Tema 6 : Docker

Ciclo Superior DAW

Asignatura: Despliegue de aplicaciones web

Curso 20/21

Introducción

- En este capítulo veremos aspectos como:
  - Introducción a Docker
  - Contenedores de Docker
  - Imágenes de Docker
  - Ejemplos de trabajo con imágenes y contenedores

Conceptos de Docker

- Los **contenedores** son una tecnología que ofrece unas ventajas similares a las VMs pero aprovechando mejor los recursos:
  - Los contenedores tardan milisegundos en arrancar
  - Consumen únicamente la memoria que necesita la app ejecutada en el contenedor. Una VM reserva la memoria completa

Conceptos de Docker

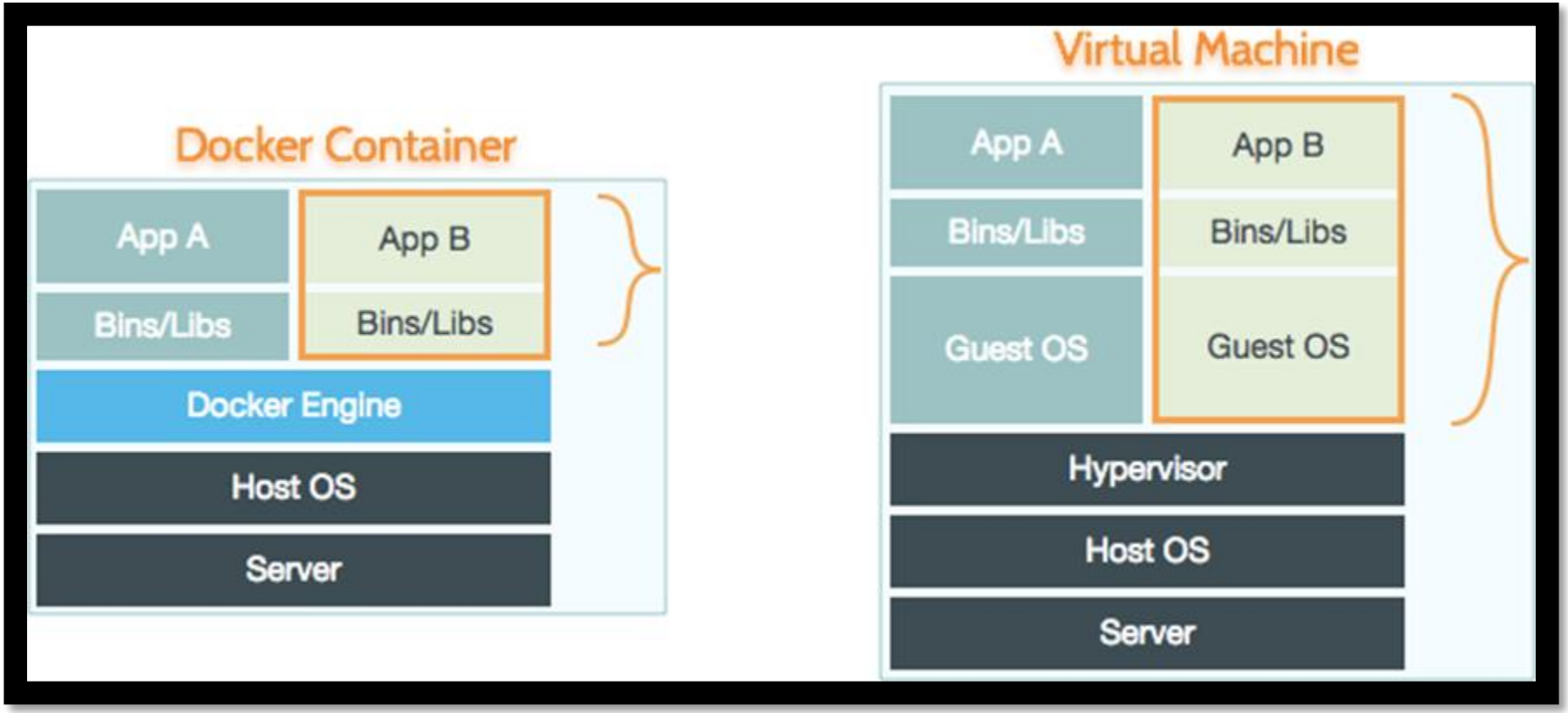
- Es la tecnología de contenedores más popular, (aunque sólo tiene 4 años y medio)
- Es para linux, aunque dispone de herramientas para desarrolladores en windows y mac
- Existe un repositorio de imágenes (hub) con contenedores públicos:

