Container Docker

A tecnologia está em constante evolução e a todo momento surgem novas ideias que aprimoram os métodos utilizados em TI. O container docker é uma delas e pode trazer vários benefícios para a sua organização.

As empresas optam por essa modalidade porque os containers trazem diferentes possibilidades. Por serem ambientes isolados e portáveis, os desenvolvedores podem empacotar aplicações com bibliotecas e links necessários. O resultado é mais eficiência no trabalho e a simplificação da implantação.

Porém, essa descrição ainda não diz todos os benefícios do uso dessa tecnologia. Por isso, criamos este post. Aqui, vamos explicar os principais detalhes sobre o docker, passando pelos seguintes aspectos:

conceito;

diferenças do sistema virtualização dessa forma;

por que se tornou uma tendência do mercado;

benefícios do uso.

Então, que tal entender melhor essa tecnologia? É só continuar lendo!

O que é um container docker?

Esse conceito precisa ser compreendido em partes. O container é um ambiente isolado. Já o docker é uma plataforma open source na linguagem de programação Go, que possui alto desempenho e é desenvolvida diretamente no Google.

Assim, o docker agrupa partes de softwares de um sistema de arquivo completo e que abrange todos os recursos necessários para a sua execução. Por isso, é uma plataforma de containers.

Isso significa que tudo pode ser instalado no servidor e é armazenado nos containers. Dessa forma, os mesmos softwares e as suas versões podem ter uma execução facilitada em qualquer ambiente de desenvolvimento.

O que ocorre na prática é que o docker destaca recursos e usa bibliotecas de kernel em comum. Os itens empacotados — ou até mesmo um ambiente inteiro — são dispostos no container e se tornam portáveis, o que torna o trabalho conjunto mais eficiente. Ao mesmo tempo, a implantação pode ser feita em ambientes não heterogêneos.

Assim, o docker é uma implementação de virtualização de containers que vem conquistando cada vez mais espaço devido à computação em nuvem. Anteriormente, o hypervisor dominava o ambiente, mas agora ele é excluído do processo.

Devido a todas as facilidades que oferece, o container é uma das principais tendências de TI. Ele simplifica a aplicação da metodologia DevOps e facilita o desenvolvimento ágil, tanto que o Google utiliza essa tecnologia há mais de 10 anos.

Já o docker, por ser open source, possibilita a execução de deploys e o escalonamento de aplicações com mais facilidade. Além disso, devido à virtualização por container, propicia um ambiente isolado e leve para rodar o programa.

Quais as diferenças do sistema virtualizado com docker?

O container exclui a virtualização pelo hypervisor e muda o processo para o docker. Essa é a principal modificação. No entanto, há vários detalhes intrincados que precisam ser compreendidos.

Para facilitar, vamos apresentar o modelo tradicional de virtualização e o novo, com o container:

Sistema virtualizado por VM

O modelo mais comum conta com uma máquina virtual (VM, do inglês virtual machine), que trabalha com um sistema operacional (SO) completo, porém separado do equipamento. A execução do software é feita em cima de um servidor físico com a finalidade de emular determinado sistema de hardware.

Esse processo é possibilitado pelo hypervisor, software que cria e efetiva a VM. Basicamente, ele se localiza entre o hardware e o SO, sendo um elemento fundamental para a virtualização do servidor. Um exemplo de hypervisor é o VMWare, Hyper-V e o VirtualBox.

Uma mesma máquina pode contar com diferentes VMs. Cada uma opera um SO exclusivo e possui seu próprio kernel, binários, aplicativos e bibliotecas, o que significa que ocupam um espaço grande no servidor.

Quando foi criada, essa tecnologia trouxe diversos benefícios devido à capacidade de consolidação de aplicativos em um sistema único. Com isso, é possível alcançar a redução de custos pelo rápido provisionamento e obter o aprimoramento da possibilidade de recuperação de desastres.

Duas áreas muito fortalecidas e beneficiadas com essa medida foi a de QA e development, porque se tornou possível testar os programas e usar os servidores liberados para montar esses ambientes.

Assim, apesar de haver vários benefícios com a virtualização por VM, ainda era necessário criar uma alternativa mais simples e leve.

