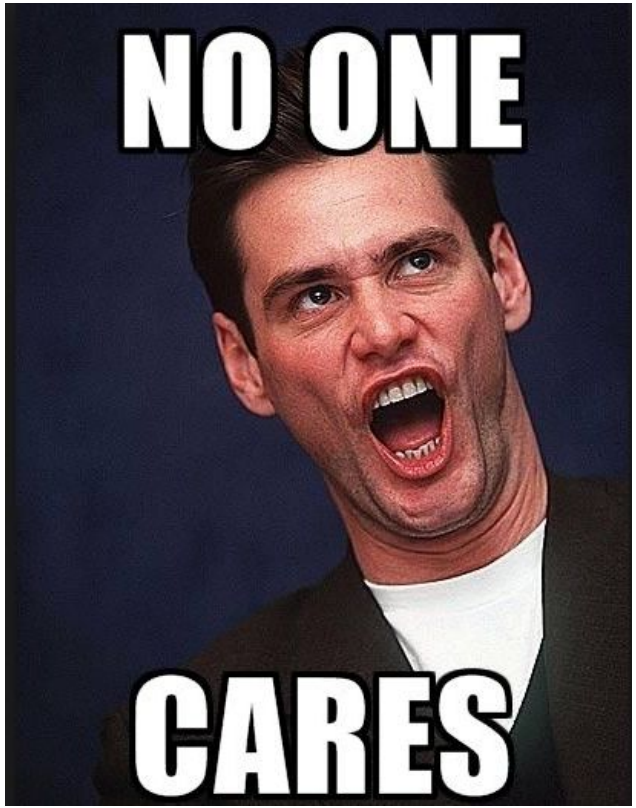
Docker: come se fosse ...


NO ONE
CARES

Linux Containers (LXC) is an operating-system-level virtualization method for running multiple isolated Linux systems (containers) on a single control host (LXC host).
It does not provide a virtual machine, but rather provides a virtual environment that has its own CPU, memory, block I/O, network, etc. space and the resource control mechanism

I Linux Container sono una virtualizzazione a livello di sistema operativo per fare funzionare diversi sistemi linux isolati su un singolo host.
Non produce una macchina virtuale ma un environment che è dotato di propria CPU, memoria, blocchi, rete ecc

https://wiki.archlinux.org/index.php/Linux_Containers

https://en.wikipedia.org/wiki/Docker_(software)

Docker (software)

From Wikipedia, the free encyclopedia

This article is about virtualization software. For the desktop icon docker, see Dock (computing). For company, see Docker, Inc. For other uses, see Docker (disambiguation).

Docker is a software technology providing containers, promoted by the company Docker, Inc.[6] Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux.[7] Docker uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others[8] to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines (VMs).[9]

Docker è una tecnologia che fornisce container, promossa da Docker inc. Fornisce un livello di astrazione e automazione a livello di sistema operativo, sia per Windows che per Linux.
Usa l'isolamento delle risorse tramite alcune feature del kernel linux, come i *cgroups*, i *namespaces* e un filesystem di unione, come *overlayfs*

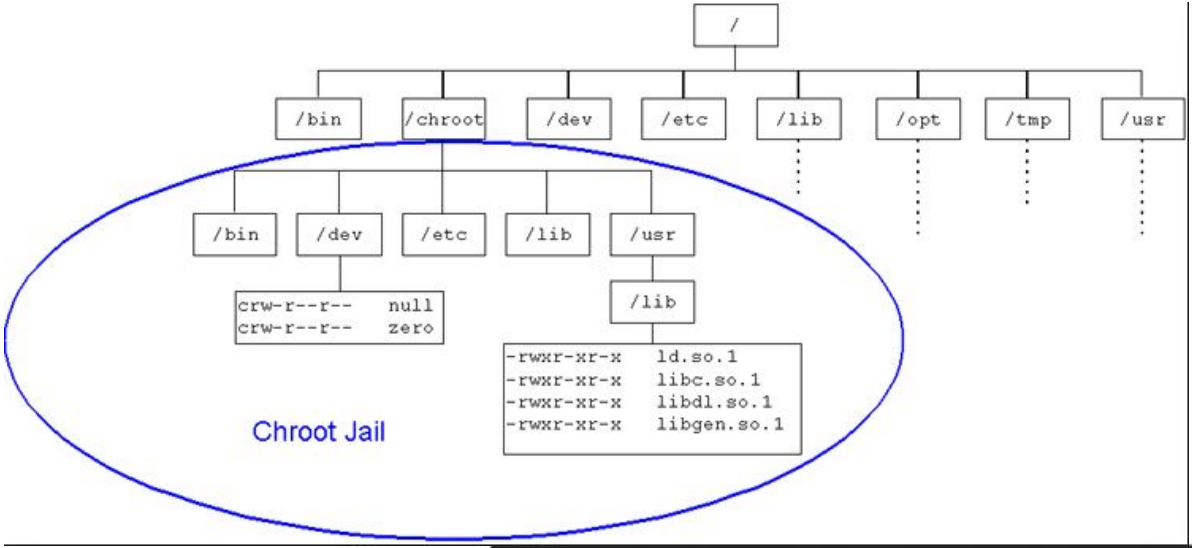
“chroot on steroids”

