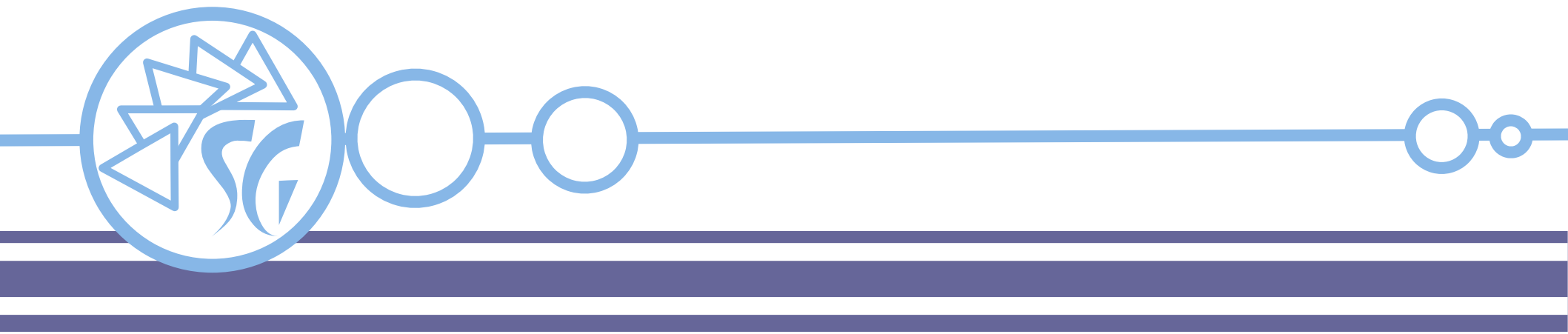

DOCKER

Ing. Simone Giustetti

www.giustetti.net

Docker è un **sistema di amministrazione di macchine virtuali** a livello di sistema operativo.

Esegue i programmi isolando le risorse hardware e software per ognuno di essi, sempre all'interno dello stesso sistema operativo.

Ottiene la compartimentazione creando molti **contenitori** distinti.

Non esistono le macchine virtuali tradizionali.

Ogni contenitore usa il medesimo kernel del sistema ospite.

Ogni programma in esecuzione all'interno di un contenitore è "isolato" e può accedere solo alle risorse assegnate al suo contenitore.

È una evoluzione di un ambiente **chroot** in grado di garantire un maggior isolamento.

Vantaggi / Svantaggi Offerti dai Contenitori

I vantaggi garantiti dall'uso di contenitori sono:

- Avvio rapido delle applicazioni.
- Prestazioni nell'ordine di quelle native. Non è utilizzata l'emulazione.
- Minor carico sulle CPU e minor consumo di risorse dato che non esiste un supervisore.

Gli svantaggi consistono in:

- Impossibilità di usare un kernel diverso da quello del sistema ospite.
- Peggior livello di isolamento rispetto ad una VM.

Confronto con Altri Sistemi Unix

Docker si appoggia sui **contenitori di Linux**.

Altri sistemi operativi della famiglia **UNIX** sono dotati di funzionalità simili. Variano il nome e le caratteristiche:

- AIX: WPAR.
- DragonFly BSD: Virtual Kernel.
- FreeBSD: Jail.
- HP-UX: SRP.
- Linux: OpenVZ (Virtuozzo).
- Solaris: Zone.

Compatibilità dei Contenitori Linux

Un contenitore può esser fatto girare su altri sistemi operativi oltre che su Linux:

- MacOsX (Tramite emulatore).
- NetBSD attraverso Podman (Non supportato).
- OpenBSD attraverso Podman (Non supportato).
- Windows (Tramite emulatore).

Podman è un programma concorrente di Docker. Ha le medesime funzionalità ed una sintassi compatibile.

libcontainer

Docker utilizza la libreria **libcontainer** per gestire i contenitori al posto della libreria nativa **LXC**.

Libcontainer supporta **funzionalità di virtualizzazione astratte** che consentono di creare un pacchetto, a partire da un software e dalle sue dipendenze, sotto forma di contenitore virtuale, che può essere eseguito su qualunque sistema operativo con la necessaria infrastruttura.

Finalità di Docker

Docker è un insieme di strumenti indirizzati agli sviluppatori.

Il loro scopo è:
Velocizzare lo sviluppo e la distribuzione di nuove applicazioni in grado di girare su più sistemi operativi oppure nel cloud.

