DOCKER

All’inizio

Una applicazione in un server fisico

Poi virtualizzazione, ma ogni macchina virtuale vuole il suo sistema operativo che consuma risorse.

Quindi sono arrivati i container che pacchettizzano una applicazione

I container all’interno dell’host fisico condividono lo stesso sistema operativo licenze ecc….

Installazione docker:

l’installazione docker installerà una macchina virtuale linux

per installare docker:

<https://store.docker.com/editions/community/docker-ce-desktop-windows>

cliccare su getdocker

dopodichè registrarsi al sito, queste le mie credenziali:

email:luca@powerapp.it

dockerid:dockerlully

password:dock20191972

Un container è un ambiente sicuro perchè isolato.

Un container è leggero.

Un container è una sorta di scatola in cui troviamo tutte le componenti fondamentali per far girare una applicazione.

Docker è uno degli engine per la gestione dei container.

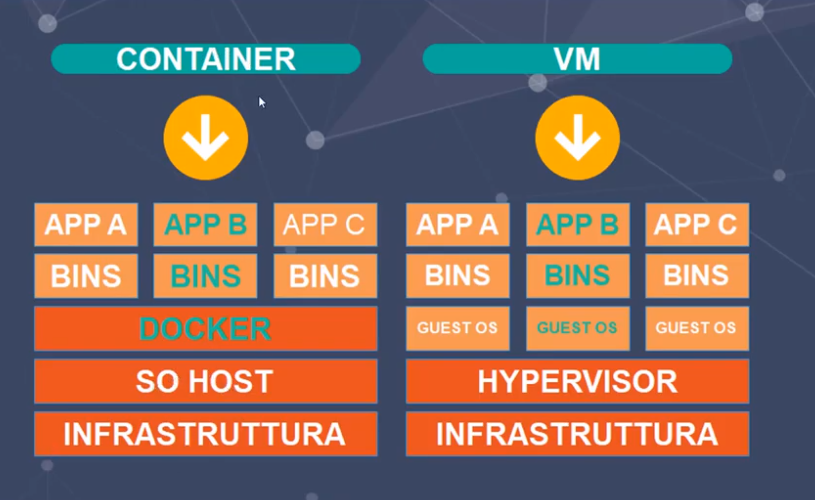
Per le macchine virtuali siamo obblicgati ad avere un hypervisor che fa da strato tra le macchine virtuali ed il sistema operativo host. E’ una sorta di orchestratore delle macchine virtuali.

In docker abbiamo il docker engine che si occupa di gestire i container.

docker run -p 8080:8080 -p 50000:50000 -–name jenkinsprimo -v /your/home:/var/jenkins\_home jenkins

docker run -p 8081:8080 -p 50000:50000 -–name jenkinssecondo -v /your/home:/var/jenkins\_home jenkins

per creare un volume sull’host fisico o agganciarne un esistente:



Docker è disponibile sia per linux che per windows.

Esistono due versioni di docker: la community (free che utilizziamo) e la enterprise

Docker è opnesource e si può trovare su github.

**Concetti di base**

Ad installazione completata troveremo un **docker client** ed un **docker daemon** (componenti principali)

**Le immagini**: sono delle entità contenenti sistema operativo ed una applicazione (una sorta di classe a partire dalla quale di creare i containers).

**Un container** è una istanza in esecuzione di una certa immagine.

L’immagine può essere scaricata esternamente o può essere costruita ad hoc.

**Dockerfile** descrive l’applicazione e descrive a docker come l’applicazione deve essere definita all’interno dell’immagine.

**Comandi base:**

**Docker images**: visualizza le immagini

Doker ps: visualizza I container

Docker ps -a (visualizza anche i container stoppati)

**Docker images pull nome immagine:tag\_immagine**

Se non specifichiamo nessun tag viene scaricata l’ultima versione.

**Docker run**

Il docker hub si trova qui:

hub.docker.com (da qui è possibile scaricare le immagini disponibili online (visualizzare il comando da lanciare, le eventuali tag da esplicitare)

Da un’immagine possiamo eseguire più container.

Se scrivo docker run hello-world lo eseguo

Per eseguire più container della stessa immagine devo dare un nome alle istanze:

docker run –name=nome immagine

docker run --name=immagine1 alphine

Si può poi entrare all’interno di un container con comandi specifici, un container poi può essere stoppato

Docker idcontainer stop.

Lo stop del container persistono fino a questo non viene eliminato, lo stop non comporta l’eliminazione dei dati.

Policies di restart:

ALWAYS, tutti i container stoppati brutalmente (non manualmente da noi) che contengono questa policies verranno riavviati all’avvio della macchina.

UNLESS STOPPED non riavvia il container anche se quest’ultimo è stato stoppato ed il docker daemon è stato riavviato.

ON FAILED: riavvia il container in stato non zero exit code.

**Come eseguire un container che contiene un webserver:**

Portmapping:

il comando -p 80:8080 associa la porta del container con la porta dell’host sul quale stiamo eseguendo docker.

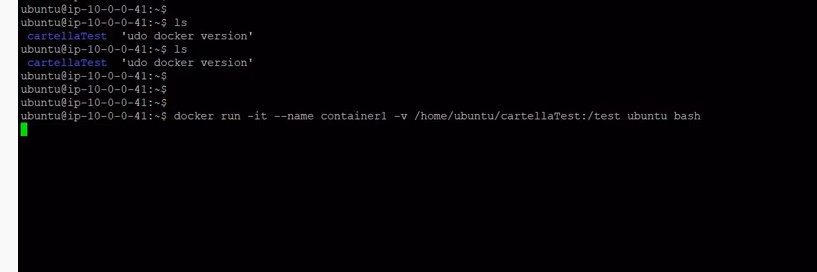
**La condivisione di volumi.**

Creiamo un container ed un volume nel container

Arrivato a lezione 13

È possibile creare un volume specificando un nume, un secondo container potrà volendo condividere lo stesso volume oppure specificando un path assoluto creeremo un volume sulla location fisica del nostro host, ad esempio sotto stiamo mappando una cartella fisica (/home/ubuntu/cartellatest su test del container

Il comando copy ci permette di copiare qualcosa dalla source al container e viceversa



**Il docker file**

Con docker image build la creeremo

Docker container run ??

Il docker file è un file di test che contiene le istruzioni per creare le immagini.

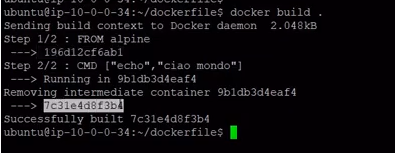
Quando definiamo una immagine è possibile partire da zero o da un’immagine esistente attraverso l’istruzione from.

Per creare una propria immagine:



Dopo il build andrebbe il dockerfiles ma se siamo nella stessa cartella basta questo: .

Una volta eseguito:



L’id dell’immagine serve poi per creare il container dall’immagine:



Il comando copy serve per copiare qualcosa dall’host al container e viceversa…

Tipo uno script dopo averlo creato…

L’inetrfaccia bridge (quella di default) si comporta come uno switch, tutti i container che eseguo senza specificare la rete vengono associati a tale rete con un indirizzo ip che appartiene a questa sottorete.

Container della stesso sottorete possono comunicare tra di loro.

Per ispezionare la rete scrivo





Per esempio potrei aver un webservice, un database ed una web application che all’interno della rete possono comunicare tra di loro.

Arrivato a 20