Universidad de Oviedo

School of Computer Science

EN
English

SOFTWARE ARCHITECTURE

Arquitectura del Software

Lab. 06
Despliegue y distribución

2020-21

Jose Emilio Labra Gayo
Pablo González
Irene Cid
Paulino Álvarez

GitHub Pages

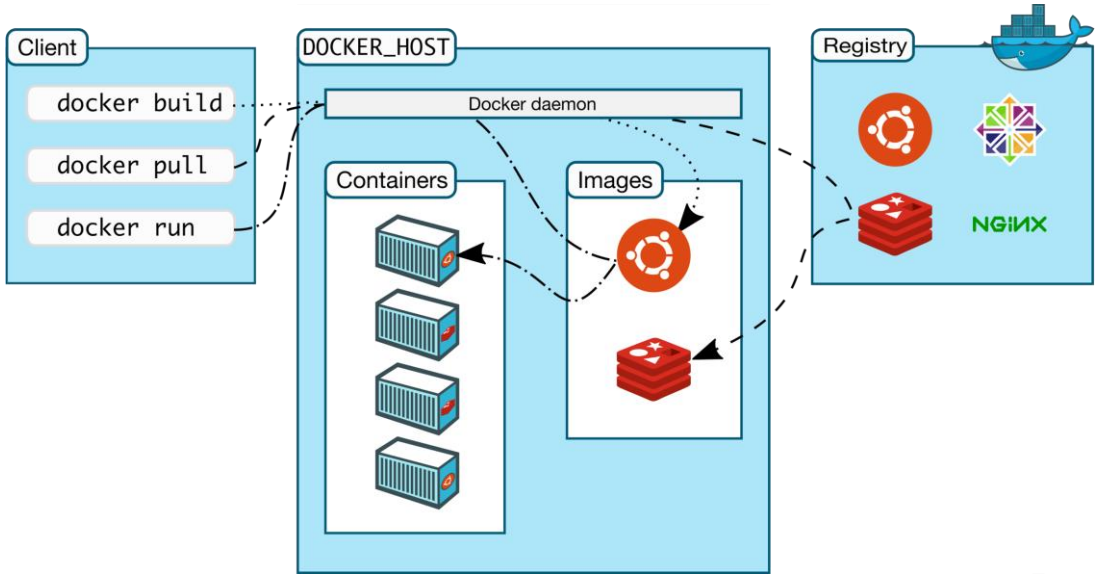
- GitHub soporta la creación de sitios web con contenido estático a través de gh-pages
- Útil para crear página personal o de proyecto
- Por defecto, se publica lo que está en la rama **gh-pages**

GitHub Pages - Ejemplos

- Ejemplo de sitio web de organización
  - Repositorio: https://github.com/Arquisoft/Arquisoft.github.io
  - Despliegue: https://arquisoft.github.io/
- Ejemplo de uso de sitio personal
  - Repositorio: https://github.com/pglez82/pglez82.github.io
  - Despliegue: http://pglez82.github.io

¿Qué es Docker?

- Plataforma para desarrolladores y administradores de sistemas.
- Basado en contenedores
- Flexible, ligero, portátil, escalable…



