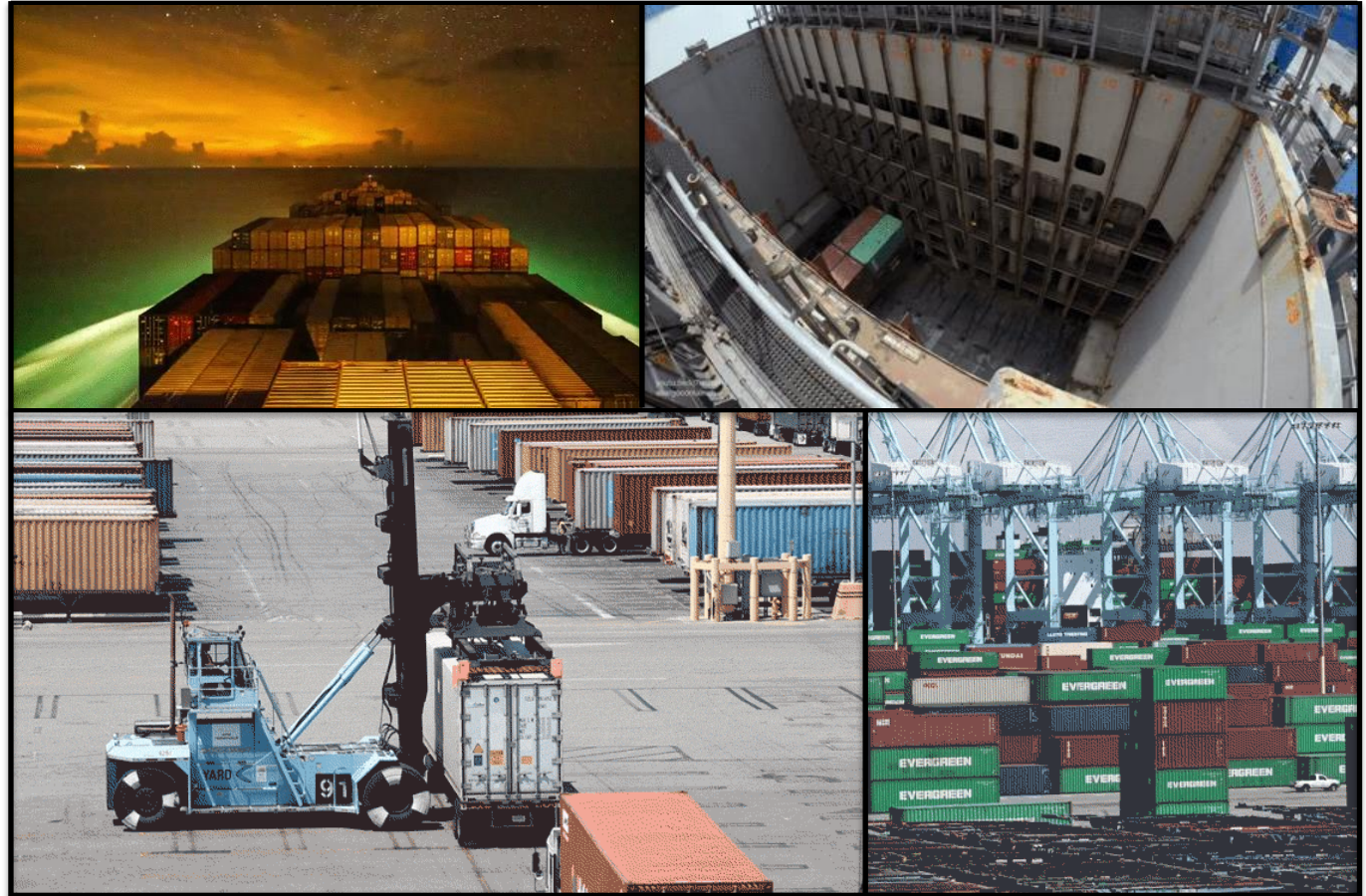




Docker

1. Introducción
2. ¿Qué es Docker?
3. Componentes Docker
4. Creación de un Dockerfile
5. Interacción con la CLI
6. Demo



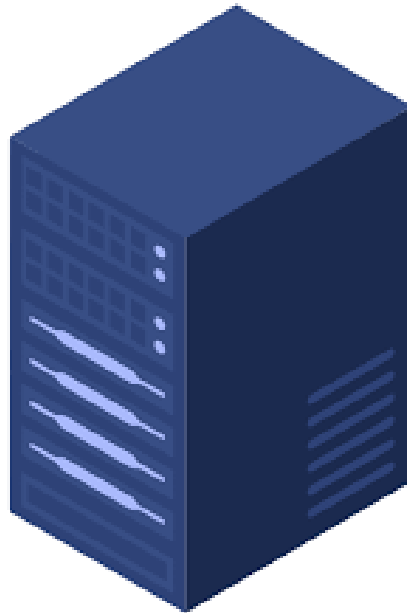
Introducción

Introducción

Despliegue en servidor físico

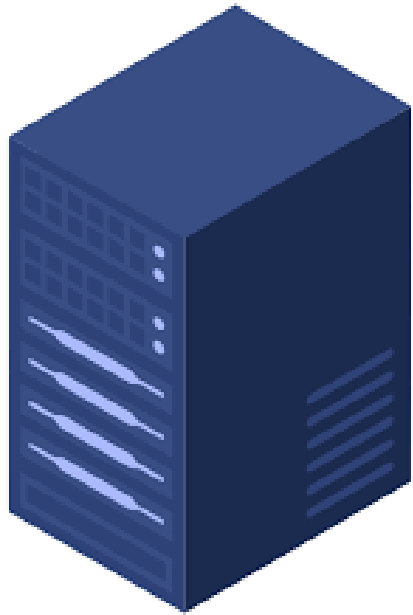


Apache Tomcat

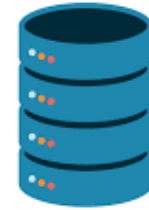


Introducción

