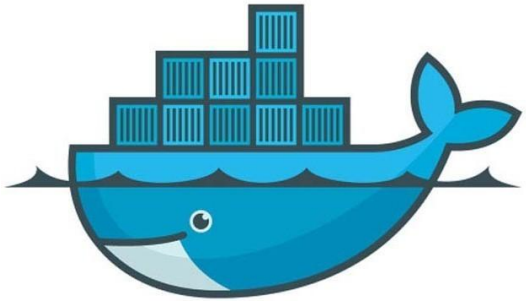
SEDAT/SEAT5

Docker Básico



SERPRO
Soluções digitais que conectam governo e sociedade

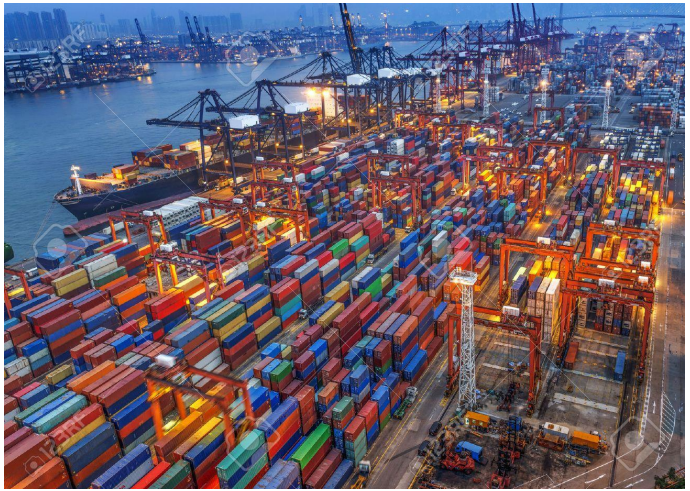
Propósito

Propósito

- Permitir empacotar uma aplicação com todas as suas dependências em uma unidade padronizada para desenvolvimento de software;

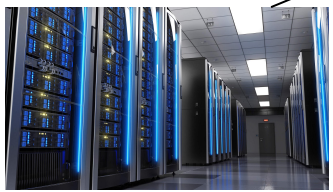
- Um paralelo com o problema de transporte de cargas (eletrônicos, grãos, carros) em navios, trens, caminhões etc;

- Engenharia de Hardware se comportando como Engenharia de Software;

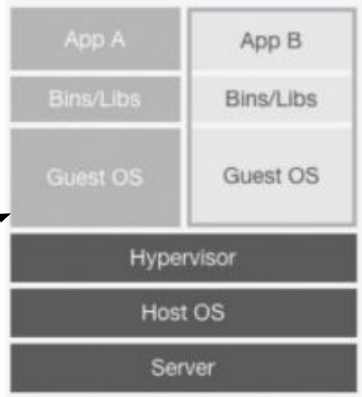


Evolução

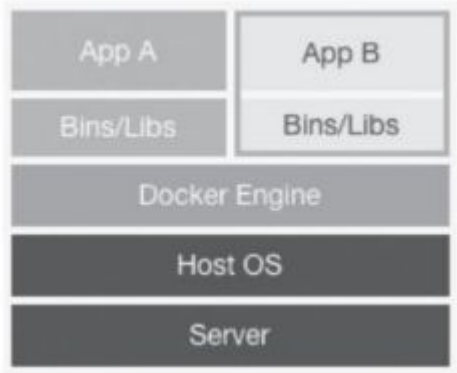
Evolução



Máquinas Físicas



Virtualização do S.O.