Docker is a Contract

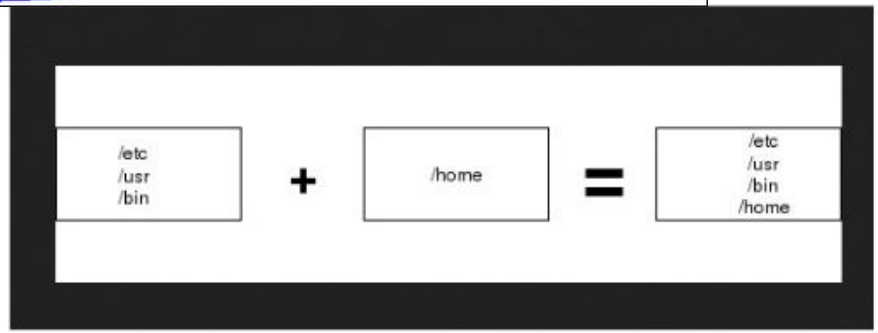
Docker is also the contract between Developers and Operations.

in brevissimo...

- chroot / lxc

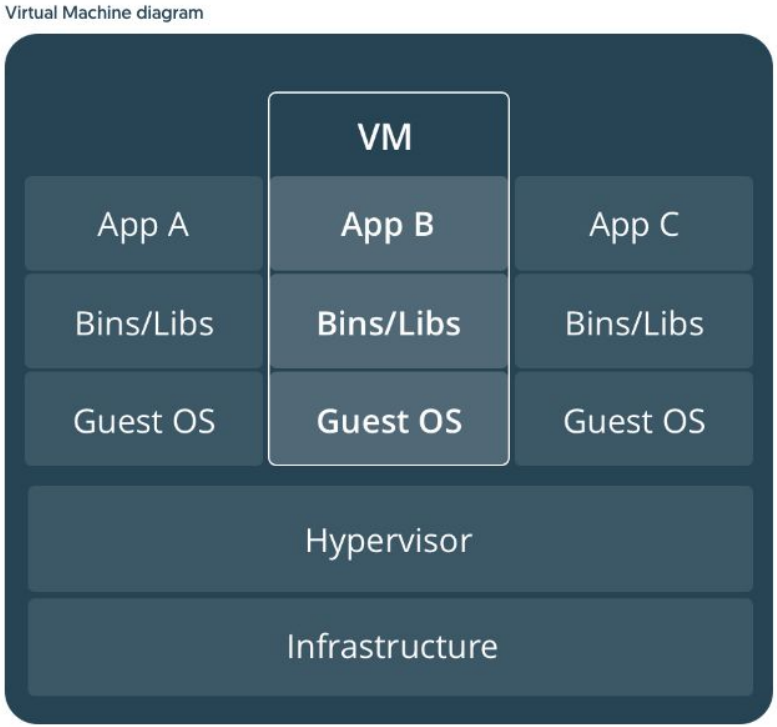


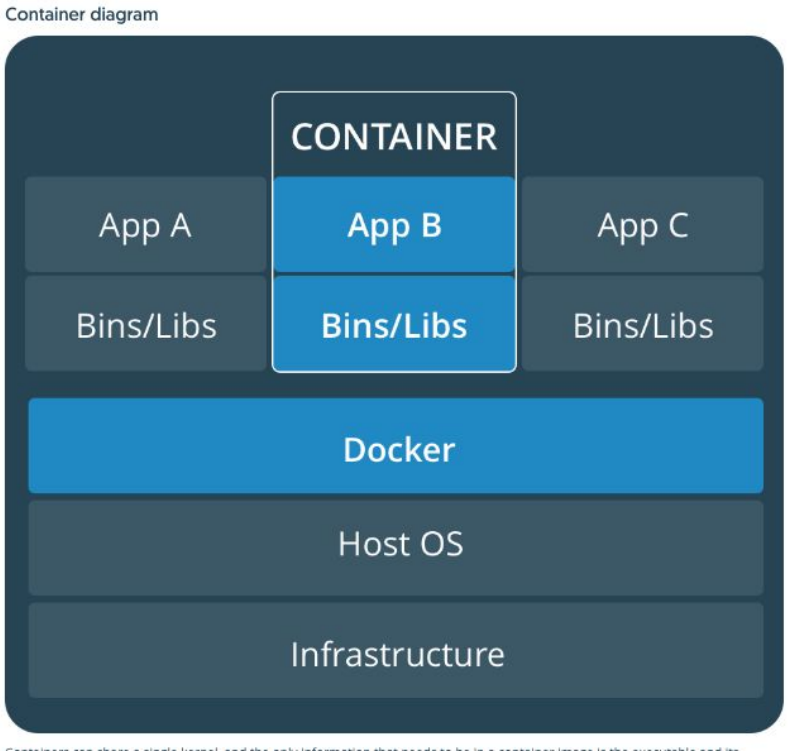
- file system di unione (aufs)
- demone e tool di gestione




STAY SAFE
KIDS
bettermeme.com

I container non sono VMs





https://docs.docker.com/get-started/

https://blog.docker.com/2016/03/containers-are-not-vms/

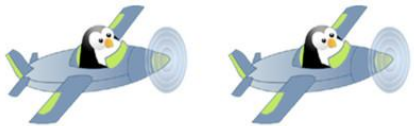
La risposta naturale delle persone quando iniziano a lavorare con Docker è tentare di definirlo in termini di macchina virtuale. Non riesco a contare il numero delle volte dove ho sentito descrivere i Docker container come “macchine virtuali leggere”

https://blog.docker.com/2016/03/containers-are-not-vms/

La risposta naturale delle persone quando iniziano a lavorare con Docker è tentare di definirlo in termini di macchina virtuale. Non riesco a contare il numero delle volte dove ho sentito descrivere i Docker container come “macchine virtuali leggere”

