Containers
(Docker e afins)

HELLO!

Gustavo Correia Gonzalez

Vinicius Ribeiro Morais

1.

Como surgiu o conceito de containers?

“

A criação e manutenção de uma máquina virtual (Virtual Box, Hyper-V ou VMWare) demanda grande quantidade de tempo, além do fato dessas máquinas virtuais consumirem uma quantidade imensa de espaço em disco.

Motivos

- Crescimento da demanda por máquinas virtuais.
- Grande dificuldade na operação desse ambiente.
- Necessidade de melhorar o modelo.

Linux Container

Mais conhecido como LXC

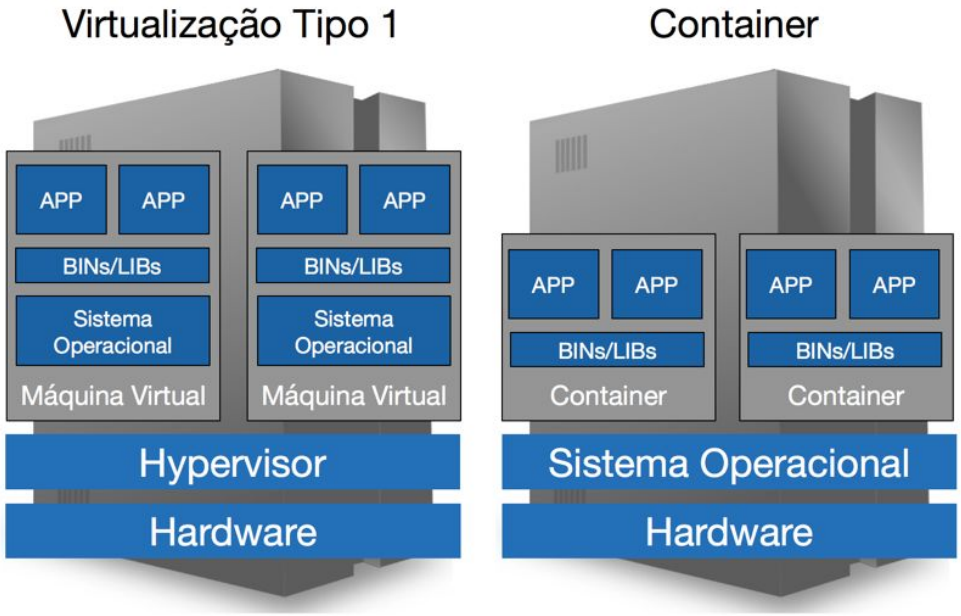
O que é um Container?

Foi lançado em 2008 e é uma tecnologia que permite a criação de múltiplas instâncias isoladas de um determinado Sistema Operacional dentro de um único hospedeiro ou em outras palavras, **é uma maneira de virtualizar aplicações dentro de um servidor GNU/Linux**.

LXC

Contêineres Linux podem ser definidos como uma combinação de várias funcionalidades de kernel (ou seja, coisas que o kernel pode fazer), que permitem o gerenciamento de aplicações (e recursos que elas utilizam) contidas dentro de seus próprios ambientes.

Virtualização X Container



Vantagens do Container

- Ao comparar com a virtualização tradicional, fica mais claro que uma aplicação sendo executada em um LXC demanda muito menos recursos, consumindo menos espaço em disco **e com um nível de portabilidade difícil de ser alcançado por outras plataformas.**

Vantagens do Container

- Aplicações podem ser portadas direto do notebook do desenvolvedor para o servidor de produção, ou ainda para uma instância virtual em uma nuvem pública.

2.

O que é o Docker?