https://www.docker.com/

Conceptos de Docker

- **¿Por qué son tan eficientes los contenedores?**
  - Para ejecutar un contenedor no se necesita hypervisor porque no se ejecuta un sistema operativo invitado y no hay que simular HW
  - Un contenedor es un paquete que contiene una app y todo el sw necesario para que se ejecute (python, Java, gcc, libs….)
  - El contenedor es ejecutado directamente por el kernel del host como si fuera una app más pero de forma aislada del resto

Conceptos de Docker



Conceptos de Docker

- **Imagen de Docker**
  - **Plantilla básica para un contenedor** (principalmente, el contenido del disco)
  - Puede contener un SO (ubuntu), alguna librería (Java) y alguna aplicación (webapp.jar)
  - El contenedor siempre es iniciado desde una imagen
  - Descargar una imagen docker desde Internet es muy sencillo

Conceptos de Docker

- **Registro de Docker**
  - Es como un repositorio (ej. Github)
  - **Servicio remoto que almacena y devuelve imágenes de docker**
  - Puede guardar varias versiones de la misma imagen
  - Todas las versiones de la misma imagen están almacenadas en repositorios
  - **Docker Hub** es un registro público manejado por Docker Inc.
  - Se pueden comprar repositorios privados en Docker Hub

Conceptos de Docker

- **Repositorios populares de Docker**

Conceptos de Docker

- **Contenedor Docker**
  - Es el equivalente de una máquina virtual, aunque la tecnología es totalmente diferente
  - **Un contenedor es creado desde una imagen de Docker**
  - Cuando se hace un cambio, **la imagen no se modifica, lo que cambia es el contenedor**
  - Puede ser **iniciado, pausado o parado**

Conceptos de Docker

- **Docker Engine**
  - **Es el servicio interno de Docker**
  - Manejar imágenes (descargar, crear, hacer pull, push...)
  - Manejar contenedores (arrancarlos, pararlos, hacer commit...)
  - Se puede usar desde el cliente Docker o usando su propia REST API

Conceptos de Docker

- **Cliente Docker**
  - Línea de comandos que nos permite controlar toda la magia de Docker Engine
  - También existen interfaces gráficas

Tutoriales

- **Tutorial para principiantes**
  - https://github.com/docker/labs/tree/master/beginner
- **Tutorial de instalación de Docker**
  - https://docs.docker.com/engine/installation/

Primeros pasos

- **Paso 1: Instalar Docker**
    - **En Windows o Mac**, utilizaremos el ejecutable “Docker Desktop Installer”
    - **En Linux**, lo instalaremos mediante el repositorio
- Nosotros utilizaremos una VM con el Docker ya instalado

Actividad 1

Lanzamos la VM con Docker instalado

Primeros pasos

- **Paso 2: Conectarse a Docker Hub**
  - Necesitaremos crearnos una cuenta en Docker Hub: https://hub.docker.com/signup
  - Configuramos nuestra cuenta:

```
docker login
```

Primeros pasos

- **Paso 2: Conectarse a Docker Hub**
  - Probaremos a ejecutar:

```
docker run hello-world
```

  - Esto nos descargará un contenedor de ejemplo, el típico “Hello World!”.

Primeros pasos

- **Paso 2: Conectarse a Docker Hub**
  - Tras esto, si usamos el comando:

```
docker images
```

  - Comprobaremos que tenemos la imagen “hello-world” descargada.