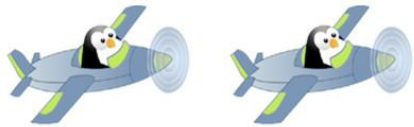
Per favore, stai lontano da:

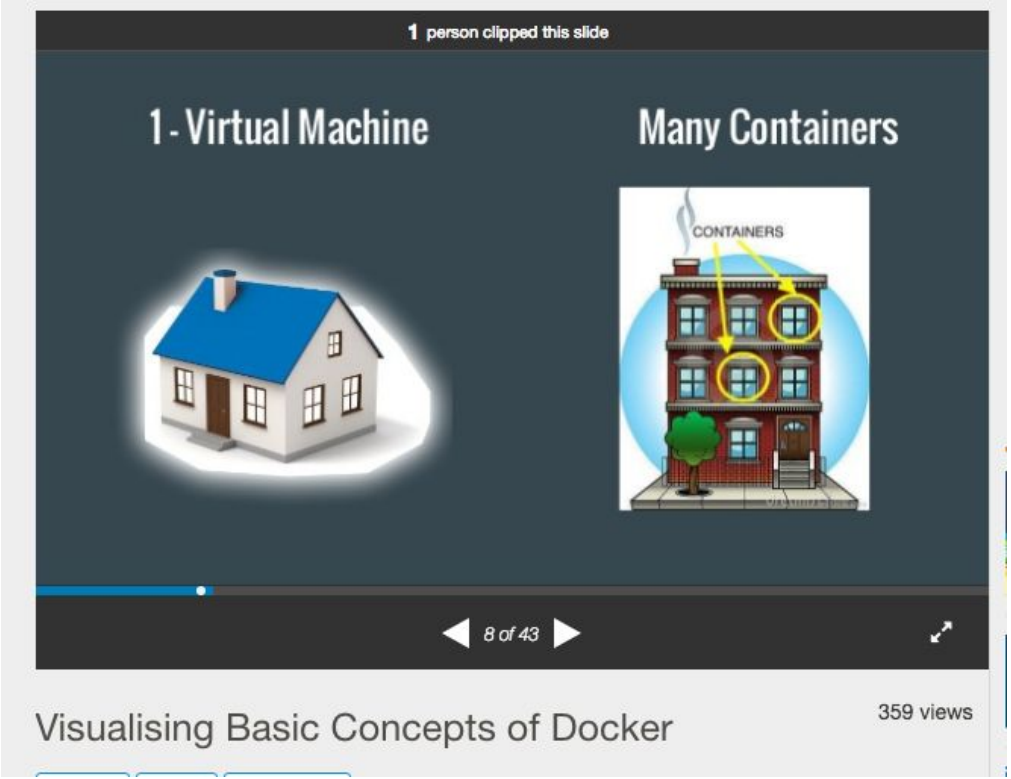
parte la “macchina docker”
“boota” il container
“reboot” del container
“parte” l’immagine
backup del container
“abilito il servizio” al container



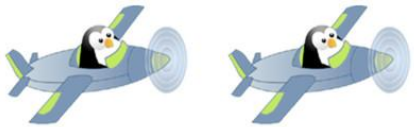
https://blog.docker.com/2016/03/containers-are-not-vms/

Gli appartamenti offrono anch’essi protezione dall’esterno, ma sono costruiti su una infrastruttura condivisa





Gli appartamenti offrono anch’essi protezione dall’esterno, ma sono costruiti su una infrastruttura condivisa



Giochiamoci un pò…

Some Docker vocabulary

Docker Image

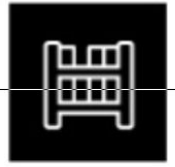
The basis of a Docker container. Represents a full application

Docker Container

The standard unit in which the application service resides and executes

Docker Engine

Creates, ships and runs Docker containers deployable on a physical or virtual, host locally, in a datacenter or cloud service provider



Registry Service (Docker Hub or Docker Trusted Registry)

Cloud or server based storage and distribution service for your images

Linux:

curl https://get.docker.com/ | bash

(non farti tentare da “apt-get install docker” o simili…)

docker

```
run

pull

stop

start

update

create

ps
```

docker

```
run
pull
stop
start
update
create
ps
```

```
kill
exec
attach
```

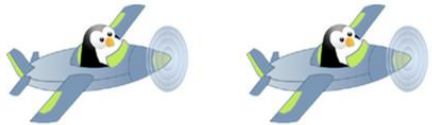
```
logfs
inspect
events
port
top
stats
diff
```

```
push
restart
pause
unpause
wait
```

```
$ docker pull nginx

$ docker run nginx nginx -v

$ docker run -n brillant -d nginx

$ docker start brillant

$ docker start e8da5d1efaca
```

Demo

```
$ docker pull nouphet/docker-php4

$ docker run -ti --rm nouphet/docker-php4 php --version

$ docker run -n brillant -v /tmp/host_dir/:/brillant nouphet/docker-php4 php /brillant/testme.php

$ docker start brillant

$ docker start e8da5d1efaca

$ docker run -e MYVAR=antani nginx bash -c 'echo $MYVAR'
```



