DE: Department of Engineering Ferrara

Ingegneria del Software Avanzata A.A. 2021/2022

Docker

Damiano Azzolini
damiano.azzolini@unife.it

Laurea Magistrale in Ingegneria Informatica e dell'Automazione - Università di Ferrara

Introduzione

Virtualizzazione

Le **macchine virtuali (VM)** sono un'astrazione di una macchina fisica.

Le macchine virtuali sono gestite da uno strato software chiamato **hypervisor**.

Ogni VM è caratterizzata da un sistema operativo ospite che viene eseguito al suo interno.

Una macchina virtuale vede come risorse hardware solo quelle che le sono state allocate dall'hypervisor.

Le macchine virtuali sono nate per controllare l'allocazione delle risorse.

Container I

Svantaggi delle macchine virtuali:

- Ogni macchina virtuale contiene un'immagine di diversi GB del sistema operativo ospite
- Overhead di spazio di archiviazione, per ogni macchina virtuale viene allocato uno spazio di archiviazione anche se non viene utilizzato
- Tempi di avvio di una macchina virtuale elevati

Perché virtualizzare un'intera macchina quando ci interessa virtualizzare una piccola parte di essa?

Container II

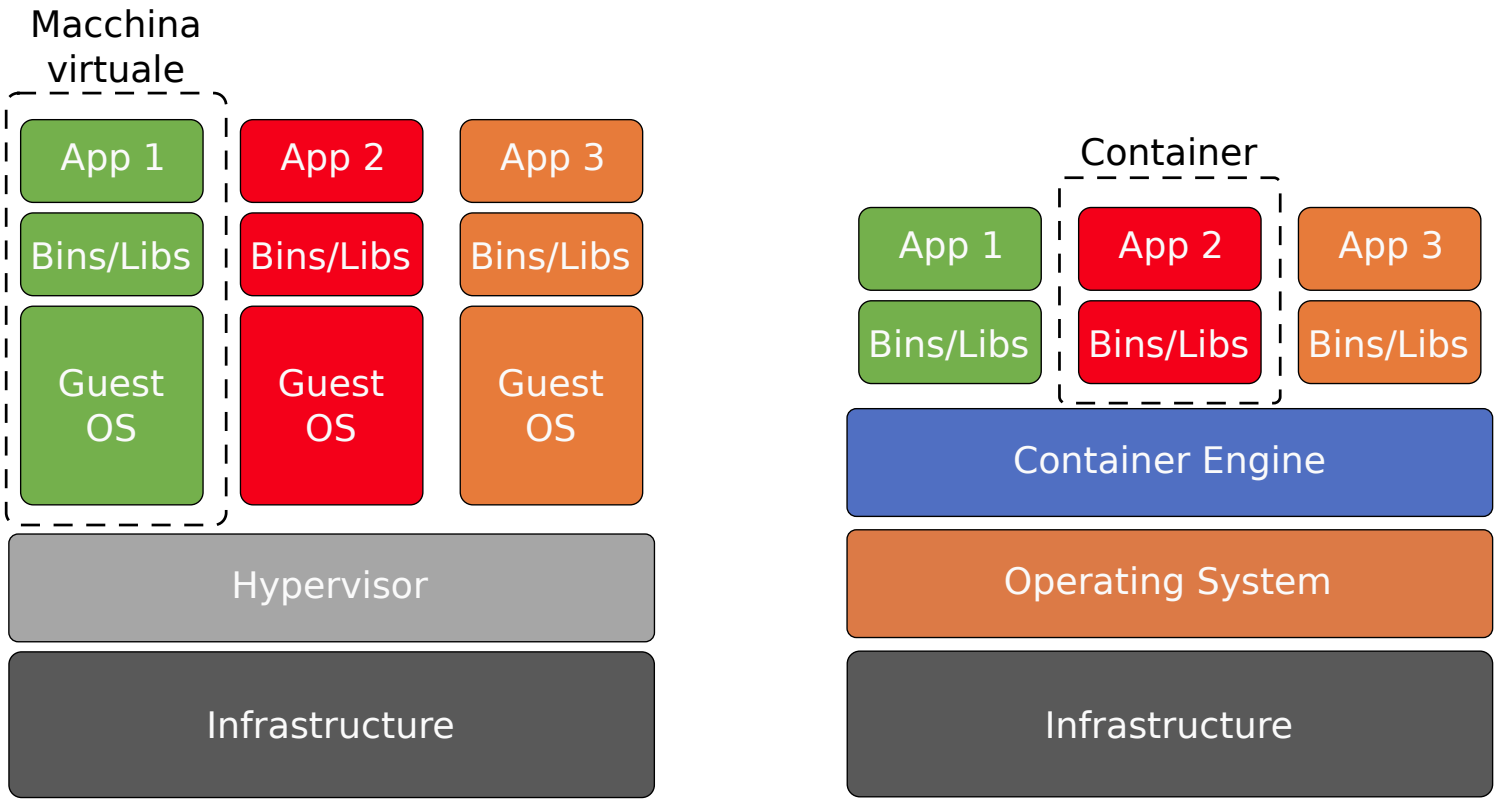
Per superare questi limiti sono stati inventati degli approcci basati sui *container*.

Questi approcci sfruttano una funzionalità del kernel del sistema operativo che consente di far esistere contemporaneamente diversi “spazi utente”, ossia diversi container.

