HowTo ASIX Docker

Curs 2018-2019

Descripció dels aprenentatges

Bàsic

1. Interactiu:
    a. conceptes generals de containers / docker.
    b. imatges.
    c. creació de containers interactius. Treball amb containers.
    d. pràctica: generar un container amb un servei httpd.
2. Generar / Desar imatges
    a. generar una imatge amb commit.
    b. gestió de les imatges.
    c. docker hub.
    d. tags / desar imatges / push / pull.
    e. pràctica: desar al dockerHub la imatge.
3. Repeat!!
    a. tornar a fer containers i containers i treballar-hi
    b. docker exec / docker cp / --rm / etc
    c. pràctica: generar interactivament un container amb un servei LDAP
4. Dockerfile
    a. generar un servidor ldap a partir de un dockerfile.
    desar totes les dades a un directori ldapserver:base.
    b. docker build.
    c. dcoker --rm
    d. pràctica: generar un servidor ldap amb dockerfile.
5. Git Hub
    a. github: usar github com a repositori de dades del desenvolupament de la imatge. push / pop.
    b. pràctica: generar un servidor ldap automatitzat lligant github i dockerhub.
6. Detach
    a. generar containers amb serveis detach.
    b. containers detach i --rm.
    c. command, entry point, pid 1.
    d. executar /bin/bash en un detach.
    e. pràctica: generar un conainer detach amb el servei ldap.
7. Ports
    a. export al dockerfile
    b. publicació de ports al container.
    c. exemple :docker amb la documentació autocontinguda amb accés navegador.
    d. pràctica: publicar al host el port amb el servei ldap del container.
    accedir des d'un host remot al servei que ofereix al container.
    e. pràctica: accedir amb qg i phpldapadmin al servei ldap de containers que estan en hosts remots.
8. Volumes
    a. usar volums amb mount.
    b. usar volums amb bind.
    c. noms de volum.
    d. exemple: usar portainer.
    e. pràctica: muntar la base de dades ldap en un volum.
    f. pràctica: muntar la configuració ldap en un volum.

Combinar

9. Network.
    a. tipus de networks.

    b. gestionar networks: crear, esborrar, usar
    c. consultar la configuració (json)
    d. pràctica: combinar: hostpam, hostunix
10. Compose
    a. docker compose
    b. pràctica: combinar: hostpam, hostunux, ldapserver.
    c. pràctica: combinar serverssh, hostunix
11. …
12. stack
13. swarm
14. machine

Presentació Docker

Nota

- ➢ prompt tipus **[root@hp01 ~]#** indica que l'ordre es realitza des del host,des d'una sessió del ordinador de treball.
- ➢ prompt tipus **[root@7d0b3c6c6c8c /]#** indica ordres efectuades des de dins del container.

Crear un container d'una imatge fedora

Es crearà un container d'una imatge **base** de fedora on s'hi instal·larà Python i vim. Els passos que es mostren en aquest apartat són:

- Crear un container interactiu
- Instal·lar software dins del container interactivament
- Llistar els containers executant-se, tots, l'últim executat.
- Llistar les imatges, totes, les d'un usuari, un repositori o una en concret.

- Crear una imatge a partir de l'estat concret d'un container.
- Observar el mecanisme d'anomenar les imatges, els tags i el tag per defecte latest.
- Quan una imatge que es genera trepitja el mateix nom que una d'existent, la existent queda sense nom, s'identifica pel ID.

- Generar nous tags per a les imatges (sobrenoms). Gestió intel·ligent dels tags per part del docker.

- Esborrar imatges.
- Esborrar containers.

Recull d'ordres:

```
# docker run -i -t fedora /bin/bash
# docker run --name nomContainer -i -t fedora /bin/bash

# docker ps [-a] [-l]
# docker top nomContainer

# docker images
# docker images fedora
# docker images usernew/*

# docker history imageName

# docker commit containerName imageName
# docker commit -m “text commit” --author “nomaAutor” containerName imageName
```

```
# docker tag imageName:tag newImageName:newTag

# docker rm nameContainer
# docker rmi imageName

nomConainer → cont.prova01
imageName → usernew/prova
imageName → usernew/prova:tag
```

```
# docker version
# docker info
# docker help
# docker <command> --help
# man docker-<ordre>
```

Generar una imatge a partir d'un container

```
# Generar un container (nom aleatori)
[root@hp01 ~]# docker run -i -t fedora /bin/bash
[root@7d0b3c6c6c8c /]#
[root@7d0b3c6c6c8c /]# yum install python vim

[root@hp01 ~]# docker ps -l
CONTAINER ID  IMAGE   COMMAND     CREATED        STATUS         PORTS
NAMES
7d0b3c6c6c8c  fedora  "/bin/bash" 6 minutes ago  Up 6 minutes
desperate_cori

[root@hp01 ~]# docker top desperate_cori
UID   PID   PPID  C  STIME  TTY
TIME      CMD
root  8118  6511  0  18:02  pts/4
00:00:00  /bin/bash

[root@7d0b3c6c6c8c /]# exit

[root@hp01 ~]# docker ps
CONTAINER ID  IMAGE   COMMAND     CREATED        STATUS         PORTS
NAMES
[root@hp01 ~]# docker ps -l
CONTAINER ID  IMAGE   COMMAND     CREATED        STATUS         PORTS
NAMES
7d0b3c6c6c8c  fedora  "/bin/bash" 9 minutes ago  Exited (0) 8 seconds ago
desperate_cori
```

```
# Generar una imatge
[root@hp01 ~]# docker commit desperate_cori usernew/img.prova
35c3761af7ac5c20ed52f93edb5a3188099cfa4d3f4659e0812e115d4633e1bd

[root@hp01 ~]# docker images usernew/img.prova
REPOSITORY          TAG      IMAGE ID      CREATED             VIRTUAL SIZE
usernew/img.prova   latest   35c3761af7ac  27 seconds ago      400.5 MB
```

Trepitjar la imatge, esborrar inicial, mirar tags

[root@hp01 ~]# docker commit desperate_cori usernew/img.prova
a3016c5e49d4b29d777fece2c4e352ea04fe94bb9daa958aa402256e24ca02b9

[root@hp01 ~]# docker images usernew/*

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
usernew/img.prova	**latest**	a3016c5e49d4	16 seconds ago	400.5 MB

[root@hp01 ~]# docker images

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
usernew/img.prova	latest	a3016c5e49d4	27 seconds ago	400.5 MB
<none>	**<none>**	**35c3761af7ac**	2 minutes ago	400.5 MB

Eliminar la inicial 'trepitjada' pel mateix tag

[root@hp01 ~]# docker rmi 35c3761af7ac5c20ed52f93edb5a3188099cfa4d3f4659e0812e115d4633e1bd
Deleted: 35c3761af7ac5c20ed52f93edb5a3188099cfa4d3f4659e0812e115d4633e1bd

Generar vàries imatges (tonteria perquè totes son iguals!)

