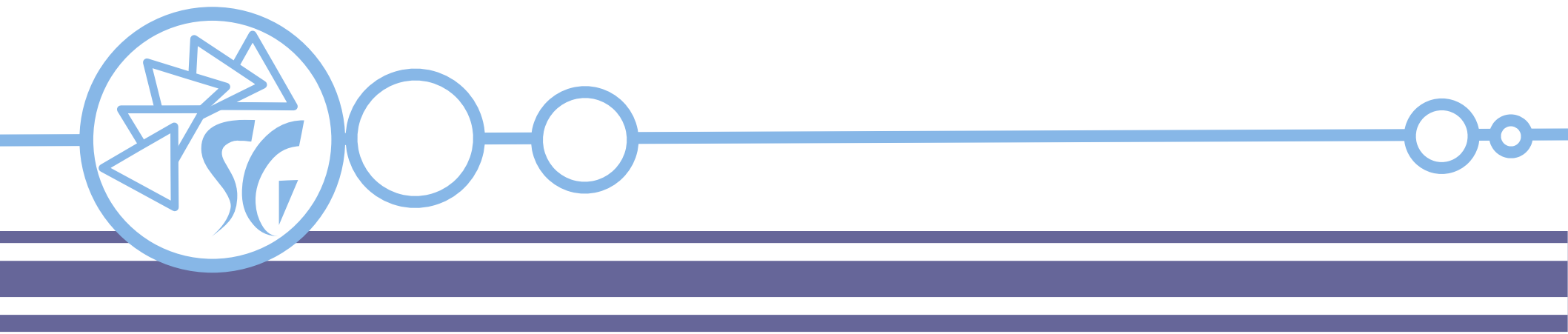

DOCKER CLI

Ing. Simone Giustetti

www.giustetti.net

Docker CLI

Docker CLI o **Docker Command Line Interface** è l'interfaccia a riga di comando che consente di gestire le immagini ed i contenitori attraverso un terminale.

La sintassi generica di ogni comando ha la forma:

```
docker <comando> <sotto-comando> <opzioni>
```

Docker CLI – Ottenere Aiuto

Il comando **Docker** dispone di un help in linea che può essere interrogato tramite:

```
docker help
```

Un breve manuale per ogni comando può essere interrogato con la sintassi:

```
docker help <comando>
docker <comando> --help

docker ps --help
docker images --help
```

Creare una Immagine

Dopo aver popolato una cartella con il contenuto opportuno ed aver aggiunto il file Dockerfile, l'immagine viene creata con il comando **build**:

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

L'opzione *-t* assegna un nome univoco all'immagine.

Una immagine può essere creata a partire da un URL.

Creare una Immagine

Lo URL può far riferimento a:

- Un repository GitHub da cui prelevare i file;
- Un archivio in formato *.tar.gz;
- Un file Dockerfile condiviso da un server remoto.

```
docker build
 https://github.com/docker/rootfs.git#container:docker

docker build github.com/creack/docker-firefox

docker build http://<server>/<archive>.tar.gz

docker build http://<server>/<docker file>
```

Importare una Immagine

Una immagine può essere importata a partire da un archivio in formato **tgz**. L'archivio può essere importato da una macchina remota.

```
docker import https://<server>/<image>.tgz
```

```
cat <image>.tgz | docker import - <image local>:new
```

Ottenere un elenco di Immagini

In una macchina possono essere configurate molte immagini. Il comando **images** rende l'elenco delle immagini esistenti:

```
docker images
docker images -a  # Mostra tutte le immagini
docker images -q  # Mostra solo gli ID
docker images <text>  # mostra le immagini il cui nome
   contiene il testo impostato
docker images --filter <filter>  # Mostra le immagini
   che verifichino il filtro impostato
   # Il filtro ha formato: "<chiave>=<valore>"
```

Storia di una Immagine

Le immagini possono essere alterate durante il loro ciclo di vita. Per ottenere informazioni circa la creazione di una immagine e le successive modifiche si usa il comando **history**:

docker history <image>

```
docker history debian
IMAGE          CREATED         CREATED BY                                      SIZE
           COMMENT
3e23a5875458   8 days ago      /bin/sh -c #(nop) ENV LC_ALL=C.UTF-8            0 B
8578938dd170   8 days ago      /bin/sh -c dpkg-reconfigure locales && loc      1.245 MB
be51b77efb42   8 days ago      /bin/sh -c apt-get update && apt-get install    338.3 MB
4b137612be55   6 weeks ago     /bin/sh -c #(nop) ADD jessie.tar.xz in /        121 MB
511136ea3c5a   9 months ago                                                     0 B
           Imported from -
```

Rimuovere una Immagine

Come ogni programma, anche le immagini hanno un ciclo di vita. Le immagini divenute superflue devono essere rimosse per liberare risorse.

