Docker - Build, Ship and Run Any App, Anywhere.

Federico Naldini

Alma Mater Studiorum - Università di Bologna, Cesena.

federico.naldini3@studio.unibo.it

30/05/2018

Overview

1. La piattaforma Docker
   - Introduzione ai *containers*
   - Docker: le basi
   - Architettura del sistema Docker
   - Costruzione di un *container* Docker
2. Live demo

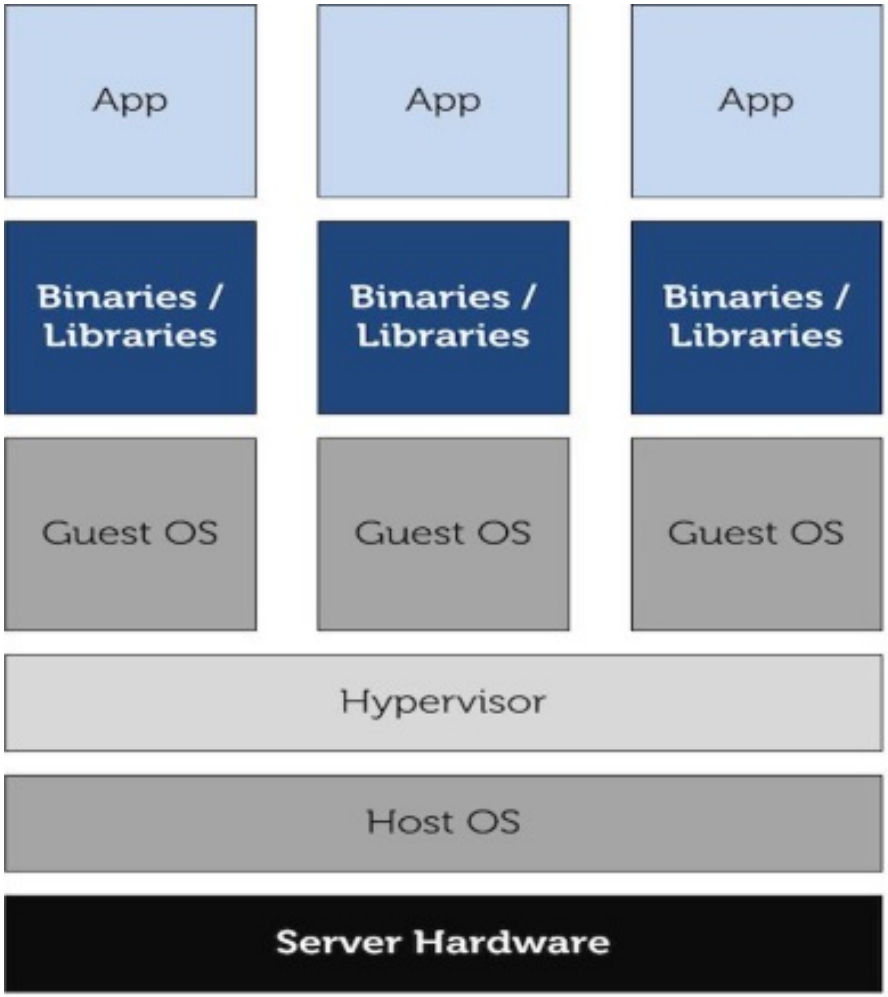
Che cos'è un *container*?

Un *container* è una forma di virtualizzazione a livello di sistema operativo, in alternativa alle classiche macchine virtuali(*VM*).
A differenza delle *VM* , l'idea dietro all'approccio a *containers* è di virtualizzare solo i componenti necessari, condividendo le restanti risorse con altri *containers*, *VM* e sistemi operativi fisici.