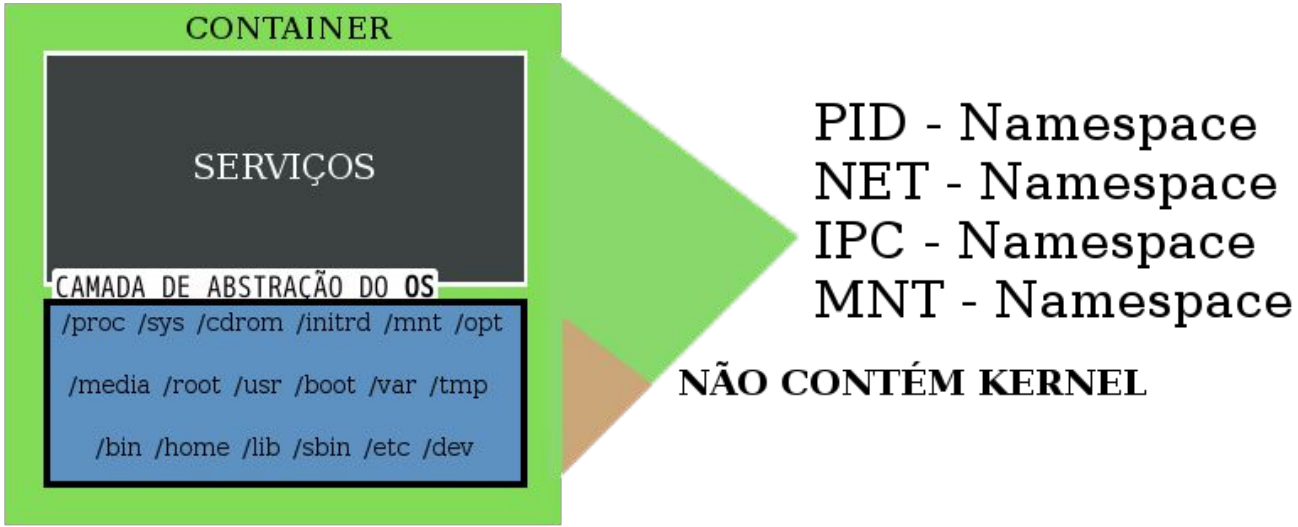
O começo!

Hoje um dos mais conhecidos LXC's do mercado é o **Docker**, escrito em **GO**, que nasceu como um projeto open source da **DotCloud**, uma empresa de **PaaS (Platform as a Service)** que apesar de estar mais interessada em utilizar LXC apenas em suas aplicações, acabou desenvolvendo um produto que foi muito bem aceito pelo mercado.

Como funciona

O Docker é uma ferramenta que pode empacotar um aplicativo e suas dependências em um recipiente virtual que pode ser executado em qualquer servidor Linux. Isso ajuda a permitir flexibilidade e portabilidade de onde o aplicativo pode ser executado, quer nas instalações, nuvem pública, nuvem privada, entre outros.

Como funciona



Quando um container é criado ...

O Docker tira proveito do recurso de **Namespaces** para prover um espaço de trabalho isolado para os contêineres. Sendo assim, quando um contêiner é criado, automaticamente um conjunto de namespaces também é criado para ele como mostra a imagem a cima. O namespace cria uma camada de isolamento para grupos de processos.

Grupos de processos

- PID – Isolamento de processos.
- NET – Controle de interfaces de rede.
- IPC - Controle dos recursos de IPC (InterProcess Communication).
- MNT - Gestão de pontos de montagem.

Partes principais do Docker

- Docker Daemon: usado para gerenciar os contêineres docker (LXC) no host onde ele roda.
- Docker CLI: usado para comandar e se comunicar com o Docker Daemon.
- Docker Image Index: um repositório (público ou privado) para as imagens do Docker.

Elementos do Docker

Os seguintes elementos são usados pelas aplicações que formam o projeto docker.

- Contêineres Docker
- Imagens Docker
- Dockerfiles

Contêineres Docker

Os contêineres docker são basicamente, diretórios que podem ser empacotados como qualquer outro, e então, compartilhados e executados entre várias máquinas e plataformas (hosts). A única dependência é ter os hosts ajustados para executar os contêineres (ou seja, ter o docker instalado). A contenção aqui é obtida através de Contêineres Linux (LXC).

Imagens Docker

As imagens docker constituem a base para os contêineres docker de onde tudo começa a se formar. Elas são muito similares às imagens de disco padrão de sistema operacional que são utilizadas para executar aplicações em servidores e computadores de mesa.