Containers

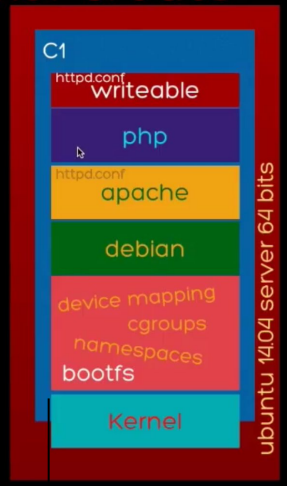
O Docker

http://joao-parana.com.br Curso - Introdução ao Docker LS LINUX SOLUTIONS

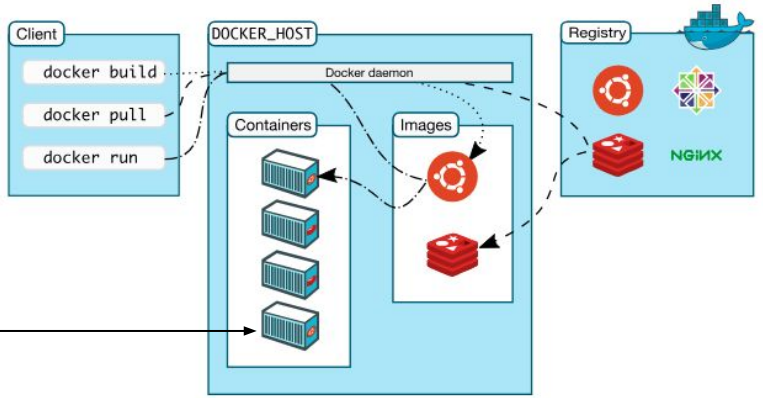
Copy-on-write - docker create

Entenda como funciona

```
$ docker create -t -i --name C1 debian bash
6ba326753db2a7365 . . .
$ docker start -a -i 6ba3
bash# apt-get install apache2
bash# apt-get install php
bash# vi httpd.conf
bash# exit
$ docker ps -a
```



1



Componentes:

- **Engine:** É um daemon que gerencia a construção e execução dos contêineres. Controla os recursos de rede, CPU, memória e demais recursos do host que os contêineres utilizam.

- **Cliente:** Passa os comandos expostos pela API da engine para criar, executar, parar contêineres, além de listar imagens etc.

- **Registry:** Repositório onde hospedamos as imagens dos contêineres. Pode ser público (Docker Hub) ou privado.

- **Compose:** Ajuda a organizar a execução de diversos contêineres e a forma como eles irão utilizar recursos de rede, persistência de dados e se comunicarem.

- **Swarm:** Faz a orquestração de contêineres espalhados em máquinas distintas, numa espécie de cluster.

Docker

Vantagens sobre as máquinas virtuais:

- Padronizar ambientes de desenvolvimento, testes, produção;

- Melhora a utilização de recursos físicos de infraestrutura;

- Melhora também a reutilização, migrando um contêiner de um host para outro;

- Como se inicia só o processo e não a pilha toda do sistema, a inicialização é praticamente instantânea;

- Utilizando o Docker Hub ou registry privado é possível distribuir as imagens;

- Limitar o uso de memória e CPU no comando de execução do contêiner;

Desvantagens:

- Por causa do sistema de imagens em camada, o overhead de IO no disco é muito maior;

- Dificulta troubleshooting, já que se adiciona mais uma camada na investigação do problema;

- Por mais isolado que os processos estejam, como há compartilhamento de recursos, há a possibilidade de um ataque sofisticado;

Porque o desenvolvimento foi pra esse caminho?

- Ambientes semelhantes promovem entrega contínua

- Facilitar a distribuição da aplicação diminuindo necessidade de configurações de ambiente

- Padronização e replicação de ambientes de desenvolvimento

- Testar a aplicação sobre uma nova infraestrutura fica muito mais fácil

- Promove um idioma comum entre dev e op e facilita a troca de ideias sobre boas práticas

- É possível reusar bons modelos de infraestrutura

Principais comandos:

- docker pull: comando que baixa imagens de um registry para a máquina local. A sintaxe básica é **docker pull nome_da_imagem:versão** (docker pull postgres:9.5.4).

- docker push: comando que envia imagens da máquina local para um registry.

- docker images: comando que exibe as imagens já baixadas localmente.

- docker ps: comando que mostra os containers rodando atualmente.

