

Microserveis amb Docker

Visió general del curs

Guió del curs

- Primer dia:
 - Evolució de les architectures d'aplicacions TIC.
 - Introducció als microserveis.
 - Introducció als contenidors.
 - Introducció a Docker.
- Segon dia:
 - Creació d'imatges amb Docker.
 - Ús d'imatges.
 - Volums i persistència de dades.
 - Enllaç senzill entre contenidors.
- Tercer dia:
 - Composició de contenidors.
 - Configuració de connexions.
 - Orquestració de contenidors.
- Quart dia:
 - Docker swarm.
 - Kubernetes.
 - Rancher.

Microserveis amb Docker

Introducció

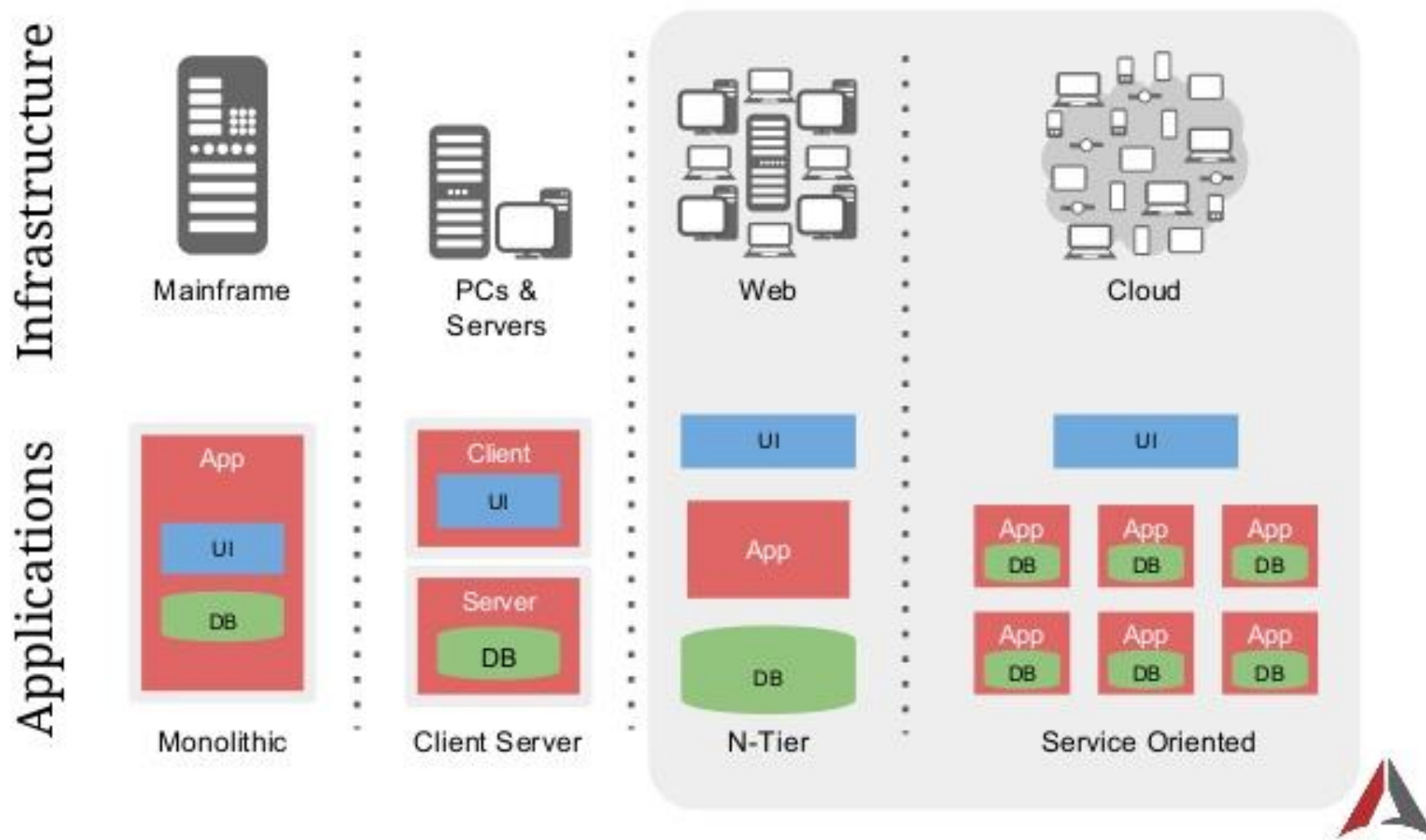
Índex

- Evolució de d'arquitectures d'aplicacions TIC.
- Introducció als microserveis.
- Microserveis usant contenidors.
- Contenidors amb Docker.
- Instal·lació de Docker.
 - Primers passos.
 - Ús d'imatges per a crear contenidors.
 - Creació d'imatges. Union File System.
 - Volumes i persistència.
 - Eliminació de recursos.
- Enllaç entre contenidors amb Docker.

Per començar: Anem instal·lant Docker...

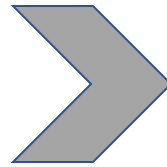
- Instruccions Ubuntu: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>
- Recomanacions:
 - Per aquest curs:
 - Instal·lació manual dels paquets descarregats que us lliurem.
 - Instal·lar els paquets .deb "docker-ce" "docker-ce-cli" i "containerd.io".
 - Descarregats de <https://download.docker.com/linux/ubuntu/dists/cosmic/pool/stable/amd64/>
 - Ús general:
 - Instal·lar utilitzant apt.
 - Servidors en producció
 - Instal·lació manual o per script, controlant les actualitzacions.

Evolució d'arquitectures d'aplicacions TIC



Arquitectura mainframe

- Pocs recursos computacionals, molt cars i difícils de gestionar.
- Pocs sistemes operatius i llenguatges de programació, i lenta evolució.
- Els usuaris no disposen de dispositius.
- Poca capacitat de interconnexió.




- Es concentra les aplicacions en recursos centralitzats.
