INF-555
Virtualização de Redes e Sistemas Computacionais

Confinamento de processos (container).
Software para container (LXC / Docker)

William Lima Reiznautt
william@ic.unicamp.br

Confinamento de processos (container)
softwares para container

Ferramentas para deploy/gerenciamento de container

LXC

lxd

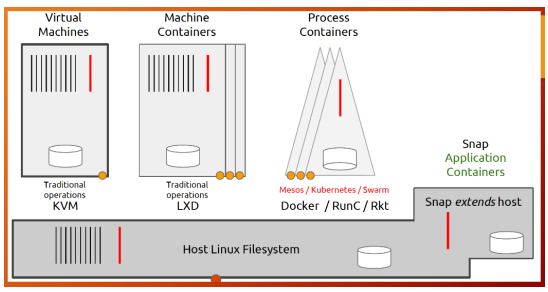
docker

Virtuozzo

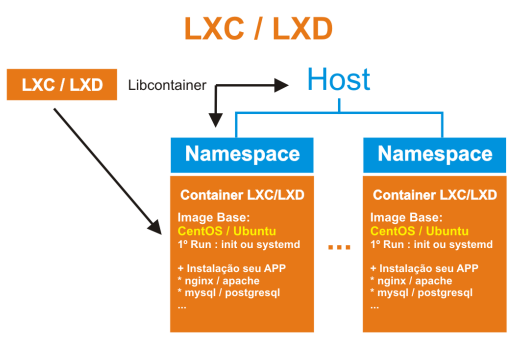
OpenVZ
Linux Containers

Confinamento de processos (container)
VMs vs Linux Container vs App Container

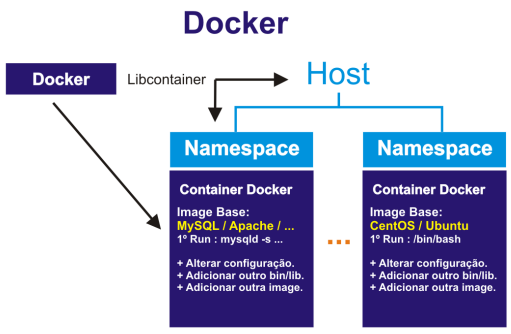
VM / Containers



Confinamento de processos (container)
LXC / LXD



Confinamento de processos (container)
Docker



Software para Contêiner : LXC / LXD e Docker
LXC / LXD

LXC / LXD

Developer(s)
Kernel: Virtuozzo, IBM, Google, Eric Biederman and others
Userspace: Daniel Lezcano, Serge Hallyn, Stéphane Graber and others
Initial release: August 6, 2008;
Stable release: 2.1.1 / 19 October 2017;
Repository: https://github.com/lxc/lxc
Written in: C, Python, Shell, Lua
Operating system: Linux
Platform: x86, IA-64, PowerPC, SPARC, Itanium, ARM
Type: OS-level virtualization
License: GNU LGPL v.2.1 (some components under GNU GPL v2 and BSD)
Website: linuxcontainers.org

Software para Contêiner : LXC / LXD e Docker
LXC / LXD

LXC / LXD

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

Suas imagens é baseada em imagens pré-criadas disponíveis para um grande número de distribuições Linux e é construída em torno de uma API REST muito poderosa, mas bastante simples.

O que é o LXD?
O LXD é um gerenciador de contêineres de próxima geração.

Software para Contêiner : LXC / LXD e Docker
LXC / LXD

Caractéristicas do LXC

LXC usa as seguintes features do kernel para conter os processos:

- Kernel namespaces (ipc, uts, mount, pid, network and user)
- Apparmor e SELinux profiles
- Seccomp policies
- chroot
- Kernel capabilities
- CGroups (control groups)

Os contêineres LXC geralmente são considerados algo intermediário entre um chroot e uma máquina virtual completa. O objetivo do LXC é criar um ambiente o mais próximo possível de uma instalação padrão do Linux, mas sem a necessidade de um kernel separado.

https://linuxcontainers.org/lxc/

Software para Contêiner : LXC / LXD e Docker
LXC / LXD

Caractéristicas do LXD

Alguns dos maiores recursos do LXD são:

- Seguro pelo seu design (contêineres não privilegiados, restrições de recursos e muito mais)
- Escalavel (de contêineres em seu laptop até diversos nodes)
- Intuitivo (API simples e clara e experiência de linha de comando)
- Imagem baseada (com uma ampla variedade de distribuições Linux publicadas diariamente)
- Suporte para contêiner entre hosts e transferência de imagens (incluindo live-migration com CRIU)
- Controle avançado de recursos
  (cpu, memória, E / S de rede, E / S de bloco, uso de disco e recursos do kernel)
- Passagem de dispositivos
  (USB, GPU, unix character e dispositivos de bloco, NICs, discos e path do sistema host)
- Gerenciamento de rede (criação e configuração de bridges, túneis, ...)
- Gerenciamento de storages
  (suporte para vários back-ends de armazenamento, pools e volumes de armazenamento)

https://linuxcontainers.org/lxd/

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - instalação

LXC / LXD

Instalação do LXC no Ubuntu 16.04 LTS

```
$ sudo -i
```

Atualizando metadata do mirrors e instalando LXC.

```
# apt update
# apt install lxc
```

Checando a configuração do sistema se atende todas as necessidades.

```
# lxc-checkconfig
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - utilização

LXC / LXD

Iniciando o primeiro contêiner LXC :

- Imagens disponível **https://us.images.linuxcontainers.org/**

Criando um primeirolxc com ubuntu xenial a partir de um download no mirrors de images;

```
# lxc-create -t download -n primeirolxc -- -d ubuntu -r xenial -a amd64
```

Listando todos contêineres criados no LXC, iniciando e se tachando ao bash.

```
# lxc-ls -f
NAME        STATE   AUTOSTART GROUPS IPV4 IPV6
primeirolxc STOPPED 0         -      -    -
```

```
# lxc-start -n primeirolxc
```

```
# lxc-attach -n primeirolxc
```

* criar usuário e setar senha de acesso root. (passwd root ; adduser <your-username>)

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - utilização

LXC / LXD

Acessando o console de seu primeiro container.

```
# lxc-console -n primeirolxc

<login>
```

* Informações sobre seu container.

```
# lxc-ls -f
```

```
# lxc-info -n primeirolxc
```

```
# lxc-top -n primeirolxc
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - utilização

LXC / LXD

Administrando LXC container:

```
# start   : lxc-start -n <lxc-name>
# console : lxc-console -n <lxc-name>
# list : lxc-ls -f
# info : lxc-info -n <lxc-name>
# stop : lxc-stop -n <lxc-name>
# destroy : lxc-destroy -n <lxc-name>
# top  : lxc-top -n <lxc-name>
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - utilização

Vamos iniciar dois ambiente LXC com o Centos (escolha a mesma opção)

```
# lxc-create -t download -n segundolxc -- -d centos
```

```
# lxc-create -t download -n terceirolxc -- -d centos
```

Qual foi o tempo de fazer o deploy do segundo ambiente ?

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

Onde fica o **rootfs** do container ? (/)

 # /var/lib/lxc/primeirolxc

f: config
d: rootfs

```
# ls -l /var/lib/lxc/primeirolxc/
-rw-r--r--  1 root root  823 Apr 23 15:39 config
drwxr-xr-x 21 root root 4096 Apr 23 15:41 rootfs
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

Subprocesso do Host

Execute o comando sleep 9000 dentro do container.

```
# ps -aux | grep sleep
```

```
# pstree -aun
```

Qual é o problema aqui ?

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

Rede no Host

```
# ip a
```

```
12: vethLHX81L@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc noqueue master lxcbr0 state UP group default qlen 1000
    link/ether fe:13:70:db:43:38 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::fc13:70ff:fedb:4338/64 scope link
       valid_lft forever preferred_lft forever
```

Outro comando para ver a bridge: ***# brctl show***

```
# ip netns list
```

* Criado dentro do kernel (sem intereção do usuário root)

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

Rede no Host

Como acessar o namespace de rede criado para o LXC.

- Pegar um numero de processo que estaja rodando sobre o netns.

```
# lxc-info -n primeirolxc -p
PID:            16426
```

- Criar diretorio temporiario de netns (estrutura de leitura ip netns)

```
# mkdir /var/run/netns
```

- Criar um link simbolico do estado do processo (ns/net)

```
# ln -sf /proc/16426/ns/net /var/run/netns/primeirolxc
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

- Agora podemos acessar o namespace que o processo está utilizando

```
# ip netns list
primeirolxc (id: 1)
```

- Vamos inciar um processo bash sobre o namespace de rede do container LXC.