Dockerfiles

São scripts contendo uma série sucessiva de instruções, orientações e comandos que devem ser executados para formar uma nova imagem docker. Cada comando executado traduz-se para uma nova camada da cebola, formando o produto final. Elas basicamente substituem o processo de se fazer tudo manualmente e repetidamente.

Containers Docker permitem

- Portabilidade de aplicação.
- Isolamento de processos.
- Prevenção de violação externa.
- Gerenciamento de consumo de recursos.

e mais, requerendo muito menos recursos do que máquinas virtuais tradicionais usadas para a implantação de aplicações isoladas.

Containers Docker não permitem

- Mexer com outros processos.
- Causar "*dependency hell*".
- Ou não trabalhar com um sistema diferente.
- Ser vulnerável a ataques e abusar de todos os recursos do sistema.

E mais ...

Sendo baseado e dependendo do LXC, a partir de um aspecto técnico, estes contêineres são como um diretório (moldado e formatado). Isso permite portabilidade e construção gradual de contêineres.

3.

Principais Funcionalidades do Docker

Contêineres facilmente portáveis

Você pode criar uma imagem de toda a configuração e aplicativos instalados em seu contêiner, transferir e instalar em um outro host desde que tenha um Docker previamente instalado.

Versionamento

Docker permite que você versione as alterações de um contêiner de uma forma muito semelhante ao feito pelo **GIT** ou **SVN**. Isto permite portanto verificar as diferenças entre versões, fazer commit de novas versões e fazer **rollback** de uma dada versão.

Reutilização de componentes

As imagens criadas podem ser reutilizadas, como por exemplo, se diversas de suas aplicações utilizam um stack com Java 8, Tomcat 8 e Oracle 12 você poderá criar uma uma imagem base contendo estes itens com sua instalação e configuração. Desta maneira esta imagem poderá ser reutilizada em diversos Contêineres diferentes. Podemos construir imagens Docker usando um arquivo **Dockerfile** e o comando de montagem **docker build**.

Compartilhamento

O **Docker Hub** já possui milhares de contêineres com as mais diversas aplicações instaladas e configuradas, desta maneira você pode rapidamente criar sua aplicação com uma base desenvolvida por outra pessoa, ou ainda criar sua base e compartilhá-la.

CLI e API

CLI(**Command Line Interface**) e API(**Application Program Interface**) permitem a criação de Programas e Scripts que interagem com o Docker para provisionar serviços nos Contêineres.

Automatização de Implantação dentro dos Contêineres

Usando os provisionadores que por sua vez usam a API do Docker, podemos automatizar a implantação dos ambientes de software.

Licença Open Source

Licenciado como Apache License, Version 2.0 mantém os códigos fonte disponíveis para facilitar o desenvolvimento colaborativo. Teve o código aberto pela dotCloud em Março de 2013.

Evita Dependency Hell

Um dos maiores problemas em múltiplos ambientes com os quais os desenvolvedores de software convivem diariamente é o gerenciamento de dependências. O Docker evita problemas neste gerenciamento.

Demanda Poucos Recursos de Hardware

Exige poucos recursos de processos, memória e espaço em disco.

Performance inigualável

Por exemplo, é possível baixar uma imagem do Fedora ou Debian do repositório público na Internet em menos de um minuto e executar um comando simples num contêiner criado com esta imagem, à partir do computador Host, em menos de um segundo.

Ligação entre Contêineres

Conectar contêineres via mapeamentos de porta TCP/IP não é a única forma de disponibilizar recursos entre eles. Um contêiner Docker pode se conectar a um outro via um sistema de ligação e enviar informações de um para o outro de forma eficiente e segura. Quando os contêineres estão ligados, a informação sobre o contêiner origem pode ser enviada para um contêiner destino.

4.

Tutorial do Docker


Let’s do this.

Instalação

https://docs.docker.com/engine/installation/

Hello World

- $ docker run ubuntu /bin/echo “Hello World!”

