Contingut

DOCKER.

Introducció:

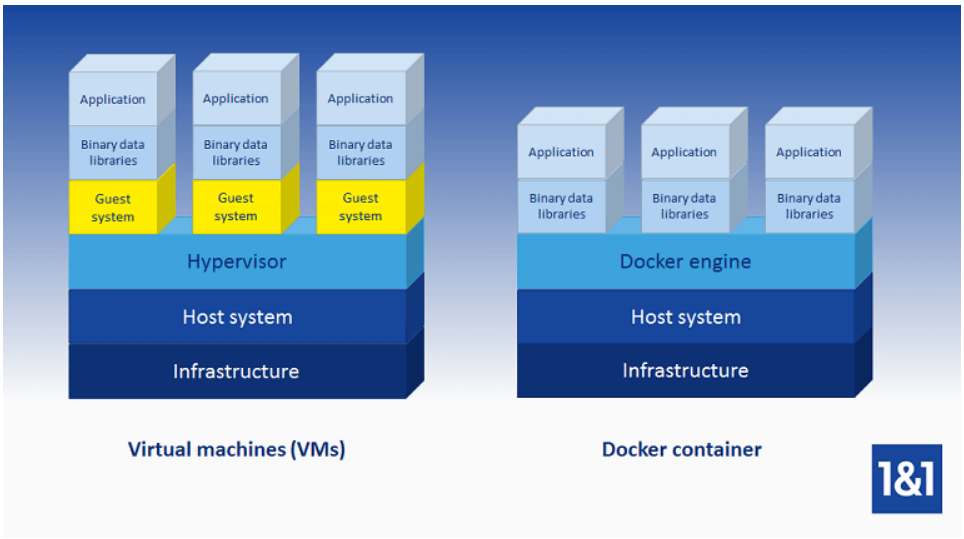
Docker és un projecte de codi obert que automatitza el desplegament d'aplicacions dins de contenidors de programari, proporcionant una capa addicional d'abstracció i automatització de virtualització d'aplicacions en múltiples sistemes operativos.
Docker utilitza característiques d'aïllament de recursos del nucli Linux, com ara cgroups i espais de noms (namespaces) per permetre que "contenidors" independents s'executen dins d'una sola instància de Linux, evitant la sobrecàrrega d'iniciar i mantenir màquines virtuals.

- Els Cgroups limiten l'accés de processos a la memòria, la CPU i els recursos I/O de manera que eviten que les necessitats pel que fa a recursos d'un projecte concret afectin a altres processos en execució.
- Els Namespaces (espais de noms) limiten els processos i els seus processos fill a una secció específica del sistema subjacent i són utilitzats per Docker per encapsular projectes en cinc camps concrets:
  - Identificació de sistemes (UTS): els espais de noms UTS s'utilitzen en la virtualització basada en contenidors per assignar als contenidors el seu propi nom de domini i d'equip.
  - ID de procés (PID): cada contenidor Docker fa servir un espai de noms independent per als identificadors (ID) de procés. Tots aquells processos que tinguin lloc fora del contenidor no es visualitzen des de l'interior d'aquest, el que permet que els processos encapsulats en contenidors dins d'un mateix sistema host posseeixen el mateix PID sense que això causi cap problema.
  - Comunicacions entre processos (IPC): els espais de noms IPC aïllen processos en un contenidor, impedint la comunicació amb processos fora del contenidor.
  - Recursos de xarxa (NET): els espais de noms network permeten adjudicar a cada contenidor recursos de xarxa separats, com adreces IP o taules d'enrutament.
  - Punts de muntatge dels sistemes de fitxers (MNT): gràcies a aquests sistemes de noms, un procés aïllat no veu mai el sistema de fitxers de l'amfitrió complet, sinó que accedeix només a una petita part d'aquest, en general una imatge creada concretament per a aquest contenidor.

El suport del nucli Linux per als espais de noms aïlla la vista que té una aplicació del seu entorn operatiu, incloent arbres de procés, xarxa, ID d'usuari i sistemes d'arxius muntats, mentre que els cgroups del nucli proporcionen aïllament de recursos, incloent la CPU, la memòria, el bloc d'E/S i de la xarxa. Des de la versió 0.9, Docker inclou la biblioteca libcontainer com la seva pròpia manera d'utilitzar directament les facilitats de virtualització que ofereix el nucli Linux, a més d'utilitzar les interfícies abstretes de virtualització mitjançant libvirt, lxc (Linux Containers) i systemd-nspawn.

Mentre que la virtualització tradicional de maquinari es basa en iniciar diferents sistemes convidats en un mateix sistema amfitrió (host), amb Docker les aplicacions s'executen com a processos aïllats dins d'un mateix sistema gràcies als denominats contenidors. Es parla llavors d'una virtualització basada en contenidors i, per tant, també d'una virtualització a nivell de sistema operatiu.

