

# Installare Docker

- Caveat: eventuali versioni più antiche non sono più supportate e vanno rimosse
- Seguendo l'ottima documentazione in <https://docs.docker.com/> si installerà *docker-ce*, Docker Community Edition, garantita come “edizione” ufficiale e più recente, cf. <https://docs.docker.com/install/> (sezione *About Docker CE*)
- Esiste anche l'edizione Docker EE ( Enterprise edition), a pagamento, mentre la Community Edition è gratuita
- Altra guida (rapida e efficace): <https://computingforgeeks.com/installing-docker-ce-ubuntu-debian-fedora-arch-centos/>
- Fino a pag. 5, si illustra l'installazione su CentOS, versione free di Red Hat, e, un tempo, distro di “prima scelta” per Docker
- Al 2021, però, Docker gira egregiamente su Debian/Ubuntu e CentOS è *discontinued*; si consiglia quindi di saltare a pag. 6

# Installazione e avvio di Docker-CE: in a nutshell

- Per altre distro (Ubuntu, Debian, ...) e per il cloud (AWS, Azure), seguire: <https://docs.docker.com/install/>
  - Per Ubuntu... **non** installare con *apt* prima di aver letto la guida sopra!
- Ottima sintesi, concisa ed efficace, per installare Docker CE: <https://computingforgeeks.com/installing-docker-ce-ubuntu-debian-fedora-arch-centos/>
- Installato Docker-CE, si può avviare (da root) il servizio Docker:

```
# systemctl start docker
# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor
   preset: disabled)
   Active: active (running) since Fri 2018-06-01 03:28:10 UTC; 30s ago
```

- Il daemon Docker è, a volte, “delicato”. Se risponde in modo strano o non risponde, sempre da super-user:

```
# systemctl restart docker    # e, se ancora non bastasse:
# killall dockerd             # e, se ancora non bastasse: killall -9 dockerd
```

# Immagine/Container *Hello world!*

- Lanciandolo, verifichiamo il buon funzionamento di Docker (da utente comune, ma appartenente al gruppo *docker*):

```
$ docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally
```

```
latest: Pulling from library/hello-world
```

```
# immagine scaricata dall'hub di
```

```
Docker
```

```
9bb5a5d4561a: Pull complete
```

```
Digest:
```

```
sha256:f5233545e43561214ca4891fd1157e1c3c563316ed8e237750d59bde73361e77
```

```
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
```

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

```
To generate this message, Docker took the following steps:
```

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

- Alcune keyword: Docker daemon, client, image, container

# Elementi di base: CLI e daemon/servizio

Da riga di comando, come utente comune nel gruppo docker, si invoca il client (CLI) docker:

```
$ docker
```

```
Usage: docker [OPTIONS] COMMAND
```

```
A self-sufficient runtime for containers
```

```
Options:
```

```
-H, --host list      Daemon socket(s) to connect to
```

```
...
```

```
Management Commands:
```

```
container  Manage containers
```

```
...
```

```
Commands:
```

```
attach      Attach local standard input, output, and error streams to a running container
```

Verifichiamo ora la presenza del daemon dockerd e del servizio docker:

```
$ ps -A | grep dockerd
```

```
367 ?        00:06:04 dockerd
```

```
$ systemctl status docker
```

```
● docker.service - Docker Application Container Engine
```

```
Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
```

```
Active: active (running) since Fri 2019-06-07 13:56:13 CEST; 1 day 18h ago
```

```
Main PID: 367 (dockerd)
```

```
Tasks: 24 (limit: 2370)
```

```
Memory: 176.2M
```

```
CGroup: /system.slice/docker.service
```

```
├─367 /usr/bin/dockerd -H fd://
```

```
├─375 containerd -config /var/run/docker/containerd/containerd.toml -log level info
```

# Interazione di base

Come spiegato dall'output di *docker run hello-world*:

- il cliente CLI *docker* si rivolge al daemon/servizio/engine *dockerd*
- questo crea un container (dal template/immagine *hello-world*)
- il container esegue il codice dell'app incapsulata nel container
- il daemon *dockerd* media l'input/output tra il container in esecuzione e il client CLI *docker*, che a sua volta comunica con il terminale dell'utente
  - nel caso di *hello-world*, l'output generato dall'app/container va al daemon *dockerd*, che lo passa al client CLI *docker*, che lo emette sulla standard output dell'utente

La comunicazione daemon-client avviene via una socket del kernel:

```
$ ls -l /var/run/docker.sock
```

```
srw-rw---- 1 root docker 0 7 giu 13.55 /var/run/docker.sock
```