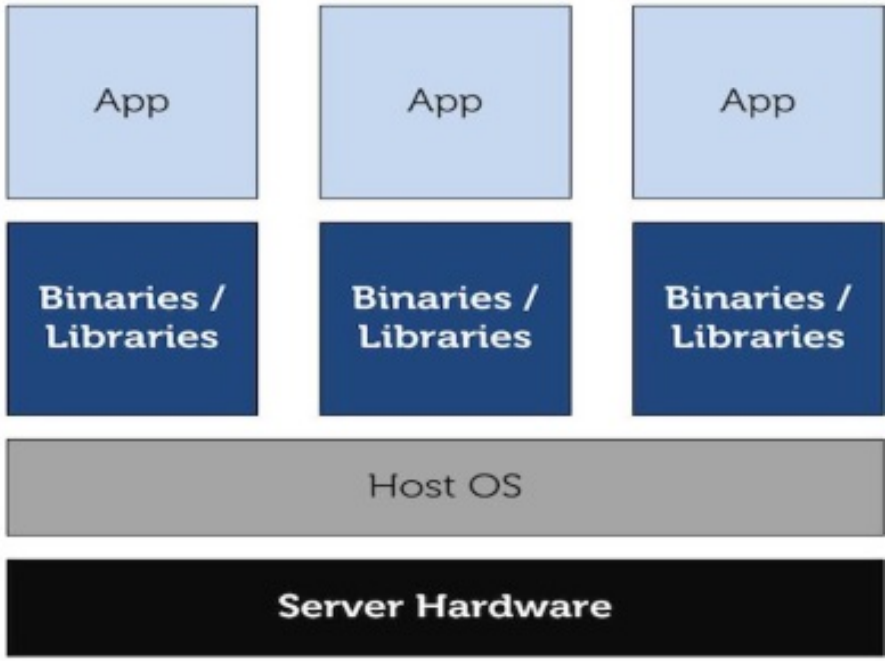
Vantaggi dell'approccio a *containers*

- Buon supporto alla scalabilità.
- Isolamento tra i vari *containers*,anche se in esecuzione sullo stesso insieme di risorse(utilizzando feature come *Namespaces*, presente nel Kernel Linux).
- Controllo rigoroso sull'utilizzo delle risorse fisiche(sfruttando *CGroups*, altra feature del Kernel Linux)
- Overhead di virtualizzazione minimo, grazie alla riduzione del numero di layer necessari.

Layers di virtualizzazione necessari



Virtualization



Containers

Docker, che cos'è?

Il progetto open source Docker viene rilasciato nel 2013 da una compagnia chiamata dotCloud, che lavorava su software di tipo PAAS (Platform as a service) per il cloud computing.
Docker si pone come obbiettivo di automatizzare lo sviluppo di applicazioni all'interno di containers software, sfruttando tutti i vantaggi di una virtualizzazione a livello di sistema operativo; per riuscire a fare ciò, si avvale delle funzionalità di isolamento presenti nel kernel Linux, come ad esempio cgroups, utilizzato per la gestione di risorse fisiche, e namespaces, che invece garantiscono un isolamento a livello di processo.

I vantaggi della piattaforma Docker

- Open Source.
- Ogni applicazione è eseguita in una *Sandbox*
- Applicazioni leggere da eseguire.
- Ogni applicazione è eseguita in una *Sandbox*
- Ogni *container* è isolato da ogni altro sistema, salvo diverse istruzioni.
- Supporto diretto al *Clustering*, utilizzando un modulo apposito chiamato `Swarm Module`

Chi utlizza Docker? E come?

Gli svantaggi della piattaforma Docker

- Vincolato al kernel Linux.
- Supporto limitato allo storage permanente delle modifiche.
- Open Source.

I moduli di Docker

Lo scheletro dell'Architettura di Docker si compone di diversi moduli, ciascuno con il suo compito ben definito:

- **Docker Engine**: Cuore del sistema Docker, si occupa di mandare in esecuzione i singoli container.
- **Docker Machine**: Modulo utilizzato per creare macchine virtuali minimali per portare Docker su sistemi senza kernel Linux(Windows, MacOs), oggi soppiantato da applicazione native per i sistemi.
- **Docker Swarm**: si occupa della gestione della *Docker Swarm Mode*, ovvero una modalità per gestire cluster di *containers* Docker.

Docker Engine

Cuore della piattaforma è il modulo *Docker Engine*, composto da tre sottomoduli:

Docker Client

Fornisce un insieme di comandi all'utente per poter istruire il *Docker Daemon* sul da farsi, utilizzando `API RESTful` proprietarie.

