Tutorato Progettazione, Algoritmi e Computabilità

Docker

Dott. Michele Beretta

Prof.ssa Patrizia Scandurra

Università degli Studi di Bergamo

2023 – 2024

Outline

Container

Container – perché

- Popolari
- Maggiore tecnologia dei servizi cloud più grandi
- Tecnologie popolari offrono anche dei container
- Configurazioni diverse in fase di sviluppo e produzione
- "*It works on my machine*"
- Deployment veloce

Container – definizione

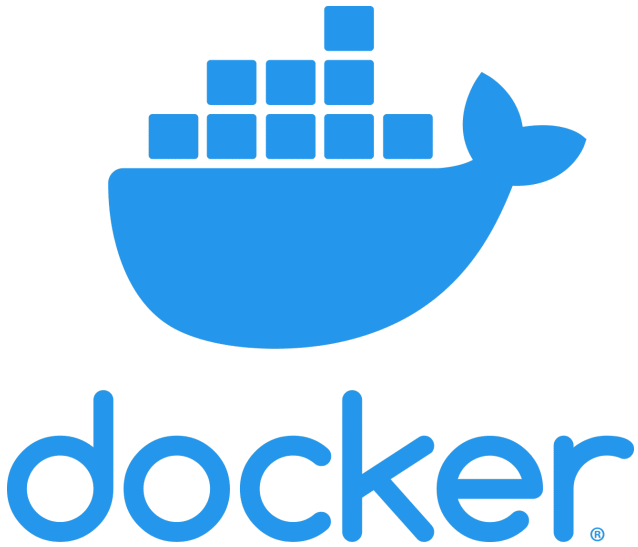
Un container è un *pacchetto software* che contiene tutti gli elementi necessari alla sua esecuzione.

Sono creati tramite la virtualizzazione dal sistema operativo. Sono essenzialmente dei "semplici processi" con una visibilità limitata di quello che è il sistema host.

Più info approfondite qui.

Docker

Docker



Docker è una tecnologia che permette di definire dei container in modo semplice.

Docker – quando usarlo

- Per standardizzare gli ambienti di sviluppo: distribuendo un dockerfile tutti gli sviluppatori hanno la certezza di lavorare esattamente sullo stesso ambiente evitando situazioni di incompatibilità
- Per utilizzare simultaneamente versioni differenti della stessa libreria/linguaggio
- Per isolare le applicazioni e le risorse (sicurezza dei dati, ambiente più pulito)

Docker – Dockerfile

Docker crea automaticamente immagini basandosi sulle istruzioni inserite all'interno di un dockerfile.

```
# syntax=docker/dockerfile:1

FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

Docker – Dockerfile

Comandi principali:

- Indicazione dell'immagine parent da cui creare il docker (`FROM`)
- Directory principale da cui eseguire i comandi (`WORKDIR`)
- Copia di file locali ad immagine docker (`COPY`)
- Esecuzione di comandi (`RUN`)
- Comando principale da eseguire (`CMD`)
- Esposizione di porte (`EXPOSE`)
- Argomenti da passare (`ARG`)
- Variabili d'ambiente (`ENV`)

Docker compose

Docker Compose

Nell'ambito di architetture basate su servizi sta sempre più prendendo piede il *Compose file*:

- Yaml (docker-compose.yml)
- Definisce servizi software, di rete e volumi (file, directory, etc.) che devono essere utilizzati da un container Docker
- Permette di specificare anche impostazioni hardware:
  - Numero/Percentuale di CPU che ogni container deve usare
  - Specifica CPU su cui lanciare il container
  - Parametri di allocazione della CPU, e.g., come periodo di esecuzione per sistemi real time

Docker Compose – Esempio guidato (1)

Vogliamo realizzare un semplice container Docker, tramite Docker Compose, che esegue un semplice programma Java[1](e.g., somma tra numeri).

Preparazione:

1. Installare Docker
2. Procurarsi il JAR del programma
3. Creare una cartella con dentro un file `docker-compose.yml`

[1] Per altri linguaggi è simile

Docker Compose – Esempio Guidato (2)

Scrivere il contenuto di `docker-compose.yml`:

```
version: "2"
services:
  sommatore:
    image: openjdk:14
    container_name: "sommatore"
    volumes:
      - ./sommatore.jar:/tmp/sommatore.jar
    working_dir: /tmp
    command: ["java", "-jar", "sommatore.jar"]
```

Docker Compose – Esempio Guidato (3)

Facciamo partire il container:

```
$ docker-compose up
```

Se il comando dà un errore simile a “no matching manifest ...”, significa che l’immagine che state caricando nel docker (in questo caso openjdk) non è disponibile sul vostro PC. Potete risolvere scaricandola con il comando:

```
$ docker pull -platform linux/arm64/v8 openjdk:14
```

Docker Compose – Esempio Guidato (4)

A questo punto è possibile vedere a terminale l'output del programma.

Infine, si può rimuovere il container non più usato

```
$ docker rm /sommatore
```

Docker Compose – Esercizio

Modificare l'esempio appena fatto per fare in modo che il sommatore riceva i numeri in input leggendoli da un file `numeri.txt`, il cui contenuto può essere variato ad ogni esecuzione dall'utente.