Il comando **rmi** rimuove una o più immagini:

```
docker rmi <image>
docker rmi $(docker images -a -q)  # Rimuove tutte le
   immagini configurate su una macchina.
```

Se il contenitore corrispondente stesse girando, la rimozione fallirebbe.

Creare un Back-up di una Immagine

Il comando **save** crea una copia dell'immagine, in formato **tar**, su stdout:

```
docker save <image> > image_backup.tar
```

```
docker save <image> -o image_backup.tar
```

L'opzione *-o* consente di salvare l'immagine in un file anziché su stdout.

L'immagine salvata può essere caricata in seguito con il comando **load**.

Cercare una Immagine Pubblicata

Molte immagini già pronte all'uso sono messe a disposizione sia dai produttori di Docker che da terzi. Le immagini possono essere cercate in Internet oppure con il comando **search**.

```
docker search <target>
docker search -f <filter> <target>
docker search --limit <int> <target>

docker search busybox
docker search -f stars=3 busybox
```

Scaricare / Pubblicare una Immagine

Il comando **pull** consente di scaricare una immagine pubblicata in un repository (Sia Docker Hub che uno privato).

```
docker pull <target>
docker pull debian:bullseye
docker pull <registry>:5000/testing/debian
```

Il comando **push** pubblica una immagine in un repository.

```
docker push <registry>:5000/<path>/deb_httpd:latest
```

Creare ed Avviare un Contenitore

Il comando **run** crea ed avvia un contenitore a partire da una immagine ed esegue l'applicazione ivi contenuta:

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

Il nome del contenitore è obbligatorio e deve essere univoco. È possibile usare lo ID al posto del nome.

Ottenere un elenco di Contenitori

In una macchina possono coesistere molti contenitori. Il comando **ps** ne rende l'elenco.

```
docker ps -a
```

L'opzione *-a* mostra anche i contenitori inattivi.

```
docker ps -q
```

L'opzione *-q* mostra solo l'identificatore univoco di ogni contenitore.

Fermare / Riavviare un Contenitore

I contenitori possono essere fermati utilizzando il comando **stop** ed il nome univoco oppure lo ID:

```
docker stop <container>

docker stop $(docker ps -a -q)  # Ferma tutti i
      contenitori che girano su una macchina.
```

Un contenitore fermo può essere avviato con il comando **start**:

```
docker start <container>
```

Forzare il Fermo di un Contenitore

Quando non è possibile fermare un contenitore con il comando **stop**, lo si può terminare con **kill**. Il comando invia un segnale sigkill ad ogni processo che sta girando nel contenitore.

```
docker kill <container>
```

L'esecuzione di kill ha un timeout predefinito della durata di 10 secondi.

Riavvio forzato di un Contenitore

È possibile forzare il riavvio di un contenitore che non risponda più ai comandi con **restart**. Tutti i processi che girano nel contenitore vengono fermati preventivamente inviando loro un segnale sigkill.

```
docker restart <container>
```

Mettere in Pausa un Contenitore

Il comando **pause** sospende tutti i processi di un contenitore. Ad ogni processo del contenitore viene inviato un segnale sigstop. Il contenitore non può interagire con il mondo fino a che non viene riavviato con il comando **unpause**.

```
docker pause <container>

docker unpause <container>
```

Creare un Back-up di un Contenitore

Il comando **export** crea una copia del file system contenuto in un contenitore in formato **tar**:

```
docker export <container> > container_backup.tar
```

```
docker export <container> -o container_backup.tar
```

L'opzione *-o* consente di salvare l'immagine in un file anziché su stdout.

Attenzione: export non gestisce i volumi associati ad un contenitore.

Rimuovere un Contenitore

I contenitori possono essere rimossi solo dopo esser stati fermati. Per rimuovere un contenitore si usa il comando **rm**:

```
docker rm <container>

docker rm $(docker ps -a -q)  # Rimuove tutti i
   contenitori fermi.
```

Se il contenitore stesse girando, la rimozione fallirebbe.

