que haja a necessidade de configuração de uma máquina virtual para cada um desses aplicativos. O objetivo final é segregar e facilitar a portabilidade dessas aplicações.

Um container Linux é um conjunto de um ou mais processos organizados isoladamente do sistema. Todos os arquivos necessários para executá-los são disponibilizados por uma imagem distinta. Na prática, os containers Linux são portáteis e consistentes durante toda a migração entre os ambientes de desenvolvimento, teste e produção. Essas características os tornam uma opção muito mais rápida de usar do que os pipelines de desenvolvimento, que dependem da replicação dos ambientes de teste tradicionais. Os containers também são uma parte importante da segurança da TI por conta da popularidade e da facilidade de uso deles.

O software de TI "Docker" é uma tecnologia de conteinerização para criação e uso de containers Linux.

A empresa Docker inc se baseia no trabalho realizado pela comunidade do Docker, tornando-o mais seguro, e compartilha os avanços com a comunidade em geral. Depois, ela oferece aos clientes corporativos o suporte necessário para as tecnologias que foram aprimoradas e fortalecidas.

Com o Docker, é possível lidar com os containers como se fossem máquinas virtuais modulares e extremamente leves. Além disso, os containers oferecem maior flexibilidade para você criar, implantar, copiar e migrar um container de um ambiente para outro. Isso otimiza as aplicações na cloud.

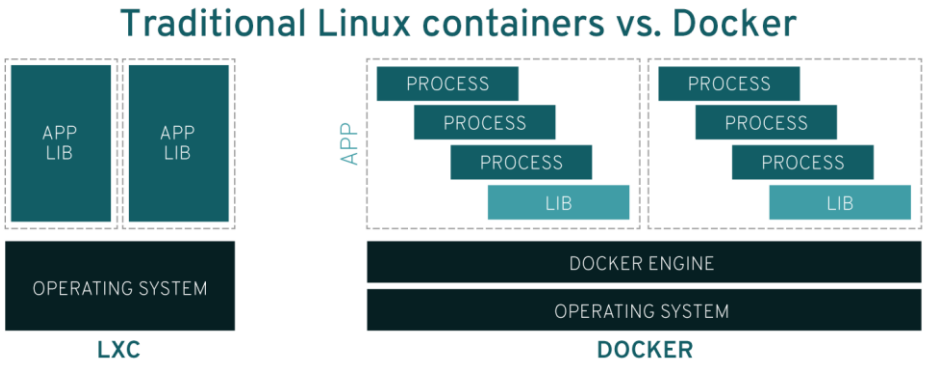
Como docker funciona:

A tecnologia Docker usa o kernel do Linux e recursos do kernel como Cgroups e namespaces para segregar processos. Assim, eles podem ser executados de maneira independente. O objetivo dos containers é criar essa independência: a habilidade de executar diversos processos e aplicações separadamente para utilizar melhor a infraestrutura e, ao mesmo tempo, manter a segurança que você teria em sistemas separados.

As ferramentas de container, incluindo o Docker, fornecem um modelo de implantação com base em imagem. Isso facilita o compartilhamento de uma aplicação ou conjunto de serviços, incluindo todas as dependências deles em vários ambientes. O Docker também automatiza a implantação da aplicação (ou de conjuntos de processos que constituem uma aplicação) dentro desse ambiente de container.

Essas ferramentas baseadas nos containers Linux (o que faz com que o Docker seja exclusivo e fácil de usar) oferecem aos usuários acesso sem precedentes a aplicações, além da habilidade de implantar com rapidez e de ter total controle sobre as versões e distribuição.

Não, a tecnologia Docker foi desenvolvida inicialmente com base na tecnologia LXC, que a maioria das pessoas associa aos containers Linux "tradicionais". No entanto, desde então, essa tecnologia tornou-se independente. O LXC era útil como uma virtualização leve, mas não oferecia uma boa experiência para usuários e desenvolvedores. A tecnologia Docker oferece mais do que a habilidade de executar containers: ela também facilita o processo de criação e construção de containers, o envio e o controle de versão de imagens, dentre outras coisas.