Fuente:https://docs.docker.com/get-started/overview/

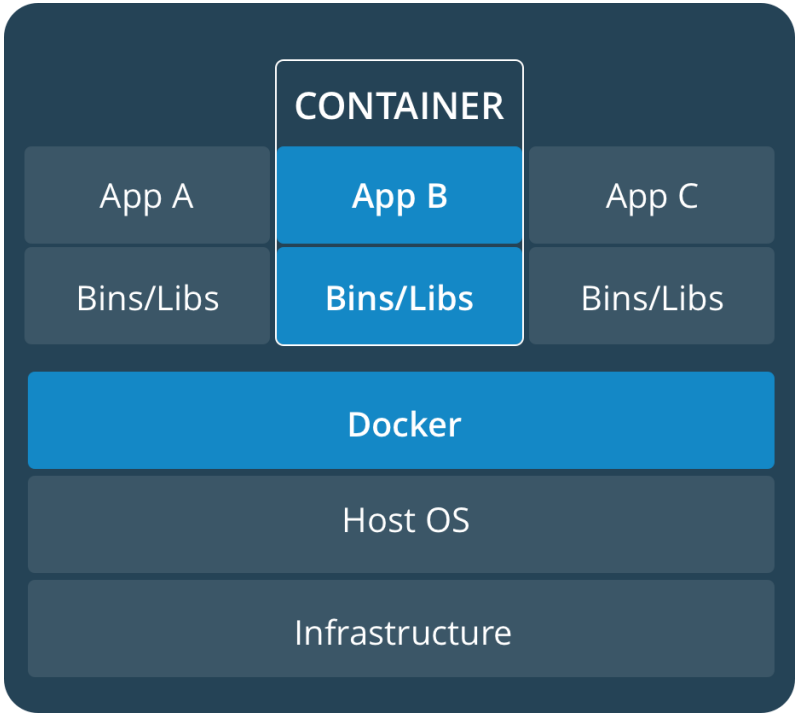
Imágenes en Docker

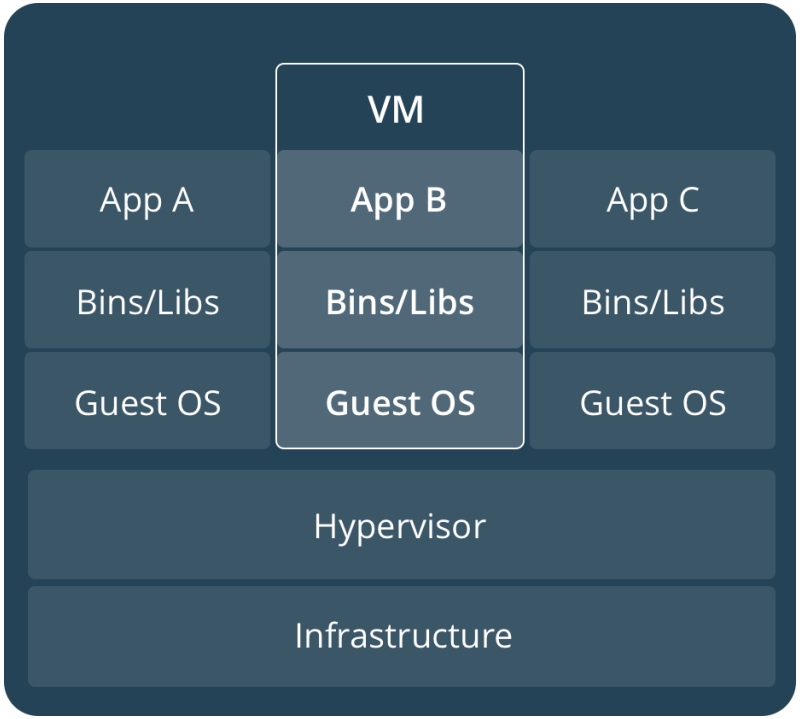
- **Definición**: plantilla de solo lectura que sirve de base para crear un contenedor.
- Incluye todo lo necesario para ejecutar una aplicación:
  - Código
  - Sistema Ejecutable
  - Librerías
  - Variables de ejecución
  - Ficheros de configuración
- No tiene estado y nunca cambia

¿ Qué es un contenedor?

- Es una instancia de una imagen
- Docker está basado en contenedores que encierran aplicaciones
- Docker permite la organización entre contenedores
- Enlazando varios contenedores podemos simular arquitecturas complejas.

Es una máquina virtual?

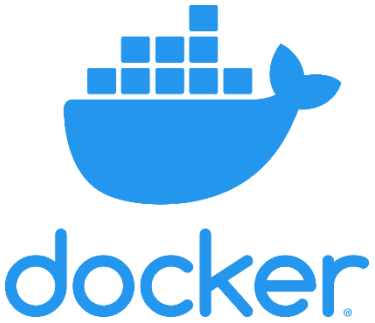




Fuente: https://docs.docker.com/get-started/#containers-and-virtual-machines
https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine

Descargar docker

- URL : https://www.docker.com
- Disponible para Linux, Windows and Mac
- Versión Docker Desktop para (Windows/Mac)
- En caso de incompatibilidad en Windows véase faq#issue3