[root@hp01 ~]# docker commit -m "descripció del motiu del commit" --author="usuari newuser" desperate_cori usernew/img.prova:v2
c61c901448e09b175b9df927faebe38d2c84186574bd1a26a9f43dba920de9f6

[root@hp01 ~]# docker commit -m "perque si" --author="usuari newuser" desperate_coriusernew/img.prova:v3
5f037321ad082bb1ec8446c2d65d338f15b2ad21ff5c69793127491b22d7f3b6

[root@hp01 ~]# docker images usernew/*

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
usernew/img.prova	v3	5f037321ad08	30 seconds ago	400.5 MB
usernew/img.prova	v2	c61c901448e0	50 seconds ago	400.5 MB
usernew/img.prova	latest	a3016c5e49d4	9 minutes ago	400.5 MB

Observar l'historial d'evolució d'una imatge (latest, v2 i v3 són la mateixa)

[root@hp01 ~]# docker history usernew/img.prova:v3

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
5f037321ad08	2 minutes ago	/bin/bash	214 MB	perque si
ded7cd95e059	5 months ago	/bin/sh -c #(nop) ADD file:4be46382bcf2b095fc	186.5 MB	
48ecf305d2cf	6 months ago	/bin/sh -c #(nop) MAINTAINER Lokesh Mandvekar	0 B	

[root@hp01 ~]# docker history usernew/img.prova:v2

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
c61c901448e0	3 minutes ago	/bin/bash	214 MB	descripció del motiu del commit
ded7cd95e059	5 months ago	/bin/sh -c #(nop) ADD file:4be46382bcf2b095fc	186.5 MB	
48ecf305d2cf	6 months ago	/bin/sh -c #(nop) MAINTAINER Lokesh Mandvekar	0 B	

* mirar la part de tags

Eliminar imatges

```
[root@hp01 ~]# docker rmi usernew/img.prova:newtag
Untagged: usernew/img.prova:newtag
Deleted: 5f037321ad082bb1ec8446c2d65d338f15b2ad21ff5c69793127491b22d7f3b6

[root@hp01 ~]# docker rmi usernew/img.prova:v2
Untagged: usernew/img.prova:v2
Deleted: c61c901448e09b175b9df927faebe38d2c84186574bd1a26a9f43dba920de9f6

[root@hp01 ~]# docker images usernew/*
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
usernew/img.prova   latest              a3016c5e49d4        22 minutes ago      400.5 MB
```

```
# Eliminar containers
root@hp01 ~]# docker rm desperate_cori
desperate_cori
```

Gestió dels tags de les imatges

```
# Gestió  dels tags
[root@hp01 ~]# docker tag usernew/img.prova:v3 usernew/img.prova:newtag
[root@hp01 ~]# docker tag usernew/img.prova:v2 usernew/img.nownom:v2
[root@hp01 ~]# docker images usernew/*
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
usernew/img.prova   newtag              5f037321ad08        7 minutes ago       400.5 MB
usernew/img.prova   v3                  5f037321ad08        7 minutes ago       400.5 MB
usernew/img.prova   v2                  c61c901448e0        7 minutes ago       400.5 MB
usernew/img.nownom  v2                  c61c901448e0        7 minutes ago       400.5 MB
usernew/img.prova   latest              a3016c5e49d4        16 minutes ago      400.5 MB

[root@hp01 ~]# docker rmi usernew/img.nownom:v2
Untagged: usernew/img.nownom:v2

[root@hp01 ~]# docker rmi usernew/img.prova:v3
Untagged: usernew/img.prova:v3

[root@hp01 ~]# docker images usernew/*
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
usernew/img.prova   newtag              5f037321ad08        10 minutes ago      400.5 MB
usernew/img.prova   v2                  c61c901448e0        11 minutes ago      400.5 MB
usernew/img.prova   latest              a3016c5e49d4        20 minutes ago      400.5 MB
```

```
[root@hp01 ~]# docker tag --help
Usage:    docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]
Tag an image into a repository
  -f, --force=false           Force
  --help=false      Print usage
```

Crear un container partint d'una imatge pròpia

A partir d'una imatge pròpia es poden crear diversos containers i tenir-los en execució. Examinar-ne l'execució, engegar, parar i attach (per a containers interactius). També consultar la configuració d'un container.

Recull d'ordres:

```
# docker run --name nomContainer -i -t imageName /bin/bash
# docker ps -n=nº

# docker [-a] start nomContainer [nomContainer]
# docker stop nomContainer [nomContainer]
# docker attach nomContainer
```

```
# docker inspect containerName
# docker inspect nameInage

# docker inspect --format '{{.NetworkSettings}}' nomContainer
```

En aquest apartat observarem com:

- Crear diversos containers interactius de sessió de *bash*.
- Sortir amb *exit* d'un container, que s'atura però no es destrueix.
- Observar l'estat dels containers que s'estan executant o dels n últims executats.
- Engegar containers amb *start*,
- Connectar a la consola del terminal virtual d'un container interactiu amb *attach*.
- Aturar containers amb *stop*.
- Engegar i connectar a la sessió interactiva d'un container (en un sol pas)

- Consultar la configuració d'un container o una imatge, en format text o json.
- La configuració es pot mostrar tota o amb un filtre de selecció

Gestió de containers

```
# Creació de dos containers
[root@hp01 ~]# docker run --name cont.prova01 -i -t usernew/img.prova /bin/bash
[root@e8df071cfbf9 /]# exit
exit
[root@hp01 ~]# docker run --name cont.prova02 -i -t usernew/img.prova /bin/bash
[root@81570ca14090 /]# exit
exit

[root@hp01 ~]# docker ps -n=2
CONTAINER ID    IMAGE           COMMAND         CREATED         STATUS                  PORTS
```

```
NAMES
81570ca14090        usernew/img.prova   "/bin/bash"         About a minute ago   Exited (0) About a minute ago
cont.prova02
e8df071cfbf9        usernew/img.prova   "/bin/bash"         2 minutes ago        Exited (0) 2 minutes ago
cont.prova01
```

Mes d'un container en execució

[root@hp01 ~]# docker start cont.prova01

cont.prova01

[root@hp01 ~]# docker start cont.prova02

cont.prova02

[root@hp01 ~]# docker ps

```
CONTAINER ID        IMAGE               COMMAND             CREATED              STATUS              PORTS
NAMES
81570ca14090        usernew/img.prova   "/bin/bash"         6 minutes ago        Up 5 seconds
cont.prova02
e8df071cfbf9        usernew/img.prova   "/bin/bash"         6 minutes ago        Up 14 seconds
cont.prova01
```

[root@hp01 ~]# docker top cont.prova01

```
UID                 PID                 PPID                C                   STIME               TTY
TIME                CMD
root                10781               6511                0                   19:14               pts/4
00:00:00            /bin/bash
```

