Docker Desktop

- Installate **Docker Desktop** dal sito:
  `https://www.docker.com/products/docker-desktop`
- Avviate **Docker Desktop.**
- Aprite un terminale (o prompt dei comandi) e digitate:
  `> docker run -d -p 80:80 docker/getting-started`
- Provate ad entrare nella directory dell'immagine appena creata:
  `> cd getting-started`
- Analizzate i file che ci sono e guardate come è fatto il `dockerfile`!

Docker Desktop

- Per vedere il file:
  `> type Dockerfile`

- Provate anche i comandi:

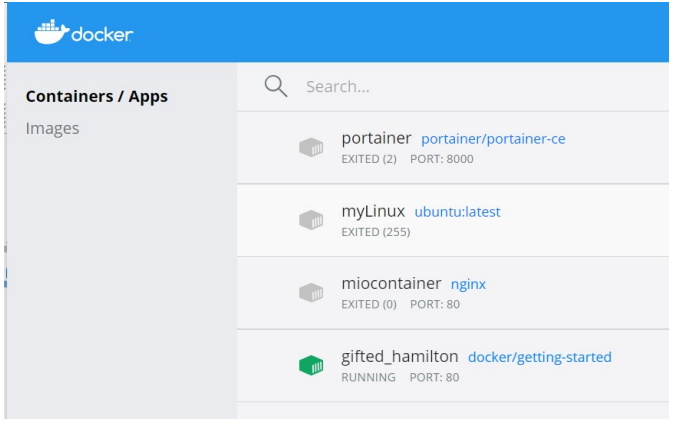
```
> docker ps
> docker images
```

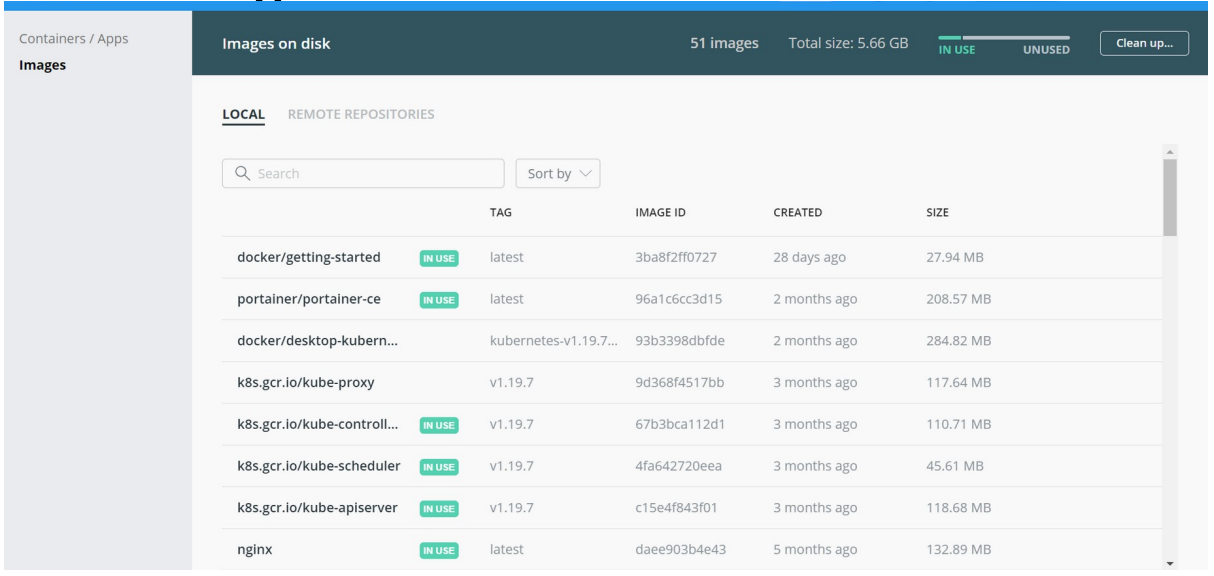
Anaconda Prompt (Miniconda3)

```
(base) C:\Users\Utente\getting-started>type Dockerfile
# Install the base requirements for the app.
# This stage is to support development.
FROM python:alpine AS base
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

# Run tests to validate app
FROM node:12-alpine AS app-base
WORKDIR /app
COPY app/package.json app/yarn.lock ./
RUN yarn install
COPY app/spec ./spec
COPY app/src ./src
RUN yarn test

# Clear out the node_modules and create the zip
FROM app-base AS app-zip-creator
RUN rm -rf node_modules && \
    apk add zip && \
    zip -r /app.zip /app

# Dev-ready container - actual files will be mounted in
FROM base AS dev
CMD ["mkdocs", "serve", "-a", "0.0.0.0:8000"]

# Do the actual build of the mkdocs site
FROM base AS build
COPY . .
RUN mkdocs build

# Extract the static content from the build
# and use a nginx image to serve the content
FROM nginx:alpine
COPY --from=app-zip-creator /app.zip /usr/share/nginx/html/assets/app.zip
COPY --from=build /app/site /usr/share/nginx/html

(base) C:\Users\Utente\getting-started>
```