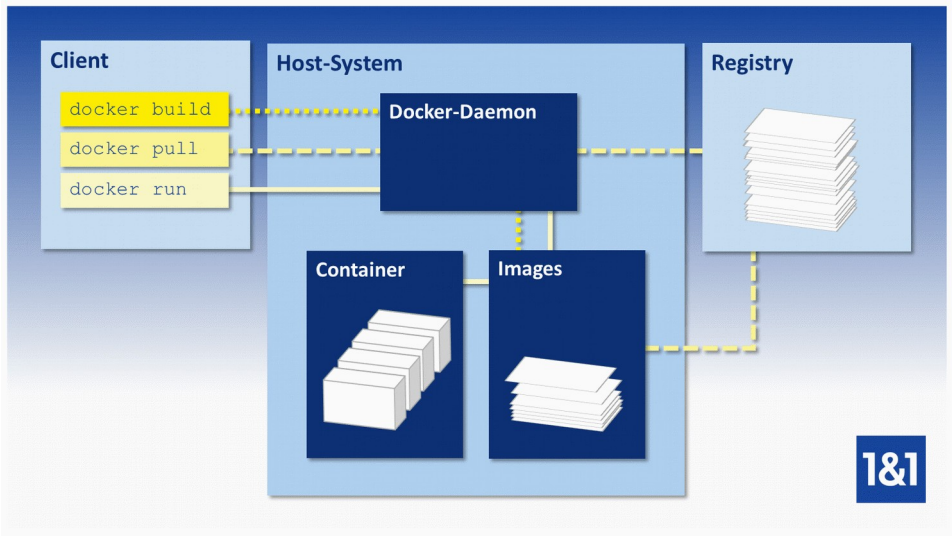
En el gràfic es poden veure les diferències principals en l'arquitectura de les dues tècniques de virtualització:



Imatges Docker

De forma molt similar a les màquines virtuals, els contenidors de Docker es basen en imatges, que són plantilles de només lectura amb totes les instruccions que necessita el motor de Docker per crear un contenidor. Com còpia portàtil d'un contenidor, una imatge Docker es descriu en forma d'arxiu de text (Dockerfile). Abans d'iniciar un contenidor en un sistema, es carrega un paquet amb la imatge corresponent si aquesta no està ja guardada de forma local. La imatge carregada prepara tots els sistemes d'arxius amb els paràmetres necessaris per a l'execució. Un contenidor pot considerar-se com un procés en execució d'una imatge.

A la següent imatge es pot veure la forma en què els components individuals de Docker es combinen prenent com a exemple les ordres docker build, docker pull i docker run:



La comanda docker build dóna instruccions al dimoni per crear una imatge (línia puntejada), per a això ha d'estar disponible el Dockerfile corresponent. Si l'usuari no ha creat la imatge, sinó que la presa d'un repositori en Docker Hub, llavors s'executa la comanda docker pull (línia discontínua). Quan se li ordena al dimoni iniciar un contenidor amb l'ordre docker run, el programa comprova primer si la imatge requerida està emmagatzemada de forma local. En cas afirmatiu, el contenidor s'inicia (línia contínua). També pot passar que el dimoni no trobi la imatge, a partir de la qual cosa s'extreu una directament del repositori.

Instal·lació de Docker:

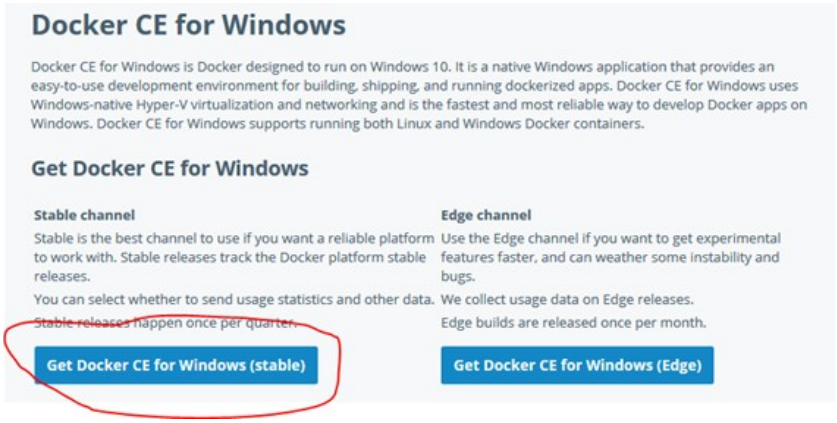
Actualment docker es pot instal·lar sobre qualsevol sistema operatiu Linux, Windows o IOS.
Docker maneja dues versions una que és de pagament per a empreses EE (Enterprise Edition) i l'altra és la versió gratuïta que és la de la comunitat CE (Community Edition)