Attach a la sessió de terminal d'un container

[root@hp01 ~]# docker attach cont.prova01

<prémer return>

[root@e8df071cfbf9 /]#

[root@e8df071cfbf9 /]#

Aturar un container des d'una sessió externa

[root@hp01 ~]# docker stop cont.prova01

cont.prova01

[root@hp01 ~]# docker ps -n=2

```
CONTAINER ID        IMAGE               COMMAND             CREATED              STATUS              PORTS
NAMES
81570ca14090        usernew/img.prova   "/bin/bash"         13 minutes ago       Up 6 minutes
cont.prova02
e8df071cfbf9        usernew/img.prova   "/bin/bash"         13 minutes ago       Exited (0) 43 seconds ago
cont.prova01
```

Engegar i connectar (attach) a un container aturat)

[root@hp01 ~]# docker start -a cont.prova01

[root@e8df071cfbf9 /]#

[root@hp01 ~]# docker stop cont.prova01 cont.prova02

cont.prova01

cont.prova02

```
[root@hp01 ~]# docker start cont.prova01  cont.prova02
cont.prova01
cont.prova02
```

* Consultar l'apartat de configuració d'un container

```
# Eliminar Containers
[root@hp01 ~]# docker rm cont.prova01  cont.prova02
cont.prova01
cont.prova02
```

Altres ordres de gestió

Resum d'ordres:

```
# docker create --name nomContainer imageName  command
# docker pause nomContainer
# docker unpause nomContainer
# docker restart nomContainer
```

```
# docker logs nomContainer
# docker stats nomContainer
# docker ports nomContainer
```

```
# docker rename nomContainer  nomNouContainer
```

L'ordre "*docker run*" realitza en realitat dues accions:

- docker create
- docker start

```
# Crear container
[root@hp01 ~]# docker create  --name noucontainer -i -t fedora /bin/bash
e3dc5152a4b7b907bb81c8e42cf6caef06ded7178f4836c2e047b73952317b96

[root@hp01 ~]# docker start -a   noucontainer
[root@e3dc5152a4b7 /]# bash
```

```
# Monitoritzar els processos/recursos de dins del container
[root@hp01 ~]# docker top noucontainer
UID         PID         PPID        C           STIME       TTY
TIME        CMD
root        3753        2989        0           19:44       pts/4
00:00:00    /bin/bash
```

```
root         4087         3753         0         19:47         pts/4
00:00:00     bash

[root@hp01 ~]# docker stats noucontainer
CONTAINER           CPU %               MEM USAGE/LIMIT     MEM %               NET I/O
noucontainer        0.00%               7.926 MB/4.04 GB    0.20%               648 B/738 B
```

```
# Modificar el nom a un container
[root@hp01 ~]# docker rename noucontainer NEWcontainer
```

```
# Restart i eliminar-lo
[root@hp01 ~]# docker restart NEWcontainer
NEWcontainer
[root@hp01 ~]# docker stop NEWcontainer
NEWcontainer
[root@hp01 ~]# docker rm NEWcontainer
NEWcontainer
```

```
# Crear un nou container amb ports
[root@hp01 ~]# docker run -p 13 --name noucontainer -i -t fedora /bin/bash
[root@c36157016c85 /]#
```

```
# Mirar els ports
[root@e3dc5152a4b7 /]# yum install xinetd
[root@e3dc5152a4b7 /]# vi /etc/xinetd.d/daytime-stream
[root@e3dc5152a4b7 /]# /usr/sbin/xinetd
[root@e3dc5152a4b7 /]# chkconfig
xinetd based services:
    chargen-dgram:    off
    chargen-stream:   off
    daytime-dgram:    off
    daytime-stream:   on
    ….
```

```
# Observar el port del servei daytime-stream
[root@hp01 ~]# docker port noucontainer
13/tcp -> 0.0.0.0:32768

[root@hp01 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
          PORTS                     NAMES
c36157016c85        fedora              "/bin/bash"         3 minutes ago       Up 3 minutes
0.0.0.0:32768->13/tcp   noucontainer
```

```
# Observar que el port funciona
root@hp01 ~]# telnet localhost 32768
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
05 NOV 2015 19:09:22 UTC
Connection closed by foreign host.

[root@hp01 ~]# ncat localhost 32768
05 NOV 2015 19:12:05 UTC
^d
```

```
# Pausa d'un cpontainer
[root@hp01 ~]# docker pause noucontainer
noucontainer

[root@hp01 ~]# ncat localhost 32768
^d
[root@hp01 ~]# telnet localhost 32768
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
^]
```

```
# Reanudar un container en pausa
[root@hp01 ~]# docker unpause noucontainer
noucontainer

[root@hp01 ~]# ncat localhost 32768
05 NOV 2015 19:14:40 UTC
^d

[root@hp01 ~]# telnet localhost 32768
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
05 NOV 2015 19:14:48 UTC
Connection closed by foreign host.
```

Configuració d'un container

Es poden consultar les dades 'tècniques' o llistat de tota la configuració:

```
[root@hp01 ~]# docker run --name cont.prova01 -i -t usernew/img.prova /bin/bash
[root@e4a6b7da05e3 /]# exit
exit
```

```
[root@hp01 ~]# docker inspect cont.prova01  | head -30
[
{
        "Id":
"e4a6b7da05e3d88bdfdb18be9b22fd1ed336a32e31ef923ae8d42d77681ac4e0",
        "Created": "2015-11-03T18:30:51.173996964Z",
        "Path": "/bin/bash",
        "Args": [],
        "State": {
        "Running": false,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 0,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2015-11-03T18:30:52.367071873Z",
        "FinishedAt": "2015-11-03T18:30:56.174549711Z"
        },
        "Image":
"a3016c5e49d4b29d777fece2c4e352ea04fe94bb9daa958aa402256e24ca02b9",
        "NetworkSettings": {
        "Bridge": "",
        ….

[root@hp01 ~]# docker inspect cont.prova01 | grep "{"
{
        "State": {
        "NetworkSettings": {
        "HostConfig": {
        "PortBindings": {},
        "RestartPolicy": {
        "LogConfig": {
        "Config": {}
        "GraphDriver": {
        "Data": {
        "Config": {
        "Labels": {}
```

```
[root@hp01 ~]# docker inspect --format '{{.NetworkSettings}}' cont.prova01
{       0 false  0   0   map[] map[]  [] []}
```

Gestió del compte de docker.io

Un repositori públic de docker és el gestionat pel senyor docker mateix anomenat **Docker HUB** amb adreça web: https://hub.docker.com/

Aquest repositori públic de docker correspon al repositori docker.io que podem observar quan llistem imatges.