- Facilitat de manteniment d'aplicacions basades en un únic SO i llenguatge.
- Com respondre a augment de demanda?



Solucions monolítiques

Arquitectura client/servidor

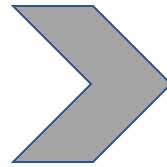
- Popularització de l'ordinador personal.
 - Es pot delegar responsabilitat al dispositiu final.
 - Molta més varietat de SO i llenguatges de programació.
 - Augmenta la capacitat de connexió.
- 
- Es delega la visualització al dispositiu final.
 - Es complica el manteniment de diverses versions d'SO i llenguatge.
 - Es pot donar servei a més clients sense un augment lineal de recursos.



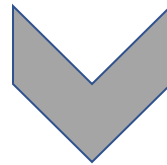
Separació entre visualització i procés/còmput

Arquitectura de capes

- Molts més dispositius de treball i visualització.
- Més potència de càlcul i emmagatzemament al client.
- Hardware especialitzat en còmput o emmagatzemament.
- Augment important de la capacitat de connexió.



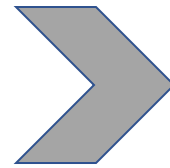
- Es poden delegar més operacions als clients.
- Es poden separar lògica de negoci i emmagatzemament.
- Es pot escalar “horitzontalment” segons les necessitats.
- S’especialitzen les funcions.



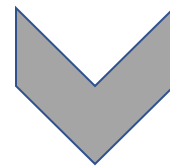
Separació funcional

Arquitectura orientada a servei

- Molta capacitat de connexió.
- Gran oferta de serveis especialitzats per funció.
- Creixement exponencial de l'ús de les aplicacions.



- Delegació de funcions a serveis especialitzats.
- Escalat per demanda de cada servei individual.
- Dificultat de gestió de la interconnexió.
- Dificultat de gestió de les dependències.



Separació funcional, física i lògica

Introducció als microserveis

- Definició:
 - S'anomena microserveis a una manera de desenvolupar i compondre sistemes software basada en la creació de petits components independents que interactuen entre ells a través de la xarxa.
 - Suposen un pas més en la orientació a servei, ja que porten el concepte al "límit".