Docker Desktop

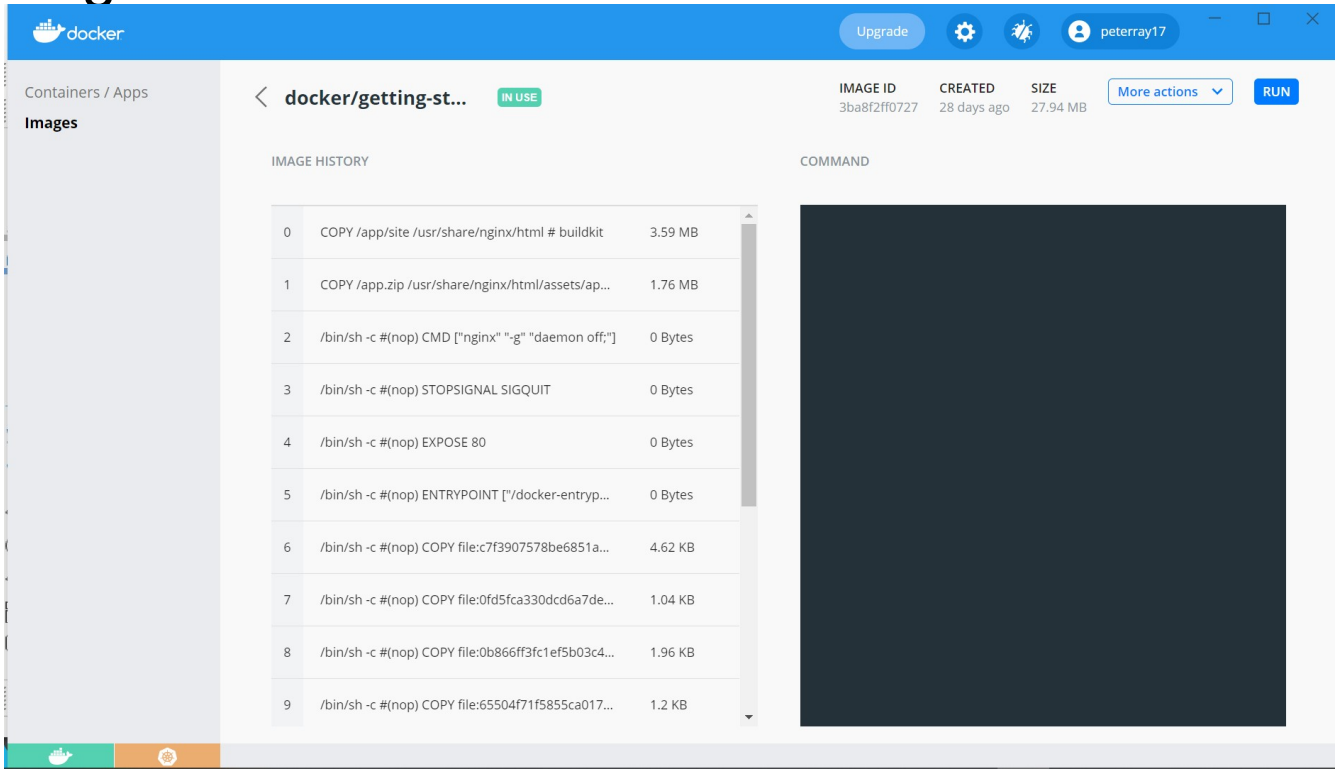
- Torniamo su **Docker Desktop.**
- Sulla finestra di **Docker Desktop** potete vedere l'elenco dei contenitori attivi:
- Cliccando su **images**, vedrete le imagini installate nel sistema:





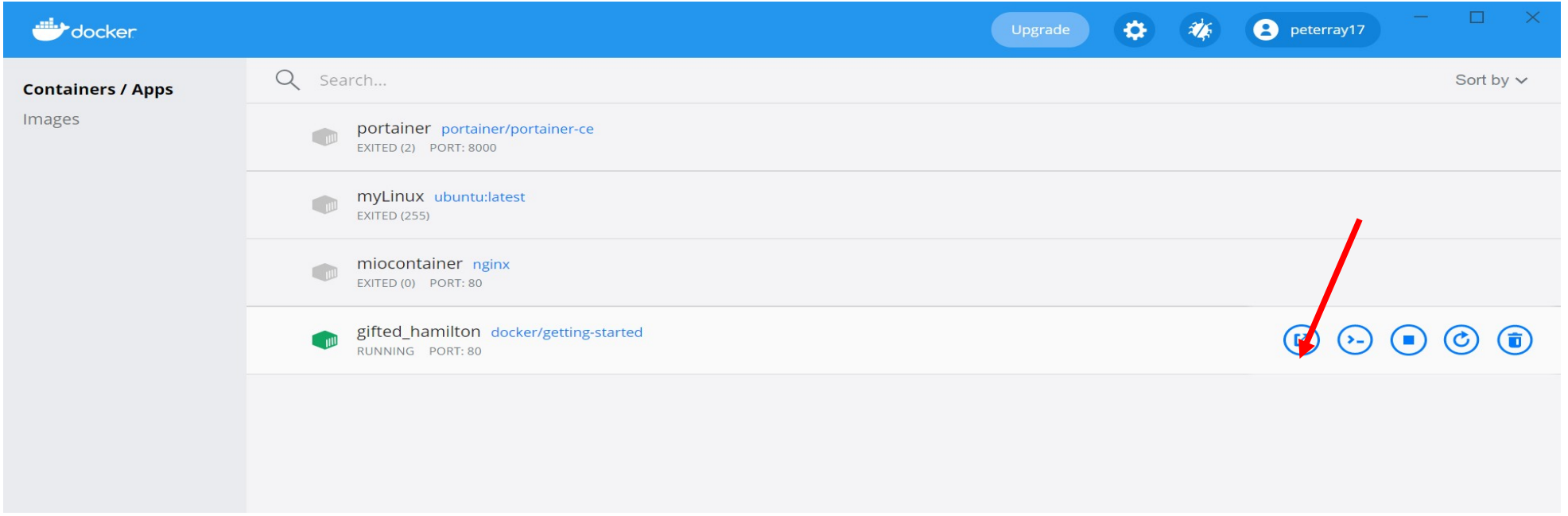
Docker Desktop

- Cliccando sul **container** `docker/getting-started` vedrete i **layer** che compongono i **container** con la storia delle istruzioni usate per crearlo.



Docker Desktop

- Tornate indietro e passando sopra l'immagine `docker/getting-started` vi appariranno dei pulsanti.
- Cliccate sul primo, *open in browser* per inziare il tutorial.



Docker Desktop

- Seguite il tutorial e… buon divertimento!

- Mettete like sulla pagina e seguitemi per altre interessanti lezioni di laboratorio!

Docker – Installazione su Linux Ubuntu

- Anzitutto aggiorniamo l'elenco dei pacchetti e installiamo alcuni pacchetti richiesti:

```
# sudo apt update
# sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

- Aggiungiamo al sistema il repository ufficiale di **Docker** e la relativa chiave GPG:

```
# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
# sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

Docker – Installazione su Linux Ubuntu

- Aggiorniamo di nuovo l'elenco dei pacchetti:

```
# sudo apt update
```

- Installiamo Docker:

```
# sudo apt install docker-ce
```

- Controlliamo lo stato del demone Docker:

```
# sudo systemctl status docker
```

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2018-07-05 15:08:39 UTC; 2min 55s ago
     Docs: https://docs.docker.com
 Main PID: 10096 (dockerd)
    Tasks: 16
   CGroup: /system.slice/docker.service
           ├─10096 /usr/bin/dockerd -H fd://
           └─10113 docker-containerd --config /var/run/docker/containerd/containerd.toml
```

Docker – Sintassi del comando docker

- Sinteticamente usare Docker significa effettuare chiamate al comando omonimo creando una o più catene formate da comando sottocomando e parametri seguendo la sintassi:

```
# docker [opzioni] [comando] [argomenti]
```

- Ad esempio per ottenere le informazioni sul sistema docker:

```
# docker info
```

- Per ottenere un elenco dei comandi disponibili digitare:

```
# docker
```

- Per ottenere un aiuto per ogni singolo comando:

```
# docker comando --help
```

Docker – Gestire i container

- I **contenitori Docker** vengono creati da **immagini Docker**.
- Per impostazione predefinita, **Docker** estrae queste immagini da **Docker Hub**,.
- Verifichiamo di poter accedere al **Docker Hub** avviando il tutorial:

```
# docker run -d -p 80:80 docker/getting-started
```

- NB: ricordatevi di disattivare nginx o apache…
- NB: ora potete svolgere il tutorial collegandovi col browser all' ip della vostra vm, ad esempio: `http://<ipvostravm>`
- Cosa succede?

Docker – Gestire i container

- E’ possibile cercare le immagini disponibili su **Docker Hub** utilizzando il comando `docker` con il sotto-comando di ricerca. Ad esempio, per cercare l'immagine di Ubuntu:

```
# docker search ubuntu
```

- Dovreste ottenere un elenco di tutte le immagini che coinvolgono `ubuntu` presenti nel **Docker Hub**.
- Per scaricare ed installare l’immagine ufficiale di Ubuntu:

```
# docker pull ubuntu
```

- Per eseguire un **container** con l’immagine appena scaricata è sufficiente utilizzare il comando `run`:

```
# docker run ubuntu sleep 100
```

Docker – Gestire i container

- Vediamo i container in esecuzione con `docker ps`

```
root@las:~# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
d4030db3a461   ubuntu    "sleep 10"    3 seconds ago    Up 2 seconds              eager_euler
```

- Per spegnere un container utilizzate il comando `docker stop` specificando l'id del container o il suo nome:

```
# docker stop d403db3a461
```

- Per vedere quali immagini sono scaricate sul server **Docker** è sufficiente usare il comando `docker images`:

```
# docker images
```

```
Output
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              113a43faa138        4 weeks ago         81.2MB
hello-world         latest              e38bc07ac18e        2 months ago        1.85kB
```

Docker – Gestire i container

- Per avviare un container in modo iterativo e quindi poterci accedere:

```
# docker -it run ubuntu
```

- Dovreste ottenere un accesso a terminale del tipo:

```
root@d9b100f2f636:/#
```

  - Dove d9b100f2f636 indica l'id del container.

Docker – I container sono oggetti isolati

- Ad esempio supponiamo di voler salvare dei file creati all'interno di un **container** con immagine Ubuntu:

```
# mkdir data_volume

# docker run -it -v /home/las/data_volume:/docs
ubuntu /bin/bash

root@b4afd75514bd:/# cd doc

root@b4afd75514bd:/# touch prova.txt
```

- Alla chiusura del **container** in `data_volume` troveremo tutti gli elementi creati e/o modificati in `/docs`.

Docker – Gestire i container

- Ora, all'interno del container potete fare ciò che volete, ad esempio installare nodejs:

```
# apt update
# apt install nodejs
# node -v
```

- Una volta finito, potete uscire dal container digitando:

```
# exit
```

- E' possibile sapere quali container sono in esecuzione nel sistema tramite il comando:

```
# docker ps
```

Docker – Gestire i container

- Per sapere quali **container** sono presenti nel sistema tramite il comando:

```
# docker ps -a
```

- Per avviare un container che non è in esecuzione si utilizza il comando `start` seguito dall'id del **container**:

```
# docker start d9b100f2f636
```

- Per fermare un container di utlizza il comando `stop` seguito dal l'id del **container**:

```
# docker stop d9b100f2f636
```

- Per cancellare un **container**:

```
# docker rm idcontainer
```

- Per "ispezionare" un **container**:

```
# docker inspect idcontainer
```

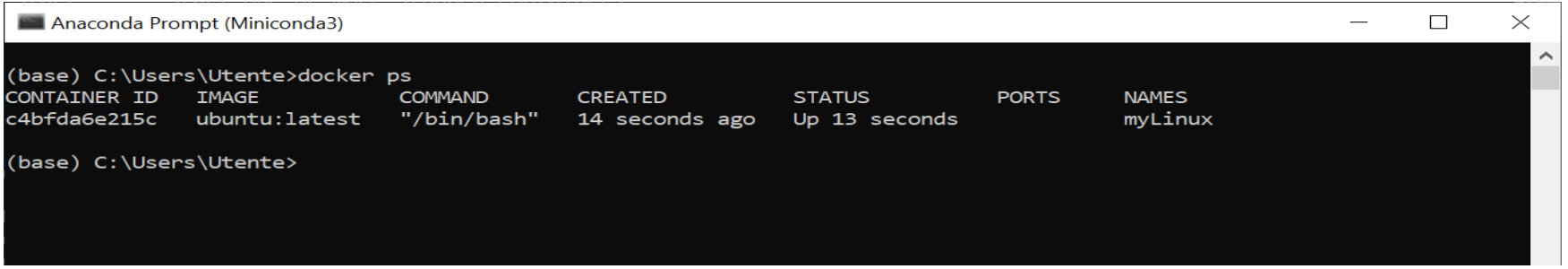
Docker – Gestire i container

- E’ possibile **assegnare un nome** ad un **container** che lo identifica all’interno del sistema. Tale nome, assegnato tramite la clausola `--name` può sostituire l’id del **container** nelle varie operazioni.