Per poder usar aquest repositori public de docker cal enregistrar-se proporcionant les dades següents:

- nom *d'usuari* (usernew)
- *password* (secret)
- *email* de l'usuari usernew@gmail.com (requerit que sigui un compte de correu vàlid perquè caldrà rebre i contestar un email en el procés d'enregistrament)

Un cop enregistrar es pot consultar l'espai propi dins del Docker Hub. Aquest espai permet desar imatges en repositoris públics i un repositori privat. Un repositori és el conjunt d'imatges que es desenvolupen en diversos tags sobre una imatge base… És a dir, per a cada propòsit de desenvolupament diferent caldrà un repositori diferent. Exemples:

- usernew/apache → s'hi aniran acumulant les evolucions de les imatges de docker on posem apache a sobre una base de per exemple fedora
- usernew/ldap → desenvolupem imatges per implemetar un servei d'autenticació ldap sobre fedora.
- usernew/python → imatges per poder generar containers usant versions de python diferents.

Recull d'ordres:

```
# docker login
# docker logout

# docker push username/nameImage
# docker pull   username/nameImage

# docker search nameImage
# docker search username/*
```

```
# Exemple de fer login com a user usernew
# docker login
Username (usernew):
WARNING: login credentials saved in /root/.docker/config.json
Login Succeeded
```

```
# Pujar una imatge al repositori públic de Docker Hub (usuari ecanet)
# docker push ecanet/asix.m06.ldap
The push refers to a repository [docker.io/ecanet/asix.m06.ldap] (len: 0)
79d5e5f09c47: Image already exists
ded7cd95e059: Image already exists
48ecf305d2cf: Image already exists
01: digest: sha256:7eb915dea960eeab2acd7fb305c9d750b6b99ed538189a6e6510e42c91b28317 size: 5046
5950ad6506bc: Image successfully pushed
```

```
# Descarregar una imatge del repositòri públic Docker Hub de l'usuari ecanet
# docker pull ecanet/asix.m06.ldap
```

```
# Tancar una sessio de consola al Docker Hub
# docker logout
```

```
# Llistar les imatges públiques de l'usuari ecanet
[root@hp01 ~]# docker search ecanet
INDEX     NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
docker.io   docker.io/ecanet/apache2          repositori temporal de prova: documentació...   0
docker.io   docker.io/ecanet/asix.m06.ldap    Imatge base (F22) d'exemple organització ...    0
```

Dockerfile

Una altra forma de generar imatges és fer-ho a través d'un Dockerfile. En lloc de generar-la a partir d'un container com fa *docker commit* es pot generar una imatge directament seguint un conjunt d'instruccions, sense necessitat de cap container ni de cap imatge prèvia.

S'utilitza un fitxer anomenat Dockerfile que conté el conjunt d'instruccions necessàries. Usualment es crea un directori amb aquest fitxer i la resta de material (altres fitxers) que cal per generar automatitzadament la imatge usant l'ordre ***docker build***.

Resum de comandes:

```
# docker  build -t="edt.asix.m06.ldap:b01" .
# docker history edt.asix.m06.ldap:b01

# docker run --name cont.ldap.01 -p 389 -i -t edt.asix.m06.ldap:b01 /bin/bash
# docker run --name cont.ldap.01 -P -i -t edt.asix.m06.ld:qap:b01 /bin/bash

# docker ps -l

# docker port edt.asix.m06.ldap:b01
```

01 Dockerfile: Exemple imatge host-base

En aquest exemple es construirà una imatge base de host incorporant utilitats bàsiques de consulta del sistema com per exemple nmap, ip, tree, etc. Utilitats que no són necessàries per fer un Docker amb una finalitat concreta però que són útils per treballar amb containers d'exemple a classe.

Llistat:

```
[root@hp01 2016-01-host]# ll
```

```
-rwxr-xr-x 1 root root  255  1 des 19:45 bash_profile
-rwxr-xr-x 1 root root  332  1 des 19:44 bashrc
-rwxr-xr-x 1 root root  405  1 des 19:47 Dockerfile
-rwxr-xr-x 1 root root 1214  1 des 19:43 Docker-help.md
-rwxr-xr-x 1 root root  511  1 des 19:48 README.md
-rwxr-xr-x 1 root root  235  1 des 19:48 startup.sh
```

Dockerfile:

```
# cat Dockerfile
# Version: 0.0.1
# Dockerfile per generar una imatge host base amb utilitats GNU/linux
FROM fedora
MAINTAINER @edt "ASIX M06/M11 @edt 2015-16"
# software
RUN yum -y install vim nmap procps psmisc iproute iputils tree passwd less
# directoris
RUN mkdir /opt/docker
# fitxers
COPY * /opt/docker/
COPY bash_profile /root/.bash_profile
COPY bashrc /root/.bashrc
COPY startup.sh /usr/sbin/
CMD ["/bin/bash"]
```

Build:

```
# docker build -t "ecanet/host:base" -a "@edt ASIX-M06 2016" .
flag provided but not defined: -a
See 'docker build --help'.
[root@hp01 2016-01-host]# man docker builf
No hi ha una entrada de manual per a builf
[root@hp01 2016-01-host]# man docker build
[root@hp01 2016-01-host]# docker build -t "ecanet/host:base" .
Sending build context to Docker daemon 8.192 kB
Step 1 : FROM fedora
 ---> cafb7975f7f0
Step 2 : MAINTAINER @edt "ASIX M06/M11 @edt 2015-16"
 ---> Running in 8578b7fc8109
 ---> fd6ba6621e25
Removing intermediate container 8578b7fc8109
Step 3 : RUN yum -y install vim nmap procps psmisc iproute iputils tree passwd less
 ---> Running in eff2fc22a204
Redirecting to '/usr/bin/dnf -y install vim nmap procps psmisc iproute iputils tree passwd less' (see 'man yum2dnf')
....
Complete!
 ---> 988d27718704
Removing intermediate container eff2fc22a204
Step 4 : RUN mkdir /opt/docker
 ---> Running in 443b890bbdac
 ---> 05774b6ec65e
Removing intermediate container 443b890bbdac
Step 5 : COPY * /opt/docker/
 ---> 4fedde883196
Removing intermediate container 387899c71d34
Step 6 : COPY bash_profile /root/.bash_profile
 ---> e46ba96f4e33
Removing intermediate container ea38d599ce1a
Step 7 : COPY bashrc /root/.bashrc
```

```
 ---> 967724177fba
Removing intermediate container 394ddfcfe512
Step 8 : COPY startup.sh /usr/sbin/
 ---> a74cd020799c
Removing intermediate container cddbdc68316b
Step 9 : CMD /bin/bash
 ---> Running in 986e75979bd4
 ---> 2934836d41c7
Removing intermediate container 986e75979bd4
Successfully built 2934836d41c7
```

