

https://docs.docker.com/get-started/docker_cheatsheet.pdf

Comandos mas utiles and utilizados

Docker brinda la capacidad de empaquetar y ejecutar una aplicación en un entorno vagamente aislado llamado contenedor.

El aislamiento y la seguridad le permiten ejecutar muchos contenedores simultáneamente en un host determinado. Los contenedores son

liviano y contiene todo lo necesario para ejecutar la aplicación, por lo que no necesita confiar en lo que está actualmente

instalado en el host. Puede compartir contenedores fácilmente mientras trabaja y asegurarse de que todas las personas con las que comparte obtengan el mismo contenedor que funciona de la misma manera.

GENERAL COMMANDS

Start the docker daemon

```
docker -d
```

Get help with Docker. Can also use `-help` on all subcommands

```
docker --help
```

Display system-wide information

```
docker info
```

IMAGES

Las imágenes de Docker son un paquete ligero, independiente y ejecutable de software que incluye todo lo necesario para ejecutar una aplicación: código, tiempo de ejecución, herramientas del sistema, bibliotecas del sistema y configuraciones.

Build an Image from a Dockerfile:

```
docker build -t <images_name>
```

Build an Image from a Dockerfile without the cache:

```
docker build -t . --no-cache <images_name>
```

List local images:

```
docker images
```

```
Delete an Image:
docker rmi <images_name>
Remove all unused images:
docker image prune
```

Ejemplo

Dockerfile

```
FROM debian

RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf
/var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data

ENV APACHE_RUN_GROUP www-data

ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

ADD ["index.html","/var/www/html/"]

ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Index.html

```
<h1>Prueba de funcionamiento contenedor docker</h1>
```

Ejecutar

```
Build de image:
docker build -t lleida/apache:latest .
Run the image
docker run -p 80:80 --name servidor lleida/apache
```

Prueba de funcionamiento contenedor docker

```
docker images
```

```
Esteban (-zsh)
esteban ~ % docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
lleida/apache        latest       3e47964eb33a  27 minutes ago 265MB
hello-world          latest       b038788ddb22  7 weeks ago   9.14kB
traefik              v2.9.9      81983806f35c  2 months ago  133MB
redis                alpine       a4cf5af74f5e  2 months ago  30.4MB
traefik/whoami        latest       c0befa930d25  3 months ago  6.27MB
authelia/authelia    latest       e4a633b495a8  6 months ago  53.9MB
eclipse-mosquitto    latest       0d0308dcc83d  7 months ago  11.6MB
grafana/grafana       latest       f49d00abd1b4  7 months ago  325MB
influxdb              latest       06dfc9129636  7 months ago  362MB
telegraf              latest       748da2f4f3cd  7 months ago  354MB
esteban ~ %
```

```
docker images
docker rmi lleida/apache
```

```
Esteban (-zsh)
esteban ~ % docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
lleida/apache        latest       3e47964eb33a  35 minutes ago 265MB
hello-world          latest       b038788ddb22  7 weeks ago   9.14kB
traefik              v2.9.9      81983806f35c  2 months ago  133MB
redis                alpine       a4cf5af74f5e  2 months ago  30.4MB
traefik/whoami        latest       c0befa930d25  3 months ago  6.27MB
authelia/authelia    latest       e4a633b495a8  6 months ago  53.9MB
eclipse-mosquitto    latest       0d0308dcc83d  7 months ago  11.6MB
grafana/grafana       latest       f49d00abd1b4  7 months ago  325MB
influxdb              latest       06dfc9129636  7 months ago  362MB
telegraf              latest       748da2f4f3cd  7 months ago  354MB
esteban ~ % docker rmi lleida/apache
Untagged: lleida/apache:latest
Deleted: sha256:3e47964eb33a942faa8295fa83c349846833724678831567adf1a44d82058499
esteban ~ %
```

```
docker image prune
```

```
esteban ~ % docker image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
esteban ~ %
```

```
docker system prune
```

The screenshot shows a terminal window titled "Esteban (-zsh)" with the following output for the command `docker system prune`:

```
esteban ~ % docker system prune
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all dangling images
- all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
e6d2835a697792ff6a064fa7ae59d9ef491e65a1fad8c77d6a32efa1c25c4597
dccfe313e35b40b03e2db4a694c42bb284287a0b38174f697f60cb19078ccb1b
545faa9b688a65c66470ba836f0d9596f91d8697771ef5d0dc1428ccd02e9617
2b5aee535c5f85cd8cd78b7627531df56410bdf93580efa53132518497c7eddf
35e895f4c79513c81d46f8fb188b343921517bf26b0849f4907b316e20a7d5ef
bad67c07fcb09c790bc769a1b7f961271c34bbd266dc7524b3c14b788e1b2a3b

Deleted Networks:
esteban_net

Deleted build cache objects:
tzhga20agbl6jc84dp8rsofxi
k6ho6wj8daglfan4g4w71eh6z
6e6nomo4btii3iskujst1ij72
r7rpc32pciljctnmq2em2erwq
tumgnwaoy13arv92w053xb2jk
hkewph9a287j6tx4entaya9yn
0hqt3z55nxercb4q38fwdzi1m
mt1vlymiakf76eqk7edhsx9yw
np1o93u9bcr0apei2h0m8uju7
d81atqy5n7rmzrr7ay6o7m5gt
pfhw17nts1hkrk2k9ks9d762q
tzfpt2542nwaomcq94dx5axl
qlwv7qnf1zysmo9eui76arh84
ok3rtcqg3pvx394aw7715fh5w
jy7snk9weut4v59r1ljzllia
szijby6wcxi9xzjjw3sugto1
ucwpadnzb3sm8ii96a6txt5wi
u160q0w893x10gzvxnkze21g
a46bwolio8p0nur3d74nhax9u
9yntu74s3pgsly9pols1f6elu
ks14vvbbyqthjemjvkkcklvh2
ty3uogt1lwec1g6awtby68hkw
r2prrdgda82w44dtqojzx4ffv
8imnyvji9r5kr41ha5bbwhoco
zuw1m2c1deko1oc5wpqcdxxaf
ejdkiwlc090l7argjf4wxzwfj
kxzroyz78l7lvnz81gn5x4cad
j1gjj01qht5e3cmx9h4a6trp
```

In the background, a chat window titled "OEM.IL - 24/7 with MHP, CGI, PS, Ex..." is visible, showing messages from users like Schwarzbach, Petra and Krause, Peter, along with status updates about QA stages and AWS switch completion.

y más ...

CONTAINERS

Un contenedor es una instancia en tiempo de ejecución de una imagen acoplable. Un contenedor siempre funcionará igual, independientemente de la infraestructura. Los contenedores aíslan el software de su entorno y aseguran que funciona uniformemente a pesar de las diferencias, por ejemplo, entre desarrollo y puesta en escena.

Create and run a container from an image, with a custom name:

```
docker run --name
```

Run a container with and publish a container's port(s) to the host:

```
docker run -p
```

Run a container in the background:

```
docker run -d
```

Start or stop an existing container:

```
docker start|stop
```

Remove a stopped container:

```
docker rm
```

Open a shell inside a running container:

```
docker exec -it sh
```

Fetch and follow the logs of a container:

```
docker logs -f
```

To inspect a running container:

```
docker inspect
```

To list currently running containers:

```
docker ps
```

List all docker containers (running and stopped):