Actividad 2

Nos conectamos a Docker Hub

Descargamos una imagen y comprobamos que se ha descargado correctamente

Primeros pasos

- **Paso 3: Descargar imágenes Docker**
  - **Los contenedores Docker se forman a partir de imágenes de Docker.** De forma predeterminada, Docker extrae estas imágenes de Docker Hub.
  - Podremos buscar imágenes disponibles en Docker Hub usando el siguiente comando:

```
docker search [IMAGEN]
```

Primeros pasos

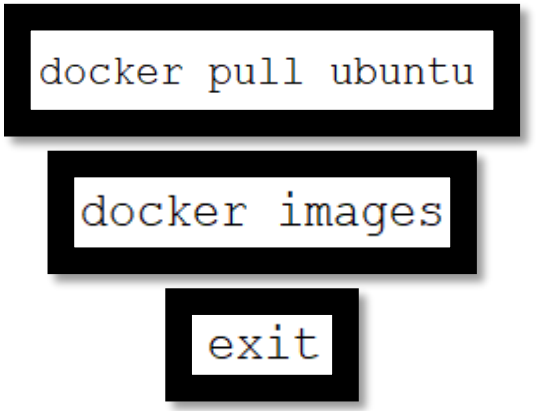
- **Paso 3: Descargar imágenes Docker**
  - **Por ejemplo, para buscar la imagen de Ubuntu, pondremos:**

```
docker search ubuntu
```

  - Nos entregará una lista de todas las imágenes que tengan un nombre que concuerde con la cadena de búsqueda.

Primeros pasos

- **Paso 3: Descargar imágenes Docker**
  - **Para descargar una imagen al ordenador**, ejecutaremos:
  - **Para ver las imágenes que se descargaron**, ejecutaremos:
  - **Para salir y PARAR un contenedor**, ejecutaremos:



Actividad 3

Probaremos a descargarnos algunas imágenes

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker**
  - El contenedor hello-world que vimos antes es un ejemplo de un contenedor que se ejecuta y se va tras emitir un mensaje de prueba.
  - Los contenedores pueden ser mucho más útiles que eso, y pueden ser interactivos.
  - A continuación, veremos un ejemplo de acceso a un contenedor mucho más interactivo

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker**
  - Por ejemplo, ejecutemos un contenedor utilizando la última imagen de Ubuntu.
  - La combinación de los flags -i y -t ofrece acceso interactivo a shell en el contenedor:

```
docker run -it ubuntu
```

Actividad 4

Probemos a lanzar el contenedor Ubuntu de forma interactiva

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker**
  - Probaremos un ejemplo más completo y complejo:
    - Crear un contenedor con el siguiente comando:

```
docker run -it -p 80:80 --name ubuntu_apache ubuntu
```

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker. Ejemplo completo**
  - Crearemos un contenedor con el siguiente comando:

```
docker run -it -p 80:80 --name ubuntu_apache ubuntu
```

  - El **parámetro -p 80:80** redirige el puerto 80 de la máquina al puerto 80 del contenedor.
  - El **parámetro --name ubuntu_apache** pone ese nombre al contenedor

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker. Ejemplo completo**
  - Instalamos Apache2 y nos aseguramos que el servicio está creado:

```
apt-get update
apt-get install apache2
```

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker. Ejemplo completo**
    - Salimos del contenedor, pero queremos que siga funcionando, por lo que salimos pulsando **CTRL-P y CTRL-Q.**

Actividad 4

Entramos con el navegador en localhost y veremos la página de Apache que se ejecuta realmente en el contenedor

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker. Ejemplo completo**
  - Volvemos a entrar en el contenedor:

```
docker attach ubuntu_apache
```

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker. Ejemplo completo**
  - Salimos con exit con lo que provocamos que el contenedor se pare
  - Entramos de nuevo en localhost con el navegador (puede que tengamos que refrescar) y comprobaremos que no se puede conectar. Esto es evidente ya que el contenedor está parado

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker. Ejemplo completo**
  - Ponemos en marcha el contenedor:

```
docker start ubuntu_apache
```

Actividad 5

Probamos a entrar otra vez en localhost.