 # docker history ecanet/host:base

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
2934836d41c7	2 minutes ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0 B	
a74cd020799c	2 minutes ago	/bin/sh -c #(nop) COPY file:96405045edd19610e	235 B	
967724177fba	2 minutes ago	/bin/sh -c #(nop) COPY file:a3e2717fd73ed0509	332 B	
e46ba96f4e33	2 minutes ago	/bin/sh -c #(nop) COPY file:1db6c15eb36b0313c	255 B	
4fedde883196	2 minutes ago	/bin/sh -c #(nop) COPY multi:7e9b69ce129c277d	2.952 kB	
05774b6ec65e	2 minutes ago	/bin/sh -c mkdir /opt/docker	0 B	
988d27718704	3 minutes ago	/bin/sh -c yum -y install vim nmap procps psm	239.4 MB	
fd6ba6621e25	8 minutes ago	/bin/sh -c #(nop) MAINTAINER @edt "ASIX M06/M	0 B	
cafb7975f7f0	8 days ago	/bin/sh -c #(nop) ADD file:e676494478be611b2d	199.9 MB	
68004564f4ef	8 days ago	/bin/sh -c #(nop) ENV DISTTAG=f25docker FGC=	0 B	
8289b4c23708	3 months ago	/bin/sh -c #(nop) MAINTAINER [Adam Miller <m	0 B	

```
# docker run -it ecanet/host:base
[root@94a71a47b577 /]# documentation
[root@94a71a47b577 /]# manuals
[root@94a71a47b577 /]# alias
```

```
[root@hp01 ~]# docker tag  ecanet/host:base edtasixm06/host:base
[root@hp01 ~]# docker login
[root@hp01 ~]# docker push edtasixm06/host:base
The push refers to a repository [docker.io/edtasixm06/host] (len: 1)
2934836d41c7: Pushed
a74cd020799c: Pushed
967724177fba: Pushed
e46ba96f4e33: Pushed
4fedde883196: Pushed
05774b6ec65e: Pushed
988d27718704: Pushed
fd6ba6621e25: Pushed
cafb7975f7f0:  Pushed
base: digest: sha256:6b654fecf5390cafbc07b0c776c9986ad8e3d125c36a83e024a5c2640c55e55f size: 16460
```

02 Dockerfile: Exemple imatge Samba

```
[root@hp01 2016-samba-01]# ll
total 144
-rwxr-xr-x 1 root root 12499  1 des 20:09 01-smb.conf
-rwxr-xr-x 1 root root   255  1 des 20:07 bash_profile
-rwxr-xr-x 1 root root   332  1 des 20:07 bashrc
-rwxr-xr-x 1 root root   616  1 des 20:22 Dockerfile
-rwxr-xr-x 1 root root  1214  1 des 20:07 Docker-help.md
-rwxr-xr-x 1 root root   710  1 des 20:09 README.md
```

```
-rwxr-xr-x 1 root root  1574  1 des 20:08 SAMBA-help.md
-rwxr-xr-x 1 root root  1379  1 des 20:13 smb.conf
-rwxr-xr-x 1 root root   131  1 des 20:08 startup.sh
```

```
# cat Dockerfile
# Version: 0.0.1
# Dockerfile per generar una imatge base de SAMBA amb 4 shares
#FROM fedora
FROM edtasixm06/host:base
MAINTAINER @edt "ASIX M06/M11 @edt 2016-17"
# software
RUN yum -y install samba  samba-client samba-common samba-libs cifs-utils
# directoris
#RUN mkdir /opt/docker
RUN mkdir /var/lib/samba/public /var/lib/samba/privat /var/lib/samba/test
RUN chmod 777 /var/lib/samba/public
RUN chmod 755 /var/lib/samba/privat
# fitxers
COPY * /opt/docker/
COPY bash_profile /root/.bash_profile
COPY bashrc /root/.bashrc
COPY startup.sh /usr/sbin/
COPY smb.conf /etc/samba/
COPY *.md /var/lib/samba/public/
COPY *.md /var/lib/samba/privat/
CMD ["/bin/bash"]
EXPOSE 139 445
```

```
# docker build -t "ecanet/samba:base" .

[root@hp01 ~]# docker tag  ecanet/samba:base edtasixm06/samba:base
[root@hp01 ~]# docker tag  ecanet/samba:base edtasixm06/samba:latest

[root@hp01 ~]# docker login
[root@hp01 ~]# docker push edtasixm06/samba:base
[root@hp01 ~]# docker push edtasixm06/samba:latest
```

```
# docker run --name "samba01" --hostname "host01" -it edtasixm06/samba:base
[root@host01 /]# startup
[root@host01 /]# documentation
[root@host01 /]# manuals
[root@host01 /]# ps ax
[root@host01 /]# testparam
[root@host01 /]# smbtree
```

03 Dockerfile: Exemple imatge Ldap

En aquest exemple es construirà una imatge base amb un servidor LDAP i labase de dades *edt.org*.

Dockerfile

```
# Version: 0.0.1
FROM fedora
MAINTAINER @edt "ASIX M06/M11 @edt"
# software
RUN yum -y install openldap openldap-servers openldap-clients
# directoris
RUN mkdir /opt/edt-ldap
RUN mkdir /var/tmp/home
RUN mkdir /var/tmp/home/1asix
RUN mkdir /var/tmp/home/2asix
# fitxers
COPY * /opt/edt-ldap/
EXPOSE 389
```

Procés de creació

```
[root@hp01 ldap-01]# docker build -t="edt.asix.m06.ldap:b01" .
Sending build context to Docker daemon 26.62 kB
Step 0 : FROM fedora
 ---> ded7cd95e059
Step 1 : MAINTAINER @edt "ASIX M06/M11 @edt"
 ---> Using cache
 ---> c1a7070c2fd8
Step 2 : RUN yum -y install openldap openldap-servers openldap-clients
 ---> Running in e82df35fcc69
Complete!
 ---> 9da07accffbd
Removing intermediate container e82df35fcc69
Step 3 : RUN mkdir /opt/edt-ldap
 ---> Running in b4973672731f
 ---> ed47bee98a1b
Removing intermediate container b4973672731f
Step 4 : RUN mkdir /var/tmp/home
 ---> Running in 017cfd28c27b
 ---> ba1841bce215
Removing intermediate container 017cfd28c27b
Step 5 : RUN mkdir /var/tmp/home/1asix
 ---> Running in 02ce00444c71
 ---> 65bc905895c3
Removing intermediate container 02ce00444c71
Step 6 : RUN mkdir /var/tmp/home/2asix
 ---> Running in 82b732728101
 ---> 2cd8c02dff3b
```

```
Removing intermediate container 82b732728101
Step 7 : COPY * /opt/edt-ldap
When using COPY with more than one source file, the destination must be a directory and
end with a /
```

```
[root@hp01 ldap-01]# docker build -t="edt.asix.m06.ldap:b01" .
Sending build context to Docker daemon 26.62 kB
Step 0 : FROM fedora
 ---> ded7cd95e059
Step 1 : MAINTAINER @edt "ASIX M06/M11 @edt"
 ---> Using cache
 ---> c1a7070c2fd8
Step 2 : RUN yum -y install openldap openldap-servers openldap-clients
 ---> Using cache
 ---> 9da07accffbd
Step 3 : RUN mkdir /opt/edt-ldap
 ---> Using cache
 ---> ed47bee98a1b
Step 4 : RUN mkdir /var/tmp/home
 ---> Using cache
 ---> ba1841bce215
Step 5 : RUN mkdir /var/tmp/home/1asix
 ---> Using cache
 ---> 65bc905895c3
Step 6 : RUN mkdir /var/tmp/home/2asix
 ---> Using cache
 ---> 2cd8c02dff3b
Step 7 : COPY * /opt/edt-ldap/
 ---> 89fdfdc0f219
Removing intermediate container 75d1449b29bf
Step 8 : EXPOSE 389
 ---> Running in 51a48fe427c9
 ---> 4c8a5be716a8
Removing intermediate container 51a48fe427c9
Successfully built 4c8a5be716a8
```