```
# docker run -it --name myLinux ubuntu:latest
```

```
(base) C:\Users\Utente>docker rm b16100c7ba86f6c07ed2e3f47994717eaece822a3a644c37f6d5b8eb45b197b0
b16100c7ba86f6c07ed2e3f47994717eaece822a3a644c37f6d5b8eb45b197b0

(base) C:\Users\Utente>docker run -it --name myLinux ubuntu:latest
root@c4bfda6e215c:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@c4bfda6e215c:/#
```

```
# docker ps
```



Docker – Building images: Esempio

- Vogliamo costruire un immagine **Docker** che generi un container per servire pagine web statiche html.
- Supponiamo di creare queste pagine nella nostra home al percorso:

  `/home/$USER/sitoweb/www`

  o

  `/C/Users/$USER/sitoweb/www`
- Per comodità vogliamo partire da un immagine già pronta di **Engine-X**.
- **NB** copiate eventuali pagine web statiche da `/var/www/html`.

Docker – Building images: Esempio

- Cominciamo creando il file “`dockerfile`” nella cartella `/home/$USER/sitoweb/`:

```
# touch dockerfile
```

- Inseriamo nel `dockerfile` la seguente configurazione:

```
FROM nginx:latest
LABEL Author="memedesimo" Version="1.0"
EXPOSE 80
COPY ./www/ /var/www/html
```

Docker – Building images: Esempio

- Così facendo stiamo definendo un'immagine a partire dall'immagine di Engine-X `nginx:latest`.
- Il container risultante esporrà la porta 80 in modo da poter mostrare all'esterno i file che abbiamo copiato nella directory `/var/www/html` grazie all'istruzione `COPY`.
- Per effettuare il build dell'immagine, ci sposteremo dove è posizionato il `dockerfile`, ovvero in `/home/$USER/sitoweb/` , e digiteremo il comando:

```
docker build -t mywebserver_1:v1 .
```

- Notate che abbiamo specificato anche la versione dell'immagine (v1).

Docker – Building images: Esempio

- Controlliamo se c'è la nuova immage:

  `# docker image ls.`

- Per eseguire l'immagine appena costruita, poichè il `dockerfile` espone la porta utilizzata da `Engine-X` (ovvero la classica 80), basterà utilizzare il parametro `-P` nel comando `docker run`:

  ```
  # docker run -d --name mywebserver_1 -P mywebserver_1
  ```

- Oppure volendo specificare una porta:

  ```
  # docker run -d --name mywebserver_1 -p 8088:80 mywebserver_1
  ```

- *(**Windows**) Come faccio ad accedere al server web appena creato? Una volta scoperto l'ip con il comando `docker-machine ip`, basta puntare il browser a quell'indirizzo aggiungendo eventualmente la porta se specificata in `docker run`. Vedi `https://docs.docker.com/machine/install-machine/`*

Docker – Esempio: Docker Compose e Wordpress

- Installiamo **Docker Compose**:

```
# apt install docker-compose
```

- Creiamo una directory vuota:

```
# mkdir wordpress
```

- Questa directory conterrà l'ambiente dell'immagine dell'applicazione. La directory dovrebbe contenere solo le risorse per costruire quell'immagine tra cui un file `docker-compose.yml` che definisce un'applicazione wordpress di base.
- Creiamo nella directory wordpress un file di nome: `docker-compose.yml`:

```
# cd wordpress

# touch docker-compose.yml
```

- Inseriamo nel file il seguente contenuto:

Docker – Esempio: Docker Compose e Wordpress

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

Docker – Esempio: Docker Compose e Wordpress

- Il volume **docker** `db_data` mantiene gli aggiornamenti effettuati da WordPress al database.
- WordPress funziona solo sulle porte 80 e 443.
- Eseguiamo `docker-compose up -d` dalla directory del progetto (vedi slide successiva).
- Se si utilizza **Docker Machine**, puoi eseguire il comando `docker-machine ip MACHINE_VM` per ottenere l'indirizzo della macchina, quindi aprire in un browser web.
- Se si utilizza Docker Desktop per Mac o Docker Desktop per Windows, utilizzare `http://127.0.0.1` come indirizzo IP e aprire `http://127.0.0.1:8000` in un browser web.

Docker – Esempio: Docker Compose e Wordpress

Questo comando esegue `docker-compose` in modalità **detach**, estrae le immagini **Docker** necessarie e avvia i contenitori

wordpress

e database..

- Provate:

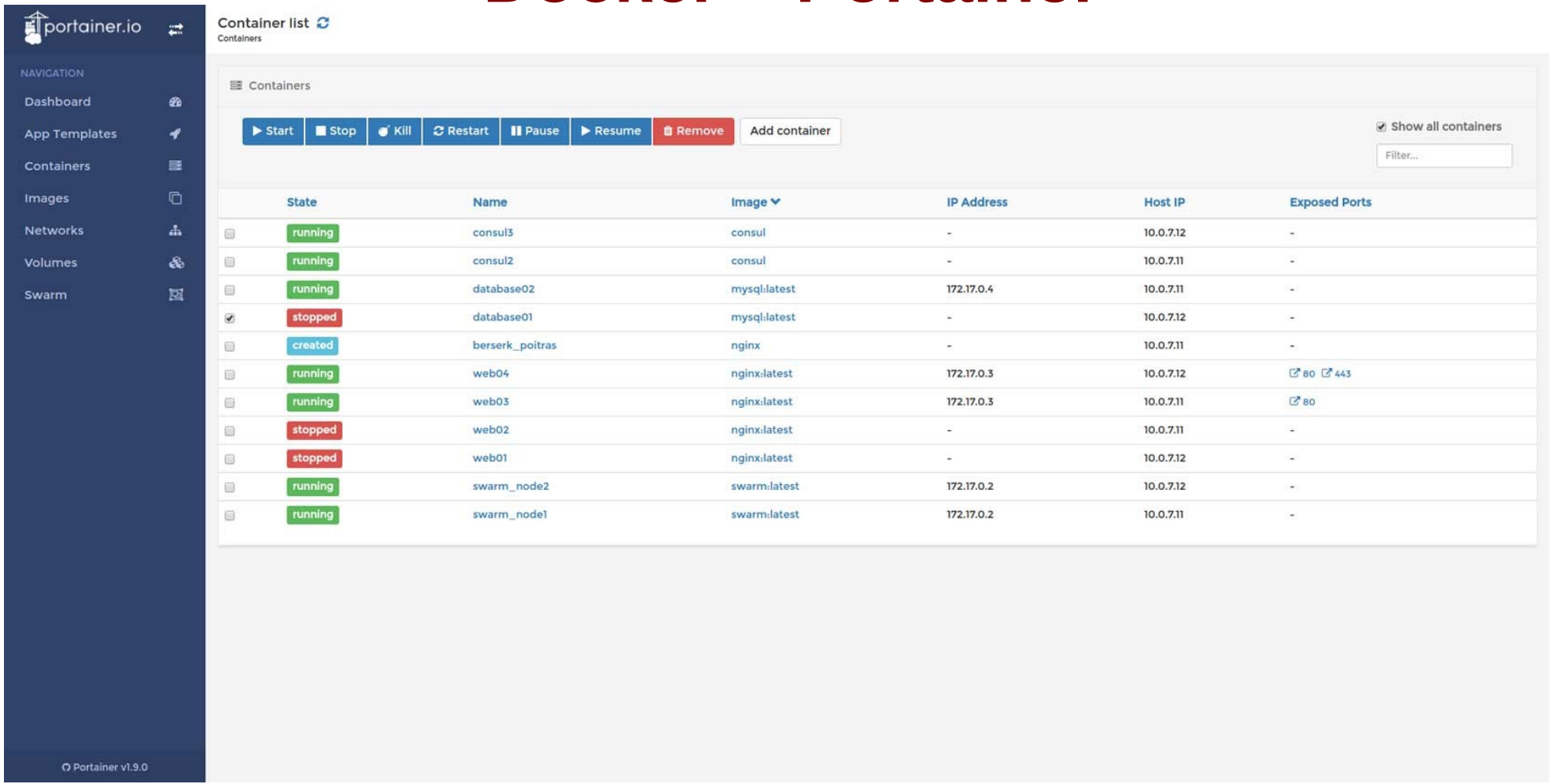
  `# docker ps`