Azzerare il tempo necessario ad installare e configurare un ambiente di lavoro. Il tempo coincide con l'avvio di un contenitore.

Finalità di Docker

Mantenere ogni ambiente di sviluppo separato ed autonomo.

Condividere velocemente l'ambiente di sviluppo con altri membri del gruppo.

Gestire la distribuzione del software attraverso il servizio **Docker Hub** che consiste in un magazzino (registry) centralizzato da cui scaricare i contenitori registrati dagli sviluppatori.

Componenti di Docker

Docker è formato da 3 componenti logici:

- Il demone / servizio **dockerd**;
- Oggetti utilizzati per “costruire” le applicazioni in Docker (Container, Image, Service);
- Magazzini (Registry) in cui caricare o da cui scaricare immagini pubbliche.

Strumenti di Docker

Docker CLI: Interfaccia carattere per amministrare dockerd e gestire i contenitori.

Docker Compose: Strumento per configurare e far girare applicazioni distribuite su più contenitori.

Docker Swarm: Fornisce funzionalità di cluster per i contenitori di Docker.

Docker Volume: Gestisce spazio disco persistente per i contenitori.

Installazione di Docker su Debian GNU/Linux

- Aggiornare il sistema:

```
apt-get update -y
apt-get upgrade -y
reboot
```

- Rimuovere eventuali versioni installate:

```
apt-get remove docker docker-engine docker.io \
    containerd runc
```

- Installare i certificati ed altri prerequisiti:

```
apt-get install ca-certificates curl gnupg \
   lsb-release
```

Installazione di Docker su Debian GNU/Linux

- Aggiungere il repository ufficiale di Docker:

```
mkdir -p /etc/apt/keyrings
curl -fsSL
    https://download.docker.com/linux/debian/gpg |
    gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg –print-architecture)
   signed-by=/etc/apt/keyrings/docker.gpg]  \
   https://download.docker.com/linux/debian \
   $(lsb_release -cs) stable" | tee
   /etc/apt/sources.list.d/docker.list > /dev/null
```

- Installare Docker:

```
apt-get install docker-ce docker-ce-cli \
   containerd.io docker-buildx-plugin \
   docker-compose-plugin -y
```

Installazione di Docker su Red Hat Linux

- Aggiungere il repository ufficiale di Docker:

```
yum-config-manager --add-repo \
https://download.docker.com/linux/rhel/docker-ce.repo
```

- Installare Docker:

```
yum install docker-ce
```

- Configurare Docker perché parta all'avvio:

```
systemctl start docker
systemctl enable docker
```

Installazione di Docker su Slackware Linux

Slackware Linux non fornisce pacchetti di Docker, ma gli script di compilazione sono scaricabili dal sito https://www.slackbuilds.org.

Per le istruzioni di compilazione di ogni pacchetto fare riferimento ai file Readme e Readme.Slackware contenuti negli archivi.

Installazione di Docker su Slackware Linux

Un’installazione completa per sviluppatori richiede i seguenti pacchetti:

- Tini;
- Google-go-lang;
- Runc;
- Containerd;
- Docker-proxy;
- Docker;
- Docker-cli;
- Docker-buildx;
- Docker-compose;
- Go-md2man;
- Buildah;
- Skopeo.

Installazione di Docker su MacOsX

- Scaricare il file *Docker.dmg* per la piattaforma di destinazione: Apple Silicon oppure Intel.
- Nel caso di piattaforma **Apple Silicon**, installare **Rosetta 2** da terminale:

```
softwareupdate --install-rosetta
```

- Avviare la procedura di installazione mediante doppio click sul file *Docker.dmg*.
- Avviare l'applicazione *Docker.app* e scorrere le schermate introduttive accettando la licenza d'uso.

Installazione di Docker su Windows

- Abilitare il supporto alla virtualizzazione hardware nel BiOS.
- Scaricare il file *Docker Desktop Installer.exe*.
- Avviare la procedura di installazione via doppio click sul file *Docker Desktop Installer.exe*.
- Preferire **WSL 2** a **Hyper-V** per piattaforme che supportino entrambi.
- Aggiungere le utenze degli sviluppatori al gruppo **docker-users**.
- Avviare l'applicazione **Docker-Desktop** ed accettare la licenza d'uso.