Avantatges dels microserveis

- Especialització (Separació funcional):
 - Cada component s'especialitza en una única funció i la fa de la millor forma possible.
- Separació tecnològica:
 - El component utilitza la tecnologia més favorable al seu propòsit: Hardware, SO i llenguatge de programació.
- Aïllament (Separació física):
 - Cada component funciona de forma separada.
 - Utilitza els recursos necessaris i pot créixer de forma independent.
- Independència (Separació operacional):
 - Escalat en funció de la demanda.
 - Desplegament independent de noves versions: Agilitat.

Exemple de transformació a microserveis

- Aplicació de gestió d'infraccions de trànsit
 - Mainframe:
 - Sistema de gestió basat en AS400.
 - Tota la gestió es fa en un únic sistema.
 - Els operadors del sistema han de ser els funcionaris de l'administració.
 - La interfície per entrar la informació, el procés de còmput i la base de dades estan en un mateix sistema.
 - És senzill programar nous serveis, sempre que no es vulgui donar accés als usuaris finals.
 - En cas de requerir més processament, s'ha de comprar una màquina més gran (scale up).
 - El creixement serà independent de quina és la part que requereix més capacitat (interfície, lògica de negoci o emmagatzemament).

Exemple de transformació a microserveis

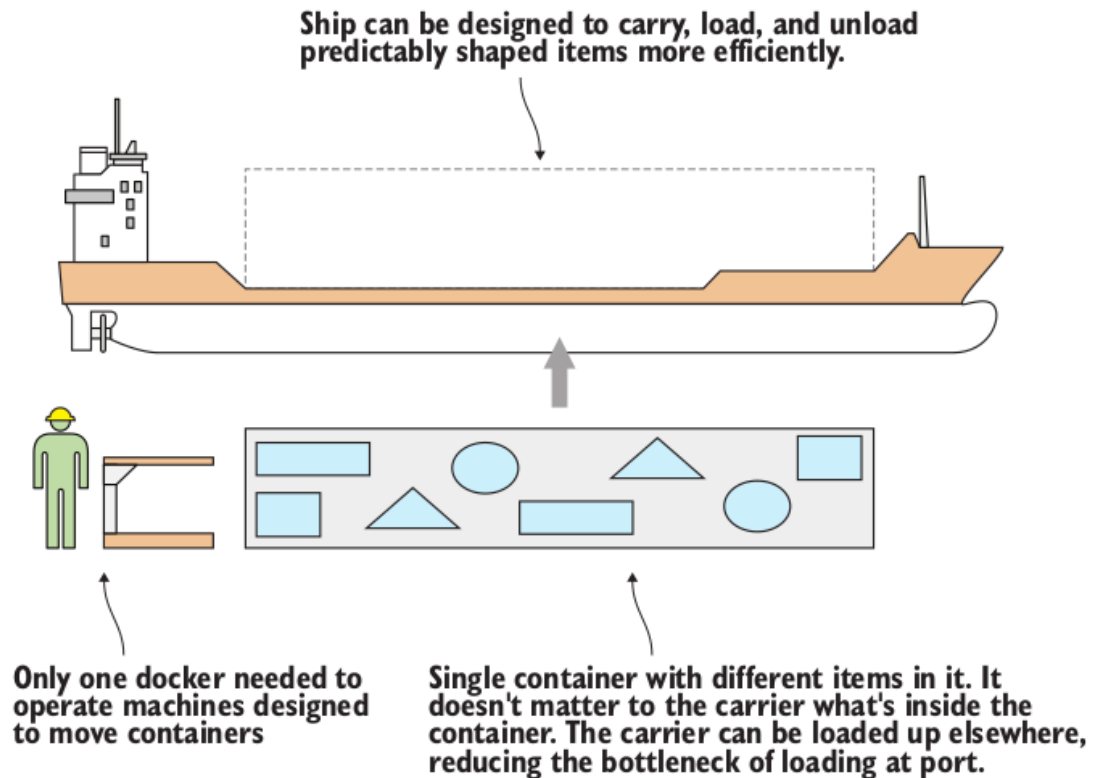
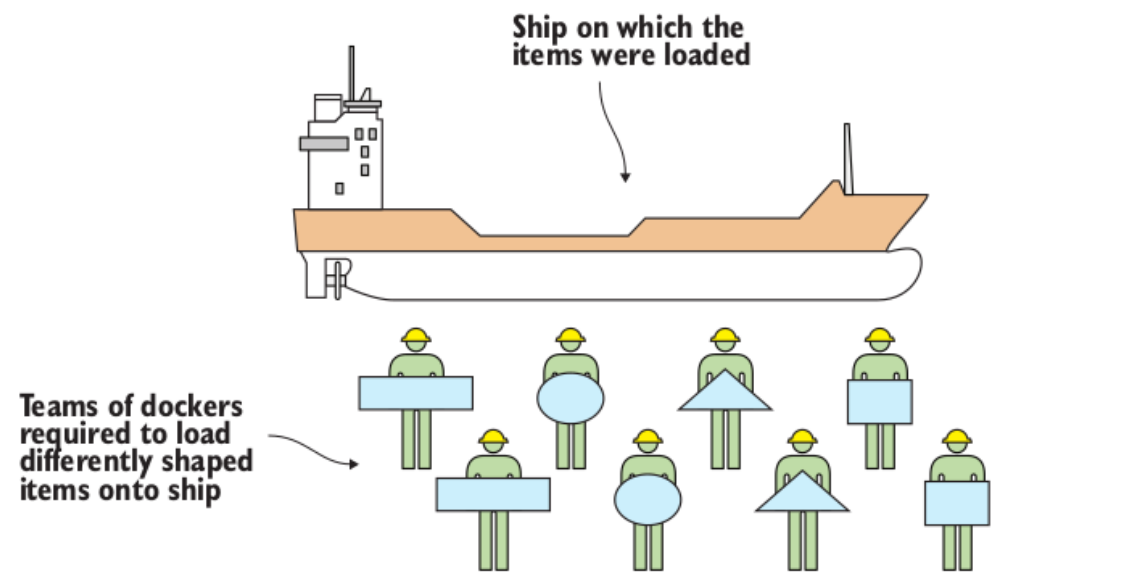
- Aplicació de gestió d'infraccions de trànsit
 - Microserveis:
 - Serveis:
 - Servei de base de dades central.
 - Servei de processament de pagaments en línia.
 - Servei de processament de pagaments bancaris.
 - Servei de processament de l'autenticació.
 - Servei d'enviament de missatges Push a clients.
 - Servei d'enviament de cartes postals.
 - Servei de gestió dels impagaments.
 - Servei de gestió de rebuts.
 - Avantatges:
 - Cada servei funciona per separat.
 - En cas de requereix més processament, es pot escalar el servei que ho requereix.
 - Es pot escalar utilitzant hardware més potent o més recursos (scale up) o afegint rèpliques del servei i repartint la càrrega (scale out).