1.Docker sobre windows:

Microsoft no volia quedar-se fora d'aquesta tecnologia i arriba a un acord amb Docker el 2015 per convertir-se en partner oficial i d'aquesta manera, incorporar aquesta tecnologia als seus sistemes operatius Windows Server 2016 i Windows 10 (encara que no en totes les versions està suportat) .
De fet, podem fer funcionar Docker en Windows 7 i Windows 8 fent servir VirtualBox, la novetat ara és que podem usar Docker de forma nativa a Windows 10 sense necessitat d'altres eines com Docker ToolBox.
si volem instal·lar Docker hem de complir certs requisits en Windows 10:

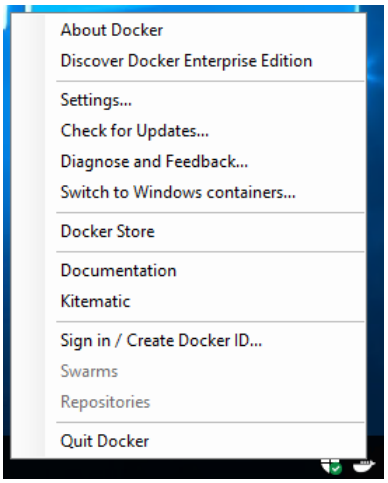
- Windows 10 versió Professional o Enterprise.
- Actualitzat amb Anniversary Edition o Creators Update.
- Updates crítics per a Windows Containers.
- Compilació igual o superior a 14393.222.
- Virtualització activada en el nostre equip

En el cas de no tenir virtualització activada, Docker detectarà aquesta característica en el procés d'instal·lació i ens preguntarà per instal·lar-la de forma automàtica.

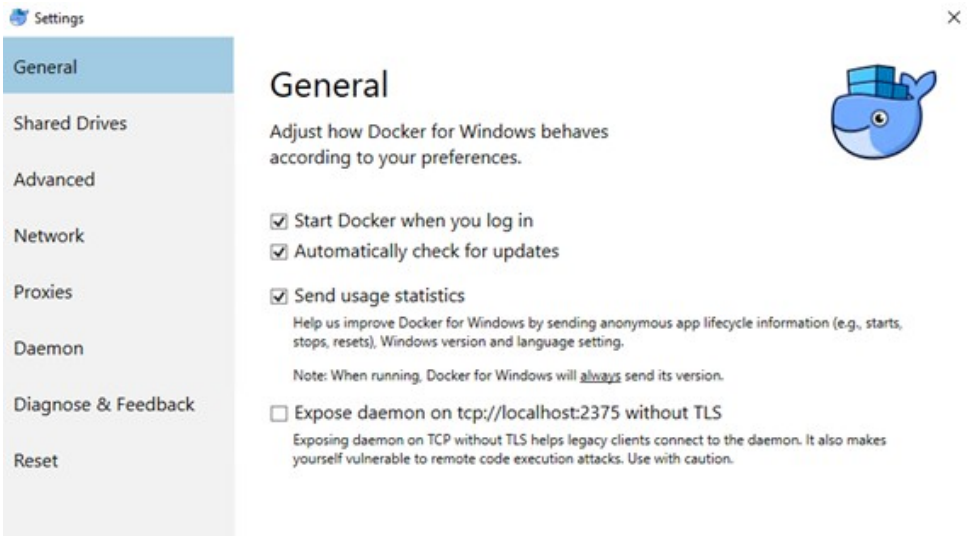


l'instal·lem i reiniciem, apareix la icona:

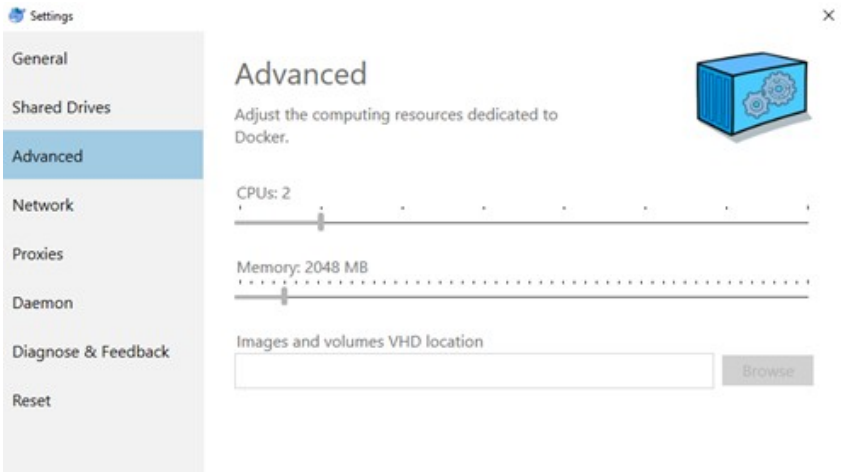
Finalment, un cop acabat el procés anterior, ja estem preparats per utilitzar Docker, Si premem botó dret sobre la icona de la barra de tasques veurem les opcions de Docker:



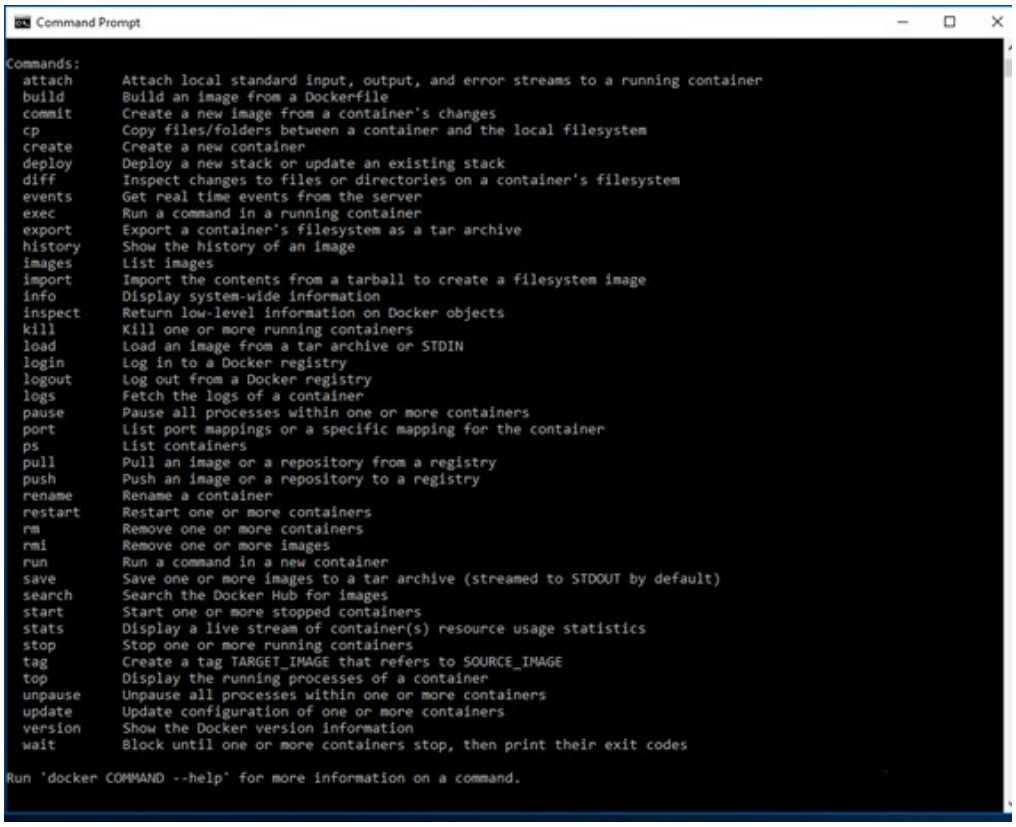
Si ara premem a "Settings ..." veurem la configuració inicial definida de Docker. Si volem que Docker no arranque amb el nostre Windows 10 el podem desmarcar aquí, la resta d'opcions ja vénen configurades i enllestides per a funcionar.



En l'opció "Advanced", veurem els ajustos per defecte que Docker reserva de manera dedicada per gestionar les seves imatges i contenidors, també ho podem canviar si fos necessari:



Obrim el interpret de comandes de Windows i executem docker –help, ens mostrarà totes les comandes disponibles.



2.Docker sobre IOS:

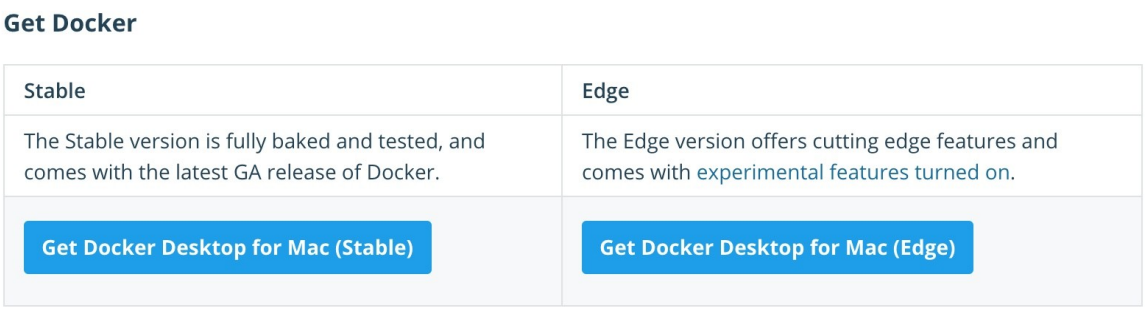
Docker Desktop per a Mac és una aplicació d'escriptori fàcil d'instal·lar per construir, depurar i provar aplicacions dockeritzades en un Mac. Docker Desktop per a Mac és un entorn de desenvolupament complet integrat amb el framework, xarxa i sistema de fitxers de l'Hipervisor de MacOS.
Docker Desktop: és la manera més ràpida i fiable per executar Docker en un Mac.

Requisits del sistema: Docker Desktop for Mac només es llança si es compleixen tots aquests requisits.

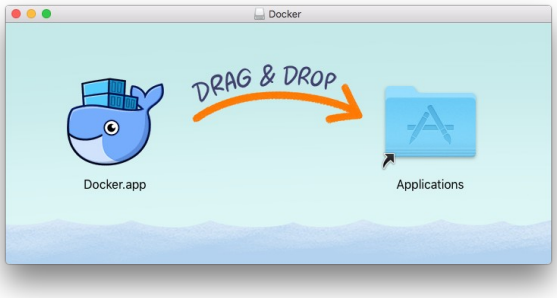
- El maquinari de Mac ha de ser un model de 2010 o més recent, amb el suport de maquinari d'Intel per a la virtualització de la unitat de gestió de memòria (MMU), incloses les taules de pàgines esteses (EPT) i el mode sense restriccions. Podeu comprovar si la vostra màquina té aquest suport executant el següent comandament en un terminal: sysctl kern.hv_support i ha de retornar 1.
- S'admeten macOS Sierra 10.12 i versions més recents de macOS. Recomanem actualitzar a la versió més recent de macOS.
- Almenys 4 GB de RAM
- VirtualBox abans de la versió 4.3.30 NO s'ha d'instal·lar (és incompatible amb Docker Desktop per a Mac). Si teniu instal·lada una versió més recent de VirtualBox, estarà bé.

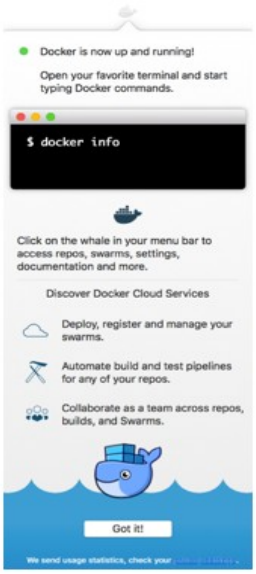
Nota: si el vostre sistema no compleix aquests requisits, podeu instal·lar Docker Toolbox, que utilitza Oracle VirtualBox en lloc d'HyperKit.

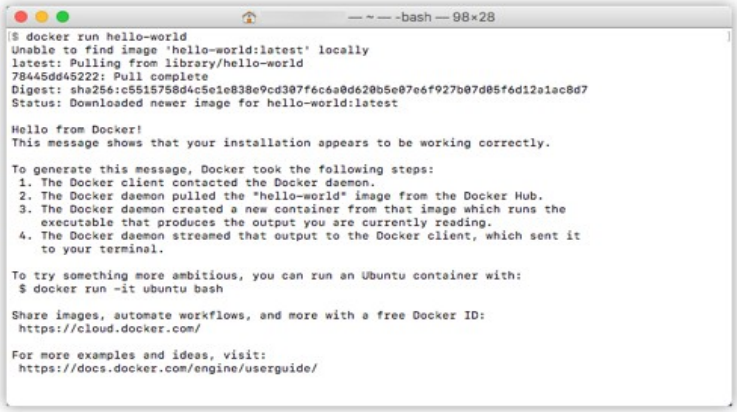
https://www.docker.com/products/docker-desktop descarregar d'aquesta pàgina docker desktop.



Fer doble click sobre docker.dmg i apareix la imatge següent, fer click sobre Docker.app







Comnecem a instal·lar i a la barra d'estat apareix la imatge següent, fem click i vorem totes le opcions que tenim.