```
# ip netns exec primeirolxc bash
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:88:10:14
          inet addr:10.0.3.2  Bcast:10.0.3.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe88:1014/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2137 (2.1 KB)  TX bytes:1374 (1.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

Entendemos como roda o container dentro do host (processo e rede).

* Processos
* Rede

Estado de Kernel do processo: `/proc/<numero pid>/ns`

Segurança (do host)

* Como mapear os UID e GID para dentro do LXC.

* Como configurar unprivileged user para rodar LXC.

https://linuxcontainers.org/lxc/security/

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

LXC / LXD

```
# cat /var/lib/lxc/<nome-lxc>/config
# Template used to create this container: /usr/share/lxc/templates/lxc-download
# Parameters passed to the template: -d ubuntu -r xenial -a amd64
# Template script checksum (SHA-1): 9748088977ba845f625e45659f305a5395c2dc7b
# For additional config options, please look at lxc.container.conf(5)

# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (Be aware this has security implications)


# Distribution configuration
lxc.include = /usr/share/lxc/config/ubuntu.common.conf
lxc.arch = x86_64

# Container specific configuration
lxc.rootfs = /var/lib/lxc/primeirolxc/rootfs
lxc.rootfs.backend = dir
lxc.utsname = primeirolxc

# Network configuration
lxc.network.type = veth
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00:16:3e:1f:93:e1
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

LXC / LXD

Configuração padrão do LXC (ira setar em todos os novos LXC criados)

```
# cat /etc/lxc/default.conf
lxc.network.type = veth
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00:16:3e:xx:xx:xx
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

LXC / LXD

* Como mapear os UID e GID para dentro do LXC. (Segurança)

```
# cat /etc/lxc/default.conf
lxc.network.type = veth
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00:16:3e:xx:xx:xx
lxc.id_map = u 0 100000 65536
lxc.id_map = g 0 100000 65536

# cat /etc/subuid
root:100000:65536
ubuntu:100000:65536

# cat /etc/subgid
root:100000:65536
ubuntu:100000:65536
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

LXC / LXD

* Como mapear os UID e GID para dentro do LXC. (Segurança)