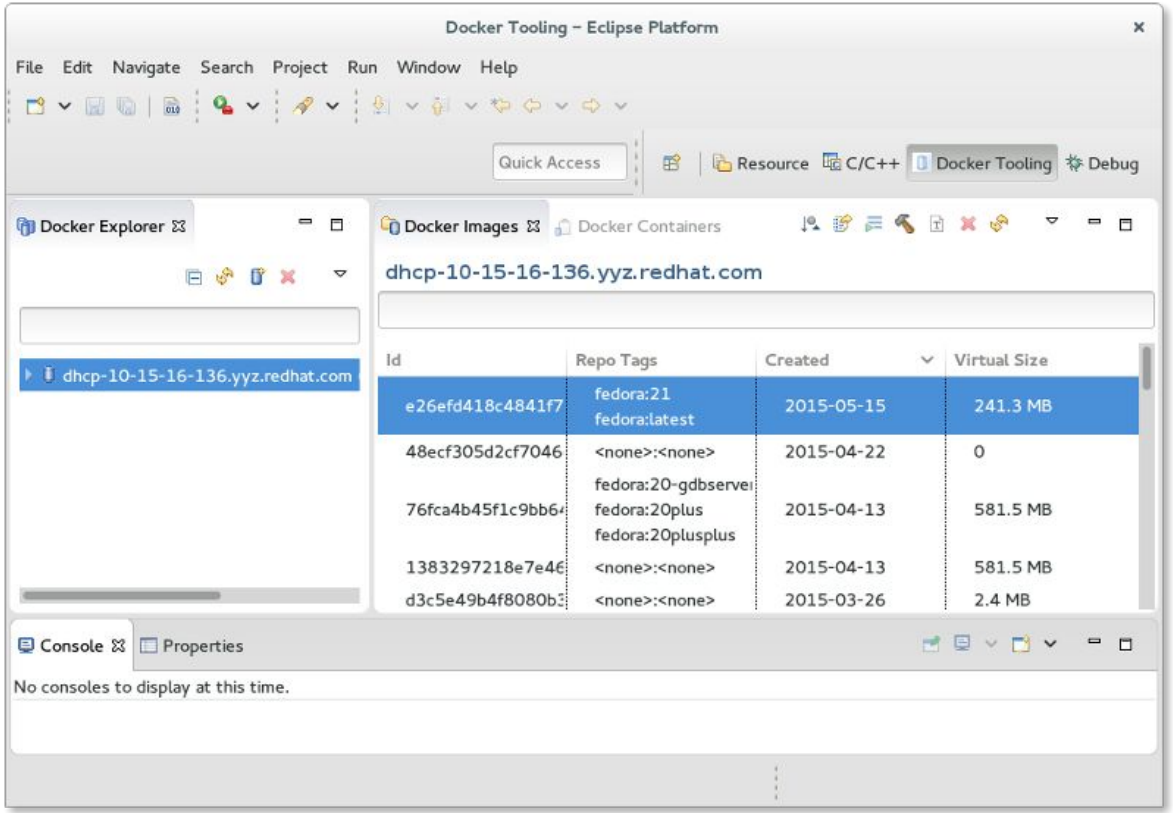
- **docker run:** comando que cria um container a partir de uma imagem. Se a imagem não existir localmente o docker tenta baixá-la.

- docker exec: comando que executa algo num container, muito usado para "entrar" no container. (docker exec -ti meu_apache bash)

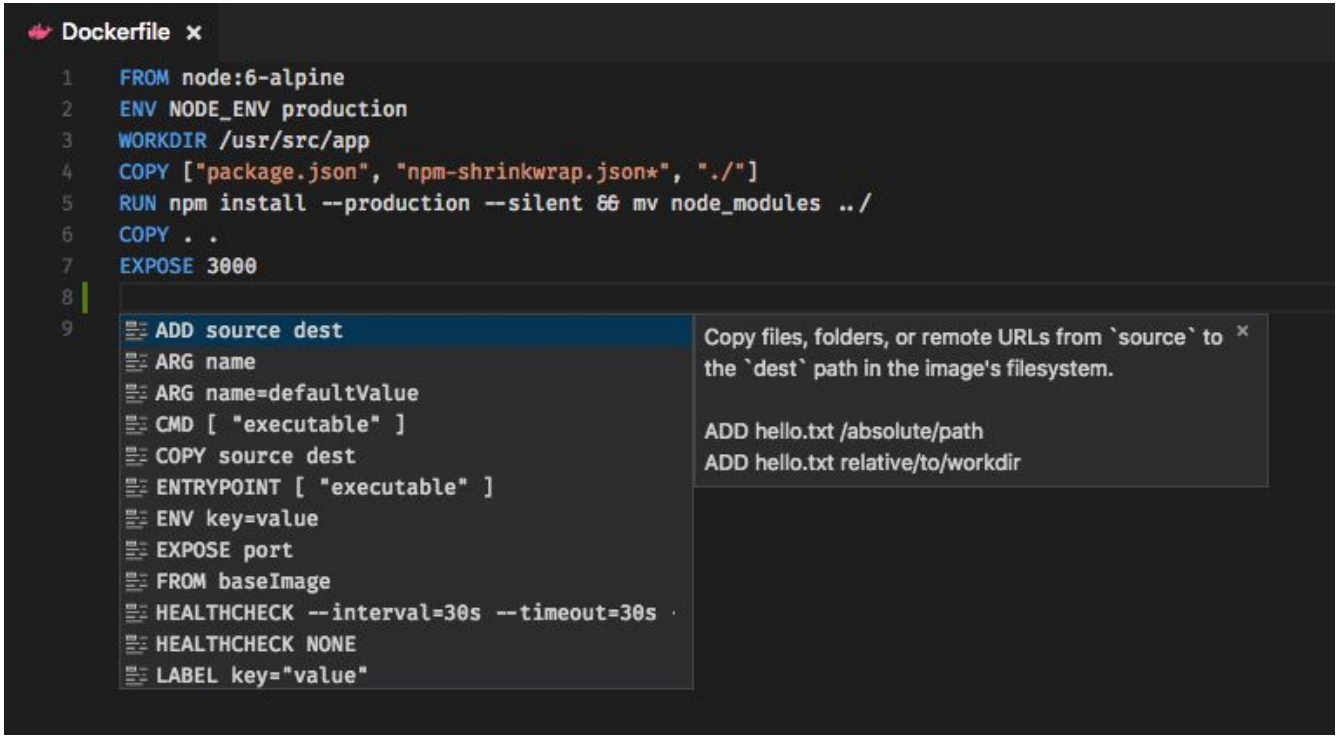
- docker rm: comando que remove um container, a sintaxe básica é **docker rm (container_id|nome)**, caso o container ainda esteja rodando é preciso usar o "-f".