Comproveu les versions

Assegureu-vos que les vostres versions de docker, docker-compose i docker-machine estiguin actualitzades i compatibles amb Docker.app. La vostra sortida pot ser diferent si esteu executant diferents versions.

```
$ docker --version
Docker versió 18.09, c97c6d6
$ docker-compose --version
docker-compose version 1.24.0, build 8dd22a9
$ docker-machine --version
versió docker-machine 0.16.0, construir 9ba6da9
```

3.Docker sobre Linux:

Instal·larem Docker sobre ubuntu 18.04, Necessitarem: Un servidor Ubuntu 18.04 configurat, incloent un usuari sudo no root. Un compte en Docker Hub, si desitja crear les seves pròpies imatges i fer el push a Docker Hub.

És possible que el paquet d'instal·lació de Docker que està disponible al repositori oficial d'Ubuntu no sigui l'última versió. Anem a instal·lar Docker des del repositori oficial de Docker per assegurar-nos que tenim l'última versió. Per fer això, anem a afegir una nova font de paquet, la clau GPG de Docker per assegurar que les descàrregues siguin vàlides i després anem a instal·lar el paquet.
Primer, actualitzem la llista de paquets existent:

```
$ sudo apt update
```

A continuació, instal·lem alguns paquets de requisits previs que li permeten a apt utilitzar paquets mitjançant HTTPS:

```
$ sudo apt install apt-transport-https ca-certificates curl programari-properties-common
```

Després, afegim la clau GPG per al repositori oficial de Docker al nostre sistema:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Afegim el repositori de Docker a les fonts d'APT:

```
$ sudo add-apt-repository "deb  https://download.docker.com/linux/ubuntu bionic stable"
```

Actualitzar de nou:

```
$ sudo apt update
```

I instal·lar docker:

```
$ sudo apt install docker-ce
```

Ara hauriem de tenir Docker instal·lat, el dimoni iniciat, i el procés habilitat per iniciar durant l'arrencada. Comproveu que s'està executant:

```
$ sudo systemctl status docker
```

Executar docker sense sudo, per a la qual cosa hauríem d'afegir l'usuari sudo al grup de docker.

```
$ sudo usermod -aG docker username    (on username és l'usuari administrador)
```

Utilització bàsica de docker:

Utilitza docker consisteix en passar-li una cadena d'opcions i comandaments seguits d'arguments. La sintaxi seria la següent:

```
$docker [option] [command] [arguments]
```

Per veure tots els subcomandes disponibles, utilitzem:
$docker

Si desitja veure les opcions disponibles per a una ordre específica, utilitzem:
$docker docker-subcommand --help (on docker-subcommand és qualsevol de les comandes de la comanda anterior).

Si volem veure la informació sobre Docker de tot el sistema, utilitzem:
$docker info

Els contenidors Docker es formen a partir d'imatges de Docker. Per defecte, Docker extrau aquestes imatges de Docker Hub, un registre de Docker administrat per Docker, l'empresa responsable del projecte Docker.
Qualsevol persona és capaç d'allotjar les seves imatges Docker en Docker Hub, per tant, la majoria de les aplicacions i distribucions de Linux que necessitem tindran les imatges allotjades aquí mateix.
Per verificar si docker està funcionant, utilitzem:
$docker run hello-world

El resultat li indicarà que Docker està funcionant correctament, si no troba la imatge en local la buscarà en remot.

Podem buscar les images de qualsevol sistema amb l'ordre:
$docker search NomImage
L'script rastrejarà Docker Hub i li lliurarà una llista de totes les imatges que tinguin un nom que concordi amb la cadena de cerca.

Per a descarregar-la amb l'ordre:
$docker pull NomImage

Després de descarregar una imatge, es pot executar un contenidor fent servir la imatge descarregada amb la comanda run. Com vam veure amb l'exemple de hello-world, si no s'ha descarregat una imatge en executar docker amb el subcomando de run, el client Docker primer descarregarà la imatge i després executarà un contenidor utilitzant la mateixa. Per veure les imatges que es tenim descarregades al nostre ordinador, utilitzarem:
$docker images

Crear contenidors i treballar amb ells:

1.Crear, llançar, parar un nou contenidor a partir d'una imatge

$docker run -opcions --name NomContainer (Aquesta opció crea el contenidor amb les característiques donades en opcions)

Opcions:

-i Manté una connexió interactiva amb el "standard input"

-t Gestiona un "pseudo TTY"

-d Corre el contenidor en mode "background" imprimeix el ID.

-a Associa standard input or output a la sessió oberta

-cpus Nombre de CPUs assignades

-IP Assigna un IP

-Mac-address Assigna un mac address especial al contenidor.

-m Fixa un límit de memòria per a aquest contenidor (solen ser uns pocs megues)

-name Assigna un nom al contenidor

-p Publica ports del contenidor a la xarxa assignada.

-rm Al parar el contenidor s'esborrarà automàticament.

-tmpfs Munta un directori en mode tmpfs (temporal que s'esborrarà, no té persistència).