```
# chmod o+x /var/lib/lxc

# lxc-create -t download -n mapuserlxc -- -d ubuntu -r xenial -a
  amd64

# ls -l /var/lib/lxc/mapuserlxc/rootfs/

total 76
drwxr-xr-x  2 100000 100000 4096 Apr 23 03:56 bin
drwxr-xr-x  2 100000 100000 4096 Apr 12  2016 boot
drwxr-xr-x  3 100000 100000 4096 Apr 23 03:57 dev
drwxr-xr-x 66 100000 100000 4096 Apr 23 16:32 etc
....
```

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

LXC / LXD

Creating unprivileged containers as a user
Creating unprivileged containers as root
Creating privileged containers

https://linuxcontainers.org/lxc/getting-started/

LXD (more secutiry about your container and ad)

https://linuxcontainers.org/lxd/getting-started-cli/

Software para Contêiner : LXC / LXD e Docker
LXC / LXD - funcionamento

Outras configurações e limites para LXC

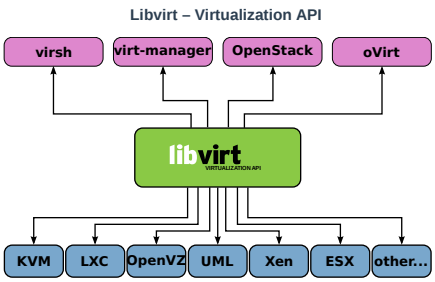
https://linuxcontainers.org/lxc/manpages/man5/lxc.container.conf.5.html

https://linuxcontainers.org/lxc/manpages/man1/lxc-cgroup.1.html

LXD

https://linuxcontainers.org/lxd/getting-started-cli/

Software para Contêiner : LXC / LXD e Docker
LXC / LXD – canivete suiço



HVM – PVM – Containers – Firewall – Network – Migration – Storage – Deploy
Access – Authentication

https://libvirt.org : Virtualization API

Software para Contêiner : LXC / LXD e Docker
Docker

Docker

O Docker é um programa que realiza virtualização no nível do sistema operacional, também conhecido como conteiner, especificamente para APPs e Programas de SO.

É desenvolvido pela Docker, Inc.

O Docker é desenvolvido principalmente para Linux, onde usa os principais recursos de isolamento do kernel Linux (cgroups / namespaces). Permitindo que "contêineres" independentes sejam executados dentro de um uma única instância do Linux, dando a visão de como se tivesse diversos sistemas isolados.

https://docs.docker.com

Software para Contêiner : LXC / LXD e Docker
Docker

Docker

O suporte do kernel Linux para namespaces principalmente isola a visão de um aplicativo (processo) do ambiente do sistema operacional.

- Incluindo árvores de processo
- Rede
- IDs de usuário
- Sistemas de arquivos montados
- CGroups do kernel fornecem recursos limitadores de memória e CPU.

Desde a versão 0.9, o Docker inclui a biblioteca libcontainer como uma maneira de usar diretamente os recursos de virtualização fornecidos pelo kernel do Linux, além de usar interfaces de virtualização abstratas via libvirt, LXC e systemd-nspawn.

Software para Contêiner : LXC / LXD e Docker
Docker

Docker

- Utiliza funcionalidades de isolamento de recursos do Kernel de Linux
- Cada container executa um processo separado
- Consome recursos nativos do hardware sem a camada de virtualização e do SO.
- O processo é “virtualizado”, mas o processamento não é virtualizado.

O suporte no Windows e no MAC atualmente não é nativo:
- Windows usa o Hyper-V
- MAC-OS xhyve

(Virtualiza o Docker Engine e as características do kernel de Linux para o Docker daemon)

Software para Contêiner : LXC / LXD e Docker
Docker

Introdução Docker

- Ambientes de desenvolvimento comuns (Desenvolvimento / Infraestrutura)
- Deploy das aplicações com maior facilidade e de forma automatizada.
- Agilidade, controle e portabilidade na gestão dos ambientes (Ex. MySQL, PHP, Solr, etc)

Cada serviço é representado por um container ou até mesmo em um unico container especifico.

Software para Contêiner : LXC / LXD e Docker
Docker

Docker

Original author(s)	Solomon Hykes
Developer(s)	Docker, Inc.
Initial release	13 March 2013; 5 years ago
Stable release	18.03.0-ce / 21 March 2018;
Repository	github.com/docker/docker-ce (Community Edition)
Written in	Go
Operating system	Linux, Windows and MAC
Platform	x86-64, ARM (experimental)
Type	Operating-system-level virtualization
License	Binaries: Freemium software as a service. Source code: Apache License 2.0
Website	docker.com

*** Wikipedia**

Software para Contêiner : LXC / LXD e Docker
Docker

Conceitos de Docker

Docker, Docker API, Images, Containers, DockerHub e Dockerfile.

Compose e Swarm.

Vantagens e Desvantagens de utilizar Docker.

Software para Contêiner : LXC / LXD e Docker
Docker

Imagem Docker

Imagens Docker prove as aplicações
(binarios, bibliotecas, execuções e pre-configuração dos ambientes)
podem ser criadas por qualquer pessoas, ou disponibilizada pela equipe de desenvolvimento oficial.

* Imagens podem ser criadas somente com suas aplicações.
* Ou existir uma imagem base -> seu conteudo -> sua imagem.

Software para Contêiner : LXC / LXD e Docker
Docker

Docker Hub (https://hub.docker.com/)
Dev-test pipeline automation, 100,000+ free apps, public and private registries

O lugar ideal para encontrar imagens atualizadas e com suporte oficial ou até mesmo imagens criada pela comunidade é o DockerHub, devido à popularidade do Docker.

Esta plataforma tem repositórios oficiais de Apache, MongoDB, Nginx, WordPress entre outros, prontos para serem utilizados.

Além disso, nesta plataforma, você poderá criar repositórios para suas imagens, se assemelhando ao GitHub ou GitLab, onde você pode criar múltiplos repositórios públicos, mas apenas um repositório privado. No caso de você precisar de mais, existe a possibilidade de pagar uma assinatura.

Software para Contêiner : LXC / LXD e Docker
Docker

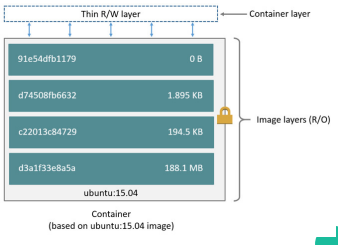
Imagem Docker e Container

Uma imagem refere-se a uma lista de camadas, que são empilhadas uma acima da outra.

Note que a imagem é imutável, mas facilmente estendida.

O container, por outro lado, é uma instância no tempo de execução de uma imagem.
Quando um novo container é criado, uma nova camada de escrita é criada no topo das camadas adjacentes.
Todas as alterações feitas no container em execução são feitas na mesma.

Uma vantagem dos containers Docker é a portabilidade, pois nos permite utilizar a mesma imagem em diferentes distribuições Linux com configurações distintas de hardware sem alterar as imagens.



Software para Contêiner : LXC / LXD e Docker
Docker

Exemplo Dockerfile: (meu sistema)

```
FROM httpd
COPY index.html /var/www/html/
RUN httpd-foreground
```

Compilando minha imagem com index.html de uma imagem base.