Despliegue en máquina virtual



Apache Tomcat

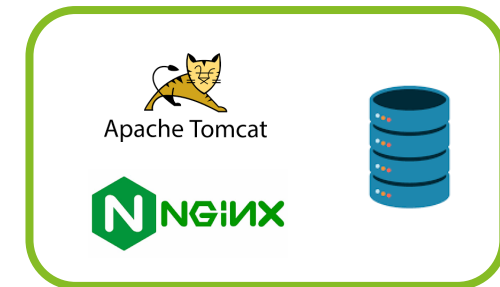
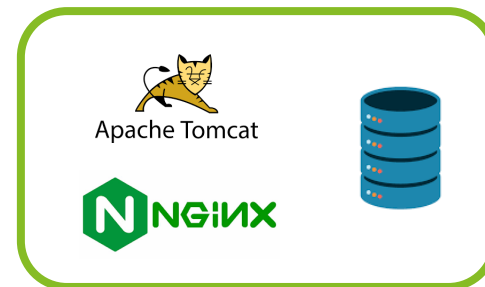
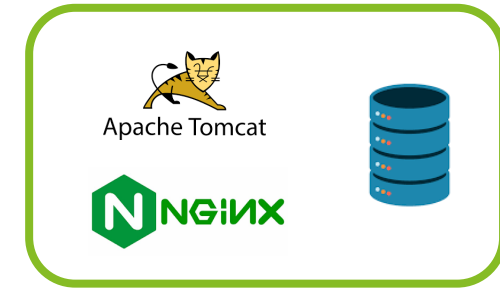
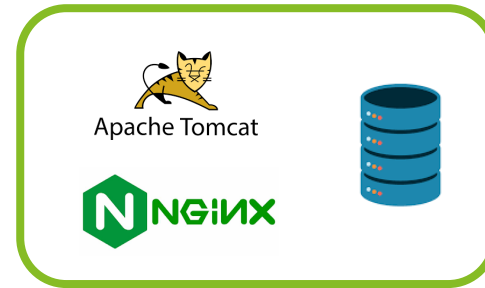
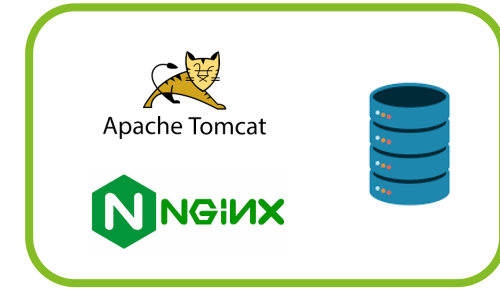
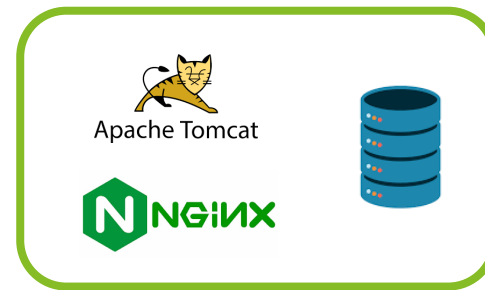
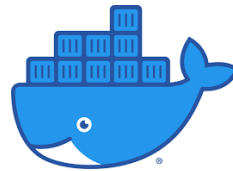
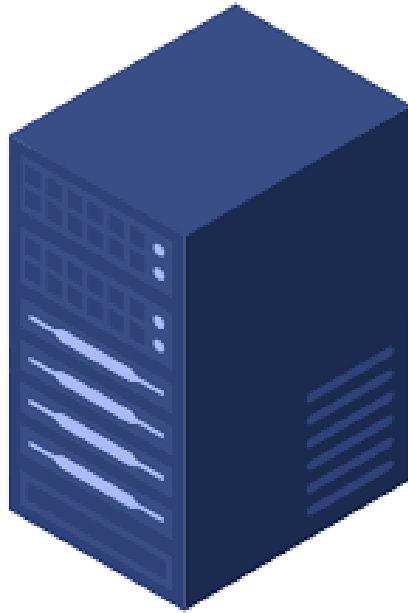