Os containers Linux tradicionais usam um sistema init capaz de gerenciar vários processos. Isso significa que aplicações inteiras são executadas como uma. A tecnologia Docker incentiva que as aplicações sejam segregadas em processos separados e oferece as ferramentas para fazer isso. Essa abordagem granular tem algumas vantagens.

Vantagens

Modularidade
A abordagem do Docker para a containerização se concentra na habilidade de desativar uma parte de uma aplicação, seja para reparo ou atualização, sem interrompê-la totalmente. Além dessa abordagem baseada em microsserviços, é possível compartilhar processos entre várias aplicações da mesma maneira como na arquitetura orientada a serviço (SOA).

Camadas e controle de versão de imagens
Cada arquivo de imagem Docker é composto por uma série de camadas. Elas são combinadas em uma única imagem. Uma nova camada é criada quando há

alteração na imagem. Toda vez que um usuário especifica um comando, como *executar* ou *copiar*, uma nova camada é criada.

O Docker reutiliza essas camadas para a construção de novos containers, o que torna o processo de criação muito mais rápido. As alterações intermediárias são compartilhadas entre imagens, o que melhora ainda mais a velocidade, o tamanho e a eficiência. O controle de versões é inerente ao uso de camadas. Sempre que é realizada uma nova alteração, é gerado um changelog integrado, o que fornece controle total sobre as imagens do container.

Reversão
Talvez a melhor vantagem da criação de camadas seja a habilidade de reverter quando necessário. Toda imagem possui camadas. Não gostou da iteração atual de uma imagem? Simples, basta reverter para a versão anterior. Esse processo é compatível com uma abordagem de desenvolvimento ágil e possibilita as práticas de integração e implantação contínuas (CI/CD) em relação às ferramentas.

Implantação rápida
Antigamente, colocar novo hardware em funcionamento, provisionado e disponível, levava dias. E as despesas e esforço necessários para mantê-lo eram onerosos. Os containers baseados em docker podem reduzir o tempo de implantação de horas para segundos. Ao criar um container para cada processo, é possível compartilhar rapidamente esses processos similares com novos aplicativos. Como não é necessário inicializar um sistema operacional para adicionar ou mover um container, o tempo de implantação é substancialmente menor. Além disso, com a velocidade de implantação, é possível criar dados e destruir os criados pelos containers sem nenhuma preocupação e com facilidade e economia.

Em resumo, a tecnologia Docker é uma abordagem mais granular, controlável e baseada em microsserviços que valoriza a eficiência.

Há limitações no uso do docker?
Por si só, o Docker é excelente para gerenciar containers únicos. No entanto, quando você começa a usar cada vez mais containers e aplicações em containers segregados em centenas de partes, o gerenciamento e a orquestração podem se tornar um grande desafio. Eventualmente, será necessário recuar e agrupar os containers para oferecer serviços como rede, segurança, telemetria etc. em todos eles. É aí que o Kubernetes entra em cena.

O Docker não fornece as mesmas funcionalidades parecidas com UNIX que os containers Linux tradicionais oferecem. Isso inclui a capacidade de usar processos como cron ou syslog dentro do container, junto à aplicação. O Docker também tem

algumas limitações em questões como a limpeza de processos netos (grandchild) após o encerramento dos processos filhos (child), algo que é processado de forma natural nos containers Linux tradicionais. Essas desvantagens podem ser mitigadas ao modificar o arquivo de configuração e configurar essas funcionalidades desde o início, algo que não está imediatamente óbvio em um primeiro momento.

Além disso, há outros subsistemas e dispositivos do Linux sem espaço de nomes. Incluindo os dispositivos SELinux, Cgroups e /dev/sd*. Isso significa que, se um invasor adquirir controle sobre esses subsistemas, o host será comprometido. Para manter-se leve, o compartilhamento do kernel do host com os containers gera a possibilidade dessa vulnerabilidade na segurança. Isso é diferente nas máquinas virtuais, que são mais firmemente segregadas a partir do sistema host.