Sistema Virtualizado Com Docker

Esse é um sistema desenvolvido na linguagem de programação Go. Com ele, os desenvolvedores conseguem criar e administrar diferentes ambientes isolados, fazendo com que os dockers sejam um sistema de virtualização diferente do tradicional.

Isso acontece porque esse novo sistema de virtualização usa o Linux Container (LXC) como backend. Ele não fornece uma VM, mas sim um ambiente virtual semelhante ao chroot, mas com um isolamento maior. Essa característica permite definir limitações de recursos por container, por exemplo, CPU, memória, I/O, entre outros.

Na prática, o SO convidado e o hypervisor são eliminados e o host entra em contato direto com as bibliotecas. Com essa ligação, os itens ficam portáveis para qualquer outro host que também possua o sistema de virtualização instalado. A consequência é a redução do tempo de deploy de uma aplicação ou infraestrutura.

Assim, é desnecessário ajustar o ambiente para que o serviço funcione corretamente. Ou seja, ele é sempre igual e é configurado uma vez. A partir disso, basta replicá-lo.

Outras possibilidades interessantes são criar os containers prontos para deploy (ou seja, imagens) a partir de dockerfiles, que são arquivos de definição.

É importante destacar que os containers se localizam em cima de um servidor físico e do SO hospedeiro. Cada um deles compartilha o kernel do SO host e costuma partilhar também as bibliotecas e binários.

A diferença é que os itens compartilhados servem somente para leitura, o que torna o container muito leve, especialmente se comparado à VM. Para ter uma ideia, o tamanho do primeiro é de Mb e, por isso, ele é iniciado em poucos segundos. Já a máquina virtual contém vários Gb e demora minutos para ser executada.

Perceba que esse novo modelo de virtualização prevê o fornecimento do básico para que uma aplicação seja executada em um SO hospedeiro.

Assim, escolher entre uma VM e um docker depende do contexto do projeto que se pretende realizar. A principal diferença é relativa ao consumo de recursos e espaço, porque a modalidade de virtualização mais recente possui diferentes camadas, que são reunidas com o UnionFS.

Essa característica traz mais agilidade ao processo, porque o rebuild é desnecessário para o update das imagens. De modo geral, os containers são mais interessantes devido a alguns fatores, como:

velocidade;

rapidez no boot;

economia de recursos;

entendimento dos processos do container como sendo realizados dentro do sistema host;

possibilidade de fazer o upload de vários containers simultaneamente, o que consome menos recursos do hardware virtual ou físico.

Assim, o container docker leva a um nível superior de virtualização, o que traz diferentes vantagens para a organização.

Como o docker funciona?

As médias e grandes empresas, principalmente, utilizam aplicações como ERPs e CRMs, ou seja, conjuntos de software que iniciam como projetos simples, mas que se tornam ineficientes com o tempo e impedem o progresso por contarem com um código-fonte monolítico.

O container surgiu para resolver esse problema. A solução passa por diferentes etapas. A primeira delas é a desagregação do aplicativo em componentes menores, os chamados microsserviços. A partir disso, os desenvolvedores conseguem adotar uma arquitetura que aumenta a eficiência operacional. Isso acontece porque o código-fonte é destinada para cada componente de aplicação. Assim, o software passa por vários estágios, por exemplo, vai para um ambiente de testes, para um virtual e, por último, de produção.

Em cada um desses locais, a aplicação deve ter uma performance consistente, que é garantida pelo container. Ele encapsula os componentes em um pacote único e leve, que permite executar os aplicativos com consistência independentemente do ambiente ser virtual ou físico.

Já o docker faz a comunicação entre cliente e servidor por meio de um API. Para realizar esse workflow, é necessário ter o serviço instalado em um local e apontar o cliente para esse servidor.

A plataforma em si usa alguns conjuntos de recursos para criar e administrar os containers, inclusive limitar os recursos. Dentre eles está a biblioteca libcontainer, que efetua a comunicação entre o Docker Daemon e o backend.

Vale a pena destacar que o docker está embasado no conceito de layers. O container é construído por meio de chroot, namespaces, cgroups e outras funcionalidades do kernel a fim de isolar a área para a sua aplicação.