```
docker ps --all
```

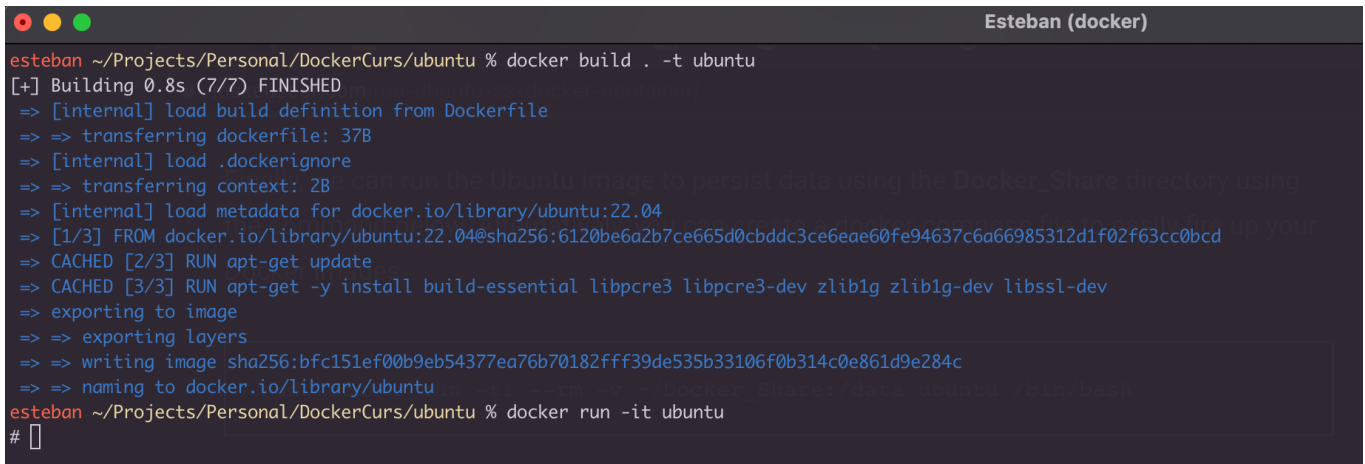
View resource usage stats:

```
docker container stats
```

Ejemplo 1:

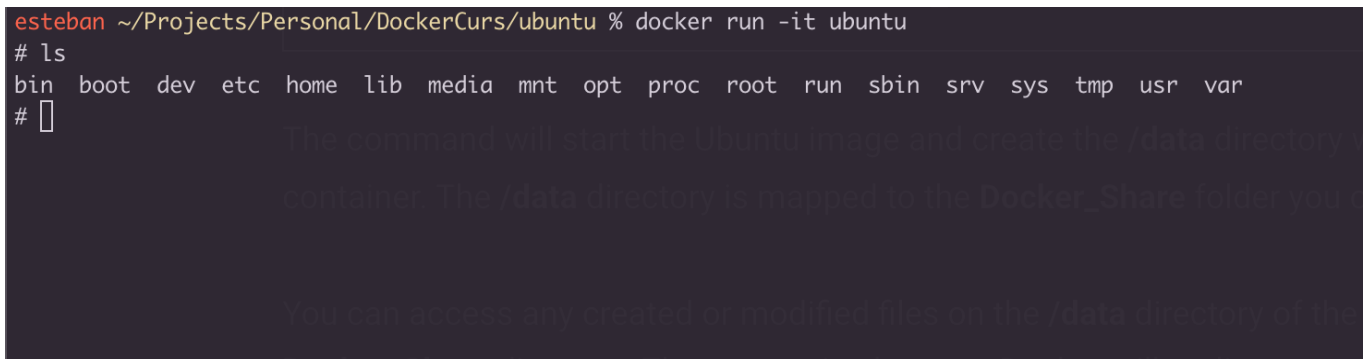
```
FROM ubuntu:22.04
USER root
RUN apt-get update
RUN apt-get -y install build-essential libpcre3 libpcre3-dev zlib1g zlib1g-
dev libssl-dev
```

```
docker build . -t ubuntu
```

A terminal window titled "Esteban (docker)" showing the execution of the Docker build command. The output indicates that the build was successful, taking 0.8 seconds. It shows the process of loading the build definition, transferring the Dockerfile, and installing the necessary packages. The final image is named "docker.io/library/ubuntu:22.04".