```
# docker build -t willreli/http-com-meu-index:latest
```

Camadas:
* httpd (tem varias camada tbm)
* copy index.html (segunda camada)
* run (terceira camada)

```
# docker history willreli/http-com-meu-index:latest
```

Software para Contêiner : LXC / LXD e Docker
Docker

Docker

```
# apt install docker.io
# docker -v
# systemctl status docker

# docker -v
Docker version 1.13.1, build 092cba3

# docker info
```

Software para Contêiner : LXC / LXD e Docker
Docker

https://hub.docker.com/

Administrando imagens

- Fazer download de uma imagem (oficial/comunidade).

```
# docker pull <nome-imagem-dockerhub>
```

* Pode-se utilizar dois pontos para definir uma versão específica da imagem.

```
# docker pull <nome-imagem-dockerhub>:<versão>
```

- Lista de imagens no local (baixados e criados).

```
# docker images
```

- Apagar imagens

```
# docker rmi <nome-imagem-dockerhub>
```

Software para Contêiner : LXC / LXD e Docker
Docker

```
# docker help image (todos os comandos docker)

Usage: docker image COMMAND

Manage images

Options:
      --help   Print usage

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

Software para Contêiner : LXC / LXD e Docker
Docker

Criar e iniciar um container docker baseada na imagem que está localmente, executar um comando:

```
# docker run -it --name <nome-container> <nome-imagem> <comando>
```

Exemplos:

```
# docker run -it --name my-container node bash
```

Criar e iniciar um container, executar comando e destruir o container

```
# docker run --rm -it --name my-container node bash
```

* É recomendável utilizar o parâmetro --rm , desta forma o container é removido automaticamente quando terminar sua execução.

```
# docker run -it -d --name teste6 centos bash
```

Software para Contêiner : LXC / LXD e Docker
Docker

Executa com terminal interativo e detach (-t terminal / -i interativo / -d *"desatachado")*

```
# docker run -it -d --name teste6 centos bash
```

Docker exec (novo processo) ou senão dependendo da aplicação você consegue atachar novamente (programa interativo ou debug)

```
# docker attach --detach-keys 'ctrl-a' teste7
```

Software para Contêiner : LXC / LXD e Docker
Docker

Docker run parametros:

```
# docker help run
...
-i, --interactive    Keep STDIN open even if not attached
...
-t, --tty            Allocate a pseudo-TTY
...
-p, --publish list   Publish a container's port(s) to the host (default [])
...
-P, --publish-all    Publish all exposed ports to random ports
```

Software para Contêiner : LXC / LXD e Docker
Docker

Docker containers:

```
# docker ps -a (lista container ativo e finalizados)

# docker top <nome-ou-id-container>

# docker exec -it 3a22d3c67adc bash
 - > sleep 9000

  ├─dockerd -H fd://
  │   ├─11*[{dockerd}]
  │   └─containerd -l unix:///var/run/docker/libcontainerd/docker-
containerd.sock --metrics-interval=0 --start-timeout 2m ...
  │       ├─10*[{containerd}]
  │       ├─containerd-shim
3a22d3c67adc3b0f7a2547d2c96cd018a08aad4d7d679c0380dd74e7dfd49338...
  │       │   ├─7*[{containerd-shim}]
  │       │   └─bash
  │       └─containerd-shim
3a22d3c67adc3b0f7a2547d2c96cd018a08aad4d7d679c0380dd74e7dfd49338...
  │           ├─7*[{containerd-shim}]
  │           └─bash
  │               └─sleep 9000
```

Software para Contêiner : LXC / LXD e Docker
Docker

Matar todos os containers em execução

```
# docker kill $(docker ps -q)
```

Parar container

```
# docker stop <id-ou-nome-container>
```

Eliminar todos os containers suspensos

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

Obter o IP de um container

```
# docker inspect container_name
```

Iniciando novamente o container no estado atual.

```
# docker start 3a22d3c67adc
# docker start -ai 3a22d3c67adc
```

Software para Contêiner : LXC / LXD e Docker
Docker

Dockerfile

Agora é o momento de criar nossas imagens customizadas (custom) e, para isso, utilizaremos Dockerfile, um arquivo de texto que contém todos os comandos que escrevemos manualmente quando criamos nossos ambientes, só que agora vamos utiliza-lo para construir uma imagem. O Docker cria as imagens lendo as instruções definidas nos arquivos Dockerfile.