Rinominare un Contenitore

Il comando **rename** cambia il nome di un contenitore.

```
docker rename <container name> <new container name>
```

File di Log

Docker salva informazioni diagnostiche per ogni contenitore in appositi file. I file possono essere acceduti usando il comando **logs**:

```
docker logs <container>
```

Controllare le Prestazioni

Docker CLI fornisce due comandi per interrogare lo stato dei contenitori attivi: **stats** e **top**. Stats rende lo stato delle risorse occupate da ogni contenitore. Top rende l'elenco dei processi che girano in un contenitore.

```
docker stats [ <container> ]

docker top <container>
```

Il prospetto reso da **stats** è aggiornato ad intervalli regolari automaticamente.

Limitare le Risorse di un Contenitore

Quando un contenitore consuma troppe risorse sul sistema che lo ospita, è possibile imporre al volo dei limiti con il comando **update**.

```
docker update <opzioni> <container>

docker update --cpu-shares 512 <container>
docker update --cpus 2 <container>
docker update --memory 80M <container>
docker update --pids-limit 250 <container>
```

Lanciare Comandi in un Contenitore

L’istruzione **exec** lancia un nuovo comando in un contenitore attivo. Se il contenitore venisse riavviato, il comando non viene rilanciato. Perdura quanto il processo **1** del contenitore.

```
docker exec [ opzioni ] <container> <command>
```

<command> deve essere un eseguibile.

```
# Funziona
docker exec -it <container> sh -c "echo a && echo b"
# Non funziona
docker exec -it <container> "echo a && echo b"
```

Lanciare Comandi in un Contenitore

Alcune opzioni mutano il comportamento del comando lanciato nel contenitore:

- *-d*: Esegue il comando in background.
- *-i*: Tiene aperto stdin anche per i comandi in background.
- *-t*: Alloca uno pseudo terminale.
- *-u*: Fa girare il comando con i privilegi dell'utenza impostata.
- *-w*: Utilizza la cartella di lavoro impostata.

Lanciare Comandi in un Contenitore

```
docker exec -i -t <container> sh # Avvia una shell nel
   contenitore

docker exec -it -w /root <container> pwd # Esegue il
   comando pwd nella cartella /root
```

Se il contenitore fosse messo in pausa, exec fallirebbe, non potendo comunicare con lo stesso.

Effetti dei Comandi Lanciati

I comandi lanciati possono modificare sia il file system del contenitore che il suo contenuto.

```
# Lancia una distribuzione Linux
docker run rockylinux
# Installa MariaDB
docker exec -it rockylinux dnf install mariadb-server
```

Le modifiche interessano solo il contenitore e non l'immagine da cui è derivato.

Le modifiche perdurano fino a che il contenitore non viene rimosso.

Copiare File da/nel Contenitore

Il comando **cp** consente di copiare file tra un contenitore ed il sistema che lo ospita. I percorsi nel contenitore sono relativi alla root (/) del contenitore. È possibile copiare sia file che directory.

```
docker cp <container>:<source> <target>
docker cp <source> <container>:<target>

docker cp slackware150:/var/logs/ /tmp/app_logs
```

Tracciare le Modifiche

Il contenuto di un contenitore muta nel tempo, è utile tener traccia delle modifiche. Il comando diff rende l'elenco delle modifiche intervenute.

```
docker diff <container>
   C /dev/stderr
   C /dev/stdin
   A /run/nginx.pid
   C /var/lib/nginx/tmp
   A /var/lib/nginx/tmp/client_body
```

A: Aggiunta di un file o una cartella.
C: Modifica di un file o una cartella.
D: Rimozione di un file o una cartella.

Salvare le Modifiche in una Immagine

È possibile salvare lo stato di un contenitore in una nuova immagine.

- Uscire dal contenitore, se collegati:

```
exit
```

- Confermare le modifiche creando una nuova immagine:

```
docker commit -m "<modifica apportata>" -a "<autore>"
<id del contenitore> <repository>/<nome della nuova
immagine>
```

Salvare le Modifiche in una Immagine

Es:

```
docker commit -m "Installato MariaDB" -a
"studiosg@giustetti.net" 59839a1b7di5
studiosg/rockylinux-mariadb
```

La nuova immagine dovrebbe ora comparire nell'elenco di quelle presenti sulla macchina.

La dimensione della nuova immagine dovrebbe differire da quella originale.

Caricare una Immagine in un Repository

Una immagine può essere caricata in ogni registry di cui si disponga delle credenziali di accesso.

I passi per caricare un'immagine su Docker Hub, dopo essersi registrati, sono:

```
docker login -u <utenza>
docker push <utenza>/<immagine>
```

Una volta caricata, l'immagine verrà elencata tra quelle associate all'utenza.

Informazioni & Licenze

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