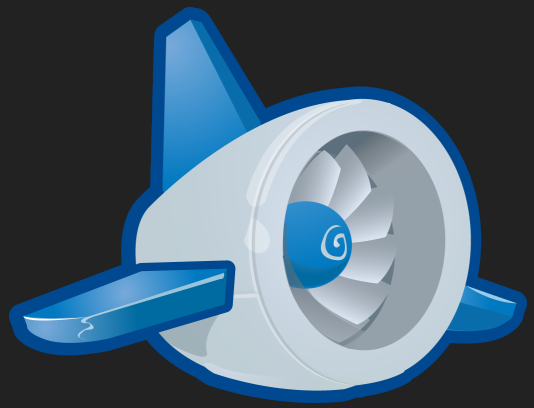
# ORIXES

## Primeiros PaaS

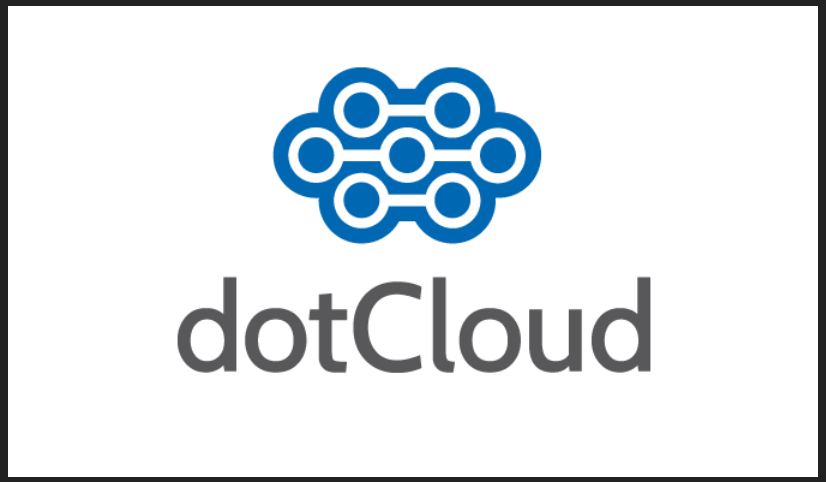
heroku



af

appfog

# ORIXES




Marzo 2013 - Docker v0.1 GitHub

Salomon Hykes

# 12 factors

## I. Base de código (Codebase)

Unha única base de código da que levar o control de versións e múltiples implantacións

## II. Dependencias

Declarar e aislar explícitamente as dependencias

## III. Configuración

Gardar a configuración no entorno

## IV. Backing services

Tratar os servicios que soportan a aplicación ("backing services") como recursos conectables