O daemon do Docker também pode representar uma vulnerabilidade à segurança. Para usar e executar os containers Docker, é provável que você use o daemon do Docker, um ambiente de execução persistente para containers. O daemon do Docker requer privilégios de raiz. Portanto, é necessário ter um cuidado maior ao escolher as pessoas que terão acesso a esse processo e o local onde ele residirá. Por exemplo, um daemon local tem menos chances de sofrer um ataque do que um daemon em um local mais público, como um servidor web.

Kubernetes, ou “k8s”, é uma plataforma open source que automatiza as operações dos containers Linux. O Kubernetes elimina grande parte dos processos manuais necessários para implantar e escalar as aplicações em containers. Em outras palavras, se você desejar agrupar em clusters os hosts executados nos containers Linux, o Kubernetes ajudará a gerenciar esses clusters com facilidade e eficiência. Esses clusters podem incluir hosts em nuvem pública, nuvem privada ou nuvem híbrida. Por isso, o Kubernetes é a plataforma ideal para hospedar aplicações nativas em nuvem que exigem escalabilidade rápida, como a transmissão de dados em tempo real por meio do Apache Kafka.

Originalmente, o Kubernetes foi criado e desenvolvido por engenheiros do Google. O Google foi um dos pioneiros no desenvolvimento da tecnologia de containers Linux. Além disso, a empresa já revelou publicamente que tudo no Google é executado em containers (inclusive, essa é a tecnologia por trás dos serviços em cloud da empresa). O Google gera mais de 2 bilhões de implantações de containers por semana, viabilizadas por uma plataforma interna: Borg . O Borg foi o antecessor do Kubernetes. As lições aprendidas ao longo dos anos de desenvolvimento do Borg foram a principal influência para o desenvolvimento da tecnologia do Kubernetes.

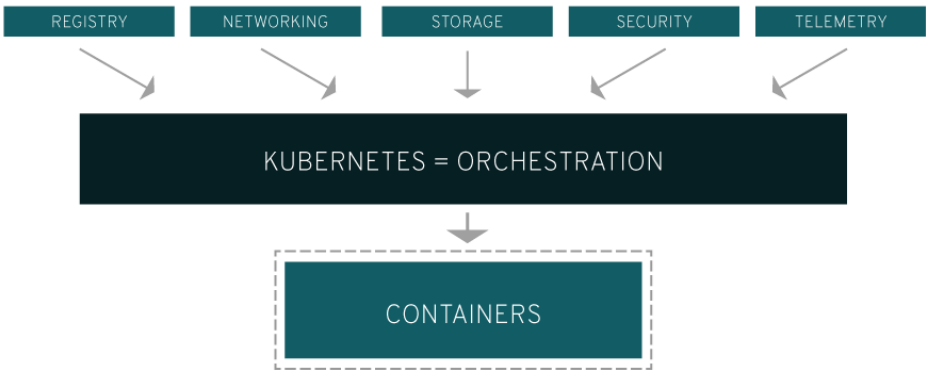
Uma curiosidade sobre o Kubernetes é que os sete raios do logotipo fazem referência ao nome original do projeto, "Project Seven of Nine" (Projeto Sete de Nove).

A Red Hat foi uma das primeiras empresas a trabalhar com o Google no desenvolvimento do Kubernetes, antes mesmo do lançamento da plataforma. Foi assim que nos tornamos o segundo maior colaborador com o projeto upstream dessa tecnologia. Em 2015, o Google doou o projeto Kubernetes à Cloud Native Computing Foundation, recém-formada na época.

Kubernetes: entenda por que ele é essencial

Aplicações de produção abrangem múltiplos containers. Eles devem ser implantados em vários hosts do servidor. A segurança dos containers tem várias camadas e pode ser complexa. É aí que o Kubernetes entra em cena. Ele oferece os recursos de orquestração e gerenciamento necessários para implantar containers em escala para essas cargas de trabalho. Com a orquestração do Kubernetes, é possível criar serviços de aplicações que abrangem múltiplos containers, programar o uso deles no cluster, escalá-los e gerenciar a integridade deles com o passar do tempo. Com o Kubernetes, você toma medidas reais para aprimorar a segurança da TI.