```
root@las:~/wordpress# docker-compose up -d
Creating network "wordpress_default" with the default driver
Creating volume "wordpress_db_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
f7ec5a41d630: Pull complete
9444bb562699: Pull complete
6a4207b96940: Pull complete
181cefd361ce: Pull complete
8a2090759d8a: Pull complete
15f235e0d7ee: Pull complete
d870539cd9db: Pull complete
7310c448ab4f: Pull complete
4a72aac2e800: Pull complete
b1ab932f17c4: Pull complete
1a985de740ee: Pull complete
Digest: sha256:e42a18d0bd0aa746a734a49cbbcc079ccdf6681c474a238d38e79dc0884e0ecc
Status: Downloaded newer image for mysql:5.7
Pulling wordpress (wordpress:latest)...
latest: Pulling from library/wordpress
f7ec5a41d630: Already exists
941223b59841: Pull complete
a5f2415e5a0c: Pull complete
b9844b87f0e3: Pull complete
5a07de50525b: Pull complete
caeca1337a66: Pull complete
5dbe0d7f8481: Pull complete
a12730739063: Pull complete
fe0592ad29bf: Pull complete
c3e315c20689: Pull complete
8c5f7fdfcedf: Pull complete
8b40a9fa66d5: Pull complete
81830aebb3f8: Pull complete
7b04d4658443: Pull complete
0e596b6c428e: Pull complete
ec84879c7faf: Pull complete
5f211a0d2061: Pull complete
47c48169dcd4: Pull complete
19b34857b097: Extracting [=====================>                             ]  6.881MB/15.58MB
78810a623bdc: Download complete
```

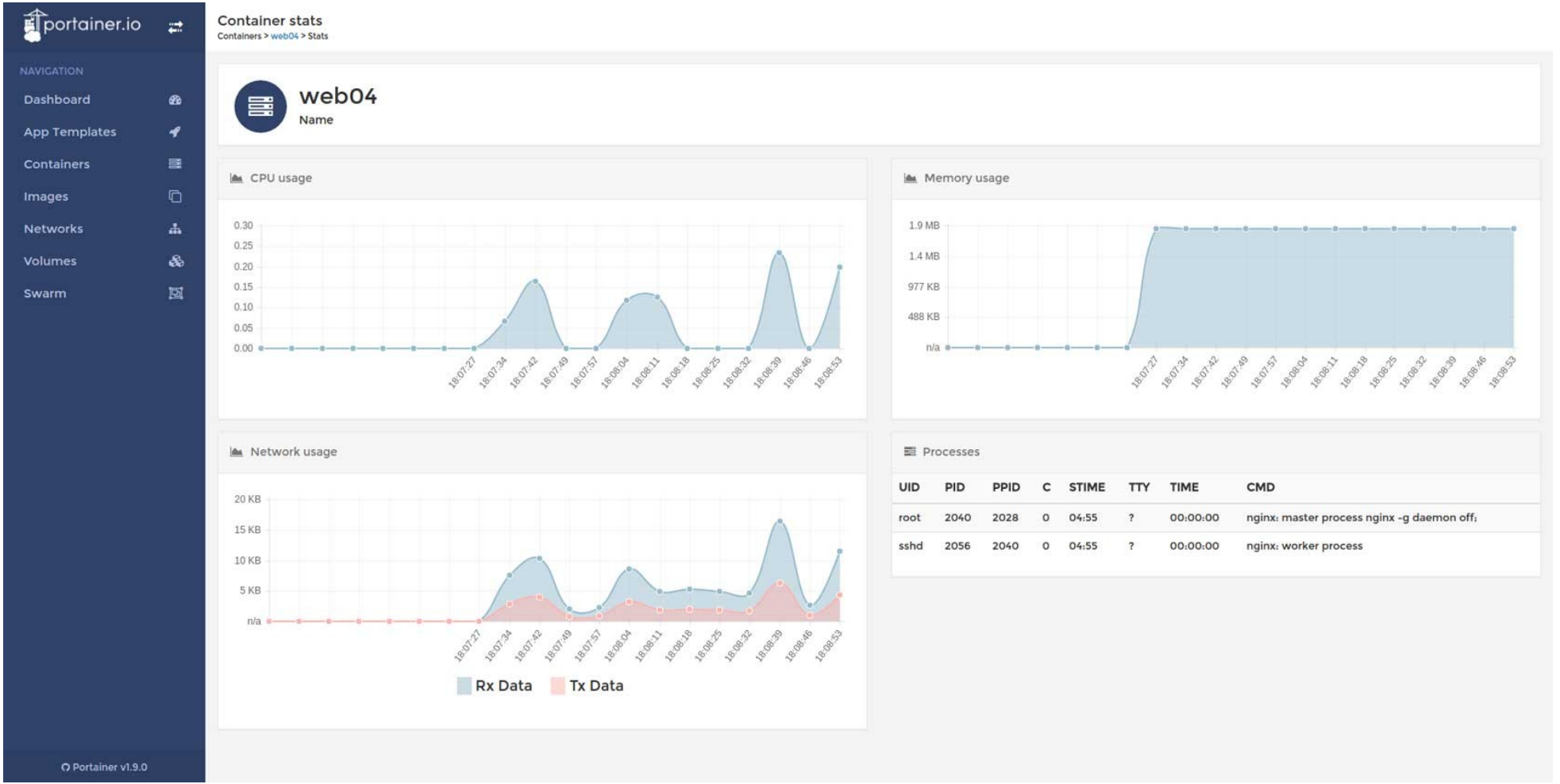
Docker – Portainer

- **Portainer** è una soluzione Open Source e multipiattaforma che mette a disposizione un'interfaccia utente con cui gestire più facilmente **Docker:** consente di amministrare **container**, **immagini**, **network** e volumi attraverso una dashboard in grado di rendere visibili i dettagli relativi a tutte le entità gestibili.
