# Introducción a Dockerfile

## ¿Qué es un Dockerfile?

Es un documento de texto que contiene todos los **comandos** que vamos a ejecutar a la hora de crear nuestra imagen. Se podría decir que nuestro Dockerfile va a ser la receta que Docker va a seguir para poder crear nuestra imagen.

## FROM

El comando **FROM** nos va a servir para basar nuestra nueva imagen en una imagen ya existente (las podemos ver en: https://hub.docker.com). Este comando **siempre** tiene que ser el primero en nuestro Dockerfile

```
FROM node:11
```

## RUN

El comando **RUN** nos va a servir para correr comandos en una terminal dentro de nuestro *container.* Esto es util para cambiar configuración a nivel sistema operativo o bien instalar paquetes de forma global.

```
RUN npm install -g pm2 --silent
```

## CMD

El comando **CMD** nos va a servir para indicarle a nuestra *imagen* que comando tiene que correr por defecto al crear nuestro *container.* En caso de que nuestro comando contenga parámetros hay que escribir el comando en formato de *array[]*

```
CMD ["node", "server.js"]
```

## EXPOSE

El comando **EXPOSE** nos va a servir para indicarle a nuestro container que puerto escuchar mientras este corriendo. Esto es especialmente útil para cuando estamos corriendo un servidor.

```
EXPOSE 3000
```

## WORKDIR

El comando **WORKDIR** nos va a servir para indicarle a nuestra imagen que directorio tiene que usar como base para los comandos que modifiquen el sistema de archivos. En caso de que no exista el directorio indicado, Docker lo va a crear.

```
WORKDIR /usr/src/app
```

# COPY

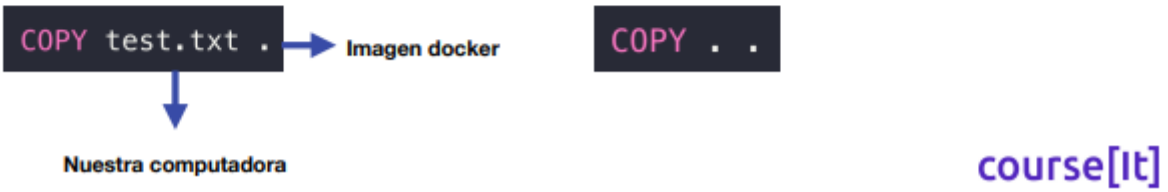
El comando **COPY** nos va a servir para copiar archivos desde nuestra computadora a nuestra imagen de Docker. Con el comando **COPY** podemos copiar archivos o directorios completos.

```
COPY test.txt .
```

```
COPY . .
```

# COPY

El comando **COPY** nos va a servir para copiar archivos desde nuestra computadora a nuestra imagen de Docker. Con el comando **COPY** podemos copiar archivos o directorios completos.



# ENV

El comando **ENV** nos va a servir para crear variables de ambiente dentro de nuestra imagen. Por ejemplo para indicarle si estamos en un ambiente de testing o productivo.

```
ENV NODE_ENV production
```

## ¿Qué hace este Dockerfile?

```
# Buscamos la imagen de node version 10.15
FROM node:10.15-alpine
# Establecemos una carpeta de trabajo
WORKDIR /usr/src/app
# Instalamos las dependencias
RUN npm install  --silent
# Copiamos todos los archivos
COPY . .
# Ejecutamos el comando de npm build
RUN npm run build
# Exponemos el puerto 3001
EXPOSE 3001
# Corremos nuestra imagen con el comando npm start
CMD ["npm", "start"]
```

# Instrucciones básicas

course[It]

## docker ps

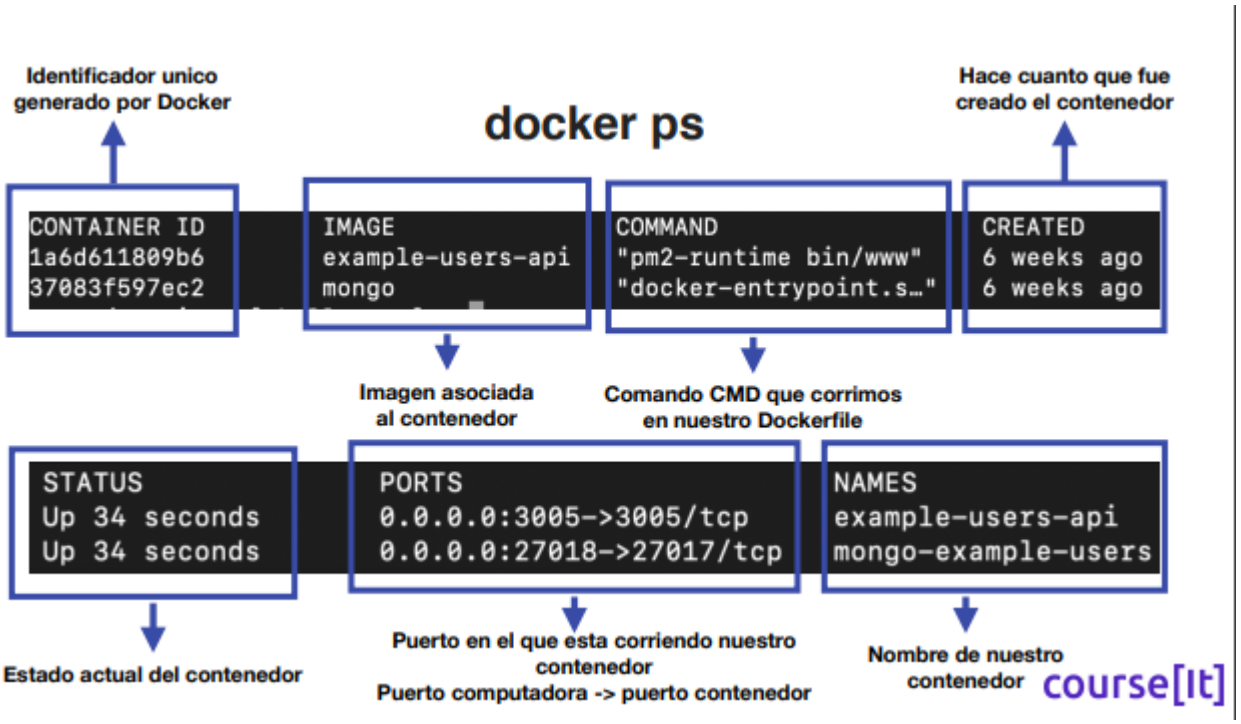
El comando **docker ps** nos sirve para listar todos los contenedores que actualmente estén corriendo.