Também é necessário integrar o Kubernetes com os serviços de rede, armazenamento, segurança, telemetria e outros para oferecer uma infraestrutura de containers global.



No entanto, isso obviamente depende do uso que cada empresa faz dos containers em seus próprios ambientes. Uma aplicação rudimentar dos

containers Linux os trata como máquinas virtuais rápidas e eficientes. Quando escalado para um ambiente de produção e diversas aplicações, fica claro que é necessário ter vários containers alocados funcionando em conjunto para disponibilizar serviços individuais. Isso multiplica substancialmente o número de containers no ambiente. À medida que eles se acumulam, a complexidade também aumenta.

O Kubernetes corrige vários problemas comuns que ocorrem com a proliferação de containers, organizando-os em "pods". Os pods adicionam uma camada de abstração aos containers agrupados. Assim, é mais fácil programar as cargas de trabalho e fornecer os serviços necessários a esses containers, como rede e armazenamento. Outros componentes do Kubernetes são úteis no balanceamento de carga entre os pods. Com isso, é possível garantir que o número de containers em execução seja suficiente para oferecer suporte às cargas de trabalho.

Com a implementação correta do Kubernetes (e a ajuda de outros projetos open source, como Open vSwitch, OAuth e SELinux), as empresas podem orquestrar todas as partes da infraestrutura de containers.

A principal vantagem que as empresas garantem ao usar o Kubernetes, especialmente se estiverem otimizando o desenvolvimento de aplicações para a cloud, é que elas terão uma plataforma para programar e executar containers em clusters de máquinas físicas ou virtuais. Em termos mais abrangentes, com o Kubernetes, é mais fácil implementar e confiar totalmente em uma infraestrutura baseada em containers para os ambientes de produção. Como o propósito do Kubernetes é automatizar completamente as tarefas operacionais, ele permite que os containers realizem muitas das tarefas possibilitadas por outros sistemas de gerenciamento ou plataformas de aplicações.

O Kubernetes possibilita:

- Orquestrar containers em vários hosts.
- Aproveitar melhor o hardware para maximizar os recursos necessários na execução das aplicações corporativas.
- Controlar e automatizar as implantações e atualizações de aplicações.
- Montar e adicionar armazenamento para executar aplicações com monitoração de estado.
- Escalar rapidamente as aplicações em containers e recursos relacionados.

- Gerenciar serviços de forma declarativa, garantindo que as aplicações sejam executadas sempre da mesma maneira como foram implantadas.
- Verificar a integridade e autor recuperação das aplicações com posicionamento, reinício, replicação e escalonamento automáticos.

No entanto, o Kubernetes depende de outros projetos para oferecer plenamente esses serviços orquestrados. Com a inclusão de outros projetos open source, é possível atingir a capacidade total do Kubernetes. Dentre esses projetos necessários, incluem-se:

Registro, como o Atomic Registry ou o Docker Registry.

Rede, como o OpenvSwitch e roteamento de borda inteligente.

Telemetria, como o heapster, o kibana, o hawkular e o elastic.

Segurança, como o LDAP, o SELinux, o RBAC e o OAUTH com camadas de multilocação.

Automação, com a adição de playbooks do Ansible para a instalação e o gerenciamento do ciclo de vida do cluster.

Serviços, oferecidos em um catálogo variado de conteúdos previamente criados de padrões de aplicações populares.

Assim como qualquer tecnologia, há vários termos específicos que podem representar uma barreira inicial. Vamos explicar alguns dos termos mais comuns para ajudar você a entender melhor o Kubernetes.

Master: a máquina que controla os nós do Kubernetes. É nela que todas as atribuições de tarefas se originam.

Nó: são máquinas que realizam as tarefas solicitadas e atribuídas. A máquina mestre do Kubernetes controla os nós.

Pod: um grupo de um ou mais containers implantados em um único nó. Todos os containers em um pod compartilham o mesmo endereço IP, IPC, nome do host e outros recursos. Os pods separam a rede e o armazenamento do container subjacente. Isso facilita a movimentação dos containers pelo cluster.

Controlador de replicações: controla quantas cópias idênticas de um pod devem ser executadas em um determinado local do cluster.