Desse modo, no docker, o kernel monta o rootfs na modalidade somente leitura. Logo em seguida, um arquivo do sistema é criado como read-write sobre o rootf. Depois, o kernel sobrepõe a partição raíz como read-only e pega um novo file system para colocá-lo sobre o rootfs.

Observe que o container está pronto para executar a imagem (layer somente leitura) após o carregamento do rootfs. Ele também é uma camada read-write criada a partir de uma imagem somente leitura.

Por isso, ele é uma ferramenta excelente para DevOps. No caso dos administradores de sistemas há o benefício da flexibilidade, custos menores e menos áreas ocupadas. Para os desenvolvedores, é mais liberdade para que esses profissionais foquem a atividade principal.

Por que ele vem se tornando uma tendência no mercado?

O docker tem se popularizado devido a diferentes facilidades proporcionadas pela plataforma. Em comparação com a VM, a estrutura é menor, porque o hypervisor é substituído pelo docker engine e o SO hospedeiro é excluído.

Ambos os modelos permitem o isolamento de recursos e incluem as bibliotecas, aplicação e arquivos necessários. A diferença é que a VM precisa do SO e, com isso, exige espaço e possui custo de manutenção.

Essa especificação evidencia que o docker tem uma estrutura mais portátil e leve. É mais fácil mantê-la, porque os containers compartilham o mesmo SO da hospedagem. Apenas os processos são executados isoladamente e, portanto, o custo e a necessidade de espaço são menores.

A portabilidade é outra vantagem do docker, o que traz ainda mais benefícios para os ambientes de development. Assim, uma aplicação é mais facilmente executada no ambiente de homologação e/ou de produção.

Além disso, há uma garantia maior de que não haverá erros ou imprevistos durante o deploy. Por fim, gasta-se menos tempo para configurar os ambientes e se perde menos tempo analisando e identificando as diferenças existentes. O resultado é a integração das equipes de desenvolvimento e SysAdmin, que aumentam a produtividade.

Para entender melhor a importância, basta observar um exemplo prático. Imagine que a sua empresa realiza um projeto em WordPress com o PHP versão 5.*, mas deseja atualizar a linguagem para a sétima.

Esse processo de update da versão se torna mais simples com o docker. Você pode realizar testes para definir a versão que permanecerá sendo utilizada e assinalar os ajustes necessários.

Devido a todas essas peculiaridades, centenas de empresas utilizam o docker em todo o mundo. Alguns exemplos são Uber, The Washington Post, PayPal, General Electric, eBay, Spotify etc. Um dos motivos para isso é o fato de diversas soluções de hospedagem adotaram a tecnologia.

Outra justificativa é que, no começo, ele pode representar uma demanda muito grande de recursos, mas a recompensa são as vantagens em agilidade, já que a velocidade de implantação de containers vai de milissegundos a alguns segundos.

Quais são as vantagens de utilizar?

O container docker é uma tecnologia que apresenta diferentes benefícios às empresas. As principais são:

Economia Significativa de Recursos

Essa questão pode ser explicada pela contextualização da diferença existente entre imagens e containers. As primeiras são um ambiente read-only, enquanto os segundos são definidos como uma imagem em execução, na qual é gerada um layer extra que armazena os dados relativos à determinada operação.

É possível configurar a dependência de uma imagem em relação a outra. Nessa situação é criada uma pilha de imagens, sendo que cada uma delas é somente leitura. Assim, pode-se montar um amontoado para diferentes containers, o que ocasiona uma economia de recursos sobre o disco, que é compartilhado entre os ambientes.

Ainda é possível exemplificar essa situação por um ambiente composto por uma imagem Apache, Debian e módulo PHP. Se forem necessários 20 containers da pilha de imagens, não é necessário utilizar o recurso multiplicado por 20.

Isso porque a utilização do espaço em disco se refere somente aos logs e arquivos temporários de cada container. Desse modo, os dados são armazenados no layer extraindividual.

Disponibilidade Maior do Sistema

Esse benefício é ocasionado pelo fato de o docker virtualizar o sistema de maneira diferente ao método da VM. Por compartilhar o SO e outros componentes, há mais espaço livre, o que deixa os processos mais ágeis e oferece uma disponibilidade maior.