Apache Tomcat



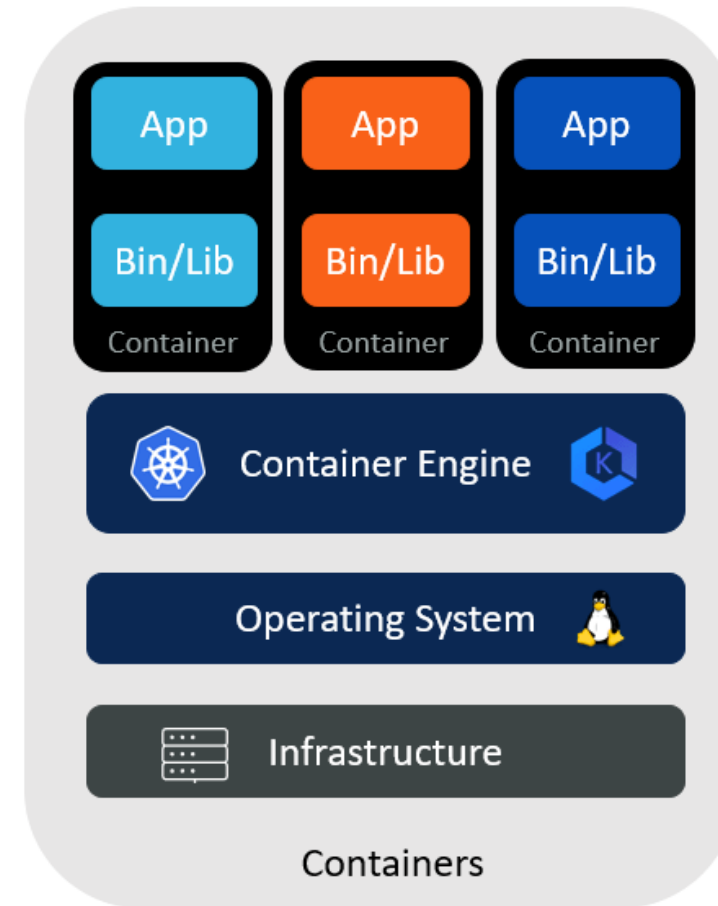
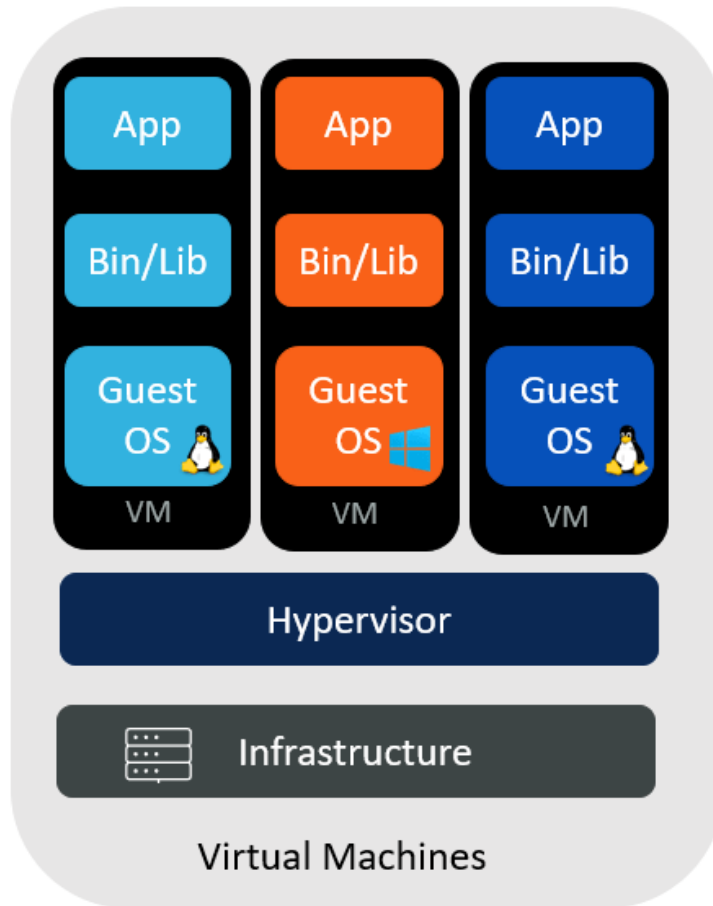
Introducción

