SLIDE 12

* docker run hello-world
* docker run -it alpine /bin/sh

SLIDE 18

docker inspect redis

SLIDE 28

* + docker image ls -> Per visualizzare la lista delle immagini presenti nel nostro ambiente Docker locale
  + docker image oppure docker images --help -> help
  + docker image pull oppure docker pull ->Per effettuare il pulling di un’immagine da un registro
  + docker images -q -> Restituisce solo l’ID numerico dell’immagine.
  + docker images -f -> permette di inserire un filtro e selezionare solo le immagini richieste.
  + docker rmi oppure docker image rm -> Per rimuovere un’immagine dall’ambiente Docker locale.
  + docker inspect nome\_immagine:tag -> Per avere delle informazioni dettagliate su una certa immagine Docker.
  + docker search stringa -> Ricerca all’interno del Docker Hub tutti i repository che contengono la stringa specificata nel comando.
  + docker run nome\_immagine -> Permette di eseguire un container a partire da una certa immagine specificata. Vedremo nel dettaglio
  + docker history nome\_immagine -> Permette di visualizzare lo storico delle operazioni svolte relativamente ad un determinata immagine.

SLIDE 31

* > docker run –it alpine /bin/sh
* # ps –elf

SLIDE 32

* > docker ps (o docker container ls) vedo che non ci sono container attivi
* > docker run alpine sleep 10 -> lo tiene attivo per 10 sec
* Avviare più container a partire da un’immagine
  + > docker run --name=docker1 alpine sleep 10
  + > docker run --name=docker2 alpine sleep 10
  + > docker ps
* Interazione
  + > docker run -it alpine /bin/sh ->crea una console interattiva di tipo bash
  + si entra quindi nella shell e posso interagire
  + per uscire dal container mantenendolo attivo ctrl+p e ctrl+q
  + > docker ps
  + > docker stop [ID\_CONTAINER]
  + Per rientrare nel container: > docker container exec –it [ID\_CONTAINER] /bin/bash
  + Oppure: > docker attach [ID\_CONTAINER]

SLIDE 33

* > docker ps -> elenco container
* > docker ps -a -> elenco container, anche quelli stoppati
* > docker container run [image] [app] -> avvia un nuovo container a partire da un’immagine con il servizio indicato
* > docker rm [ID\_CONTAINER] -> per rimuovere il container (stoppare un container prima di rimuoverlo per mantenere una corretta gestione dei processi)
* > docker container start -i [ID\_CONTAINER] -> start il container stoppato
* > docker container stop [ID\_CONTAINER] -> stop del container stoppato
* > docker container prune -> rimuove tutti i container stoppati
* Per avviare un container in background utilizzare il flag –d
  + > docker run -d -it alpine /bin/sh
* Per rientrare nel container:
  + > docker attach [ID\_CONTAINER]
  + > docker container exec –it [ID\_CONTAINER] [app] (crea un processo dedicato e migliora l’interattività col container)

SLIDE 35

* > docker run -d -it ubuntu /bin/bash
* > docker ps
* > docker attach [ID\_CONTAINER]
* Creiamo un file
* Stoppiamo il container
  + Ctrl+p e Ctrl+q
  + > docker stop [ID\_CONTAINER]
  + Verifichiamo con > docker ps
* Riavviare il container con > docker start [ID\_CONTAINER]
* Entriamo e verifichiamo che ci sia ancora il file

SLIDE 36

* > docker run –d –it –restart always ubuntu /bin/bash
* > docker ps
* Stoppiamo il servizio docker e riavviamo: si dovrebbe riavviare automaticamente

SLIDE 38

* > docker run –d –p 80:80 nginx
* > docker ps
* 0.0.0.0. -> qualsiasi host
* Andare con un browser su http://localhost
* Stoppare il container
* > docker run –d –p 8080:80 nginx
* > docker ps
* Andare con un browser su http://localhost:8080

SLIDE 39

* > docker system info

SLIDE 43

* > docker volume –help
* > docker volume create primovolume
* > docker volume ls
* > docker volume inspect primovolume
* > docker volume rm primovolume
* > docker volume ls
* > docker images
* > docker volume create volumeprimo
* > docker volume ls
* > docker run -it –name testContainer –v volumeprimo:/test ubuntu bash
* > docker ps
* > docker rm [ID\_CONTAINER]
* > docker ps –a
* Verificare che il volume esista ancora (> docker volume ls)

SLIDE 44

* > docker run –it --name container1 –v volumetre:/test ubuntu bash
* Usciamo dal container
* > docker volume ls
* > docker run –it --name container2 –v volumetre:/test ubuntu bash
* Volume3 è condiviso