Observar l'historial de la imatge generada:

```
[root@hp01 ldap-01]# docker history edt.asix.m06.ldap:b01
IMAGE          CREATED         CREATED BY                                      SIZE                COMMENT
4c8a5be716a8   2 minutes ago   /bin/sh -c #(nop) EXPOSE 389/tcp                0 B
89fdfdc0f219   2 minutes ago   /bin/sh -c #(nop) COPY multi:09ca1d222d477781   17.62 kB
2cd8c02dff3b   3 minutes ago   /bin/sh -c mkdir /var/tmp/home/2asix            0 B
65bc905895c3   3 minutes ago   /bin/sh -c mkdir /var/tmp/home/1asix            0 B
ba1841bce215   3 minutes ago   /bin/sh -c mkdir /var/tmp/home                  0 B
ed47bee98a1b   3 minutes ago   /bin/sh -c mkdir /opt/edt-ldap                  0 B
9da07accffbd   4 minutes ago   /bin/sh -c yum -y install openldap openldap-s   194.8 MB
c1a7070c2fd8   10 minutes ago  /bin/sh -c #(nop) MAINTAINER @edt "ASIX M06/M   0 B
ded7cd95e059   5 months ago    /bin/sh -c #(nop) ADD file:4be46382bcf2b095fc   186.5 MB
48ecf305d2cf   6 months ago    /bin/sh -c #(nop) MAINTAINER Lokesh Mandvekar   0 B
```

Pràctiques

Pràctiques de m11:

Primera pràctica: Dockerfile

Dockerfile:

- crear ldapserver:base i pujar-lo al git. explicat els elements bàsics de dockerfile: from, maintainer, run, copy, workdir i cmd.
- crear ldapserver:dataDB i pujar-lo al git. automatitzar que es comporti com un daemon el container (detached amb slapd en foreground).

Conceptes:

- elements del Dockerfile
- procés amb PID 1 de un container: el programa principal que es crida. Usar CMD o posar /bin/bash a la comanda.
- docker run --rm per executar containers com a ordres que finalitzen.
- usar docker exec amb /bin/bash per entrar a containers, també quan hem fet malament el proceś 1 (un daemon) i es mor i volem engegar-lo manualment.
- En un container s'hi executa un servei, el servei s'executa en primer pla!! perquè té el pid 1 i ha de mantenir l'execució del container.

Segona pràctica: Container detach

Documentació:

- descarregar un docker autocontingut amb documentació que permet ser consultada localment al port 4000.

Practicar les comandes de dockerfile:

- practicar ADD, EXPOSE, CMD i ENTRYPOINT

Pràctiques:

- 01-ldapserver:base
- 02-ldapserver:dataDB container amb servei slapd i base de dades full. Ha de poder funcionar en mode deatach. Si ha de serlleuger, un cop verificat el funcionament eliminar els dnf install innecessaris.
- 03-hostbase:base incorporar-hi tots els serveis (xinetd, ftp, http, pop, imap pops, imaps, http-switch, altres serveis -bis, tftp, telnet, ssh i sendmail). Posar 'xixa' de publicació usar un tar-gz amb ADD.
- 04-postgres:base.

Tercera pràctica: Automated Builds

automated builds

branch master /ldapserver:base base

branch master /ldapserver:dataDB dataDB

observar els logs un cop el build (que tarda!) és success

Ports: EXPOSE dels ports 389 i 636 Només declara, no publica. Usar -P o -p per publicar.
-P publica tot l'exposat, -p permet definir-ho amb el format: ip:hostport:containerport
Si no es posa ip són totes les ips del host. Si no es posa hostport se n'assigna un de dinàmic, estàndard 32768. Es poden indicar rangs a hostport i containerport. A containerport es pot indicar /tcp o /udp.
Amb les ordres docker ps mirar els ports. Amb docker port s'examina per a un conainer donat.
Practicar: assignar un expose, usar -P, -p 389, -p 389.389, -p ip-publica:389:389
Docker network

crear xarxes mynet01, mynet02, mynet02. docker network ls i inspect de cada xarxa.
Esborrar mynet03 i fer-la de nou.
Crear els hosts h1 (389) h2 (1389) i h2 (2389) verificar que des del host funciona ldapsearch amb cada ip dels containers (h1, h2 i h3) i amb cada port local (389, 1389, 2039).
Entrar amb docker exec a h1 i verificar que són vàlids els noms de host h1 h2 i h3. Mirar /etc/hosts i fer ldapserch -h h1...

Quarta pràctica: Port - Network

xx
xx

Cinquena pràctica: Version

xx
xx

Sisena pràctica: Service

xx
xx

Setena pràctica: Volumes

xx
xx

Vuitena pràctica: Portainer

xx
xx

Novena pràctica: Swarm

xx
xx

Desena pràctica: Stack

xx
xx

Desena pràctica: Machine

xx
xx

Onzena pràctica: Cloud

xx
xx

Dotzena: AMI AWS EC2

xx
xx

Pràctiques imatges

hostbase// hostservices

imatge amb serveis de xarxa. incloure un puserver.py servidor que retorna uname, host i ip.

postgres

imatge postgres amb un volume per les dades i compose per múltiples instàncies.

nfs

imatge nfs amb un volume per les dades i compose. Incorporrar un script de creació automàtica dels homes dels usuaris.

Apèndix

Apèndix A: Tricks

ctrl-p + ctrl-q

- per sortir d'un container sense aturar-lo, es pot tornar a reenganxar amb attach.

container run command

- es poden usar containers de docker com a containers per a la execucio de una sola ordre (isolada) i que en finalitzar el container s'esborri:
  $ docker run -rm -it fedora /usr/bin/who

docker logs [-f] [contname]

- mostra la sortida stdout que generen els containers daemon.

eliminar volums no referenciats

- **Note:** Docker will not warn you when removing a container *without* providing the -v option to delete its volumes. If you remove containers without using the -v option, you may end up with "dangling" volumes; volumes that are no longer referenced by a container. You can use docker volume ls -f dangling=true to find dangling volumes, and use docker volume rm <volume name> to remove a volume that's no longer needed.