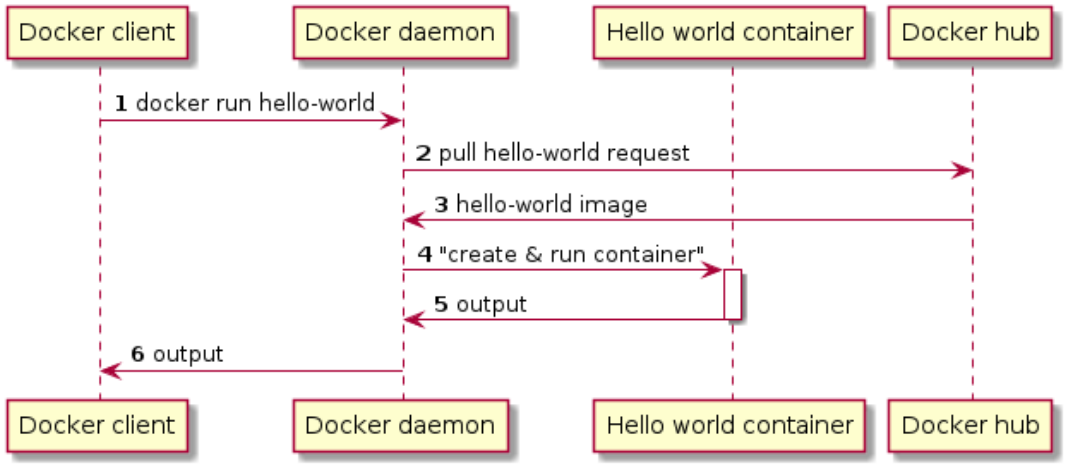
Docker Hub

- Repositorio de imágenes: https://hub.docker.com/
- Permite ejecutar servicios y aplicaciones de una manera rápida y escalable
- Existen imágenes probadas de servicios conocidos
- Ejemplo: Necesidad de un servidor http
  - `docker pull nginx`
  - `docker pull httpd`

Docker paso a paso

- Comprobamos que está instalado `$ docker -v`
- Ejejutamos “Hello World”

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest:
sha256:f9dfddf63636d84ef479d645ab5885156ae030f...
Status: Downloaded newer image for hello-world:latest
```



Ejemplo ejecución Ubuntu

```
$ docker container run -it ubuntu:latest /bin/bash
. . .
root@813cb77cebb2:/# ls -la
total 72
drwxr-xr-x   1 root root 4096 Mar 30 05:46 .
drwxr-xr-x   1 root root 4096 Mar 30 05:46 ..
-rwxr-xr-x   1 root root    0 Mar 30 05:46 .dockerenv
drwxr-xr-x   2 root root 4096 Mar 11 21:05 bin
drwxr-xr-x   2 root root 4096 Apr 24  2018 boot
drwxr-xr-x   5 root root  360 Mar 30 05:47 dev
drwxr-xr-x   1 root root 4096 Mar 30 05:46 etc
. . .
drwxr-xr-x   1 root root 4096 Mar 11 21:03 usr
drwxr-xr-x   1 root root 4096 Mar 11 21:05 var
root@813cb77cebb2:/#
```

Ejemplo comprobar estado

- Commandos para comprobar el estado

```
λ docker image ls
REPOSITORY    TAG     IMAGE ID       CREATED         SIZE
hello-world   latest  fce289e99eb9   14 months ago   1.84kB

λ docker container ls --all
CONTAINER ID  IMAGE       COMMAND  CREATED       STATUS
8b6518da11db  hello-world "/hello" 9 minutes ago Exited (0) 9 minutes ago
```

https://github.com/pglez82/docker_cheatsheet

Ejemplo ejecución Servidor Web

- Vamos a ejecutar un servidor web docker

Ejecución segundo plano

publish:puerto contenedor:puerto host.

```
$ docker run --detach --publish=80:80 --name=webserver nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
68ced04f60ab: Pull complete
28252775b295: Pull complete
a616aa3b0bf2: Pull complete
Digest: sha256:2539d4344dd18e1df02be842ffc435f8e1f699cfc55516e2cf2cb16b7a9aea0b
Status: Downloaded newer image for nginx:latest
b7e9213eb3367cd465b29701a7e6441a7210a46d439196d30e76ddc9c72ee280
```

Otros comandos docker

- `docker info`
- `docker ps`
- `docker image ls`
- `docker container ls –all`
- `docker pull`
- `docker run`
- `docker stop`
- `docker rm`

Cómo construir una imagen

- Uso de DSL (Domain Specific Language) para crear imágenes
- Creamos un fichero llamado **Dockerfile**
- Contiene commandos necesarios para construer una imagen:

Keywords: FROM, RUN, ADD, COPY, ENV, EXPOSE, CMD…

Dockerfile