-v Munta un directori en el contenidor amb persistència, pot ser una carpeta de l'equip real o un volum docker.

-e Permet crear variables d'entorn.

Parar i reiniciar una instancia d'un contenidor.

```
$ docker stop NomContainer
$ docker start -a NomContainer
```

Si tanquem la terminal el docker seguirà corrent, ara hem d'executar la comanda per llistar els contenidors actius:

```
$docker ps
```

Per a eliminar un contenidor:

```
$docker rm Nomcontainer
```

EXEMPLE 1:

```
# docker run --name u1 -it ubuntu:17.10 /bin/bash

root@d4f3a33c3bef:/#
```

Hem arrencat un contenidor utilitzant la imatge d'Ubuntu 17.10, a mode interactiu, amb nom "u1", al mateix temps ens obre una sessió de bash per interactuar.
Per a eixir del contenidor executem exit.

EXEMPLE 2:

Creació del nostre Contenidor
Crearem un contenidor amb les següents característiques:

- Amb el port 8069 obert per accedir des del amfitrió
- Amb el Port 22 obert
- Amb una terminal Interactiva
- Amb un Directori Compartit entre hoste (docker) i amfitrió (SO)

```
docker run -it -p 22 -p 8069: 8069 -v / opt / Odoo: / home / Odoo --name odoo_11_dev ubuntu:16.04
```

És important indicar la versió de la Imatge descarregada per això al final hem indicat ubuntu: 16.04
El resultat és una consolta interactiva, en la qual podem executar ordres.

EXEMPLE 3: Dos contenidors a la vegada on un d'ells utilitza l'altre

Iniciar un servidor PostgreSQL

```
$ docker run -d -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB= postgres --name db postgres:10
```

Iniciar una instancia de Odoo utilitzant l'anterior.

```
$ docker run -p 8069:8069 --name odoo --link db:db -t odoo
```

El alias del contenidor que executa Postgres deu ser db per a que Odoo puga connectar-se al servidor de Postgres.

2.Modificar un contenidor i crear una imatge.

```
$docker commit Nomcontenidor NomImage_queVolem
$docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
```

El commit permet ficar opcions per a modificar el contenidor.

```
$docker commit --help
```

-m és per al missatge de confirmació que ens ajudarà a saber quins canvis em fer.
-a s'usa per especificar l'autor.
-c Aplicar instrucció Dockerfile a la imatge creada
-p Pausa el contenidor durant la confirmació (per defecte es vertader)

Crear un usuari a la pagina oficial de docker i pujar images:

La pàgina oficial és: https://hub.docker.com i d'onar-se d'alta. Podem crear un repositori i guardar les nostres images per a poder accedir a elles des de qualsevol lloc.

Anem a pujar una imatge de Docker a Docker Hub.

```
$docker login  -u docker-registry-username
```

Se li demanarà que es certifique l'ús de la contrasenya de Docker Hub.

Podem canviar el nom de la imatge d'abans:

```
$docker tag NomImage  docker-registry-username/NomImageNova
```

Pujar la imatge:

```
$docker push docker-registry-username/NomImageNova
```

Exportar i importar els nostres contenidors.

```
$docker export Nomcontenidor > Nom.tar
$docker export Nomcontenidor -o nom.tar (son les comilles)
```

Per Utilitzar la image amb el import:

```
$docker import Nom.tar NomImage
```

Per a crear un docker del meu ordinador: fer un tar.gz del meu disc dur i importar-lo a un docker i ja estaria.

Docker compose.

Docker Compose és una eina que permet simplificar l'ús de Docker, generant scripts que faciliten el disseny i la construcció de serveis. Aquí resumim alguns tipos. Amb Docker Compose pots crear diferents contenidors i al mateix temps, en cada contenidor, diferents serveis.
Docker-compose.yml. Em permet unir més d'un docker en uno.
Hauriem de crear un arxiu anomenat docker-compose.yml amb el contingut que volem crear. Per exemple:

```
version: '3'

  services:
    jekyll:
      build: .
      command: bundle exec jekyll serve --incremental --host 0.0.0.0
      ports:
        - "4000:4000" # jekyll serve
      volumes:
        - .:/app
```