Imagens do Docker

- https://hub.docker.com/

Pesquisando uma imagem

- $ docker search <nome-da-imagem>

Baixando uma imagem

- $ docker pull <nome-da-imagem>

Exibindo todas as imagens baixadas

- $ docker images

Exibindo todos os containers

- $ docker ps
- $ docker ps -l

Exemplo prático (Blog com Wordpress)

- $ docker run --name database -e MYSQL_ROOT_PASSWORD=teste123 -d mysql

- $ docker run --name blog-teste --link database:mysql -e WORDPRESS_DB_PASSWORD=teste123 -p 80:80 -d wordpress

Entrando em um container em execução

- $ docker exec -i -t blog-teste bash
- $ docker run -i -t ubuntu bash

Excluindo um container

- $ docker rm <nome-do-container> | <id>

Parando um container que está em execução

- $ docker stop <nome-do-container>

Excluindo todos os containers com um só comando

- $ docker rm $(docker ps -q -a)

Excluindo uma imagem

- $ docker rmi <nome-da-imagem> | <ID>

Executando comandos dentro do container

- $ docker exec -it <nome-do-container> <comando>

“Matando” um container

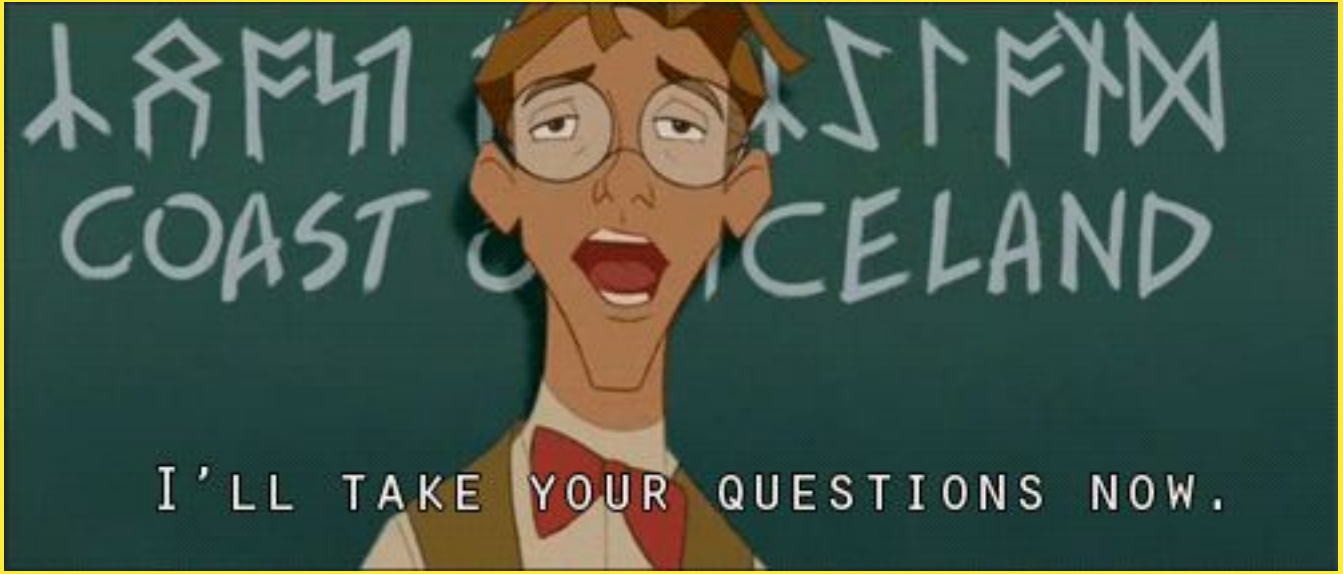
- $ docker kill <nome-do-container>

Instalando programas no container

Criando seu próprio container

- $ docker commit -m "comentário" <nome-do-container> <nome-da-imagem>

Dúvidas?



Sugestões para estudo

- https://docs.docker.com/
- https://www.digitalocean.com/community/tutorials/como-instalar-e-utilizar-o-docker-primeiros-passos-pt
- http://it-ebooks.info/book/6555/