Despliegue contenedor



Introducción

MVs vs Contenedores



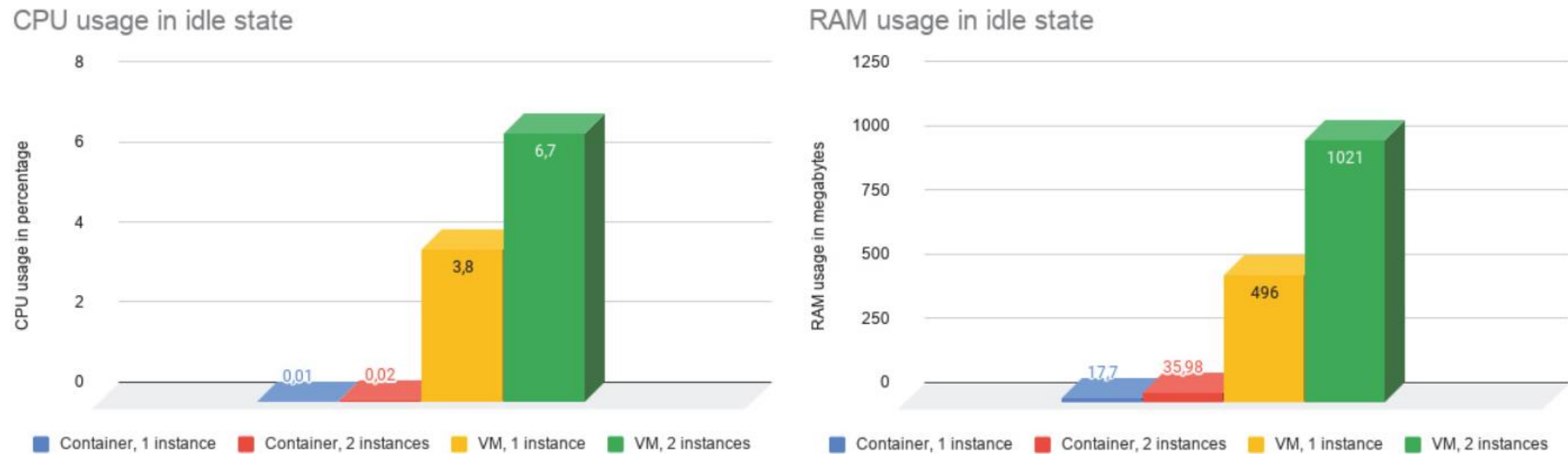


Figure 4.1: Graph overview for CPU usage in percentage, in idle state.

¿Qué es Docker?

¿Qué es Docker?