Contenidors

- Què son?
- Com funcionen.
- Pq son una bona opció per a implementar microserveis: Comparació amb VM

Què son els contenidors?

- Un container és una **unitat** lleugera, independent i estàndard de software que **empaqueta** tot allò necessari per a que funcioni una aplicació (***codi, entorn d'execució, eines de sistema, llibreries i configuracions***).
- Aquesta unitat pot funcionar sense problemes en diferents entorns de computació, i es pot moure de forma ràpida i segura entre diferents entorns.

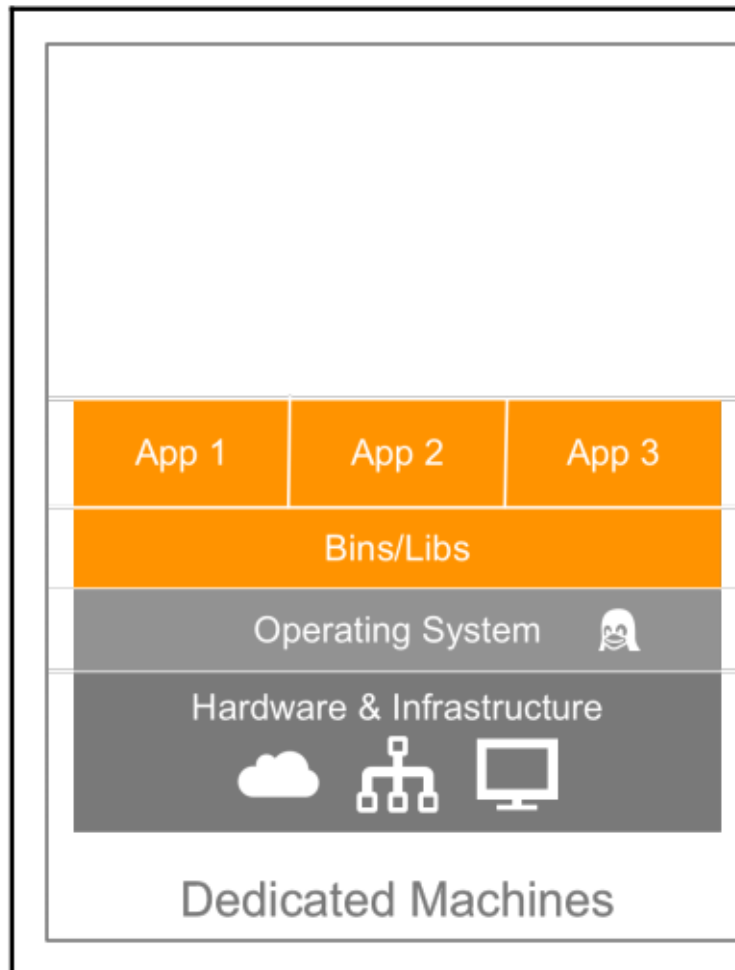


Servidors, màquines virtuals i contenidors

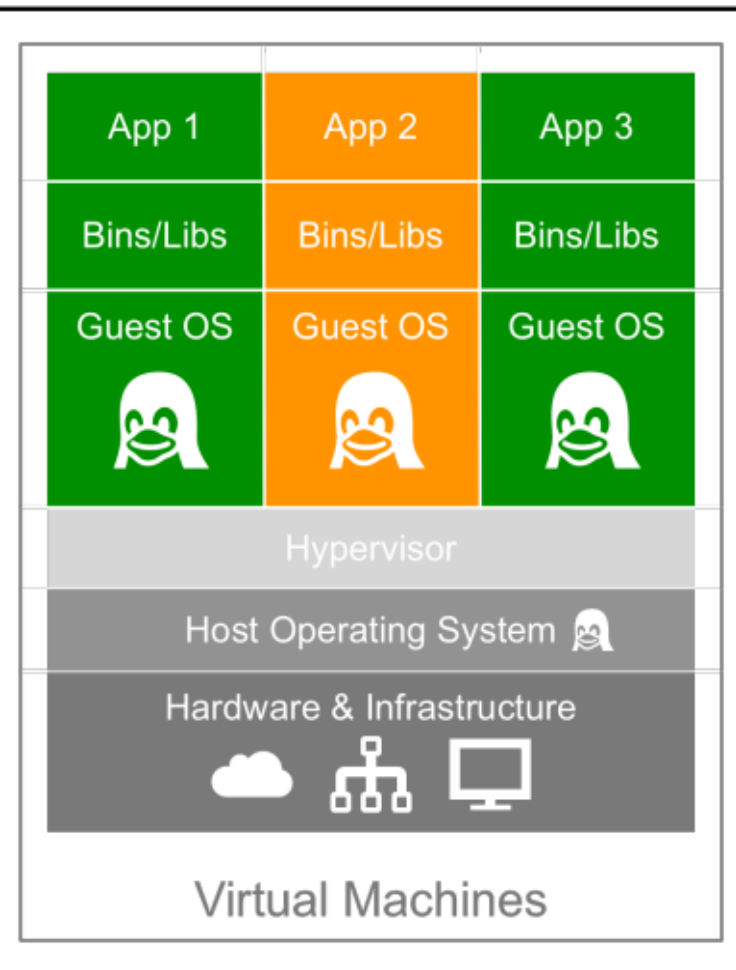
- Un servidor està compost per tot el seu hardware, un sistema operatiu i unes aplicacions que funcionen sobre aquest.
- Una màquina virtual requereix un hipervisor sobre el sistema operatiu "hoste" (host), que virtualitza el hardware del servidor i el fica a disposició dels sistemes operatius "convidats" (guest) que es creen sobre l'hipervisor.
- Un contenidor requereix un software sobre el sistema operatiu que li permet utilitzar els recursos hardware i software del servidor de base.