```
FROM ubuntu
CMD echo "Hi Software architecture students"
```

Construyendo una imagen

1. Creamos una carpeta
2. Creamos nuestro fichero Dockerfile (no extension)

```
FROM ubuntu
CMD echo "Hi ASW students"
```

3. Ejecutamos: docker build –t <<image_name>>

```
λ docker build -t "example1" .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM ubuntu
latest: Pulling from library/ubuntu
5bed26d33875: Pull complete
.....Successfully built 41784c740df4
Successfully tagged example1:latest
```

4. Listamos las imagenes : docker images

```
λ docker images
REPOSITORY TAG    IMAGE ID  CREATED        SIZE
example1   latest 41784c740 32 seconds ago 64.2MB
```

5. Arrancamos un contenedor con nuestra imagen: run <<image_name>>

```
λ docker run example1
Hi ASW students
```

Ejemplo1: Solid Server en Local

1. Ejecutamos un pull para traernos la imagen (repositorio de la imagen https://hub.docker.com/r/nodesolidserver/node-solid-server)

```
$ docker pull nodesolidserver/node-solid-server
```

2. Ejecutamos la imagen en el puerto 8443

```
$ docker run -p 8443:8443 --name solid nodesolidserver/node-solid-server
```

3. Veremos la app en http://localhost:8443

Ejemplo 2: Radarin webapp

Radarin webapp DockerFile

https://github.com/Arquisoft/radarin_0/tree/master/webapp

```
FROM node:12.14.1
COPY . /app
WORKDIR /app
#Install the dependencies
RUN npm install --production
#Create an enviroment variable to set where the api is (check src/api/api.js).
#When we deploy to heroku this will take a different value. Check .github/workflows/radarin.yml
ARG API_URI="http://localhost:5000/api"
ENV REACT_APP_API_URI=$API_URI
#Create an optimized version of the webapp
RUN npm run build
#Install software neccesary for generating the doc
RUN apt-get update && apt-get -y install ruby openjdk-8-jre
RUN gem install asciidoctor asciidoctor-diagram
#Generate the doc
RUN npm run docs
CMD [ "node", "server.js" ]
```

Ejemplo 3: Radarin restapi

Radarin restapi DockerFile

https://github.com/Arquisoft/radarin_0/tree/master/restapi

```
FROM node:12.14.1

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied
COPY package*.json ./

#In this case, the mongodb in memory dependency is quite heavy so we avoid it for the docker image
RUN npm install --production

#By default this is the address of the mongo container. If we are deploying to heroku we will get
#a mongo_uri here direct from github secrets (check .github/worflow/radarin.yml)
ARG MONGO_URI="mongodb://mongoserver:27017/api"
ENV MONGO_URI=$MONGO_URI

# Bundle app source
COPY server.js api.js ./
ADD models ./models

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

Combinando contenedores

- **Docker compose** permite el trabajo con módulos, creando arquitecturas modulares
- Se pueden definer distintos servicios que comunican entre ellos
- Cada servicio se ejecuta en su contenedor
- Fichero de configuración:

  docker-compose.yml
- Radarin docker-compose

Docker Compose

- Configuration a través del fichero
  - En el campo servicio tendremos una entrada por cada servicio que queramos ejecutar
  - Puede haber dependencia entre los servicios
  - Por defecto, todos los servicios comparten la misma red y se pueden acceder por el nombre del servicio
- Ejecución
  - Para ejecutar o parar la configuración indicada en el fichero solo temenos que llamar a

```
docker-compose (up|down)
```

Docker Compose

- Rebuild
  - Una vez se han generado los contenedores a partir de una imagen no vuelven a ser reconstruidos hasta que se detecte un cambio. Se puede forzar con

```
docker-compose up --build --force-recreate
```

Más Información

1. Pequeño repositorio con todos los comandos que se pueden usar en docker:
   https://github.com/pglez82/docker_cheatsheet
2. Documentación oficial:
   https://docs.docker.com/compose/gettingstarted/