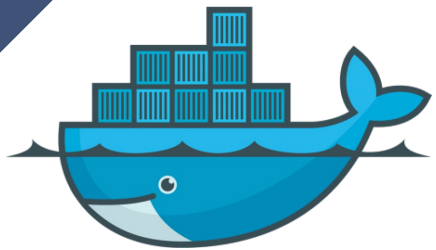
# Docker de 0 a 100

Israel Oña Ordoñez | DevOps Engineer

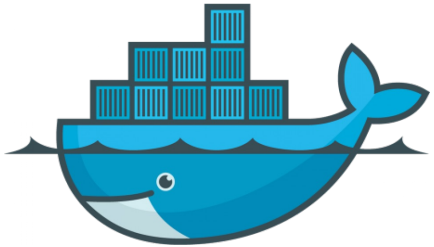


devisra

# 4

# Imágenes

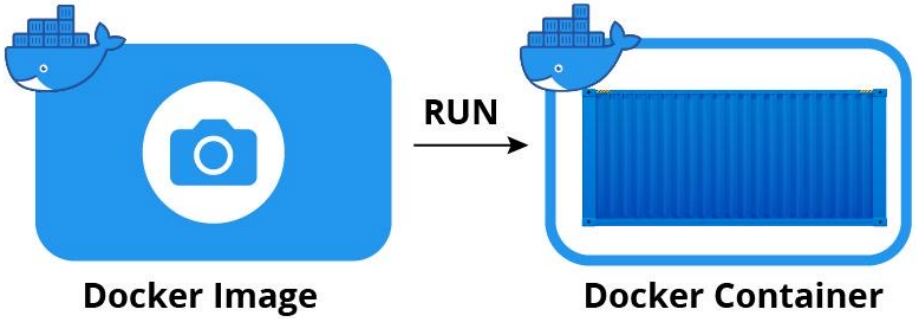
Dockerfiles y construcción de imágenes



# ¿Qué es una imagen?

Una imagen de Docker es un archivo inmutable (plantilla) de solo lectura que define cómo se creará un contenedor (código fuente, librerías, dependenciasy otros archivos requeridos) a partir de un Dockerfile.

Es comparable a una instantánea en entornos de máquinas virtuales.



# Dockerfile

Un Dockerfile es un documento de texto que contiene todos los comandos necesarios para crear una imagen.

Los comandos poseen una estructura de Instrucción con Argumentos.