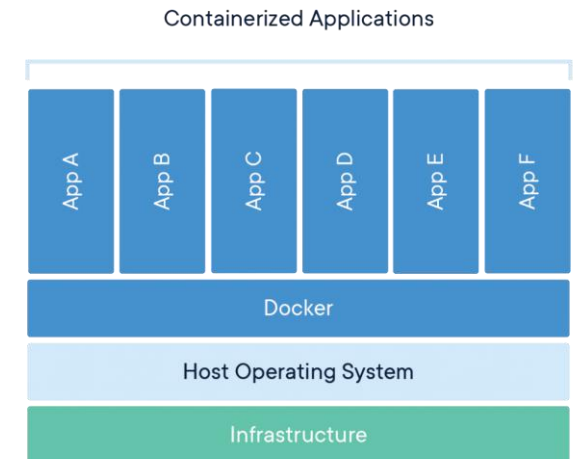
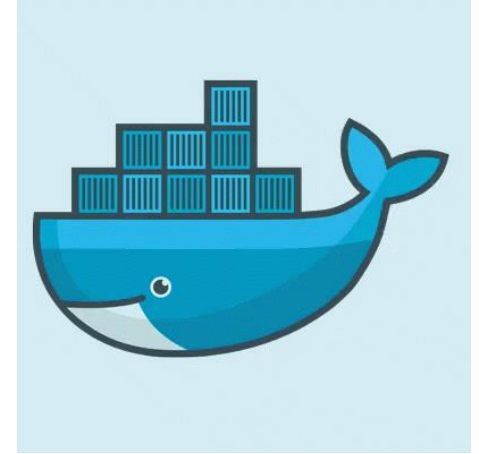
Características

Definición

- Plataforma código abierto.
- Empaquetar y desplegar aplicaciones en contenedores.

Características

- Portable
- Ligero
- Autosuficiente
- Seguro



Componentes de Docker

Componentes principales

Imagen

- Plantilla de solo lectura que define un contenedor.

Dockerfile

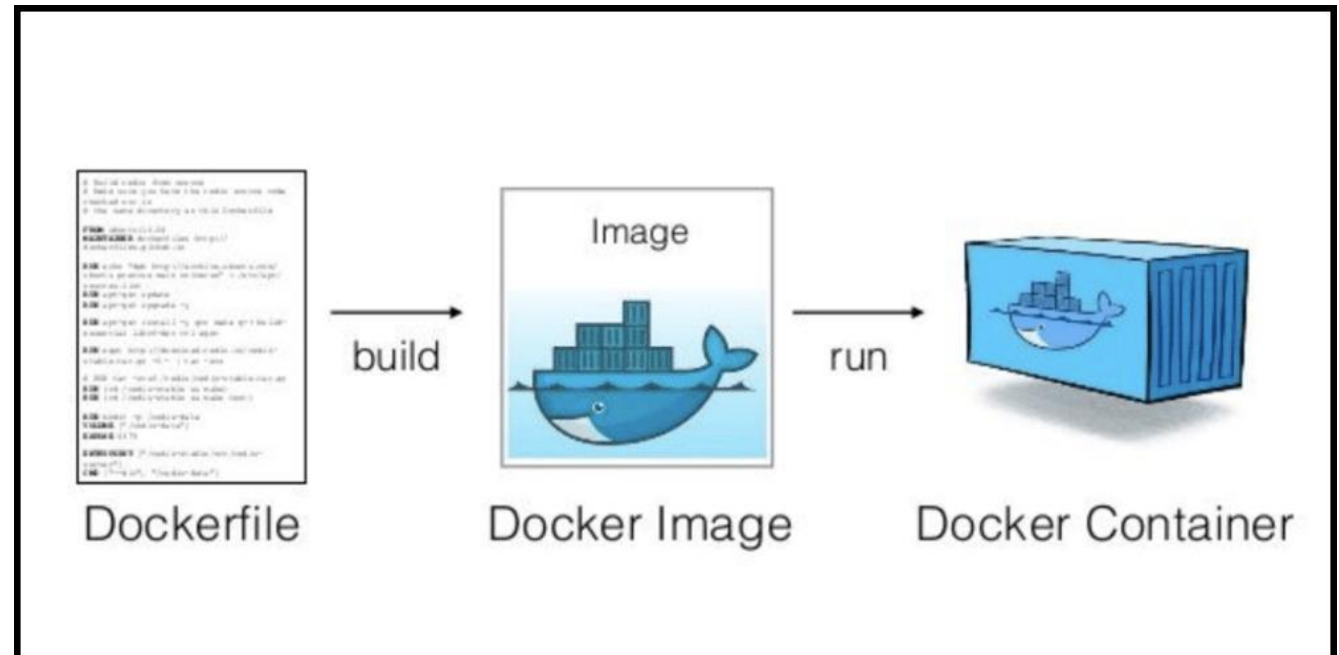
- Fichero para construir la plantilla.