Em termos bastante simples, a máquina fica menos “pesada” e fica com mais espaço para rodar outros programas e aplicações.

Possibilidade de Compartilhamento

Os arquivos podem ser compartilhados entre o host e o container ou até mesmo um volume tem a possibilidade de ser distribuído para outros. A prática da segunda situação é mais indicada nos casos em que se deseja ter persistência de dados e quando não se atrela ao host que hospeda o container.

Outro aspecto que precisa ser considerado é que o container está em um nível de virtualização operacional, ou seja, é um processo em execução em um kernel compartilhado entre outros containers.

Por meio do namespace é feito o isolamento da memória RAM, disco, processamento e acesso à rede. Ou seja, em um contexto isolado há o compartilhamento do kernel, mas a impressão é de que o SO é dedicado.

Por fim, o compartilhamento pode ocorrer pela cloud. Essa tecnologia disponibiliza um repositório de imagens e ambientes prontos, que podem ser compartilhados. Com a utilização desse serviço, o docker chega a extrapolar o limite técnico e passa para questões de gerência, processo e update do ambiente.

Com isso, torna-se mais fácil compartilhar as modificações e oferecer uma gestão centralizada em relação às definições de ambiente. O espaço para testes também se torna mais leve, o que permite baixar uma solução durante uma reunião, por exemplo, ou fornecer um padrão de melhores práticas para todos os colaboradores.

Gerenciamento Facilitado

Os containers são executados em máquinas físicas ou virtuais e o grupo delas é chamado de cluster. Esse item precisa ser monitorado constantemente. Por isso, foram criadas ferramentas para fazer o gerenciamento, por exemplo, o Kubernetes e o OpenShift.

Esses sistemas trabalham em conjunto com o docker e operam o equipamento que possibilita a execução dos containers. No entanto, os sistemas de arquivos também são gerenciados.

Essas ferramentas de monitoramento geram uma abstração, denominada pod, no nível de um componente da aplicação. Essa questão inclui um grupo de um ou mais containers, armazenamento compartilhado e alternativas de operação.

Similaridade Dos Ambientes

A transformação da aplicação em uma imagem docker permite que ela seja instanciada como container em diferentes ambientes. Essa característica garante a sua utilização, por exemplo, tanto no notebook do desenvolvedor quando no servidor de produção.

Tenha em mente que a imagem aceita parâmetros na iniciação do container, situação que indica diferentes comportamentos conforme o ambiente. Por exemplo: ele pode se conectar ao banco de dados local para testes a partir da base de dados e credenciais, mas em produção acessará um database com infraestrutura robusta, que possui as suas próprias credenciais.

É importante destacar que ambientes semelhantes impactam a análise de erros e a confiabilidade do processo de entrega contínua de forma positiva. No segundo caso, a base é a criação de um artefato único que faz a migração, que pode ser a própria imagem do docker com as dependências necessárias para a execução do código dinâmico ou compilado.

Aplicação Como Pacote Completo

As imagens do docker possibilitam o empacotamento da aplicação e as suas dependências, o que simplifica o processo de distribuição por não ser exigida ampla documentação sobre a configuração da infraestrutura com a finalidade de execução. Para isso, basta disponibilizar o repositório e permitir o acesso para o usuário.

A partir disso é possível fazer o download do pacote, que pode ser executado facilmente. O update também sofre impacto positivo, porque a estrutura de layers permite que, em caso de modificação, somente a alteração seja transferida.

Essa medida faz com que o ambiente possa ser mudado de maneira simples e rápida. Com apenas um comando existe a capacidade de fazer o update da imagem da aplicação, que é refletida no container em execução quando for desejado.

As imagens podem ser categorizadas com tags, o que facilita o armazenamento de diferentes versões de uma aplicação. Se houver algum problema na atualização, é possível retornar para a imagem com a tag anterior.

Padronização e Replicação

As imagens do docker são construídas por meio de arquivos de definição, o que assegura o seguimento de um determinado padrão e eleva a confiança na replicação. Dessa maneira, fica muito mais viável escalar a estrutura.

Com a chegada de um novo membro para a equipe de TI, por exemplo, ele pode se integrar e receber o ambiente de trabalho rapidamente, com apenas alguns comandos. Ele ainda pode desenvolver os códigos de acordo com o padrão adotado para a equipe.