Dockerfile

```
INSTRUCTION ARGUMENT
```

Dockerfile

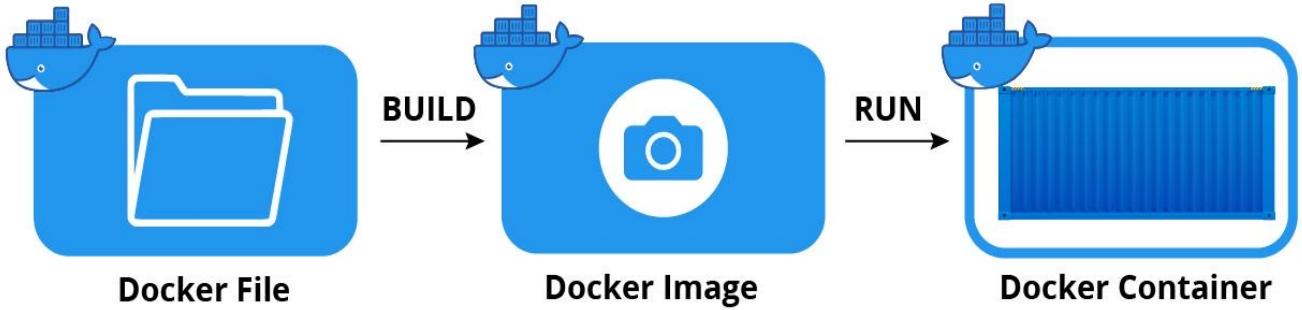
```
FROM Ubuntu

RUN apt-get update
RUN apt-get install python

RUN pip install flask
RUN pip install flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Start from a base OS or another image

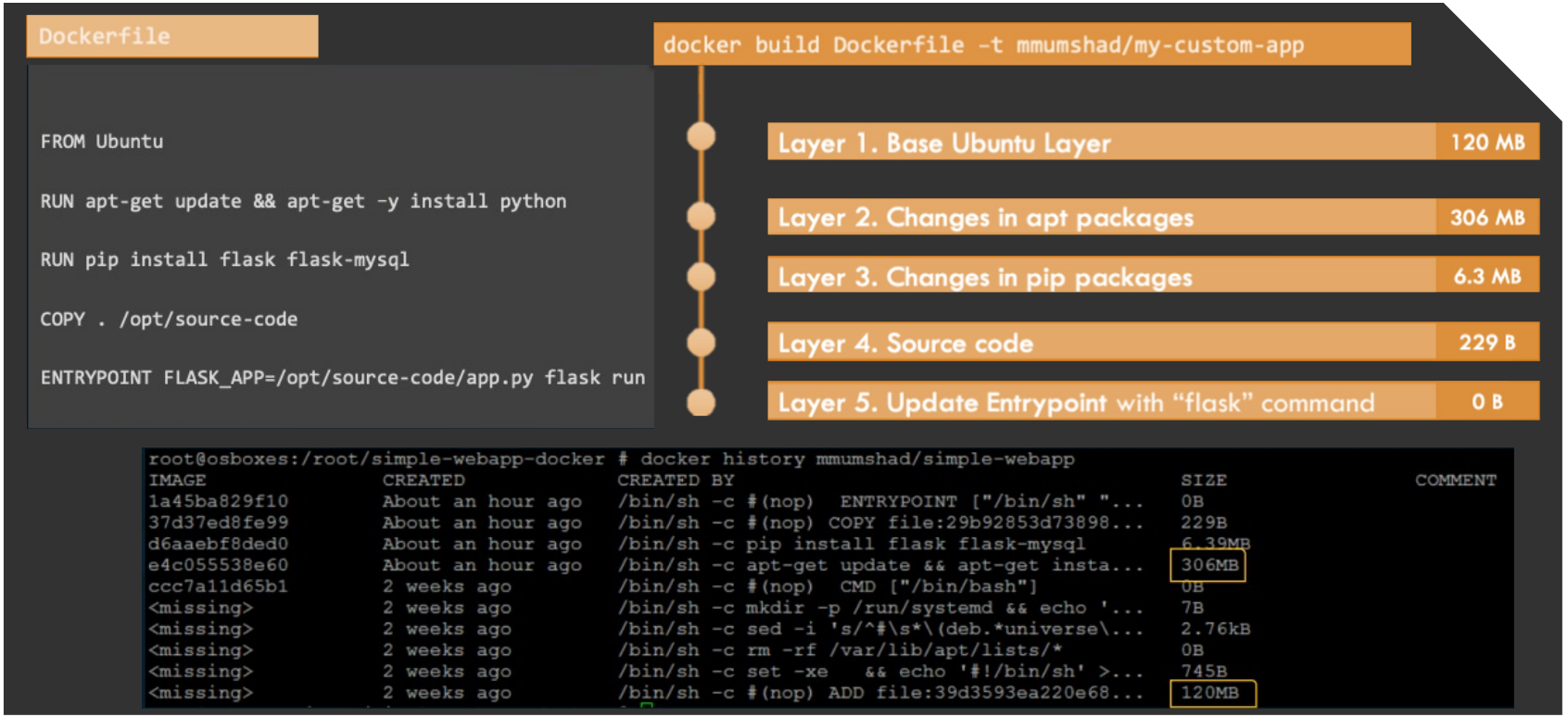
Install all dependencies

Copy source code

Specify Entrypoint



# Layered Architecture




Cada nueva instrucción crea una nueva capa solo con los cambios desde la capa anterior, todas las capas son almacenadas en cache.

# docker build

Crea una imagen a partir de un Dockerfile (el punto al final del comando es importante)

```
docker build .
```

## Opciones

--file, -f: Establece la ruta del archivo Dockerfile a usar (por defecto es 'PATH/Dockerfile')

```
docker build -f Dockerfile .
```

--no-cache: No usa la cache al construir la imagen

```
docker build --no-cache .
```

--tag, -t: Nombra y opcionalmente tagea la imagen con el formato 'nombre:tag'

```
docker build -t image:v2 .
```

# FROM

Inicializa una nueva etapa de construcción y establece la imagen base para las instrucciones posteriores.

Un Dockerfile válido debe comenzar con una instrucción FROM (excepto cuando empieza con un ARG).

La imagen base puede ser cualquier imagen válida.

```
FROM node:16-alpine

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

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

*https://docs.docker.com/engine/reference/builder/#from*

# WORKDIR

Establece el directorio de trabajo para cualquier instrucción RUN, COPY u otras posteriores.

Si el directorio no existe, se creará con toda la estructura definida.

Se puede usar tantas veces sea necesario en un Dockerfile.