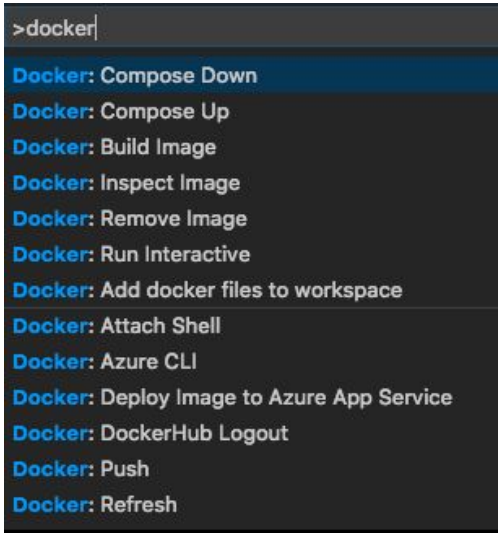
- docker build: comando que constrói uma imagem a partir de um arquivo Dockerfile. A sintaxe básica é **docker build -t nome_da_imagem:versão**

Para uma gestão visual (Plugin Eclipse):



Para uma gestão visual (Plugin Visual Studio):





Vamos para a prática!(1)

1) Rodar contêiner Java compilando e executando uma classe.

Vamos para a prática!(2)

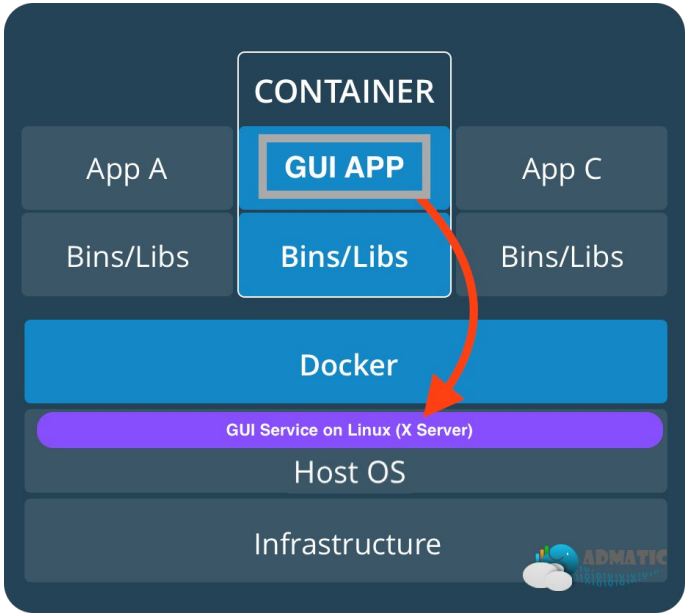
1) Rodar contêiner Tomcat mapeando porta e volume e mostrando o html no Browser

Depois iremos brincar com a imagem alterando-a e publicando.