BIND

docker run -it –v D:/ciccio:/test2 ubuntu bash

SLIDE 46

* BUILD -> costruisci
* SHIP -> condividi
* RUN -> fai girare

SLIDE 50

FROM

* Creiamo una directory
* Creare un Dockerfile nellacartella
  + FROM alpine
  + CMD ["echo", "Hello World"]
* > docker build . //ricerca il dockerfile
  + Costituzione di una prima immagine a partire dalla distro alpine
  + Aggiunta del comando cmd
  + Quindi ha creato l’immagine finale con l’id dell’immagine
* > docker image ls
* > docker run --name containerTest [ID\_IMAGE]
  + Ovviamente il container esiste in funzione del processo principale che è nel container stesso, quindi stampa Hello World e si chiude

COPY

* Creiamo una directory
* Creare un file script1.sh che contiene "echo Hello World« (nella cartella)
* Creare un Dockerfile nella stessa cartella
  + FROM alpine
  + COPY script1.sh /script1.sh
  + CMD ["sh", "script1.sh"]
* > docker build .
* > docker run --name containerTest [ID\_IMAGE]

RUN (esegue il comando in un nuovo layer)

* Creiamo una directory
* Creare un Dockerfile nella cartella
  + FROM ubuntu
  + RUN apt-get update
  + RUN apt-get install –y wget
* > docker build -t imagewget .
* > docker run -it --name containerwget imagewget
  + Posso provate il comando wget (wget www.google.com)

CMD

* Creiamo una directory
* Creare un Dockerfile nella cartella
  + FROM ubuntu
  + RUN apt-get update
  + RUN apt-get install –y curl
  + RUN touch ip.txt
  + RUN curl ifconfig.me/ip –o "ip.txt«
  + CMD ["cat","ip.txt"]
* > docker build -t imagecurl .
* > docker run imagecurl

ENTRYPOINT

* Creiamo una directory
* Creare un Dockerfile nella cartella
  + FROM ubuntu
  + RUN apt-get update
  + RUN apt-get install –y iputils-ping
  + ENTRYPOINT ["ping","-c","5"]
* > docker build .
* > docker run [IMAGE ID] [www.google.com](http://www.google.com/)

SLIDE 56

* > docker network ls
* > docker inspect network [ID\_RETE]

Installiamo il tool brctl (LINUX)

* $ brctl show

SLIDE 57

1. Verifica assegnamento IP
   * > docker run –it --name containerprimo ubuntu bash
     + $ apt -get update
     + $ apt -get install net-tools
     + $ ifconfig // e ci segnamo l’ip
   * > docker network ls
   * > docker inspect network bridge [o ID\_NETWORK]
   * Verifichiamo l’ip con la subnet
2. Comunicazione tra due Container
   * > docker run –it --name containerA ubuntu bash
     + $ apt -get update
     + $ apt -get install net-tools
     + $ ifconfig // e ci segnamo l’ip
   * > docker run –it --name containerB ubuntu bash
     + $ apt -get update
     + $ apt -get install net-tools
     + $ ifconfig // e ci segnamo l’ip
   * > docker network ls
   * > docker inspect network bridge [o ID\_NETWORK]
   * Verifichiamo che i due container abbiano la stessa sottorete
   * > docker exec –it containerA bash
     + $ apt -get install iputils-ping
     + $ ping [IP altro container]
3. Comunicazione verso l’esterno –NAT
   * > docker run –it --name containerA ubuntu bash
     + $ apt -get update
     + $ apt -get install net-tools
     + $ apt -get install iputils-ping
     + $ ifconfig // e ci segnamo l’ip
     + $ ping 8.8.8.8
     + Tramite NAT: CONTAINER (indirizzo ip privato) -> HOST (indirizzo ip privato) ->(indirizzo ip pubblico)
4. Port Mapping
   * > docker run –d --name webserver1 –p 8082:80 nginx
   * > docker ps
   * > docker port [ID\_CONTAINER]
   * Verifichiamo se il web server è accessibile tramite browser o curl
5. Creazione della propria rete
   * > docker network create reteTest
   * > docker network ls
   * > docker inspect network reteTest
   * > docker run –it --name containerA --network reteTest ubuntu bash
   * Verificare che abbia l’ip appartenente alla rete reteTest (o tramite ifconfig o tramite > docker inspect containerA)
6. Creazione della propria rete
   * Avviare 2 container sulla reteTest e 2 sulla rete di default
   * Pingare i nomi (non gli ip) dei container a 2 a 2

SLIDE 62

Creare una cartella composefile, accedere e creare un docker-compose.yml

Version: '3'

services:

web:

image: nginx

database:

image: redis

> docker-compose config (docker-compose per vedere l’help)

========================================================

> docker-compose up -d

> docker ps (i due container saranno up)

> docker-compose stop

> docker ps (i due container saranno down)

> docker ps –a (li vedo stoppati)

========================================================

Version: '3'

services:

web:

image: nginx

ports:

- 8090:80/tcp

database:

image: redis

> docker-compose up -d

> docker ps (i due container saranno up)

verificare le porte con un browser (o curl)

========================================================

Fare esempio con volume e networks

========================================================

SCALE

> docker-compose up –d --scale database=10 //scali e aumenti il numero di db a 10