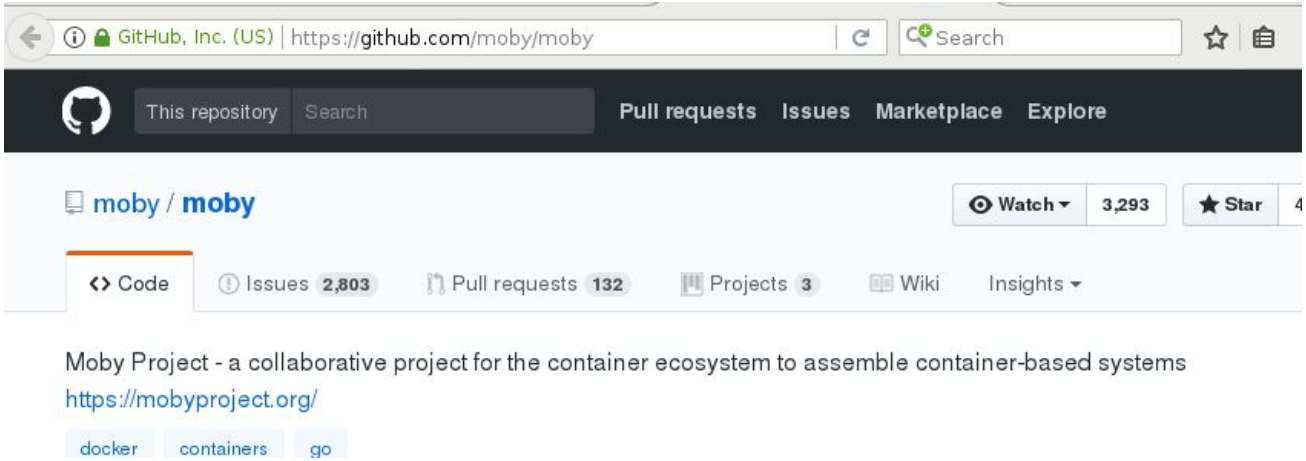
Vorrei provare l’ultima versione di redis

docker run -d -p 6969:6379 redis:latest

docker stop a1ea7fb0ec3c

docker start a1ea7fb0ec3c

Moby che?



Moby is comprised of:

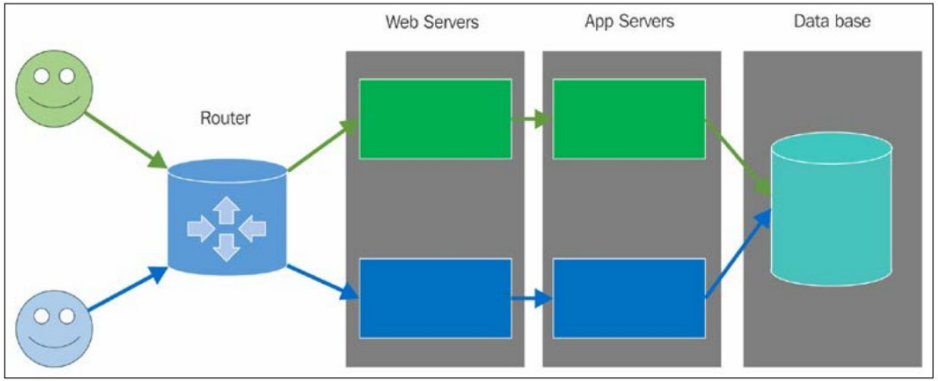
1. A **library** of containerized backend components (e.g., a low-level builder, logging facility, volume management, networking, image management, containerd, SwarmKit, ...)
2. A **framework** for assembling the components into a standalone container platform, and tooling to build, test and deploy artifacts for these assemblies.
3. A reference assembly, called **Moby Origin**, which is the open base for the Docker container platform, as well as examples of container systems using various components from the Moby library or from other projects.


WTF AM I READING
imgflip.com

Nel breve termine

- ambiente performante (docker su bare metal)
- continuous integration possibile con poche modifiche al codice
- processi facimente monitorabili
- Dockerfile / docker-compose condivisibili (e migliorabili dai dev)
- Non reinventare la ruota

Nel lungo termine



The Matrix of Hell

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

docker

Also a Matrix from Hell

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

container history

1979: Unix V7
2000: FreeBSD Jails
2001: Virtuozzo
2001: Linux VServer
2004: Oracle Solaris Containers
2005: Open VZ (Open Virtuzzo)
2006: Process Containers
2007: Control Groups merged into Linux kernel
2008: LXC
2011: Cloud Foundry Warden
2013: LMCTFY
2013: Docker
2014: Rocket
2016: Windows Containers