Por sua vez, na necessidade de testar uma versão com as imagens, é possível mudar um ou mais parâmetros de um arquivo de definição. Ou seja, é mais simples criar e mudar a infraestrutura.

Possibilidade de Acessar Comunidade

O repositório de imagens docker pode ser acessado para conseguir modelos de infraestrutura de aplicações e serviços prontos para integrações complexas. Dois exemplos são o mysql como banco de dados e o nginx como proxy reverso.

No caso de a aplicação exigir esses recursos, você pode apenas usar as imagens do repositório e configurar os parâmetros para adequação ao seu ambiente. Essa é uma vantagem principalmente porque as imagens oficiais costumam ser condizentes com as boas práticas.

Como você pôde perceber, a tecnologia do docker é um pouco complexa, mas vale a pena apostar nela. A sua empresa consegue economizar recursos, configurar parâmetros e virtualizar em um nível muito mais alto que o permitido pelas VMs atualmente.

Docker se refere a muitas coisas. Isso inclui: um projeto da comunidade open source; as ferramentas resultantes desse projeto; a empresa Docker Inc., principal apoiadora do projeto; e as ferramentas compatíveis formalmente com a empresa. O fato de que as tecnologias e a empresa têm o mesmo nome pode causar uma certa confusão.

Veja uma simples explicação:

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

A comunidade open source do Docker trabalha gratuitamente para melhorar essas tecnologias para todos os usuários.

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.

Certo. Mas o que são containers Linux?

Como o Docker funciona?

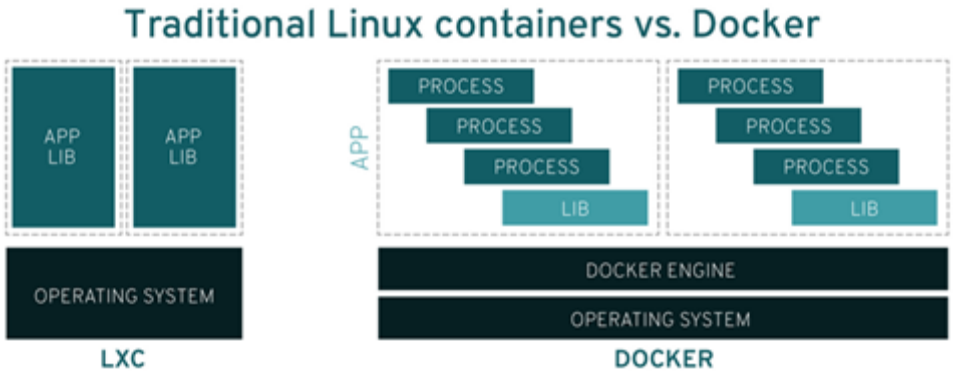
A tecnologia Docker usa o kernel do Linux e recursos do kernel como Cgroups e namespaces para segregar processos. Assim, eles podem ser executados de maneira independente. O objetivo dos containers é criar essa independência: a habilidade de executar diversos processos e aplicações separadamente para 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.

O Docker utiliza a mesma tecnologia que os containers Linux tradicionais?

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.

As Vantagens dos Containers Docker

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 funcionalidade 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.

Docker é um projeto OpenSource que fornece uma plataforma para desenvolvedores e administradores de sistemas permitindo que se crie containers leves e protrátil de diversas aplicações.

Sua funcionalidade permite adicionar e simplificar o uso, dos linux containers (LXC), que são, basicamente, uma forma de isolamento de processo e sistemas, quase como virtualização, porém mais leve e integrada ao host. O Docker permite criar aplicações e “containers” que isolam o S.O base e todo a pilha de dependências de seu app (libs, servidores e etc) de forma leve em espaço e performance.

O Docker também trabalha com um sistema de arquivos “empilháveis”, o aufs, que permite que os passos para configuração do seu container funcionem de forma incremental e “cacheable”, mais ou menos como “commits” do GIT.

Em poucas palavras, o Docker oferece a você um conjunto completo de ferramentas de alto nível para transportar tudo que constitui uma aplicação entre sistemas e máquinas – virtual ou física – e trás consigo grandes benefícios agregados.