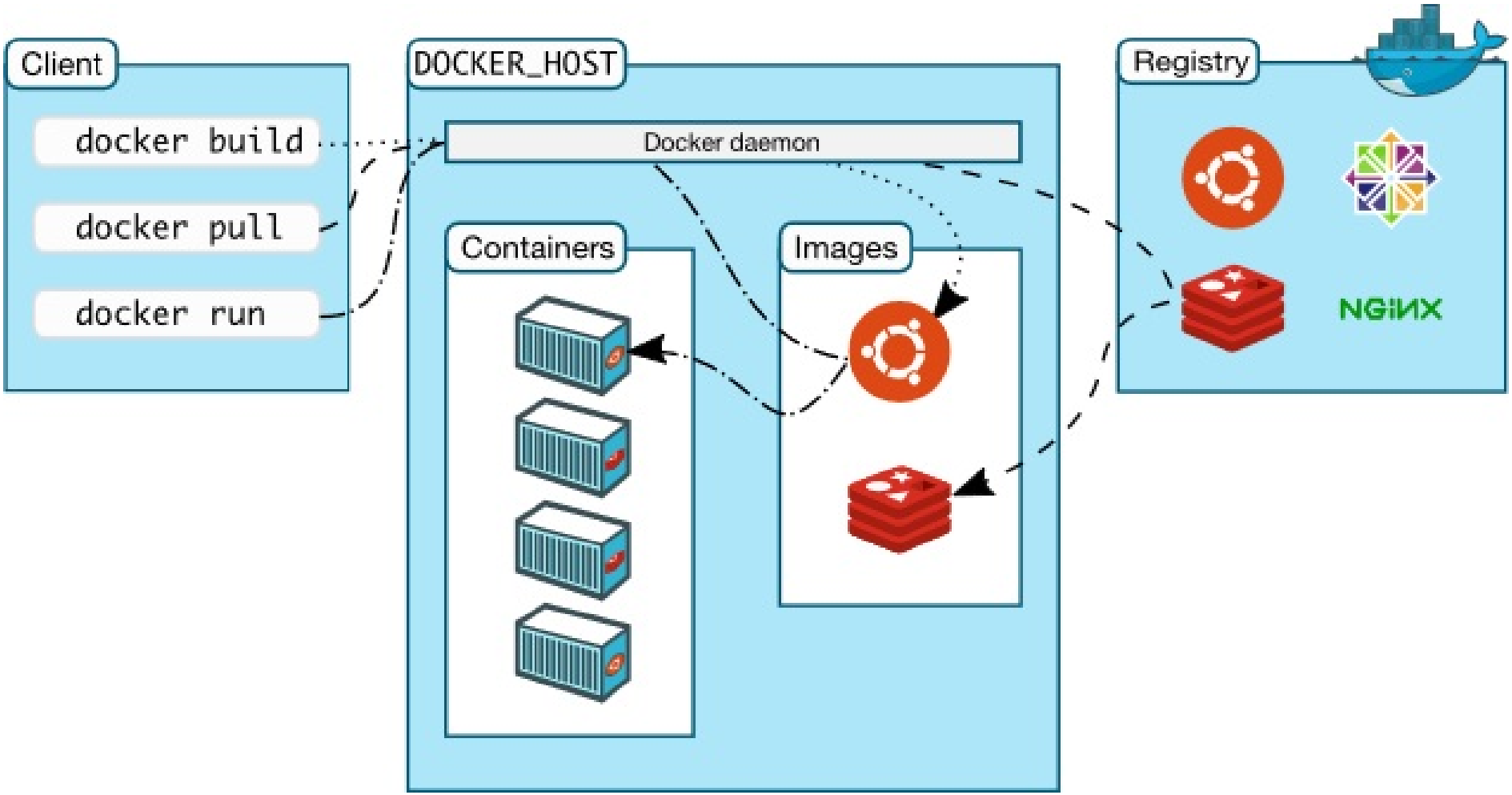
Docker Daemon

Un processo demone eseguito in background su un host locale o remoto, si occupa della maggior parte del lavoro, gestendo sotto comando tutti gli oggetti del sistema.

Docker Registry

Registro pubblico o privato, a cui *Docker Daemon* accede per recuperare le immagini necessarie al funzionamento dei container

Docker Engine



Creazione di un *container*

Un *container* deve essere sempre generato da una `Docker Image`.
Le *best practices* di Docker associano ogni immagine a un `Dockerfile`, un file testuale che contiene le istruzioni da eseguire per generare l'immagine

Dockerfile

Ogni `Dockerfile` è strutturato in tre sezioni

FROM

Specifica l'immagine da utilizzare come base.

RUN

Qui vengono indicati i comandi da eseguire sull'immagine base per ottenere il risultato desiderato.

CMD

Si piò specificare una singola istruzione, tale comando sarà eseguito all'avvio del *container*