¿Nos carga la página del Apache? ¿Por qué?

Primeros pasos

- **Paso 4: Ejecutar un contenedor Docker. Ejemplo completo**
  - Si ahora volvemos de nuevo a entrar en localhost y **NO nos deja**.
  - Lo que ocurre es que cuando se para un contenedor y se arranca de nuevo, el estado (los procesos que estaban funcionando) del contenedor desaparece.
  - En este caso lo que ocurre es que el servicio de apache está parado. Conéctate al contenedor, arranca el servicio y vuelve a probar:

Primeros pasos

- **Paso 5: Gestionando los contenedores de Docker**
  - Cuando llevemos un tiempo utilizando Docker, tendremos varios contenedores activos (siendo ejecutados) e inactivos. **Para ver los que están activos**, usaremos:

```
docker ps
```

Primeros pasos

- **Paso 5: Gestionando los contenedores de Docker**
  - Para iniciar un contenedor que se haya detenido, usaremos:

```
docker start [NOMBRE_CONTENEDOR]
```

  - Para detener un contenedor que se esté ejecutando, usaremos:

```
docker stop [NOMBRE_CONTENEDOR]
```

  - Para borrar un contenedor, usaremos:

```
docker rm [NOMBRE_CONTENEDOR]
```

Actividad 6

Detenemos todos los contenedores.

Probamos a borrar todos aquellos contenedores que no estemos usando

Primeros pasos

- **Paso 6: Hacer cambios en un contenedor a una imagen de Docker**
  - Probaremos a crear una imagen a partir del contenedor que ya tenemos de Ubuntu con Apache instalado.
  - Para ello, deberemos utilizar el siguiente comando:

```
docker commit [ID_CONTENEDOR] [NOMBRE_IMAGEN]
```

Primeros pasos

- **Paso 6: Hacer cambios en un contenedor a una imagen de Docker**
  - En nuestro ejemplo, el comando sería:

```
docker commit ff3456f4cd4 ubuntu_apache
```

Actividad 7

Probaremos que se ha creado correctamente la imagen, listando todas las imágenes

Primeros pasos

- **Paso 7: Hacer push de imágenes de Docker a un repositorio Docker**
  - El siguiente paso tras crear una imagen nueva usando una imagen existente es compartirla.
  - Si deseamos hacer push de una imagen a Docker Hub o cualquier otro registro de Docker, debemos tener una cuenta en ese sitio.

Primeros pasos

- **Paso 7: Hacer push de imágenes de Docker a un repositorio Docker**
  - Debemos seguir los siguientes pasos:
    - Iniciar sesión en Docker Hub para hacerle push a su imagen

```
docker login -u [NOMBRE_REGISTRO_DOCKERHUB]
```

    - Hacer tag a la imagen

```
docker tag [NOMBRE_IMAGEN] [NOMBRE_REGISTRO_DOCKERHUB]/[NOMBRE_IMAGEN]
```

Primeros pasos

- **Paso 7: Hacer push de imágenes de Docker a un repositorio Docker**
  - Debemos seguir los siguientes pasos:
    - Subir la imagen a DockerHub

```
docker push [NOMBRE_REGISTRO_DOCKERHUB]/[NOMBRE_IMAGEN]
```

Actividad 8

Sube a DockerHub la imagen que acabamos de crear

Primeros pasos

- **Paso 7: Hacer push de imágenes de Docker a un repositorio Docker**
  - En nuestro ejemplo, los comandos que usaríamos serían:

```
docker login -u martingarciafigueira

docker push martingarciafigueira/ubuntu_apache
```