En caso de querer ver los contenedores que ya terminaron o murieron en el pasado, podemos correr el comando **docker ps -a**

## docker ps

```
CONTAINER ID        IMAGE               COMMAND                  CREATED
1a6d611809b6        example-users-api   "pm2-runtime bin/www"    6 weeks ago
37083f597ec2        mongo               "docker-entrypoint.s…"   6 weeks ago
```

```
STATUS              PORTS                      NAMES
Up 34 seconds       0.0.0.0:3005->3005/tcp     example-users-api
Up 34 seconds       0.0.0.0:27018->27017/tcp   mongo-example-users
```




# docker build

El comando **docker build** nos sirve para crear una imagen en base a un Dockerfile

El uso *normal* del comando es: **docker build .** lo que nos va a generar una imagen en base a un Dockerfile situado en esa misma carpeta.

course[It]

# Para crear una imagen

```
FROM node:10.15-alpine
ENV NODE_ENV ci
WORKDIR /usr/src/app
RUN npm install --production --silent
COPY . .
RUN npm run build
EXPOSE 3001
CMD ["node", "server.js"]
```

+ `docker build .`

course[It]

# docker build

```
Sending build context to Docker daemon   15.8MB
Step 1/10 : FROM node:10.15-alpine
 ---> 288d2f688643
Step 2/10 : ENV NODE_ENV production
 ---> Using cache
 ---> 0cfea731a3f6
Step 3/10 : WORKDIR /usr/src/app
 ---> Using cache
 ---> f9c39ce52759
```

...

```
Successfully built c2aa558e8f74
```

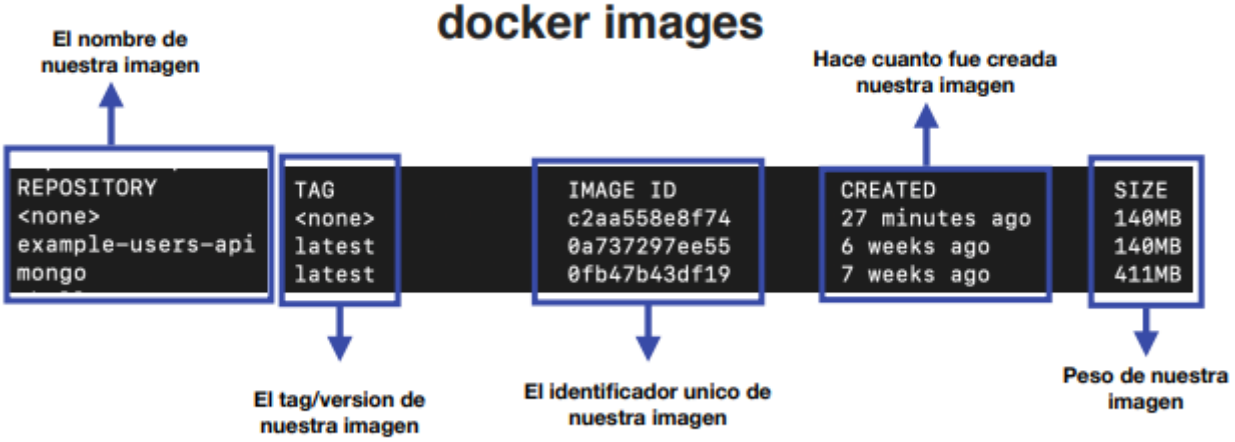
# docker images

El comando **docker images** nos sirve para listar todas las imágenes que creamos

# docker images

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|---|---|---|---|---|
| <none> | <none> | c2aa558e8f74 | 27 minutes ago | 140MB |
| example-users-api | latest | 0a737297ee55 | 6 weeks ago | 140MB |
| mongo | latest | 0fb47b43df19 | 7 weeks ago | 411MB |

# docker images



# docker run

El comando **docker run** nos sirve para crear un container a partir de una imagen

```
docker run -p PUERTO_EXPOSE:SERVIDOR IMAGE_ID
```

```
docker run -p 3000:3000 05a56348ea50
```