Contenedores

- Instancias de imágenes

Registry

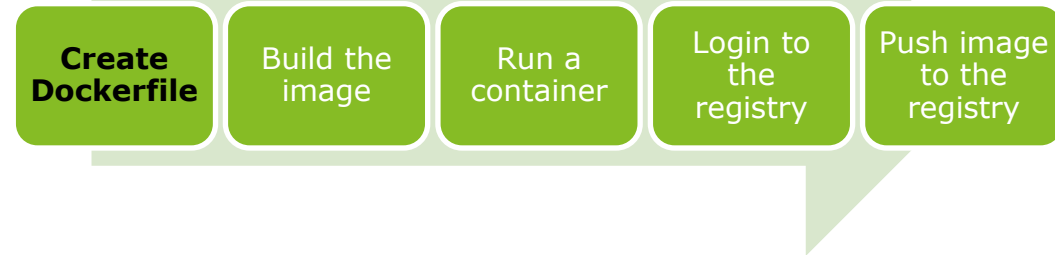
- Repositorio de imágenes



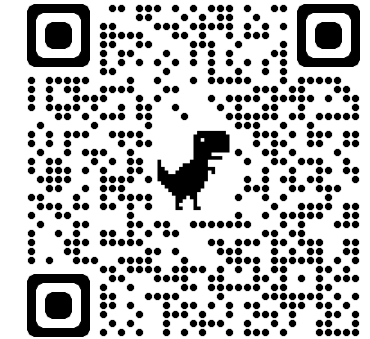
Creación de un Dockerfile

Creación de un Dockerfile

Instrucciones



- **FROM** image – Imagen base.
- **RUN** command – Comando durante la construcción.
- **CMD** command – Comando durante la ejecución
- **ARG** key[=value] – Variables usadas durante la construcción.
- **ENV** key=value – Variables usadas durante la ejecución
- **EXPOSE** port – Puertos que son accesibles desde fuera del contenedor.
- **VOLUME** path – Los volúmenes que va a usar la aplicación.
- **COPY** src dest – Copiar ficheros de interés al sistema de archivos del contenedor.



Más información

Creación de un Dockerfile

Ejemplo

```
FROM debian:stable-slim
```

```
RUN echo 'Hola Mundo desde RUN'
```

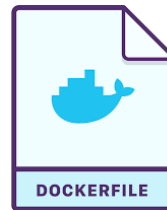
```
ARG arg1
```

```
RUN echo ${arg1}
```

```
ENV env1="Hola Mundo desde ENV"
```

```
COPY file.txt tmp/file.txt
```

```
CMD [ "sleep", "3600" ]
```



```
FROM nginx:latest
```

```
RUN apt-get update;
```

```
RUN apt-get install -y vim;
```

```
COPY ./static /usr/share/nginx/html
```

```
VOLUME "nginx-vol:/usr/share/nginx/html"
```

```
FROM openjdk:11
```

```
ARG JAR_FILE
```

```
COPY ${JAR_FILE} hero.jar
```

```
CMD ["java", "-jar", "/hero.jar"]
```

Interacción con la CLI

Interacción con la CLI

Comandos iniciales

- `docker version` - Versión de docker
- `docker info` - Contenedores en ejecución, parados...
- `docker help` - Todas las opciones del commando.

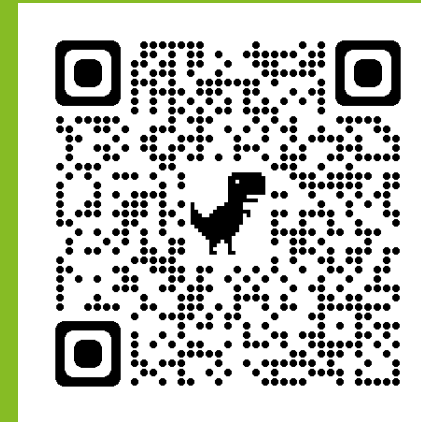
```
dxd@dxd: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
dxd@dxd:~$ docker version  
Client:  
Version:      18.09.6  
API version:  1.39  
Go version:   go1.10.8  
Git commit:   481bc77  
Built:        Sat May  4 02:35:57 2019  
OS/Arch:      linux/amd64  
Experimental: false  
  
Server: Docker Engine - Community  
Engine:  
Version:      18.09.6  
API version:  1.39 (minimum version 1.12)  
Go version:   go1.10.8  
Git commit:   481bc77  
Built:        Sat May  4 01:59:36 2019  
OS/Arch:      linux/amd64  
Experimental: false
```

```
dxd@dxd: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
dxd@dxd:~$ docker info  
Containers: 3  
  Running: 1  
  Paused: 0  
  Stopped: 2  
Images: 2  
Server Version: 18.09.6  
Storage Driver: overlay2  
  Backing Filesystem: extfs  
  Supports d_type: true  
  Native Overlay Diff: true  
Logging Driver: json-file  
Cgroup Driver: cgroupfs  
Plugins:  
  Volume: local  
  Network: bridge host macvlan null overlay  
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog  
Swarm: inactive  
Runtimes: runc  
Default Runtime: runc  
Init Binary: docker-init  
containerd version: bb71b10fd8f58240ca47fbb579b9d1028eea7c84  
runc version: 2b18fe1d885ee5083ef9f0838fee39b62d653e30  
init version: fec3683  
Security Options:  
  apparmor  
  seccomp  
    Profile: default  
Kernel Version: 4.18.0-25-generic  
Operating System: Ubuntu 18.04.2 LTS
```