Primeros pasos. Ejemplos

- **Comprobamos si Docker está correctamente instalado**

docker run hello-world

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Primeros pasos. Ejemplos

- **Lanzamos nuestro primer contenedor**

docker run alpine ls -l

```
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
c9b1b535fdd9: Pull complete
Digest: sha256:ab00606a42621fb68f2ed6ad3c88be54397f981a7b70a79db3d1172b11c4367d
Status: Downloaded newer image for alpine:latest
total 56
drwxr-xr-x    2 root     root          4096 Jan 16 21:52 bin
drwxr-xr-x    5 root     root           340 Feb 19 17:05 dev
drwxr-xr-x    1 root     root          4096 Feb 19 17:05 etc
drwxr-xr-x    2 root     root          4096 Jan 16 21:52 home
drwxr-xr-x    5 root     root          4096 Jan 16 21:52 lib
drwxr-xr-x    5 root     root          4096 Jan 16 21:52 media
drwxr-xr-x    2 root     root          4096 Jan 16 21:52 mnt
drwxr-xr-x    2 root     root          4096 Jan 16 21:52 opt
dr-xr-xr-x  127 root     root             0 Feb 19 17:05 proc
drwx------    2 root     root          4096 Jan 16 21:52 root
drwxr-xr-x    2 root     root          4096 Jan 16 21:52 run
drwxr-xr-x    2 root     root          4096 Jan 16 21:52 sbin
drwxr-xr-x    2 root     root          4096 Jan 16 21:52 srv
dr-xr-xr-x   13 root     root             0 Feb 19 17:05 sys
drwxrwxrwt    2 root     root          4096 Jan 16 21:52 tmp
drwxr-xr-x    7 root     root          4096 Jan 16 21:52 usr
drwxr-xr-x   12 root     root          4096 Jan 16 21:52 var
```

Primeros pasos. Ejemplos

- **Comprobamos las imágenes descargadas**

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
martingarciafigueira/cheers2019	latest	437e09e3639f	6 days ago	4.01MB
<none>	<none>	24a6ab06eb95	6 days ago	356MB
alpine	latest	e7d92cdc71fe	4 weeks ago	5.59MB
ubuntu	latest	ccc6e87d482b	4 weeks ago	64.2MB
martingarciafigueira/ubuntu_apache	latest	ccc6e87d482b	4 weeks ago	64.2MB
golang	1.11-alpine	e116d2efa2ab	6 months ago	312MB
hello-world	latest	fce289e99eb9	13 months ago	1.84kB

Listamos todas las imágenes descargadas en el sistema

Primeros pasos. Ejemplos

- **Ejecutando un contenedor**

docker run alpine echo “Hola mundo desde Alpine!”

```
C:\Users\MESACOCINA>docker run alpine echo "Hola mundo desde Alpine!"
Hola mundo desde Alpine!
```

Ejecutamos el comando “echo” desde Alpine

Primeros pasos. Ejemplos

- **Inspeccionamos los contenedores**

docker ps -a

```
C:\Users\MESACOCINA>docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                          PORTS               NAMES
e9eb0fcc9636        alpine              "echo 'Hola mundo de…"   About a minute ago   Exited (0) About a minute ago                       wonderful_panini
eef774fe4e17        alpine              "ls -l"                  10 minutes ago      Exited (0) 10 minutes ago                       admiring_maxwell
bc8059bef139        hello-world         "/hello"                 11 minutes ago      Exited (0) 11 minutes ago                       busy_meninsky
11748f38c55a        hello-world         "/hello"                 6 days ago          Exited (0) 6 days ago                           heuristic_goldwasser
074a4b9f39ea        ubuntu              "/bin/bash"              6 days ago          Exited (0) 6 days ago                           keen_blackwell
```

Nos muestra los contenedores del sistema

Primeros pasos. Ejemplos

- **Ejecutamos comandos interactivos en contenedores**

docker run -it alpine /bin/sh

```
C:\Users\MESACOCINA>docker run -it alpine /bin/sh
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
/ # uname -a
Linux c7f29f830136 4.19.76-linuxkit #1 SMP Thu Oct 17 19:31:58 UTC 2019 x86_64 Linux
/ # exit
```