Si se proporciona una ruta relativa, también será relativa a la instrucción WORKDIR.

```
FROM node:16-alpine

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

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

*https://docs.docker.com/engine/reference/builder/#workdir*

# COPY

Copia nuevos archivos o directorios desde un origen <src> y los agrega al sistema de archivos del contendor en una ruta destino <dest>.

Cada <src> puede o no contener comodines.

<dest> es una ruta absoluta, o una ruta relativa a WORKDIR.

- COPY file relativeDir/ → <WORKDIR>/relativeDir/file
- COPY file /absoluteDir/ → /absoluteDir/file

```
FROM node:16-alpine

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

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

*https://docs.docker.com/engine/reference/builder/#copy*

# RUN

Ejecuta cualquier comando en una nueva capa encima de la imagen actual, la imagen resultante se usará para el siguiente paso en el Dockerfile.

Existen 2 formas de utilizar esta instrucción:

- RUN <command> (preferida)
- RUN ["executable", "param1", "param2"]

```
FROM node:16-alpine

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

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

*https://docs.docker.com/engine/reference/builder/#run*

# EXPOSE

Informa a Docker que el contenedor escucha en los puertos de red especificados en tiempo de ejecución.

Se puede especificar si el puerto escucha en TCP o UDP, por defecto es TCP (3000/tcp).

Esta instrucción NO publica el puerto, simplemente funciona como un tipo de documentación para informar qué puertos pretende publicar el contenedor.

```
FROM node:16-alpine

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

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

*https://docs.docker.com/engine/reference/builder/#expose*

# CMD

Proporciona valores predeterminados para un contenedor en ejecución (como el ejecutable).

Existen 2 formas de utilizar esta instrucción:

- CMD [“executable”, “param1”, “param2”] (preferida)
- CMD command param1 param2

Se puede encontrar una tercera forma de esta instrucción (CMD [“param1”, “param2”]) cuando se pasan parámetros a ENTRYPOINT

```
FROM node:16-alpine

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm ci

COPY . .

EXPOSE 3000

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

*https://docs.docker.com/engine/reference/builder/#cmd*

# Express Dockerfile

Imagenes > API > Dockerfile > ...

You, hace 2 segundos | 1 author (You)

```
1  FROM node:16-alpine
2
3  WORKDIR /usr/src/app
4  COPY package*.json ./
5
6  RUN npm ci
7
8  COPY . .
9
10 EXPOSE 3000
11
12 CMD [ "node", "index.js" ]
13
```

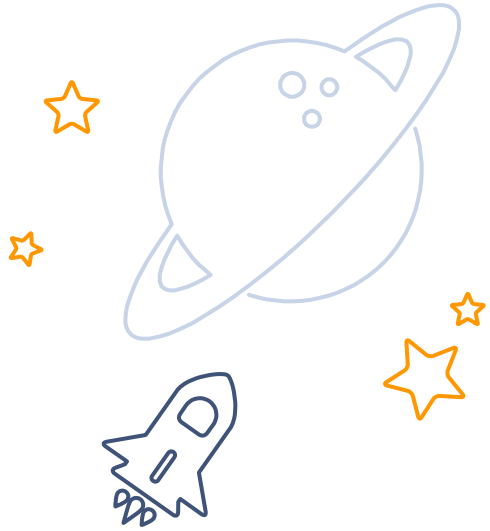
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL JUPYTER zsh - API

```
> \
> docker build -t express-api .
[+] Building 0.8s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 36B                                                                 0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 34B                                                                    0.0s
 => [internal] load metadata for docker.io/library/node:16-alpine                                   0.7s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 271B                                                                   0.0s
 => [1/5] FROM docker.io/library/node:16-alpine@sha256:f16544bc93cf1a36d213c8e2efecf682e9f4df28429a629a37aaf38ecfc25cf4  0.0s
 => CACHED [2/5] WORKDIR /usr/src/app                                                               0.0s
 => CACHED [3/5] COPY package*.json ./                                                              0.0s
 => CACHED [4/5] RUN npm ci                                                                         0.0s
 => CACHED [5/5] COPY . .                                                                           0.0s
 => exporting to image                                                                              0.0s
 => => exporting layers                                                                             0.0s
 => => writing image sha256:44ce0d5d28efed6e75f9bcca5c42fb451ca40aa0c07059f0ac6376ff434e29b7        0.0s
 => => naming to docker.io/library/express-api                                                      0.0s
```


https://github.com/devisra/taller-docker/tree/master/Imagenes/API

# Express Dockerfile