Alguns comandos do Dockerfile, para mais detalhes podem ser consultados na documentação de Dockerfile.

```
FROM
MAINTAINER
ADD
COPY
ENV
EXPOSE
LABEL
USER
WORKDIR
VOLUME
STOPSIGNAL
ENTRYPOINT
```

Todo Dockerfile começa definindo qual é a imagem que vai ser utilizada como base.

Software para Contêiner : LXC / LXD e Docker
Docker

Dockerfile (build) sua imagem (zero)

```
root@host:~/onlybash# ls
bin  Dockerfile  lib  lib64

root@host:~/onlybash# cat Dockerfile
FROM scratch
ADD /bin /bin
ADD /lib /lib
ADD /lib64 /lib64
RUN /bin/bash

root@host:~/onlybash# docker build -t onlybash .
```

Software para Contêiner : LXC / LXD e Docker
Docker

Exemplo de um Dockerfile (baseado de uma imagem):

```
FROM node:5.2
RUN npm install bower -g
RUN npm cache clear
ADD ./docker-entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
CMD [ "bash" ]

# docker build -t image_name /home/william/caminhodockerfile/
```

Software para Contêiner : LXC / LXD e Docker
Docker

Network

Expondo portas na porta publica.

```
# docker run -i --expose=22 b5593e60c33b bash
# docker run -d -p 9080:80 --name http1 httpd
# docker run -d --net=host myvnc
```

Software para Contêiner : LXC / LXD e Docker
Docker

Network

Bridge Linux

```
root@virt-01:~# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242835ab07a       no              veth3f57c33
```

```
7: veth3f57c33@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
  docker0 state UP group default
    link/ether 22:f8:94:1f:67:48 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::20f8:94ff:fe1f:6748/64 scope link
       valid_lft forever preferred_lft forever
```

Software para Contêiner : LXC / LXD e Docker
Docker

Network

Criar uma nova rede (nettest)

```
# docker network create nettest
```

Iniciando um novo container Docker com a rede criada

```
# docker run --name meucontainerapp -d --network nettest httpd
```

Pegando o endereço IP do container Docker

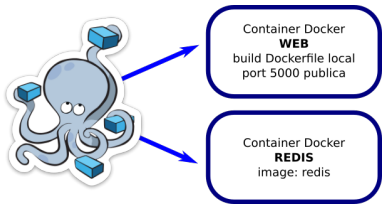
```
# docker inspect meucontainerapp | grep IPAdd
```

Software para Contêiner : LXC / LXD e Docker
Docker

Docker Composer

docker-compose.yml

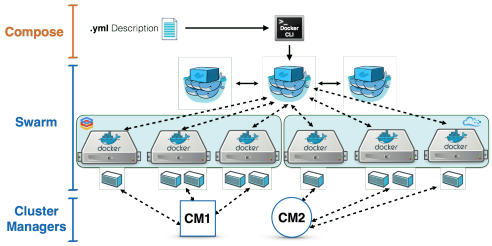
```
version: '3'
Services:

  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis

  redis:
    image: redis

volumes:
  Logvolume01: {}

# docker-compose up
```



https://docs.docker.com/compose/

Software para Contêiner : LXC / LXD e Docker
Docker

Ferramentas de Gerenciamento para Docker

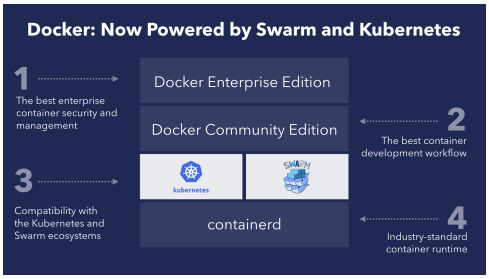


https://blog.docker.com/2015/11/deploy-manage-cluster-docker-swarm/

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

Software para Contêiner : LXC / LXD e Docker
Docker

Ferramentas de Gerenciamento para Docker



https://www.docker.com/kubernetes