- Funziona su tutti i sistemi operativi o quasi.
- L'**interfaccia di Portainer** consente di visualizzare le statistiche in tempo reale, potendo monitorare l'utilizzo delle risorse fornite da CPU e memoria mentre vengono consumate, oppure verificando lo stato delle attività di networking e dei processi funzionanti all'interno di ciascun **container**.

Docker – Portainer



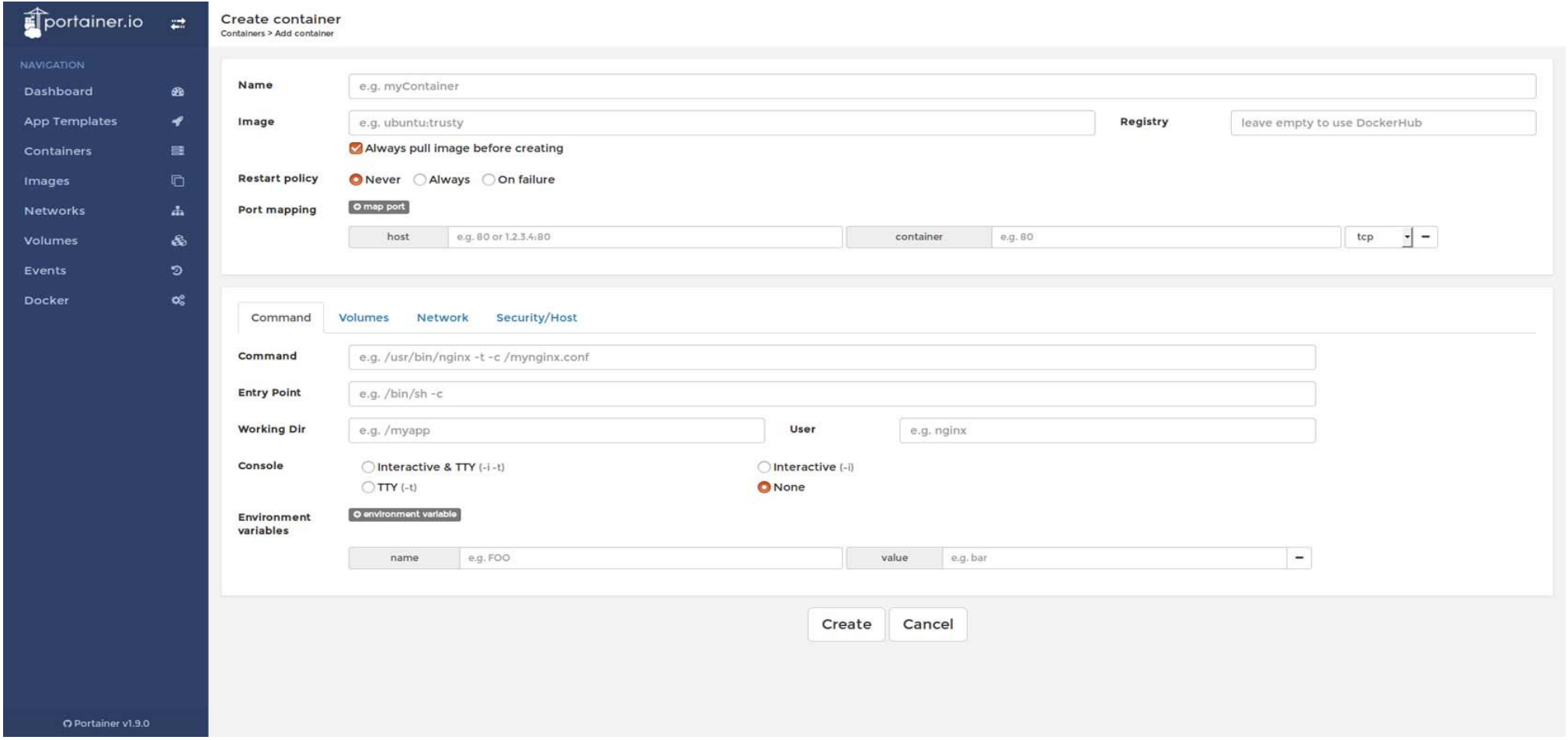
Docker – Portainer



Docker – Portainer

- **Portainer** consente di:
  - Amministrare la struttura esistente.
  - Creare nuovi **container**.
  - Effettuare fasi di deploy semplificati utilizzando un apposito sistema basato sui **template**.
- I **template** contengono diversi i modelli pronti all'uso per la messa in produzione di applicazioni e piattaforme diffusamente utilizzate: database (MySQL, MongoDB, PostgreSQL, MariaDB..), in-memory data store (Redis), CMS (WordPress, Joomla, Drupal..), Web server (Httpd, nginx..), media server e piattaforme per la messaggistica.

Docker – Portainer



Docker – Portainer, Installazione

- **Portainer** è già esistente come immagine **Docker**. Quindi l'installazione consiste nell'effettuare un pull di un nuovo **contenitore**:

```
# docker volume create portainer_data
# docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --
restart=always -v /var/run/docker.sock:/var/run/docker.sock -v
portainer_data:/data portainer/portainer-ce
```

- Ora è sufficiente accedere alla porta `9000` del server **Docker** su cui è in esecuzione **portainer** utilizzando un browser.
- Nota: la porta `9000` è la porta generale utilizzata da **Portainer** per l'accesso all'interfaccia utente. La porta `8000` viene utilizzata esclusivamente dall'agente EDGE per la funzione di tunneling inverso.