Servidores vs. Máquinas virtuais

Servidores

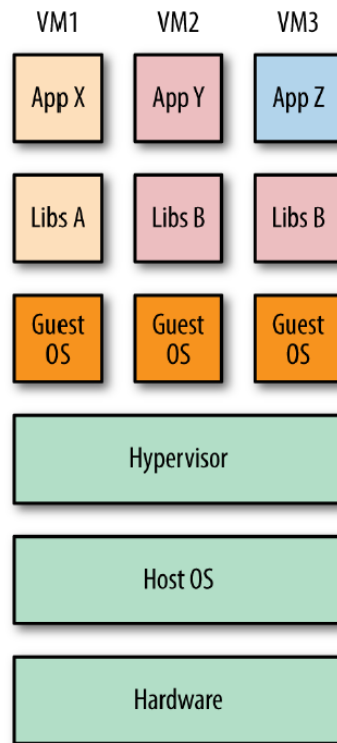


Máquinas virtuais

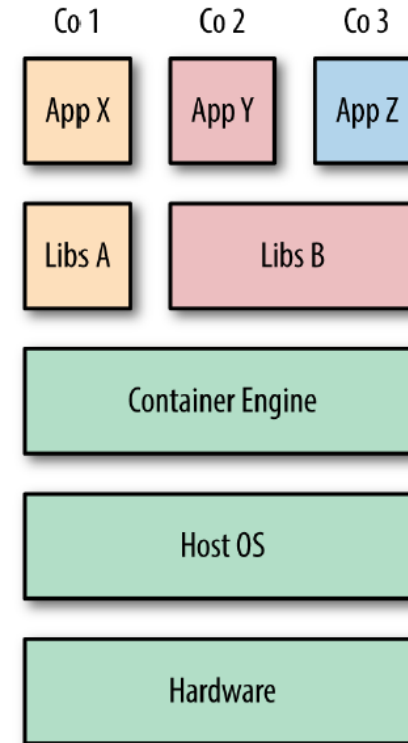


Contentidors vs. Màquines virtuals

Màquines virtuals



Contentidors



Contenidors vs. màquines virtuals

- Els contenidors comparteixen recursos amb el sistema operatiu, de forma que son més eficients.
- Una aplicació funcionant en un contenidor gairebé no comporta més “despesa” que funcionant nativament al sistema operatiu.
- Els contenidors es poden engegar molt ràpidament.
- Es poden executar molts contenidors en una única màquina.
- Es poden crear entorns molt complexos dins d'un contenidor, facilitant el lliurament i desplegament d'aplicacions.

Avantatges

- Encapsulament:
 - L'aplicació s'encapsula amb totes les seves dependències.
- Immutabilitat:
 - Les llibreries i dependències son sempre d'una determinada versió.
- Predicibilitat:
 - Tot allò que funciona en un contenidor, funcionarà igual sigui a on sigui que estigui funcionant el contenidor.
- Reaprofitament de recursos:
 - Es pot escalar recursos únicament quan son necessaris i del servei que ho requereix.

Introducció a Docker

- Imatges vs. contenidors.
- Execució de contenidors.
- Imatges.
- Creació d'imatges.
- Sistema de fitxers: Union FS i imatges inmutables.
- Execució d'imatges. Contenedor vs. Imatge.
- Registres i repositoris.
- Composició d'imatges.

Imatges vs. contenidors

- Programa vs. Procés o Objecte vs. Instància
- Els contenidors son instàncies concretes d'una imatge.
- Una imatge empaqueta un sistema de fitxers amb unes metadades.
- Un contenidor utilitza una imatge com a base i executa un procés sobre aquesta.
- Els canvis fets a un contenidor afecten a aquest contenidor, però no a la imatge que utilitza de base.