O que é um Container?

O Contanier nada mais é que um chroot. Nele é possível definir recursos como memória, rede, sistema operacional, aplicação, serviço e etc. Em um Container Docker é possível fazer testes, desenvolvimentos, estudos, etc. Além disso, também é possível utilizá-lo em um ambiente de produção.

Qual é sua base?

O Docker tem como base o LXC Linux Contaniner.

LinuX Containers (LXC) é um método de virtualização em nível de sistema operacional para executar vários sistemas Linux isolados (contêineres) em um único host de controle (LXC host).

LXC não fornece uma máquina virtual, mas fornece um ambiente virtual que tem sua própria CPU, memória, bloco I/O, rede, espaço, entre outros. Este é fornecido por cgroups recursos no kernel do Linux no host LXC. É semelhante a um chroot, mas oferece muito mais isolamento.

Linux Containers (LXC).
Permite rodar um Linux dentro de outro Linux.
Chroot on Steroids.
Dentro do container, parece uma VM.
Não é Para-Virtualização.
Fora do Containers o Docker não passa de mais um processo para o sistema operacional.

Porque usar Containers?

Velocidade;

Boot em questão de segundos;

Economia de recursos;

Os processos rodando dentro de um container são vistos como um processo no sistema Host;

É possível subir vários containers ao mesmo tempo, consumindo o minimo de recursos do hardware físico ou virtual.

Diferenças entre o Sistema Operacional comum/virtualizado com o Container Docker.

Sistema Operacional

Bootfs – Boot FileSystem;

Bottloader e Kernel;

Rootfs – Root Filesystem;

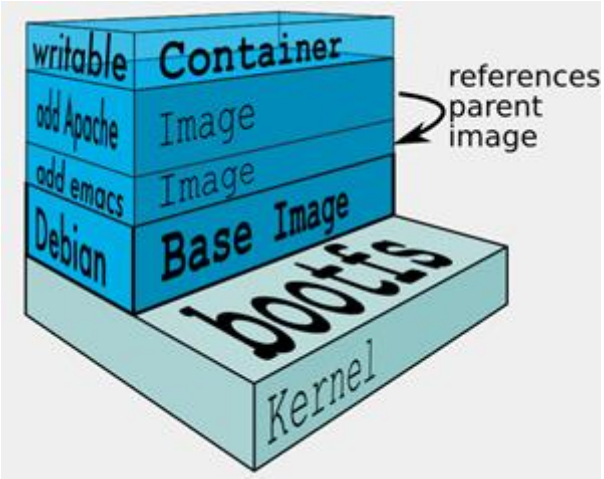
Restante dos arquivos do sistema.

Container Docker

Bootfs – (LXC, aufs/btrfs) – Mesmo Kernel da máquina.

Rootfs – Sistema de arquivos.

Com o carregamento do rootfs, o Container Docker ficará pronto para execução.
O Docker está totalmente baseado no conceito de Layers.



Em um boot tradicional do Linux:

O Kernel monta o rootfs como read-only, checa sua integridade e faz a montagem como read-write.

No Docker:

O Kernel monta o rootfs como read-only, depois outro filesystem é montado como read-write em cima do rootf

Unio File System. (O Kernel monta a partição raiz como somente leitura, só que ao invés de remontar a partição como read-write, ele se encarrega de pegar um outro FileSystem e em seguida o coloca acima do primeiro rootfs.

Observação:

No Docker após o carregando o rootfs o container estará pronto para executar a imagem;

Image – Layer somente leitura (Imagem base de um sistema operacional – 'snapshot');

O Container é uma layer read-write montada a partir de uma image que é read-only;

No Docker uma image é um layer read-only;

Uma image é uma herança de imagem;

Uma base image é quando uma imagem não possui um Pai (única).

Como é construído um Container?

O container é construído usando namepspaces, cgroups, chroot entre outras funcionalidades do kernel para construir uma área isolada para sua aplicação.

O que é o Registry (Registro)?

O registro é um repositório provido pelo Docker. Ele se econtra na nuvem, e disponibiliza uma área para envio, download e compartilhamento de imagens (snapshots) de Containers.

_____

_____