Interacción con la CLI

Listado de comandos

- **docker image ls:** Listar imagenes.
- **docker container ls:** Listar contenedores.
- **docker build:** Construir una usando un Dockerfile.
- **docker run:** Crear un nuevo contenedor.
- **docker start:** Arrancar un contenedor existente.
- **docker stop:** Parar un contenedor existente.
- **docker pull:** Descargarse una imagen de un repositorio.
- **docker login:** Iniciar sesión en un repositorio.
- **docker push:** Subir una imagen a un repositorio.
- **docker exec:** Ejecutar commando en un contenedor.
- **docker logs:** Ver los logs de un contenedor.
- **docker rm:** Eliminar contenedor.
- **docker rmi:** Eliminar imagen.
- **docker volume:** Gestionar volúmenes.



Más información

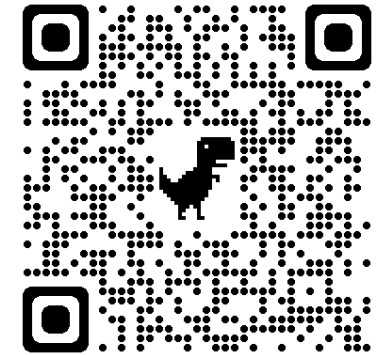
Interacción con la CLI

Build

```
docker build [OPTIONS] PATH | URL
```

- Opciones:

- **-t, --tag:** Nombre de la imagen y opcionalmente etiqueta.
- **--build-arg:** Asignar valor a los argumentos.
- **-m, --memory:** Limitar memoria.
- **--cpu-quota:** Limitar CPU.
- **--label:** Añadir metadatos.
- **--no-cache:** No usar imagenes de nuestro repositorio local.
- **--pull:** Busca si existen nuevas versiones de la imagen.



Más información

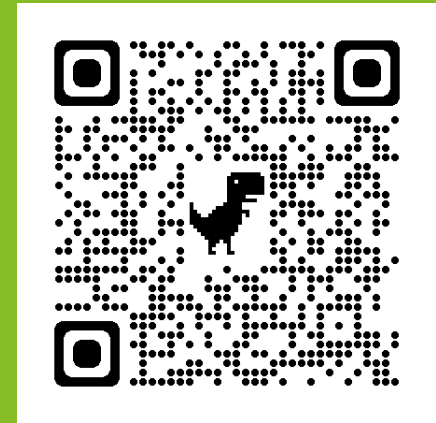
Interacción con la CLI

Run

```
docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]
```

- Opciones:

- **--name:** Nombre del contenedor.
- **-d, --detach:** Ejecutar el contenedor en segundo plano.
- **-e, --env:** Crear variables de entorno.
- **-u, --user:** Asignar usuario de ejecución del contenedor.
- **-v, --volume:** Asignar un volumen al contenedor.
- **--label:** Añadir metadatos.
- **--rm:** Eliminar el contenedor al pararse.
- **-p, --publish:** Publicar puertos del contenedor al host.
- **-it:** Proceso interactivo. Shell.



Más información

Demo

Demo

Contenido

Ejemplo básico Dockerfile



NGINX



NGINX con volumen



Spring



¿Preguntas?

¡Muchas gracias!



acanton@deloitte.es



Adrián Cantón Fernández



acastillocotan@deloitte.es



Antonio Jesús Castillo Cotán