## V. Construir, distribuir, executar

Separar completamente a etapa de construcción da etapa de execución

## VI. Procesos

Executar a aplicación como un ou máis procesos sen estado

# 12 factors

## VII. Asignación de portos

Exportar servicios mediante portos de rede

## VIII. Concurrencia

Escalar mediante o modelo de procesos

## IX. Dispoñibilidade

Maximizar a robustez con inicios rápidos e a detencións elegantes

## X. Equiparar dev/prod

Mantener desenrolo, preprodución e producción tan semellantes como sexa posible

## XI. Logs

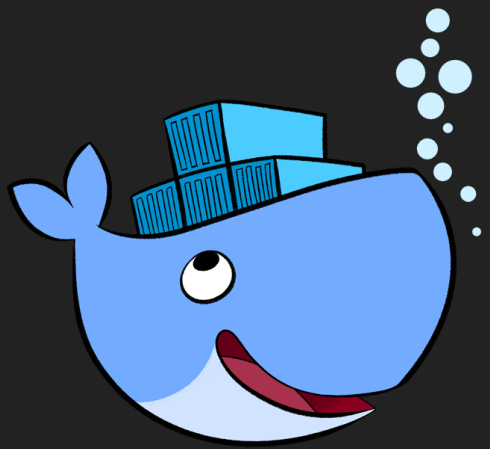
Tratar os logs como un fluxo de eventos

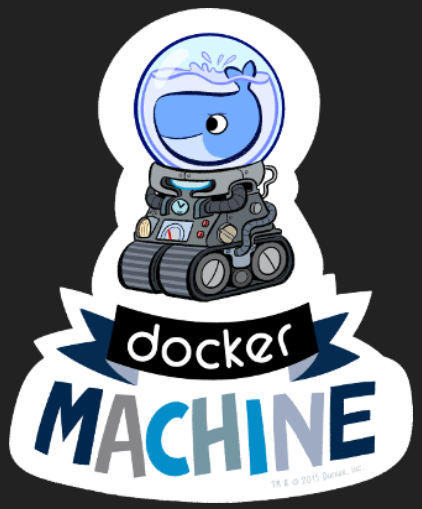
## XII. Administración de procesos

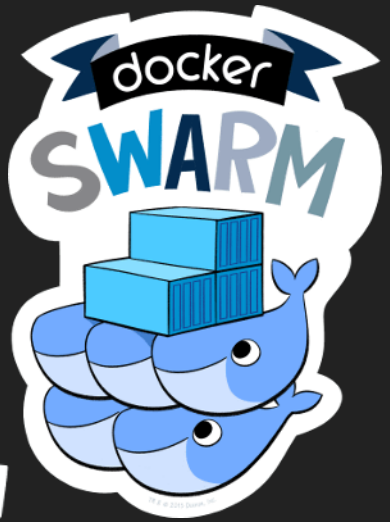
Correr as tarefas de xestión/administración como procesos de unha única execución

# Docker Toolbox

ENGINE

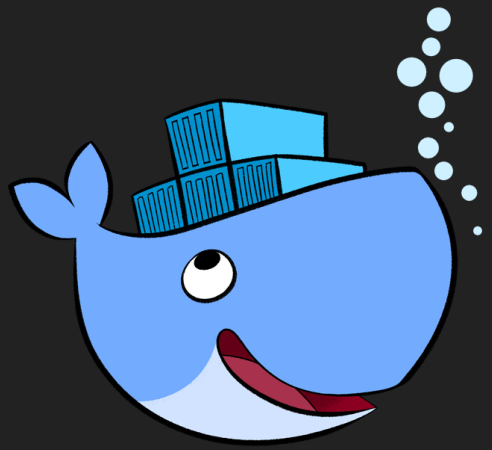





DISTRIBUTION

# Docker Engine



# Docker Engine

Ferramenta base de todo el sistema

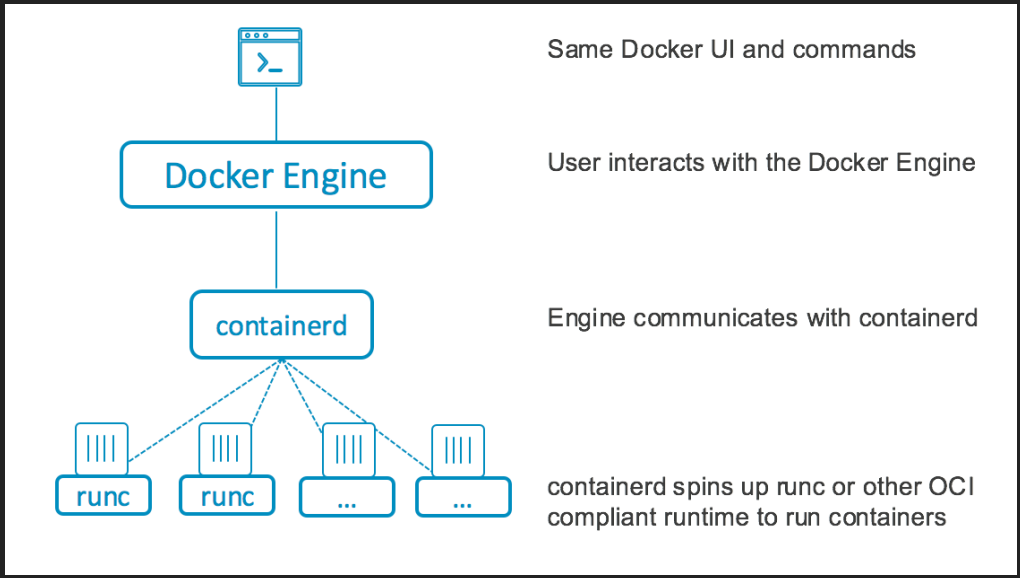
Conten tanto o cliente como o demonio

Crea os namespaces, descarga as imaxes, establece os cgroups para controlar os recursos que consume cada contaiener

Exportar unha api REST ca que se pode comunicar calquer cliente para remitirlle ordenes

# Docker Engine

Nova versión




Desde v1.11:

- engine
- containerd
- runC ou outro runtime OCI (rkt)

https://blog.docker.com/2016/04/docker-engine-1-11-runc/

# Entorno de probas

- Rexistrarse en https://panel.prefapp.es
- Activar a promo
- Generar key ssh

```
mkdir /tmp/ssh_keys && ssh-keygen -f /tmp/ssh_keys/chave -b 2048
```

- Crear unha app de **DOCKER_DEV** e configurar a parte pública da key ( **chave.pub**) no formulario de creación.
- Desplegala no servidor que se activou ca promo
- Conectarse ao server:

```
ssh -i /tmp/chave -p 2222 cargoXXX-prefapp-in.cloudapp.net
```

# Comandos básicos

https://docs.docker.com/engine/reference/run/

```
# Hello World
docker run hello-world

# ver os containers creados
docker ps -a

# ver as imaxes locales
docker images

# arrancar un container interactivo (unha ubuntu cunha shell)
docker run -ti ubuntu:14.04 bash


# Arrancar un container en segundo plano
docker run -d ubuntu:14.04 \
/bin/sh -c "while true; do echo hello world; sleep 1; done"
af4255c7574ec51b7d34654575e
```

# Comandos básicos

```
# ver o stdout/stderr do container
docker logs af4255c7574ec51b7d346545

# paralo, arrancalo de novo, matalo, borralo
stop, start, kill, rm


# arrancar un servidor web que escoite no porto 80
# e nos permita movernos por un directorio compartido

docker run -ti -d -v /home:/home -w /home \
-p 80:8000 ubuntu:14.04 python3 -m http.server

# paralo e borralo
docker stop <id_container>
```

# Comandos básicos

```
# inspeccionar un container

docker inspect <nome>

# executar un comando dentro dun container arrancado

docker run --name container -ti ubuntu:14.04 bash

docker exec -ti --name container /bin

# ver as layers dunha imaxen

docker history 4eixos/base
```

Ver gráficamente as layers dun imaxen:

https://imagelayers.io/?images=redis:latest

# Despregar unha aplicación basica

- Crear unha app chorras, en /home/code,  por ejemplo: index.php:

```
<?php

$nome = $_GET['nome'];
echo "<h1> Hello $nome </h1>";
```

- Servila co server interno de php

```
docker run -ti -p 8080:8080 \
-v $PWD:/var/www/html \
php:5-fpm php -S 0.0.0.0:8080
```

# Dockerfile e docker build

Construir unha imaxen que leve a nosa app

```
# Dockerfile

FROM php:5-fpm
COPY . /var/html/html
EXPOSE 8080
CMD ["php","-S", "0.0.0.0:8080"]
```

```
docker build . -t <nome_imaxen>
```

https://docs.docker.com/engine/reference/builder/

```
# subir a imaxen a un registry

# 1) tageala
docker tag <id_imagen> <registry>/<nome>

# 2) logearse no registry
docker login <registry>

# 3) subila
docker push <registry>/<nome>

# 4) descargar a do compañeiro
docker pull <registry>/<nome_imaxen_compi>

# 5) arrancar a imaxen do compañeiro e probala
docker run -ti <registry>/<nome_imaxen_compi>
```

# Docker compose

# Compose file reference

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

```
# Por exemplo, un wordpress

wordpress:
  image: wordpress
  links:
    - db:mysql
  ports:
    - 8080:80

db:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: example
```

# Comandos básicos

docker-compose -f <file_yml> up -d

docker-compose -f <file_yml> down

docker-compose -f <file_yml> run <cmd>

docker-compose -f <file_yml> start/stop

# Ej: Etherpad lite: http://https://hub.docker.com/r/tvelocity/etherpad-lite/

```
ep_mysql:
  image: mysql:5.6
  container_name: ep_mysql
  environment:
    MYSQL_ROOT_PASSWORD: passw0rd

etherpad:
  image: tvelocity/etherpad-lite
  container_name: etherpad
  entrypoint:
    - bash
    - -c
    - "sleep 20 && /entrypoint.sh && bin/run.sh --root"
  links:
    - ep_mysql:mysql
  ports:
    - "80:9001"
  environment:
    - ETHERPAD_TITLE=Etherpad
    - ETHERPAD_ADMIN_USER=admin
    - ETHERPAD_ADMIN_PASSWORD=admin
    - ETHERPAD_DB_PASSWORD=passw0rd
    - ETHERPAD_DB_USER=root
```