TINY TOOLS
TINY TOOLS EVERYWHERE
imgflip.com

docker-compose

```
docker run  --dns 192.168.1.5 -p 8080:80 -v /media/www:/var/www
namespace/apache_mod_php_dev  httpd -D FOREGROUND
```

```
version: '3.1'
services:
  apache:
    image: namespace/apache_mod_php_dev
    command: "httpd -D FOREGROUND"
    volumes:
      - "${APACHEVOLUME}"
    ports:
      - "${PROJECT_HTTP_PORT}"
    expose:
      - "${PROJECT_HTTP_PORT}"
    dns:
      - 192.168.1.5
    restart: always
    environment:
      - MAGE_RUN_CODE=${MAGE_RUN_CODE}
      - MAGE_RUN_TYPE=${MAGE_RUN_TYPE}
  memcached:
    image: memcached
    expose:
      - 11211
  rediscache:
    image: redis
    expose:
      - 6379
```

```
$ cat .env
PROJECT_HTTP_PORT=80
COMPOSE_PROJECT_NAME=cavalli
TESTDB=cavalli
TESTHOST=linuxdev02.namespace.local

APACHEVOLUME=../../:/var/www/html/
DOCKER_ENV=__development
```

```
version: "3.1"
services:
  apache:
    image: gcr.io/container-dev/apache_mod_php_dev
    command: "httpd -D FOREGROUND"
    volumes:
      - /media/www:/var/www/
    ports:
      - 8080:0
    dns:
      - 192.168.1.5
    restart: always
    environment:
      - VIRTUAL_HOST=${VIRTUAL_HOST}
      - VIRTUAL_UPSTREAM=${VIRTUAL_UPSTREAM}
    networks:
      - common_front
      - default
```

Commands:

build	Build or rebuild services
bundle	Generate a Docker bundle from the Compose file
config	Validate and view the Compose file
create	Create services
down	Stop and remove containers, networks, images, and volumes
events	Receive real time events from containers
exec	Execute a command in a running container
help	Get help on a command
images	List images
kill	Kill containers
logs	View output from containers
pause	Pause services
port	Print the public port for a port binding
ps	List containers
pull	Pull service images
push	Push service images
restart	Restart services
rm	Remove stopped containers
run	Run a one-off command
scale	Set number of containers for a service
start	Start services
stop	Stop services
top	Display the running processes
unpause	Unpause services
up	Create and start containers
version	Show the Docker-Compose version information

Commands:

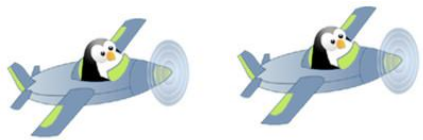
down	Stop and remove containers, networks, images, and volumes
exec	Execute a command in a running container
logs	View output from containers
pull	Pull service images
restart	Restart services
rm	Remove stopped containers
run	Run a one-off command
scale	Set number of containers for a service
start	Start services
stop	Stop services
up	Create and start containers
version	Show the Docker-Compose version information

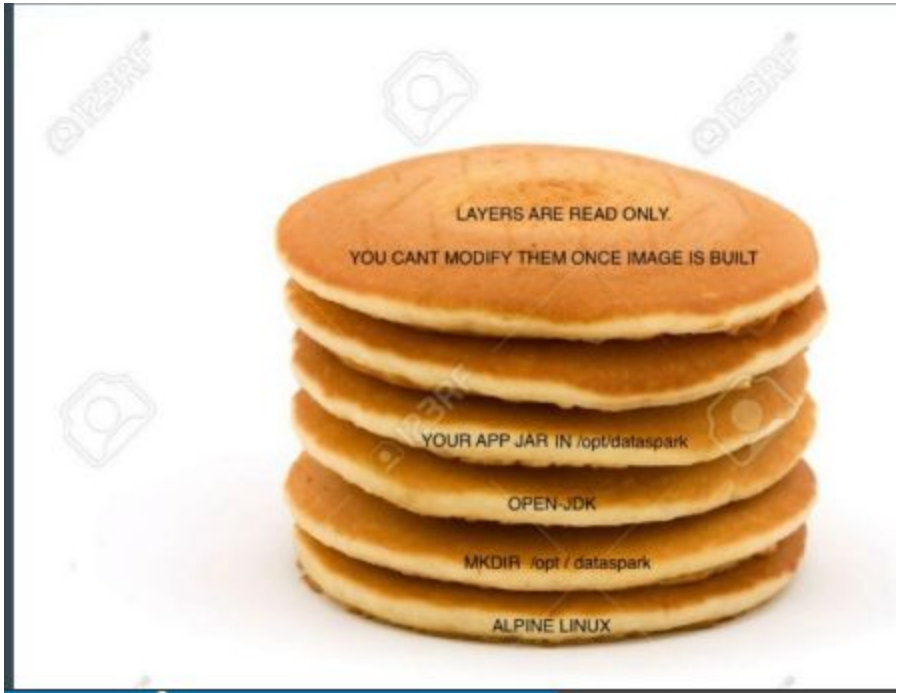
Dockerfile: esempio

https://bitbucket.org/johnnyrun/ml_toolchain_minimal

3e718d0 2014-10-05 Full commit | Blame | Embed | Raw | Edit

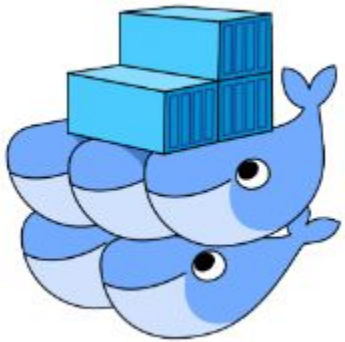
```
1  FROM resin/i386-ubuntu:14.04
2  MAINTAINER johnnyrun
3  ADD terry_guo-gcc-arm-embedded-trusty.list /etc/apt/sources.list.d/terry_guo-gcc-arm-embedded-trusty
4  RUN apt-get update
5  RUN apt-get -y install build-essential vim mercurial python zip python-docutils sudo
6  RUN apt-get -y --force-yes install gcc-arm-none-eabi make wget
7  RUN echo 'PS1="ML_build\[\033[01;34m\] \w \$\[\033[00m\]"' >> /etc/bash.bashrc
8  RUN mkdir /ml
9  RUN rm /etc/skel/.bashrc
10 RUN rm /root/.bashrc
11 RUN apt-get clean
```