Serviço: desacopla as definições de trabalho dos pods. Os proxies de serviço do Kubernetes automaticamente levam as solicitações de serviço

para o pod correto, independentemente do local do pod no cluster ou se foi substituído.

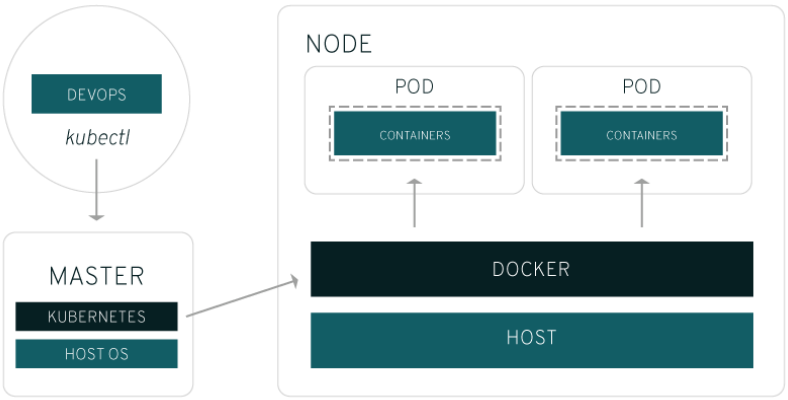
Kubelet: um serviço executado nos nós que lê os manifestos do container e garante que os containers definidos foram iniciados e estão em execução.

kubectl: a ferramenta de configuração da linha de comando do Kubernetes.

O Kubernetes é uma tecnologia open source. Por isso, ele não conta com uma estrutura de suporte formal em que as empresas podem confiar totalmente. Problemas com a implantação do Kubernetes durante a execução no ambiente de produção podem representar uma grande dor de cabeça para você e os seus clientes.

Para isso, existe o Red Hat OpenShift. Essa é uma solução de nível corporativo que oferece o Kubernetes e muito mais. O OpenShift vem com todos os elementos extras que tornam o Kubernetes potente e viável para as empresas, incluindo componentes de registro, rede, telemetria, segurança, automação e serviços. Com o Red Hat OpenShift, os desenvolvedores da sua empresa poderão criar novas aplicações em containers, hospedá-las e implantá-las na cloud. Tudo isso com a escalabilidade, o controle e a orquestração necessários para transformar boas ideias em negócios valiosos de forma rápida e fácil.

Além disso, a maior vantagem dessa solução é que essa plataforma foi desenvolvida e conta com o suporte da Red Hat, a empresa líder global em tecnologia open source.



O Kubernetes é executado em um sistema operacional (por exemplo, no Red Hat Enterprise Linux Container Host) e interage com pods de containers executados em nós. A máquina mestre do Kubernetes aceita os comandos de um administrador (ou equipe de DevOps) e retransmite essas instruções aos nós subservientes. Essa retransmissão é realizada em conjunto com vários serviços para automaticamente decidir qual nó é o mais adequado para a tarefa. Depois, são alocados os recursos e atribuídos os pods do nó para cumprir a tarefa solicitada.

Portanto, do ponto de vista da infraestrutura, são poucas as mudanças em comparação com a forma como você já gerenciam os containers. O controle sobre os containers acontece em um nível superior, tornando-o mais refinado, sem a necessidade de micro gerenciar cada container ou nó separadamente. Será necessário realizar algum trabalho, mas em sua maioria trata-se somente de uma questão de atribuir um master do Kubernetes e definir os nós e pods.

Kubernetes x Docker

A tecnologia do docker ainda realiza as mesmas tarefas do seu objetivo original. Quando o Kubernetes programa um pod para um nó, o kubelet no nó instruirá o docker a iniciar os containers especificados. O kubelet, então, continuamente coleta do docker os status desses containers e agrega as informações no master. O docker insere os containers nesse nó e os inicia e interrompe normalmente. A diferença é que um sistema automatizado solicita que o Docker realize essas tarefas em todos os nós de todos os containers, em vez do administrador fazer essas solicitações manualmente.