I container possono essere considerati dei processi e quindi sono più leggeri e veloci rispetto ad un’intera macchina virtualizzata.

I container sono gestiti da un *container engine*.

Virtualizzazione vs Container



I container non richiedono un sistema operativo ospite, ma sfruttano il kernel del sistema operativo in cui sono eseguiti, lo spazio occupato dal container è dell'ordine dei MB e i tempi di avvio sono ridotti (decimi di secondo).

Docker

DE: Department of Engineering Ferrara

Cos'è Docker

> *Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.*

È uno degli approcci basati su container più diffusi e famosi.

Concetti Fondamentali

Immagine: un'immagine docker (docker image) è un modello usato per creare un container. Un'immagine può essere basata su altre immagini (riutilizzo di immagini già definite). Analogia con classe in OOP.

Container: un container è un'*istanza* eseguibile di un'immagine. Analogia con istanza di una classe in OOP.

Container Engine: gestore dei container. Permette di creare, avviare, interrompere e cancellare un container.

Registry Service: il servizio dove si registrano e si distribuiscono le immagini. Può essere cloud (DockerHub) o locale.

Vantaggi

- **Avvio rapido**
- **Deployment semplificato**: si può distribuire un'applicazione all'interno di un'immagine senza preoccuparsi che l'utente finale abbia impostato l'ambiente di lavoro
- **Rilascio semplificato**: basta pubblicare una nuova immagine (push) e il cliente può fare il pull della nuova versione per avere tutto aggiornato
- **Testing semplificato**: è possibile testare un'applicazione in diversi ambienti (per esempio, se ho sviluppato un'applicazione utilizzando Python 3.7, posso creare un container e testarla con Python 3.2)
- **Controllo delle risorse**: gli sviluppatori possono decidere quante risorse computazionali allocare per ogni container. È possibile suddividere un'applicazione in **microservizi**, dove ogni container fornisce un servizio ben specifico (maggior controllo e protezione)

Installare Docker

Mac, Windows, Linux `https://docs.docker.com/get-docker/`

In Linux (o WSL 2) è possibile utilizzare il comando: `sudo apt install docker`

Comandi Docker

I comandi docker hanno la seguente struttura: `docker <comando-principale> [<comando-secondario>] [<argomenti>]`

È possibile ottenere la lista dei comandi docker con il comando: `docker`

È possibile ottenere la versione di docker col comando: `docker version`

Comandi per le Immagini I

Comando principale per gestire le immagini:

```
1 docker image <comando-secondario>
```

Elencare le immagini registrate in locale:

```
1 docker image ls
```

Cancellare un'immagine:

```
1 docker image rm <nome-immagine>
```

Comandi per le Immagini II

Scaricare un'immagine da DockerHub:

```
1  docker image pull <nome-immagine>
```

Creare un'immagine da un Dockerfile:

```
1  docker image build <nome-immagine> <path-dockerfile>
```

Creare un'immagine da un Dockerfile ed assegnarle un tag:

```
2  docker image build --tag <nome-immagine>:<tag> <path-dockerfile
       >
```

Il tag di un'immagine può essere inteso come una versione di quell'immagine.

Dockerfile I

È possibile creare un'immagine personalizzata con i Dockerfile.

I Dockerfile sono file con il nome `Dockerfile` e contengono le istruzioni per costruire un'immagine.

Dockerfile II

```
1  # Utilizzo l'immagine node:12.18.1 come immagine base: includo
2  # nell'immagine corrente tutte le funzionalita' dell'immagine
3  # node:12.18.1
4  FROM node:12.18.1
5
6  # Imposto una cartella di lavoro: cartella nella quale verranno
7  # eseguiti diversi comandi, tra cui COPY
8  WORKDIR /usr/src/app
9
10 # Copia i file dall'host nella posizione corrente
11 COPY package.json .
12
13 # Eseguo un comando
14 RUN npm install
15
16 # Informo Docker che il container e' in ascolto sulla
17 # porta 8080
```

Dockerfile III

```
18  EXPOSE 8080
```

Comandi per i Container I

Comando principale per gestire i container:

```
1 docker container <comando-secondario>
```

Elenco dei container attivi:

```
1 docker container ls
```

Elenco dei container nella macchina (compresi quelli terminati):

```
1 docker container ls -a
```

Creare ed eseguire un container da un'immagine:

```
1 docker container run <nome-immagine>
```

Comandi per i Container II

Creare ed eseguire un container da un'immagine e poi entrare nel suo terminale:

```
3  docker container run -it <nome-immagine> bash
```

Staccarsi (detach) da un container senza interromperlo quando siamo sul suo terminale (comando precedente), utilizzare la combinazione di tasti `Ctrl+p`, `Ctrl+q`.

Per eliminare un container:

```
1  docker container rm <nome-container>
```

Pubblicare e Scaricare un'Immagine

DockerHub è un servizio cloud per la registrazione e pubblicazione di immagini: `https://hub.docker.com`

Per pubblicare un'immagine precedentemente costruita bisogna creare un alias per quell'immagine e fare il push coi comandi

```
1  docker image tag <local-image>:<tagname> <remote-repo>:<tagname
       >
2  docker image push <remote-repo>:<tagname>
```

I repository remoti seguono la convenzione `<nome-utente>/<nome-immagine>`

Sessione Pratica

Prima di proseguire:

- Installare Docker
- Creare un account su DockerHub
- Creare un repository pubblico su Docker (Repositories > Create Repository) con il nome `isa`