- Versio '3': Els arxius docker-compose.yml són versionats, el que significa que és molt important indicar la versio de les instruccions que volem donar-li. A mesura que Docker evoluciona, hi haurà noves versions, però de totes maneres, sempre hi ha compatabilidad cap enrrere, en indicar la versió.
- Services: definim el que volem tenir al docker. I a continuació el nom del servei, al nostre cas s'anomena web.
- Jekyll: Indica el nom del servei. Podria ser qualsevol nom. La idea és que el nom indiqui el tipus de servei que estem construint. Ex: MySQL, FrontEndNGINX, etc.
- Build: S'utilitza per indicar on està el Dockerfile que volem utilitzar per crear el contenidor. En posar "." Automàticament considerarà el Dockerfile existent al directori actual.
- Comand: Un cop creat el contenidor, aquí llancem la comanda que permet executar Jekyll. La comanda "-host 0.0.0.0": serveix per mapejar el contenidor al sistema operatiu host (el qual estem utilitzant-lo per desenvolupar).
- "Ports": Aqui mapegem els ports locals 4000 (webserver Jekyll) al servidor host. Això permetrà que accedint a Localhost: 4000 puguem provar el lloc generador per Jekyll
- "Volumes": Aqui fem que el directori actual es mapeje directament amb el /app, lloc on hem creat l'aplicació. D'aquesta manera, qualsevol canvi en el directori local al host, es farà immediatament al contenidor.

Després, amb la comanda docker-compose up, li donem instruccions a Docker perquè cree el contenidor, segons Dockerfile i l'executi, amb la comanda docker-Compose.
Algunes consideracions:

- docker-compose up: dóna instruccions a Docker per crear el contenidor, i executar segons docker-compose.yml
- docker-compose down: apaga tot els serveis que va aixecar amb docker-compose up.
- docker-compose ps : permet veure els contenidors funcionant.
- docker-compose exec: permet executar una ordre a un dels serveis aixecats de Docker-compose.

El dockerfile.

És l'editor de contenidors (docker). Crear un fitxer anomenat dockerfile amb el contingut desitjat.

Vodem veure un exemple de l'estructura a els docs de dockerhub. https://docs.docker.com/get-started/part2/

Anem escribint el codi que volem executar per a crear el nostre contenidor personalitzat. El últim punt seria llançar les comandes necessaries per a l'arranc. Crear infraestructura, l'entorn i després arrancar.

Podem crear un script amb totes les ordres necessaries per a crear i executar el docker, si va parametritzat podrem canviar les versions directament, sense canviar les dades.

Els Dockerfile tenen algunes instruccions:

- FROM: indica la imatge base a partir de la qual crearem la imatge que construirà el Dockerfile.
- Maintainer: documenta el creador de la imatge.
- ENV HOME: estableix el directori HOME que faran servir les comandes RUN.
- RUN: permet executar una instrucció en el contenidor, per exemple, per instal·lar algun paquet mitjançant el gestor de paquets (apt-get, yum, rpm, ...).
- ADD: permet afegir un arxiu al contenidor, en moltes ocasions s'utilitza per proporcionar la configuració dels serveis (ssh, mysql, ...).
- VOLUME: estableix punts de muntatge que al utilitzar el contenidor es poden proporcionar, els volums són a forma d'exterioritzar un determinat directori i proporcionar persistència (les imatges de docker són de només lectura i no emmagatzemen dades entre diferents execucions).
- EXPOSE: indica els ports TCP/IP pels quals es pot accedir als serveis del contenidor, els típics són 22 (SSH), 80 (HTTP) i en aquest cas el port per defecte de mysql 3306.
- CDM: estableix la comanda del procés d'inici que es farà servir si no s'indica un a l'iniciar un contenidor amb la imatge.

Exemple:

```
1  FROM phusion/baseimage:0.9.15
 2 MAINTAINER picodotdev <pico.dev@gmail.com>
 3
 4 ENV HOME /root
 5
 6 RUN apt-get update -q
 7
 8 RUN /etc/my_init.d/00_regen_ssh_host_keys.sh
 9
10 RUN echo 'root:$6$l/PahbyY$jFhqIAuvHeK/GwjfT71p4OBBkHQpnTe2FErcUWZ8GIN1ykdI7CgL05Jkk7MYW6l.0pijAlfoifkQnLpal
11
12 ADD bashrc /root/.bashrc
13 ADD timezone /etc/timezone
14
15 EXPOSE 22
16
17 CMD ["/sbin/my_init"]
```

Aquest és el document base que després s'utilitzarà en el següent.

```
FROM picodotdev/base:1.0
MAINTAINER picodotdev <pico.dev@gmail.com>

ENV HOME /root

RUN apt-get install -y mysql-server mysql-client

ADD my.cnf /etc/mysql/my.cnf
RUN mkdir /etc/service/mysql
ADD mysql /etc/service/mysql/run
RUN chmod +x /etc/service/mysql/run

RUN rm -R /var/lib/mysql && \
     mkdir /var/lib/mysql && \
     mkdir /mnt/keys

VOLUME ["/var/lib/mysql", "/mnt/keys"]

RUN apt-get clean && \
     rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

EXPOSE 22 3306

CMD ["/sbin/my_init"]
```