Vamos para a prática!(3)

1) Mostrar que é possível rodar um contêiner de minha imagem
2) Mostrar que o mapeamento de volume provoca reflexo imediato das alterações do arquivo

Aplicações GUI em contêiner



Dockerfile:

```
FROM ubuntu:14.04

RUN apt-get update && apt-get install -y firefox

# Replace 1000 with your user / group id
RUN export uid=1000 gid=1000 && \
    mkdir -p /home/developer && \
    echo "developer:x:${uid}:${gid}:Developer,,,:/home/developer:/bin/bash" >> /etc/passwd && \
    echo "developer:x:${uid}:" >> /etc/group && \
    echo "developer ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/developer && \
    chmod 0440 /etc/sudoers.d/developer && \
    chown ${uid}:${gid} -R /home/developer

USER developer
ENV HOME /home/developer
CMD /usr/bin/firefox
```

Constrói a imagem: docker build -t firefox .

Roda a imagem:

```
docker run -ti --rm \
       -e DISPLAY=$DISPLAY \
       -v /tmp/.X11-unix:/tmp/.X11-unix \
       firefox
```

Docker

Dockerfile:

- **Arquivo responsável por realizar a criação e construção de imagens no Docker;**
- **Contém instruções que o Docker deve seguir para conseguir realizar a criação de uma imagem;**
- **As instruções são interpretadas linha a linha pelo motor do Docker;**
- **Possui uma linguagem onde devemos respeitar sua sintaxe e seus comandos existentes;**

Docker

Dockerfile (Principais comandos):

- **FROM: Define qual imagem base a ser utilizada para iniciar o estágio de compilação;**
- **RUN: Inicia uma nova camada e nele você pode executar N comandos (apt-get install);**
- **EXPOSE: Expõe as portas de escuta do container especificadas na execução;**
- **ENV: Variáveis de ambiente a serem enviadas para a imagem;**
- **COPY: Copia os arquivos do diretório especificado do host para o diretório especificado dentro do container;**
- **ADD: Além de copiar também arquivos do local, ele serve para enviar arquivos empacotados com "tar" e automaticamente descompactar no destino e baixar arquivos de uma URL, desde que não tenha autenticação;**
- **CMD: É o que vale no start do container;**
- **VOLUME: Instrução que fala qual volume será mapeado para o container;**
- **USER: Nome de usuário (ou UID) a serem usados ao executar a imagem e as instruções;**
- **WORKDIR: Diretório à partir de onde os comandos ou as ações serão realizadas;**

Dockerfile (Vamos à prática):

- Iremos construir um Dockerfile que represente a imagem de uma aplicação python junto com suas dependências.

Docker Compose:

- Ferramenta separada (não faz parte do docker em si), mas muito útil para "orquestrar" conjuntos de containers sem muita complexidade e em um único nó;
- Kubernetes, por exemplo, seria adequado para um uso em múltiplos nós;
- Já vem junto ao pacote Docker da central de pacotes;
- Recomenda-se seu uso para quando se precisa rodar mais de um container ao mesmo tempo (ex: apache + jboss)

Docker Compose:

docker-compose.yml =

```yaml
version: '2'

services:

  wildfly:
    image: jboss/wildfly:10.1.0.Final
    container_name: wildfly
    ports:
      - "9000:8080"
    volumes:
      - /tmp/teste3/ROOT.war:/opt/jboss/wildfly/standalone/deployments/ROOT.war

  postgres:
    image: postgres:9.6-alpine
    container_name: postgres
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - PGDATA=/var/lib/postgresql/data/pgdata
    ports:
      - '5432:5432'
    volumes:
      - /tmp/teste3/banco:/var/lib/postgresql/data/pgdata
```

Docker Compose (Vamos à prática):

- Subir um wildfly com um .war e um banco postgres;

Referências:

- Figueira da Silva, Wellington. Aprendendo Docker

- Gomes, Rafael. https://github.com/gomex/docker-para-desenvolvedores

- https://docs.docker.com

- **Página da DEDAT:**
  https://dedat.gitpages.serpro/deat5/deat5-gitpages/introducao-docker/

SEDAT/SEAT5

lista-deat5@grupos.serpro.gov.br

/serprobrasil

@serprobrasil

@serpro

/serpro

serpro.gov.br

SERPRO

Soluções digitais que conectam governo e sociedade