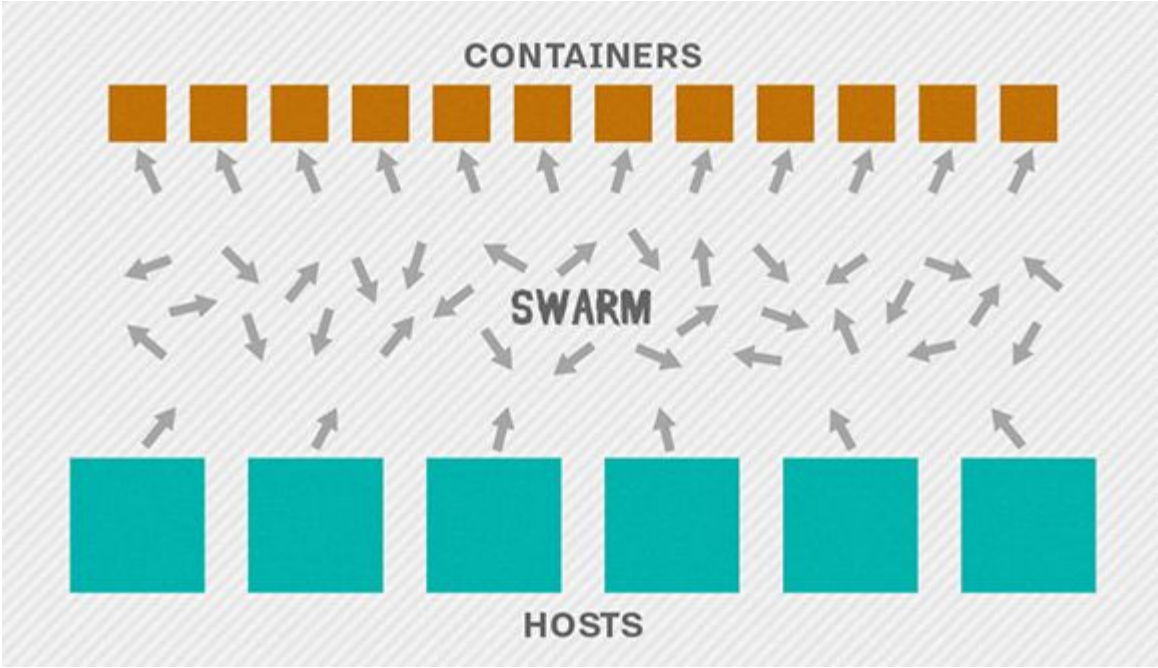


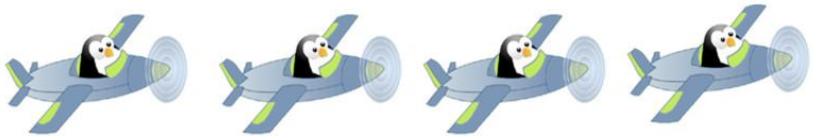
ualising Basic Concepts of Docker

swarm

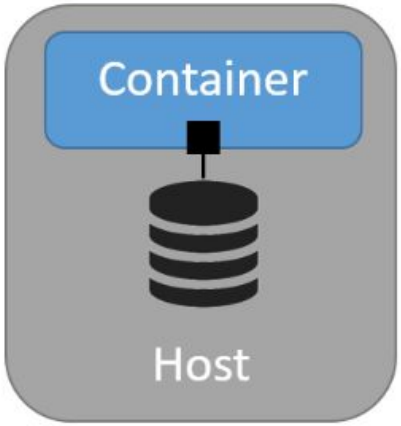


orchestration

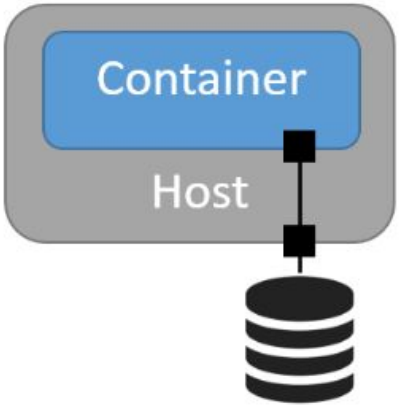

CONTAINERS
SWARM
HOSTS



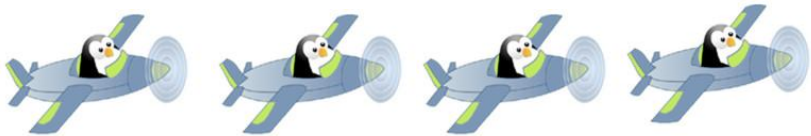
Data in the container
Lost when the container terminates