```
> docker images express-api
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
express-api   latest    44ce0d5d28ef   21 minutes ago   121MB
>
>
> docker run -d --name my_express -p 3001:3000 express-api
0c33a3b9fcd56ad47402f04e7d38870140bb54f848b025f3db60de3231dbc07c
>
>
> docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS
               NAMES
0c33a3b9fcd5   express-api   "docker-entrypoint.s…"   6 seconds ago   Up 5 seconds   0.0.0.0:3001
->3000/tcp   my_express
>
>
> curl localhost:3001
Hello World!%
>
>
> docker stop my_express
my_express
>
>
> curl localhost:3001
curl: (7) Failed to connect to localhost port 3001 after 7 ms: Connection refused
~ >
```

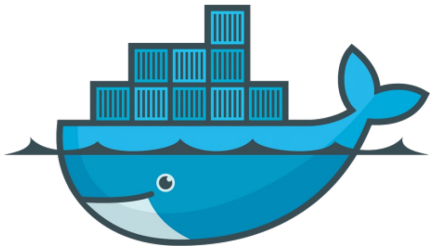



**docker run -d --name my_express –p 3001:3000 express-api**

# 5

## Registries

Docker Hub



# ¿Qué es un registry?

Un registry es donde todas las imágenes son almacenadas.

Docker Hub es un servicio de repositorio proporcionado por Docker y es el servicio por defecto al cual se conecta el demonio de Docker.

Existen otros servicios de repositorio como Azure Container Registry (ACR), Amazon Elastic Container Registry (ECR), Google Container Registry (GCR); cuando se necesita tener una solución interna de repositorio de imágenes docker es possible desplegar una version de Docker Registry.

# Image addresing

Cuando se ejecuta cualquier comando que deba descargar una imagen de un registry (pull, run, create, etc) por defecto se asume que el registry será Docker Hub (docker.io).

La siguiente es la convención para la dirección de las imágenes:

- Registry/User or Account/Image or Repository
  - httpd
  - docker.io/httpd/httpd

Cuando se necesita cargar una imagen a un registry (push) por defecto se asume que el registry será Docker Hub (docker.io) pero siempre será necesario nombrar la imagen con al menos el nombre de la cuenta y el repositorio

- devisra/custom_image

# docker login

Permite iniciar sesión con el registry de Docker Hub o del que se especifique su URL.

```
docker login docker.io
```

## Opciones

--password, -p: Establece la contraseña de la cuenta

```
docker login -p Pass123
```

--username, -u: Establece el usuario de la cuenta

```
docker build -u User
```

# docker image tag

Crea un nuevo tag para una imagen base, para ello es necesario indicar la imagen base <source_image>:<tag> y el tag deseado <target_image>:<tag>.

```
docker image tag actual_image:tag new_image:tag
```

# docker push

Permite cargar una imagen o un repositorio a un registry con el cual se haya realizado el login previamente.

```
docker push account/image:tag
```

## Opciones

--all-tags, -a: Carga todos los tags de la imagen indicada al repositorio

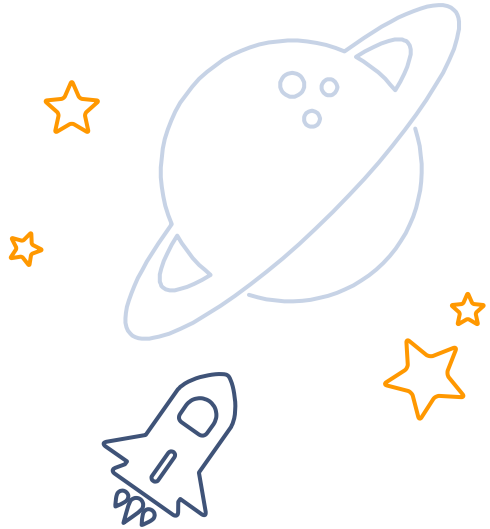
```
docker push -a account/image
```

--quiet, -q: No muestra el detalle de la carga en la consola

```
docker push -q account/image:tag
```

# Express Dockerfile

```
> docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID,
 head over to https://hub.docker.com to create one.
Username: devisra
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https:/
/docs.docker.com/go/access-tokens/
>
>
> docker push devisra/express-api:v1
The push refers to repository [docker.io/devisra/express-api]
9a01ceb0a097: Pushed
23af104f94b3: Pushed
2f209b96d3f3: Pushed
9dfcf6b89930: Pushed
a79681b2645a: Mounted from library/node
f506bc026271: Mounted from library/node
11816ca779c7: Mounted from library/node
994393dc58e7: Mounted from library/node
v1: digest: sha256:bef645744ab4fd7df06c4cd4acb8da90ae8007482619081e3e0785a55367cd93 size: 1992
```




https://hub.docker.com/r/devisra/express-api