Permessi e ownership implicano che solo gli utenti nel gruppo *docker* possono interagire direttamente col daemon (v. prossima slide)

# Docker come utente non-root

Durante l'installazione, sarà stato definito il gruppo *docker*:

```
$ grep docker /etc/group  
docker:x:999:user1,user2
```

Se l'utente corrente non è nel gruppo *docker*, si vedrà l'output:

```
$ groups | tr ' ' '\n' | grep docker  
$
```

Se invece l'utente corrente è nel gruppo *docker*, si vedrà l'output:

```
$ groups | tr ' ' '\n' | grep docker  
docker  
$ sudo deluser $USER docker # ora logout e login...
```

Qui sopra si è temporaneamente eliminato dal gruppo *docker* l'utente corrente, che non potrà più interagire col daemon/engine di Docker:

```
$ docker run hello-world  
docker: permission denied trying to connect to Docker daemon socket at unix:///var/run/docker.sock...
```

Rimettiamo quindi l'utente corrente nel gruppo *docker*, con un comando tra:

```
$ sudo adduser $USER docker # rimettiamo l'utente nel gruppo docker (logout e login di nuovo)
```

```
$ sudo usermod -aG docker $USER # rimettiamo l'utente nel gruppo docker (logout e login di nuovo)
```

# Immagine/container *Ubuntu*

```
$ docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
a48c500ed24e: Pull complete
1e1de00ff7e1: Pull complete
0330ca45a200: Pull complete
471db38bcfbf: Pull complete
0b4aba487617: Pull complete
Digest:
sha256:c8c275751219dadad8fa56b3ac41ca6cb22219ff117ca98fe82b42f24e1ba64e
Status: Downloaded newer image for ubuntu:latest
root@91f5926d6c41:/#
```

- *Da man docker-run:*

- i, --interactive, Keep container's STDIN open even if not attached
- t, Allocate a pseudo-TTY. The default is false. When true, Docker can allocate a pseudo-tty and attach to the standard input of any container
- L'immagine per il container *Ubuntu* non esisteva localmente, quindi è stata scaricata da <https://store.docker.com> (o *hub.docker.com*)

# Immagini e container

- L'immagine è un template (stampo) di container, da cui si possono generare più container in esecuzione nell'engine Docker
  - (cf. programma (file eseguibile) vs. processo (in esecuzione))
- Un'immagine ha una struttura interna stratificata (*layered*) che riflette la sua tipica creazione *bottom-up*, che parte da uno strato iniziale, esegue dei comandi di modifica dello stato di questo, aggiungendo quindi altri strati...
- Il sito [microbadger](#) consente di visualizzare la struttura dei container...



# Immagini e layer: [microbadger.com](http://microbadger.com)

- Da [microbadger](http://microbadger.com) :

Layers	4	No matching base image?
32 bytes	MAINTAINER Tianon Gravi <admwiggin@[hidden]> - mkimage-deboots...	+
	ADD file:eb15dbd63394e063b805a3c32ca7bf0266ef64676d5a6fab4801f2e81e2...	+
60.6 MB	ADD file:eb15dbd63394e063b805a3c32ca7bf0266ef64676d5a6fab4801f2e81e2...	+

immagine di ubuntu:lucid

Layers	2	No matching base image?
32.3 MB	ADD file:09d717d62608e18d79af6b6cd5aae36f675bd5c4f34452ab1693b56bfbf...	+
32 bytes	CMD ["/bin/bash"]	

immagine di debian:squeeze

# Immagini/Container: gestione da riga di comando

```
$ docker images      # immagini disponibili (scaricate prima dall'hub di Docker)
REPOSITORY          TAG                IMAGE ID           CREATED
SIZE
ubuntu              latest            452a96d81c30      4 weeks ago
79.6MB
hello-world         latest            e38bc07ac18e      7 weeks ago
1.85kB

$ docker ps          # container attivi
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS

$ docker container ls # come il precedente: container attivi
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS
# ls /var/lib/docker/image/overlay2/imagedb/content/sha256/
452a96d81c30a1e426bc250428263ac9ca3f47c9bf086f876d11cb39cf57aeec
e38bc07ac18ee64e6d59cf2eafcdddf9cec2364dfe129fe0af75f1b0194e0c96

$ docker images
REPOSITORY          TAG                IMAGE ID           CREATED
SIZE
ubuntu              latest            452a96d81c30      4 weeks ago
79.6MB
hello-world         latest            e38bc07ac18e      7 weeks ago
1.85kB
```