Testare L’Installazione di Docker

Aprire un terminale e lanciare il comando:

```
docker run hello-world
```

Se l’installazione è consistente verrete salutati da un messaggio simile a:

```
...
Hello from Docker.
This message shows that your installation appears
to be working correctly.
...
```

Le Immagini

Una **immagine** è un filesystem isolato, in sola lettura, usato come modello per creare container.

Una immagine deve contenere tutto il necessario (Programmi, librerie, file di configurazione, dipendenze) per far girare una applicazione.

Una immagine contiene anche **metadati** di configurazione necessari per inizializzare le variabili di ambiente ed impostare il comando predefinito eseguito all'avvio del contenitore.

Creare una Immagine

Una immagine richiede spazio sul disco ove copiare tutti i file necessari a far girare una applicazione.

Una immagine può essere configurata a partire da zero, oppure modificandone una preesistente.

Il sito **Docker Hub** (https://hub.docker.com) distribuisce un elenco di immagini pubbliche.

Creare una cartella dedicata per ogni immagine.

Il File DockerFile

Creare un file di testo battezzandolo: *Dockerfile*. Il file conterrà i metadati necessari a configurare l'immagine ed il **comando predefinito** da eseguire all'avvio dell'immagine.

Il primo parametro del file *Dockerfile* consiste nella immagine importata. L'immagine è specificata con l'istruzione **FROM**:

```
FROM python:latest
FROM php:8.2-cli
```

Il File DockerFile

Bisogna poi specificare il contenuto da copiare nell'immagine e la sotto-cartella di destinazione per ogni file. L'istruzione **COPY** specifica cosa copiare dentro una immagine:

```
COPY main.py /
COPY . /usr/src/myapp
```

Segue la cartella di lavoro nel file system della immagine, impostata con l'istruzione **WORKDIR**:

```
WORKDIR /
WORKDIR /usr/src/myapp
```

Il File DockerFile

Infine bisogna impostare l'applicazione da eseguire nell'immagine. L'istruzione **CMD** specifica l'applicazione:

```
CMD [ "python", "./main.py" ]
CMD [ "php",    "./script.php" ]
```

Creare una Immagine

Una volta popolato il file *DockerFile* con i parametri dell'applicazione, è necessario creare l'immagine che potrà poi essere distribuita ed eseguita.

Il comando **build** crea una immagine:

```
docker build -t my_python_application .
docker build -t my_php_application .
```

L'opzione *-t* assegna il nome all'immagine.

Eseguire una Immagine

Il comando **run** avvia una immagine eseguendo l'applicazione ivi contenuta:

```
docker run my_python_application
docker run my_php_application
```

Il nome dell'immagine è obbligatorio in quanto ogni macchina può contenere molte immagini.

Eseguire una Immagine Priva di DockerFile

È possibile lanciare una applicazione senza scrivere il corrispondente file *DockerFile*, impostando le opzioni direttamente nella riga di comando:

```
docker run -v "$PWD":/ -w / python:latest \
   pthon main.py

docker run -v "$PWD":/usr/src/myapp \
   -w /usr/src/myapp php:8.2-cli php script.php
```

Immagini ed Utenze Standard

I contenitori di Docker devono essere eseguiti da una utenza con privilegi elevati (Root).

Le utenze standard possono eseguire una immagine tramite **sudo**.

Le utenze incluse nel gruppo **docker** possono eseguire immagini senza ricorrere a sudo.

```
usermod -aG docker <utenza>
```

Informazioni & Licenze

LICENZA
Salvo dove altrimenti specificato grafica, immagini e testo della presente opera sono © Simone Giustetti.
L'opera può essere ridistribuita per fini non commerciali secondo i termini della licenza:
Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale

È possibile richiedere versioni rilasciate sotto diversa licenza scrivendo all'indirizzo: studiosg@giustetti.net

TRADEMARK
- Docker è un trademark di Docker Inc.
- FreeBSD è un trademark di The FreeBSD Foundation.
- Linux è un trademark di Linus Torvalds.
- Macintosh, OS X e Mac OS X sono tutti trademark di Apple Corporation.
- MariaDB è un trademark di MariaDB Corporation Ab.
- MySQL è un trademark di Oracle Corporation.
- UNIX è un trademark di The Open Group.
- Windows e Microsoft SQL Server sono trademark di Microsoft Corporation.
- Alcuni algoritmi crittografici citati nella presente opera potrebbero essere protetti da trademark.

Si prega di segnalare eventuali errori od omissioni al seguente indirizzo: studiosg@giustetti.net