Dockerfile:un esempio

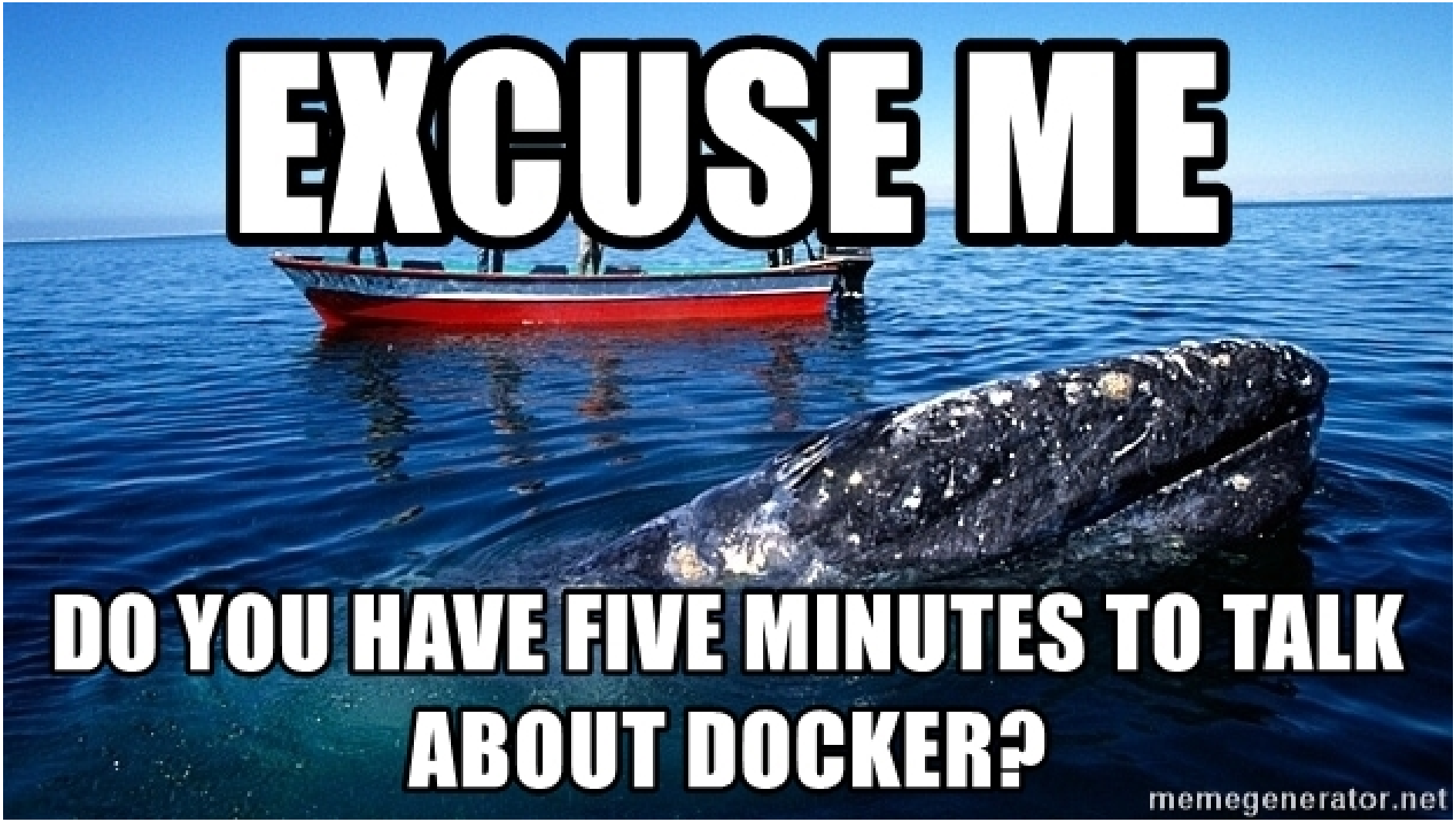
Dockerfile

```
FROM library/openjdk:latest

EXPOSE 8080
RUN git clone
 https://gitlab.com/das-lab/lpaas/lpaas-ws.git &&\
cd lpaas-ws &&\
chmod u+x gradlew &&\
printf '#!/bin/bash\n' >> script.sh &&\
echo "cd /lpaas-ws\n ./gradlew run task" >> script.sh &&\
chmod u+x script.sh &&\
./gradlew build

CMD ["/lpaas-ws/script.sh"]
```

Live Demo


EXCUSE ME
DO YOU HAVE FIVE MINUTES TO TALK ABOUT DOCKER?
memegenerator.net

Primi passi

Installazione della piattaforma(per tutti gli OS)

https://docs.docker.com/install/

Verifica della corretta installazione(da eseguire su una shell)

```
docker run hello-world
```

Una semplice bash

Ubuntu bash

```
docker run -ti ubuntu
```

NodeJS

Un semplice container che manda in esecuzione un'app

```
docker run -p 3000:3000 tirocigno/nodejssampleapp
```

Riferimenti

- Sito ufficiale Docker: https://www.docker.com/
- Architettura di Docker: https://docs.docker.com/engine/docker-overview/

Docker - Build, Ship and Run Any App, Anywhere.

Federico Naldini

Alma Mater Studiorum - Università di Bologna, Cesena.

federico.naldini3@studio.unibo.it

30/05/2018