oficial repositories

- repositoris oficials, explore: https://hub.docker.com/explore/
- fedora: https://hub.docker.com/_/fedora/
- postgres: https://hub.docker.com/_/postgres/

init script de postgres

- https://hub.docker.com/_/postgres/
- For example, the Postgres Official Image uses the following script as its ENTRYPOINT:

```
#!/bin/bash
set -e

if [ "$1" = 'postgres' ]; then
    chown -R postgres "$PGDATA"

    if [ -z "$(ls -A "$PGDATA")" ]; then
        gosu postgres initdb
    fi

    exec gosu postgres "$@"
fi

exec "$@"
```

- **Note**: This script uses the exec Bash command so that the final running application becomes the container's PID 1. This allows the application to receive any Unix signals sent to the container. See the ENTRYPOINT help for more details.
- The helper script is copied into the container and run via ENTRYPOINT on container start:
  COPY ./docker-entrypoint.sh /
  ENTRYPOINT ["/docker-entrypoint.sh"]

- This script allows the user to interact with Postgres in several ways.
- It can simply start Postgres:
  $ docker run postgres
- Or, it can be used to run Postgres and pass parameters to the server:
  $ docker run postgres postgres --help
- Lastly, it could also be used to start a totally different tool, such as Bash:
  $ docker run --rm -it postgres bash

lsblk

- llistar block devices

bridges:

- buid your own bridge:
  https://docs.docker.com/engine/userguide/networking/default_network/build-bridges/
- brctl show

Crear un bridge propi:

```
# Create our own bridge
$ sudo brctl addbr bridge0
$ sudo ip addr add 192.168.5.1/24 dev bridge0
$ sudo ip link set dev bridge0 up
$ brctl show
$ ip addr show bridge0
```

```
# systemctl stop docker

[root@hp01 ~]# brctl addbr docker2
[root@hp01 ~]# ip addr add 10.10.10.1/24 dev docker2
[root@hp01 ~]# ip link set
[root@hp01 ~]# ip link set docker2 up
```

```
[root@hp01 ~]# brctl show
bridge name     bridge id           STP enabled     interfaces
docker0         8000.0242f0955a0c   no
docker1         8000.000000000000   no
docker2         8000.000000000000   no
```

```
virbr0              8000.525400ec6fd5    yes      virbr0-nic
```

```
[root@hp01 ~]# ip a
6: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
        link/ether 02:42:f0:95:5a:0c brd ff:ff:ff:ff:ff:ff
        inet 172.17.42.1/16 scope global docker0
        valid_lft forever preferred_lft forever
        inet6 fe80::42:f0ff:fe95:5a0c/64 scope link
        valid_lft forever preferred_lft forever
13: docker1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
        link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
        inet 172.18.0.1/24 scope global docker1
        valid_lft forever preferred_lft forever
        inet6 fe80::b8f6:11ff:fef8:d6b9/64 scope link
        valid_lft forever preferred_lft forever
18: docker2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default
        link/ether fe:3e:13:39:6a:ac brd ff:ff:ff:ff:ff:ff
        inet 10.10.10.1/24 scope global docker2
        valid_lft forever preferred_lft forever
```

[root@hp01 ~]# vim $(locate docker.service)

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network.target docker.socket
Requires=docker.socket

[Service]
Type=notify
ExecStart=/usr/bin/docker daemon -b=docker2 -H fd://
MountFlags=slave
LimitNOFILE=1048576
LimitNPROC=1048576
LimitCORE=infinity

[Install]
WantedBy=multi-user.target
```

[root@hp01 ~]# systemctl daemon-reload
[root@hp01 ~]# systemctl start docker

```
[root@hp01 ~]# docker start i25
i25
[root@hp01 ~]# docker attach i25
[root@bddf16a3fefd /]#
[root@bddf16a3fefd /]#
```

```
[root@bddf16a3fefd /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
```

```
        valid_lft forever preferred_lft forever
21: eth0@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
        link/ether 02:42:0a:0a:0a:02 brd ff:ff:ff:ff:ff:ff
        inet 10.10.10.2/24 scope global eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::42:aff:fe0a:a02/64 scope link
        valid_lft forever preferred_lft forever
```

```
[root@hp01 ~]# docker start i26
i26
[root@hp01 ~]# docker attach i26
```

```
[root@hp01 ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242f0955a0c       no
docker1         8000.000000000000       no
docker2         8000.2275c7e2736a       no              veth92abd0d
                                                        vethcb20a02
virbr0          8000.525400ec6fd5       yes             virbr0-nic
```

Apèndix B : Llistat d'ordres docker

Podeu consultar la documentació de cada ordre amb:

```
# man docker-<ordre>
# docker <ordre> --help
```

attach
    Attach to a running container
    See docker-attach(1) for full documentation on the attach command.
build
    Build an image from a Dockerfile
    See docker-build(1) for full documentation on the build command.
commit
    Create a new image from a container's changes
    See docker-commit(1) for full documentation on the commit command.
cp
    Copy files/folders from a container's filesystem to the host
    See docker-cp(1) for full documentation on the cp command.
create
    Create a new container
    See docker-create(1) for full documentation on the create command.
diff
    Inspect changes on a container's filesystem
    See docker-diff(1) for full documentation on the diff command.
events
    Get real time events from the server
    See docker-events(1) for full documentation on the events command.
exec
    Run a command in a running container
    See docker-exec(1) for full documentation on the exec command.
export
    Stream the contents of a container as a tar archive
    See docker-export(1) for full documentation on the export command.
history
    Show the history of an image
    See docker-history(1) for full documentation on the history command.
images
    List images
    See docker-images(1) for full documentation on the images command.
import
    Create a new filesystem image from the contents of a tarball
    See docker-import(1) for full documentation on the import command.
info
    Display system-wide information
    See docker-info(1) for full documentation on the info command.
inspect
    Return low-level information on a container or image
    See docker-inspect(1) for full documentation on the inspect command.

kill
Kill a running container (which includes the wrapper process and everything inside it). See docker-kill(1) for full documentation on the kill command.

load
Load an image from a tar archive
See docker-load(1) for full documentation on the load command.

login
Register or login to a Docker Registry
See docker-login(1) for full documentation on the login command.

logout
Log the user out of a Docker Registry
See docker-logout(1) for full documentation on the logout command.

logs
Fetch the logs of a container
See docker-logs(1) for full documentation on the logs command.

pause
Pause all processes within a container
See docker-pause(1) for full documentation on the pause command.

port
Lookup the public-facing port which is NAT-ed to PRIVATE_PORT
See docker-port(1) for full documentation on the port command.

ps
List containers
See docker-ps(1) for full documentation on the ps command.

pull
Pull an image or a repository from a Docker Registry
See docker-pull(1) for full documentation on the pull command.

push
Push an image or a repository to a Docker Registry
See docker-push(1) for full documentation on the push command.

rename
Rename a container.
See docker-rename(1) for full documentation on the rename command.

restart
Restart a running container
See docker-restart(1) for full documentation on the restart command.

rm
Remove one or more containers
See docker-rm(1) for full documentation on the rm command.

rmi
Remove one or more images
See docker-rmi(1) for full documentation on the rmi command.

run
Run a command in a new container
See docker-run(1) for full documentation on the run command.

save
Save an image to a tar archive
See docker-save(1) for full documentation on the save command.

search
Search for an image in the Docker index

See docker-search(1) for full documentation on the search command.

start

Start a stopped container
See docker-start(1) for full documentation on the start command.

stats

Display a live stream of one or more containers' resource usage statistics
See docker-stats(1) for full documentation on the stats command.

stop

Stop a running container
See docker-stop(1) for full documentation on the stop command.

tag

Tag an image into a repository
See docker-tag(1) for full documentation on the tag command.

top

Lookup the running processes of a container
See docker-top(1) for full documentation on the top command.

unpause

Unpause all processes within a container
See docker-unpause(1) for full documentation on the unpause command.

version

Show the Docker version information
See docker-version(1) for full documentation on the version command.

wait

Block until a container stops, then print its exit code
See docker-wait(1) for full documentation on the wait command.

Apèndix C: Resum de comandes