Contenidors Docker

- Docker Engine
 - Software per a la execució de contenidors.
 - Funciona en Linux de 64 bits.
 - Pot funcionar en Mac i Windows utilitzant una màquina virtual (docker machine).
- Docker Hub
 - Servei per a la descàrrega i compartició d'imatges.

Execució de contenidors

- `# docker run hello-world`
 - El client de docker contacta amb el procés "daemon" docker.
 - El procés docker cerca la imatge `hello_world` al registre local.
 - Si no la troba, la cerca i descarrega de Docker Hub.
 - El procés docker utilitza la imatge `hello_world` per a crear un nou contenidor i executa el procés que aquella imatge executa per defecte.
 - El procés docker connecta la sortida del contenidor al client de docker, per tal que aquest la mostri pel terminal.

Execució de contenidors

- Execució no interactiva
 - # docker run alpine "Hello world"
 - # docker run alpine /bin/echo "Hola que tal?"
- Execució interactiva
 - # docker run -it alpine /bin/sh
 - / # more /etc/alpine-release
 - / # exit

Inspeccionar contenidors

Consola 1

- # docker run -it alpine /bin/sh
- /# rm -rf /bin
- /# exit

Consola 2

- docker ps
- docker inspect *nom_contenidor*
- docker diff
- docker diff
- docker logs
- docker container start *nom_contenidor*

Eliminar contenidors

- # docker ps
- # docker ps -a
- # docker rm *nom_contenidor*
- Eliminar tots els recursos no utilitzats
 - # docker container prune
 - # docker image prune

Imatges docker

- Les imatges utilitzen el sistema de fitxers Union file System.
- Sistema de fitxers que funciona per “capes”.
- Cada capa es “munta” en mode read only.
- Cada capa “amaga” els fitxers amb mateix path de la capa anterior.
- Quan una imatge es converteix en contenidor (mitjançant *docker run* o *docker create*) Docker Engine agafa la imatge i li afegeix a sobre una nova capa en mode read/write, sobre la que s’executa qualsevol canvi.
- IMPORTANT:
 - Cada canvi en una imatge (RUN apt-get update) genera una nova capa.
 - Esborrar continguts en capes superiors no estalvia espai.

Images docker

- # docker run -it --name mycontainer alpine
 - /# apk update
 - /# apk add fortune
 - /# fortune
 - /# exit
- # docker commit myalpine test/myalpine
- # docker run test/myalpine fortune

Imatges docker amb Dockerfile

- # mkdir myalpine
- # cd myalpine
- # touch Dockerfile
- # vi Dockerfile

From alpine

RUN apk update && apk add fortune

- # docker build -t test/myalpine-df .
- # docker run test/myalpine-df fortune

Registres i repositoris

- **Registre:**
 - Servei responsable d'hostatjar i distribuir imatges.
 - Per defecte és Docker Hub.
- **Repositori:**
 - Col·lecció d'imatges relacionades.
- **Tag:**
 - Identificador alfanumèric afegit a una imatge per a identificar una versió específica.
- **# docker pull dbarroso/curs-docker:2.0**
 - Baixar la versió 2.0 del repositori dbarroso/curs-docker del registre per defecte.

Registres i repositoris

- Per a descarregar imatges de Docker Hub no cal tenir usuari.
- Per a pujar imatges cal crear registrar-se i autenticar-se quan es puja.

Composició de contenidors

- `# docker run --name myredis -d redis`
- `# docker run --rm -it --link myredis:redisserver redis /bin/bash`
- `root@ca567usd:/data# redis-cli -h redisserver -p 6379`
- `redis:6379> ping`
- `PONG`
- `redis:6379> set "abc" 123`
- `Ok`
- `redis:6379> get "abc"`
- `"123"`
- `redis:6379> exit`

Volumes

- `# docker run -it -v /tmp:/data alpine`
 - `/# vi /data/prova`
 - Escriure alguna cosa
 - `/#exit`
-
- Comprovar que al directori `/tmp` del host existeix el fitxer `prova` i conté el que s'ha escrit des del contenidor.