```
esteban ~/Projects/Personal/DockerCurs/ubuntu % docker build . -t ubuntu
[+] Building 0.8s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 37B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:22.04
=> [1/3] FROM docker.io/library/ubuntu:22.04@sha256:6120be6a2b7ce665d0cbddc3ce6ae60fe94637c6a66985312d1f02f63cc0bcd
=> CACHED [2/3] RUN apt-get update
=> CACHED [3/3] RUN apt-get -y install build-essential libpcre3 libpcre3-dev zlib1g zlib1g-dev libssl-dev
=> exporting to image
=> => exporting layers
=> => writing image sha256:bfc151ef00b9eb54377ea76b70182fff39de535b33106f0b314c0e861d9e284c
=> => naming to docker.io/library/ubuntu
esteban ~/Projects/Personal/DockerCurs/ubuntu % docker run -it ubuntu
#
```

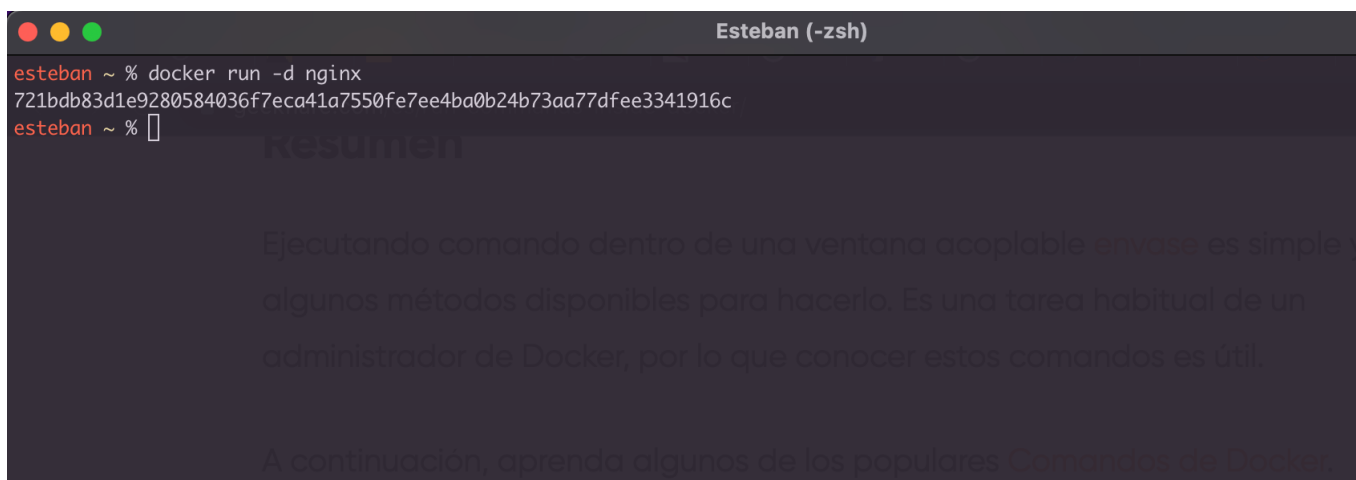
```
docker run -it ubuntu
```

A terminal window showing the execution of the Docker run command. The output shows the root directory of the container, with various system directories listed. The prompt is a hash symbol followed by a space.

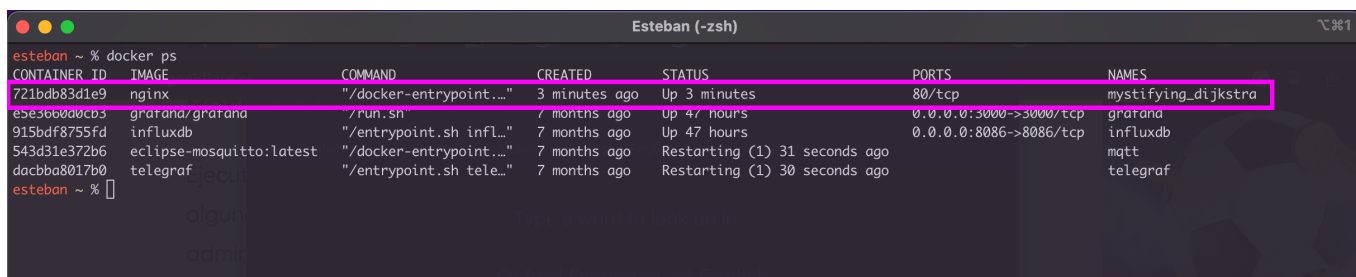
```
esteban ~/Projects/Personal/DockerCurs/ubuntu % docker run -it ubuntu
# ls
bin boot dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
#
```

Ejemplo 2:

```
docker run -d nginx
```



docker ps



docker exec -it 721 bash