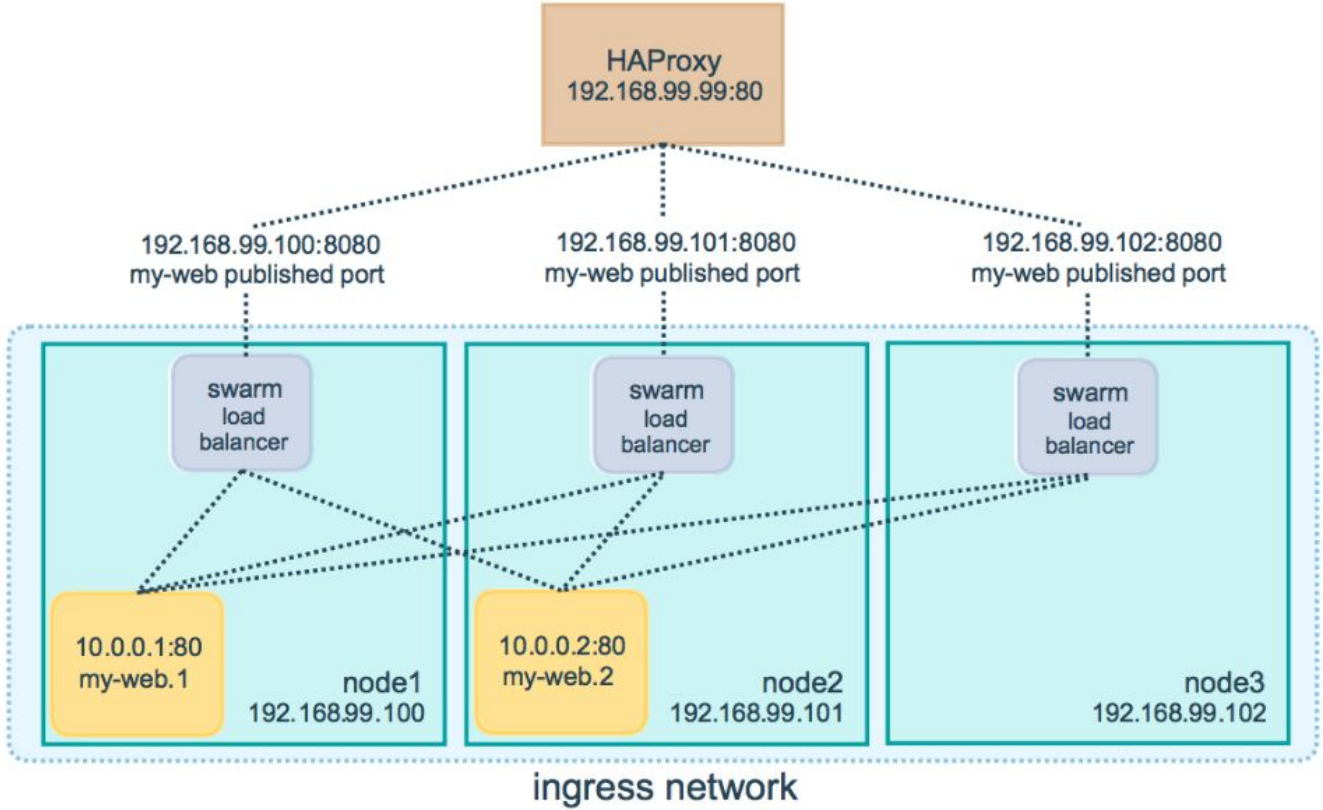


Data in a Host Volume
Lost when the host terminates

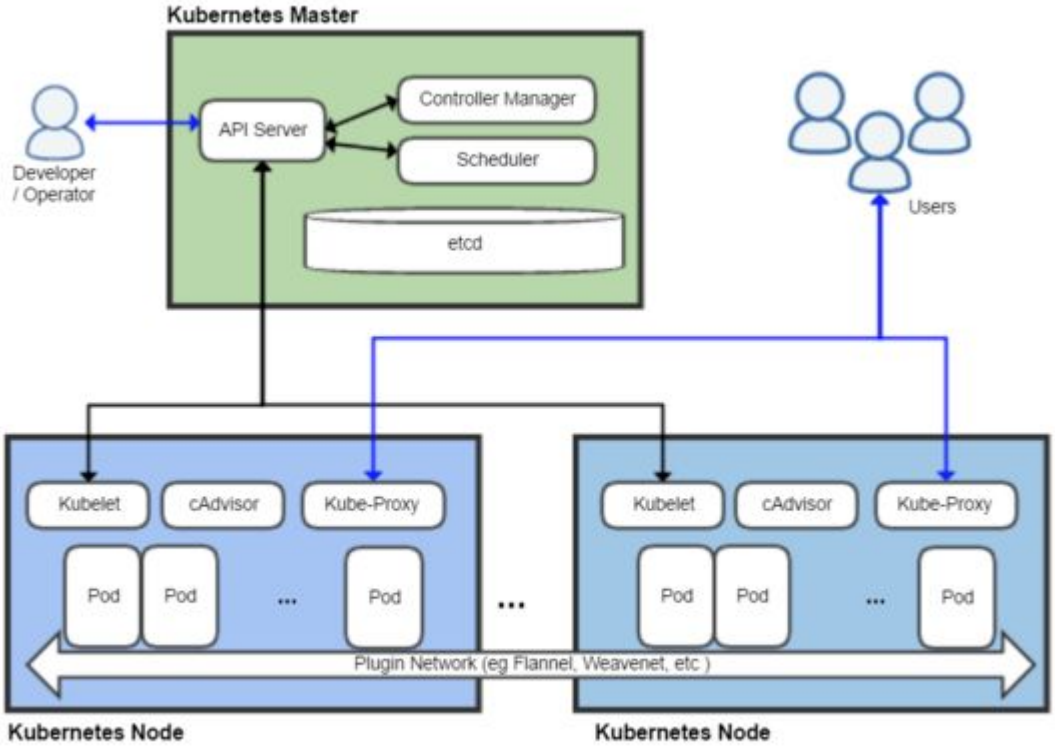


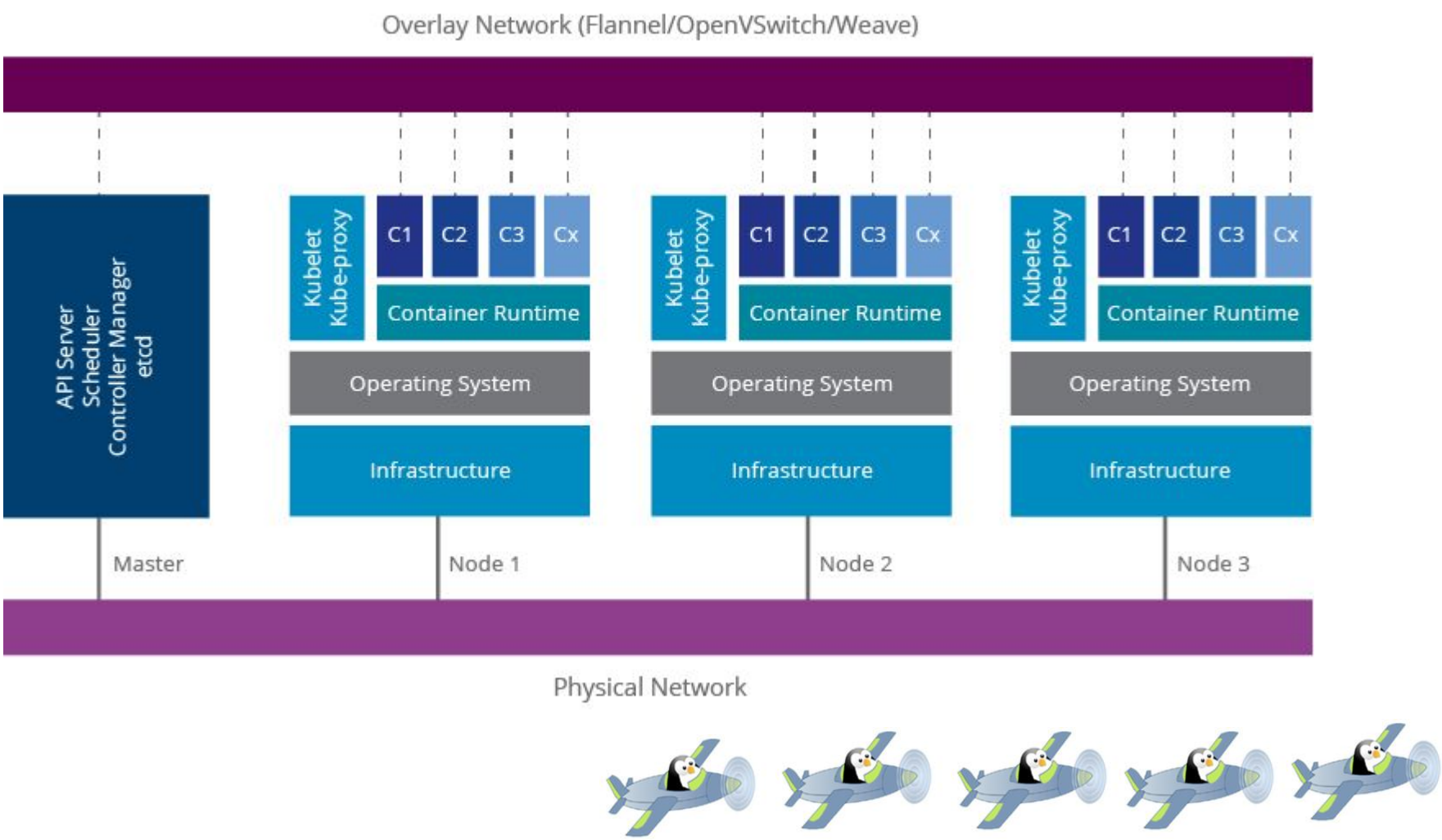
Networked Volume / File System
Independent of host and container




HAProxy
192.168.99.99:80
192.168.99.100:8080
my-web published port
192.168.99.101:8080
my-web published port
192.168.99.102:8080
my-web published port
swarm
load
balancer
swarm
load
balancer
swarm
load
balancer
10.0.0.1:80
my-web.1
node1
192.168.99.100
10.0.0.2:80
my-web.2
node2
192.168.99.101
node3
192.168.99.102
ingress network

kubernetes


Kubernetes Master
API Server
Controller Manager
Scheduler
etcd
Developer / Operator
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node


Overlay Network (Flannel/OpenVSwitch/Weave)
API Server
Scheduler
Controller Manager
etcd
Kubelet
Kube-proxy
C1
C2
C3
Cx
Container Runtime
Operating System
Infrastructure
Master
Node 1
Node 2
Node 3
Physical Network

$ more docker.txt

- Docker up and running*
- Pro Docker
- The Docker for DevOps course (udemy)