```
# docker run -i -t fedora /bin/bash
# docker run --name nomContainer -i -t fedora /bin/bash

# docker ps [-a] [-l]
# docker top nomContainer

# docker images
# docker images fedora
# docker images usernew/*

# docker history imageName

# docker commit containerName imageName
# docker commit -m “text commit” --author “nomaAutor” containerName imageName

# docker tag imageName:tag newImageName:newTag

# docker rm nameContainer
# docker rmi imageName

nomConainer → cont.prova01
imageName → usernew/prova
imageName → usernew/prova:tag
```

```
# docker version
# docker info
# docker help
# docker <command> --help
# man docker-<ordre>
```

```
# docker run --name nomContainer -i -t imageName /bin/bash
# docker ps -n=nº

# docker [-a] start nomContainer [nomContainer]
# docker stop nomContainer [nomContainer]
# docker attach nomContainer
```

```
# docker inspect containerName
# docker inspect nameInage

# docker inspect --format ‘{{.NetworkSettings}}’ nomContainer
```

```
# docker login
```

```
# docker logout

# docker push username/nameImage
# docker pull   username/nameImage

# docker search nameImage
# docker search username/*
```

```
# docker  build -t="edt.asix.m06.ldap:b01" .
# docker history edt.asix.m06.ldap:b01

# docker run --name cont.ldap.01 -p 389 -i -t edt.asix.m06.ldap:b01 /bin/bash
# docker run --name cont.ldap.01 -P -i -t edt.asix.m06.ld:qap:b01 /bin/bash

# docker ps -l

# docker port edt.asix.m06.ldap:b01
```

Apèndix D Docker Network / Bridge

Docker Network

Te dejo como hacer networks definidas y asignarlas a los dockers nuevos o añadirlas a dockers existentes

inspec. networks
docker network ls
insepec. network concreta
docker network inspect <name/id network>
Crear network (default docker +1 ip)
docker network create <namenetwork>
comprobar nueva network (comprobar cmd anteriores)

conectar un docker creado a la red
docker network connect vpn_network base
docker network inspect vpn_network

iniciar un docker
docker run -t -i --net=<namenetwork> altairisx/base:fedora /bin/bash
*** imagen muy base solo de fedora (no tool*s) dnf install iproute iputils

defined network " definir una red propia por ejemplo una para jugar con vpn's"
docker network create --driver=bridge --subnet=172.22.0.0/16 -o
com.docker.network.bridge.enable_icc=true -o
com.docker.network.bridge.enable_ip_masquerade=true -o
com.docker.network.bridge.host_binding_ipv4=0.0.0.0 foo.local

** este es un ejemplo "semiconstruido" por la fuente, opciones default de una red de docker:
"Options": {
"com.docker.network.bridge.default_bridge": "true",
"com.docker.network.bridge.enable_icc": "true",
"com.docker.network.bridge.enable_ip_masquerade": "true",
"com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
"com.docker.network.bridge.name": "docker0",
"com.docker.network.driver.mtu": "1500"
}

contrastadas con la creación de la red se nota que falta alguna opción,por lo que compruebo la red que se crea puedes acceder a Internet assignando la red nueva a un docker nuevo "RUN", así que todo funciona.

Més apartats:

Automatized Builds

Configure automated builds on dockerhub
https://docs.docker.com/docker-hub/builds/

Posar Dockerfile amb la D majúscuula!!

Disposar de compte a:

- GitHub
- Docker Hub

Configurar dockerhub per poder accedir directament a github i usar els repositoris que hi ha per generar imatges automatitzades:

- ❏ Account Settings
  - ❏ Linked accounts & services
    - ❏ Link GitHub. Seguir els passos per inicar sessió a github i cedir els permisos per deixar-hi entrar a dockerhub.
- ❏ Create
  - ❏ create Automated Builds
    - ❏ Create autobuild gityhub
      - ❏ Escollir l'usuari i el repositori, per exemple:
        edtasixm06 asixmo6-docker
        ** aqui faria una imatge, però si és un repositori que dins seu conté moltes carpetes de diferents imatges, cal seleccionar més detalladament què fer:
      - ❏ Click here to customize behavior
        - ❏ edtasixm06 zzzautobuild tag ldapserver:18base /asixm06-docker/ldapserver:18base latest
- ❏ Seleccionar el repositori creat
  A build settings desactivar: When active, builds will happen automatically on pushes.
- ❏ trigger: fer trigger manualment dels builds quan els volem generar de nou.

Exemples d'automatized builds:

a) Un directori dins d'un repositori
   namespace: edtasixm06 (és el nom del compte d'usuari)
   name: zzzautobuild (és el nom que tindrà a dockerhub el repositori)
   type: brach

name: master
dockerfile location: /ldapserver:18base (on hi ha el dockerfile. És la ruta absoluta indicant nomrepositori/directori del github que conté el dockerfile i directori de generació).
docker tag name: latest (nom del tag que volem que tingui).

Build details:

- podem observar els detalls de la creació
- també els errors, clicant, i veiem què és el que ha fallat.

dockerfile:

- es pot veure el contingut del dockerfile que ha generat el build

Portainer

Portainer és un container preparat per monitoritzar el daemon de docker del host. És un visor gràfic que permet veure tot el que fa el host, quines imatges te, quines networks, volumes, containers, etc.

Documentació:
https://medium.com/lucjuggery/about-var-run-docker-sock-3bfd276e12fd

```
# docker container run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer

# firefox → localhost:9000
```

A /var/run/docker.sock és un socket unix on escolta el daemon del docker.

Docker offline documentation

https://docs.docker.com/docsarchive/

Docs for v18.03 ***(current)*** are accessible at **https://docs.docker.com/**, or to view the docs offline on your local machine, run:
$ docker run -ti -p 4000:4000 docs/docker.github.io:latest

Docs for v1.12 are accessible at **https://docs.docker.com/v1.12/**, or to view the docs offline on your local machine, run:
docker run -ti -p 4000:4000 docs/docker.github.io:v1.12

Per visualitzar el contingut des d'un navegador: **localhost:40000**