Usamos la opción “-it” para conectar la consola con el comando del contenedor

Primeros pasos. Ejemplos

- **Controlando el ciclo de vida de los contenedores**

docker run -d seqvence/static-site

```
C:\Users\MESACOCINA>docker run -d seqvence/static-site
Unable to find image 'seqvence/static-site:latest' locally
latest: Pulling from seqvence/static-site
Image docker.io/seqvence/static-site:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image for
 better future compatibility. More information at https://docs.docker.com/registry/spec/deprecated-schema-v1/
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
716f7a5f3082: Pull complete
7b10f03a0309: Pull complete
aff3ab7e9c39: Pull complete
Digest: sha256:41b286105f913fb7a5fbdce28d48bc80f1c77e3c4ce1b8280f28129ae0e94e9e
Status: Downloaded newer image for seqvence/static-site:latest
7b0aa2ddfecacb562f467e532b24cfa9b22a20b1f2d605dfeaa41173a89458d0
```

Usamos la opción “-d” para ejecutar el contenedor en segundo plano

Primeros pasos. Ejemplos

- **Controlando el ciclo de vida de los contenedores**

docker ps

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS               NAMES
7b0aa2ddfeca        seqvence/static-site   "/bin/sh -c 'cd /usr…"   2 minutes ago       Up 2 minutes        80/tcp, 443/tcp     great_euclid
```

Usamos la opción “-d” para ejecutar el contenedor en segundo plano

Primeros pasos. Ejemplos

- **Controlando el ciclo de vida de los contenedores**

docker stop

```
C:\Users\MESACOCINA>docker stop 7b0aa2ddfeca
7b0aa2ddfeca
```

Detenemos el contenedor

Primeros pasos. Ejemplos

- **Controlando el ciclo de vida de los contenedores**

docker rm

```
C:\Users\MESACOCINA>docker rm 7b0aa2ddfeca
7b0aa2ddfeca
```

Borramos el contenedor

Docker en red. Ejemplos

- **Lanzar un contenedor en un puerto público**

docker run --name static-site

-e AUTHOR="Martin" -d

-p 9000:80 seqvence/static-site

```
C:\Users\MESACOCINA>docker run --name static-site -e AUTHOR="Martin" -d -p 9000:80 seqvence/static-site
4ff2b3bba2b549e7fc7bc564a849d26fe60b6bc65943369d9a98ac03611d5988
```

Docker en red. Ejemplos

- **Lanzar un contenedor en un puerto público**

docker run --name static-site

-e AUTHOR="Martin" -d

-p 9000:80 seqvence/static-site

```
C:\Users\MESACOCINA>docker run --name static-site -e AUTHOR="Martin" -d -p 9000:80 seqvence/static-site
4ff2b3bba2b549e7fc7bc564a849d26f[illegible]a98ac03611d5988
```

Especificamos el nombre del contenedor

Docker en red. Ejemplos

- **Lanzar un contenedor en un puerto público**

docker run --name static-site

-e AUTHOR="Martin" -d

-p 9000:80 seqvence/static-site

```
C:\Users\MESACOCINA>docker run --name static-site -e AUTHOR="Martin" -d -p 9000:80 seqvence/static-site
4ff2b3bba2b549e7fc7bc564a849d26fe60b6bc65943369d9a98ac0361[illegible]
```

Especificamos el nombre del autor

Docker en red. Ejemplos

- **Lanzar un contenedor en un puerto público**

docker run --name static-site

-e AUTHOR="Martin" -d

-p 9000:80 seqvence/static-site

```
C:\Users\MESACOCINA>docker run --name static-site -e AUTHOR="Martin" -d -p 9000:80 seqvence/static-site
4ff2b3bba2b549e7fc7bc564a849d26fe60b6bc65943369d9a98ac03611d5988
```

Especifica que se ejecuta en segundo plano

Docker en red. Ejemplos

- **Lanzar un contenedor en un puerto público**

docker run --name static-site

-e AUTHOR="Martin" -d

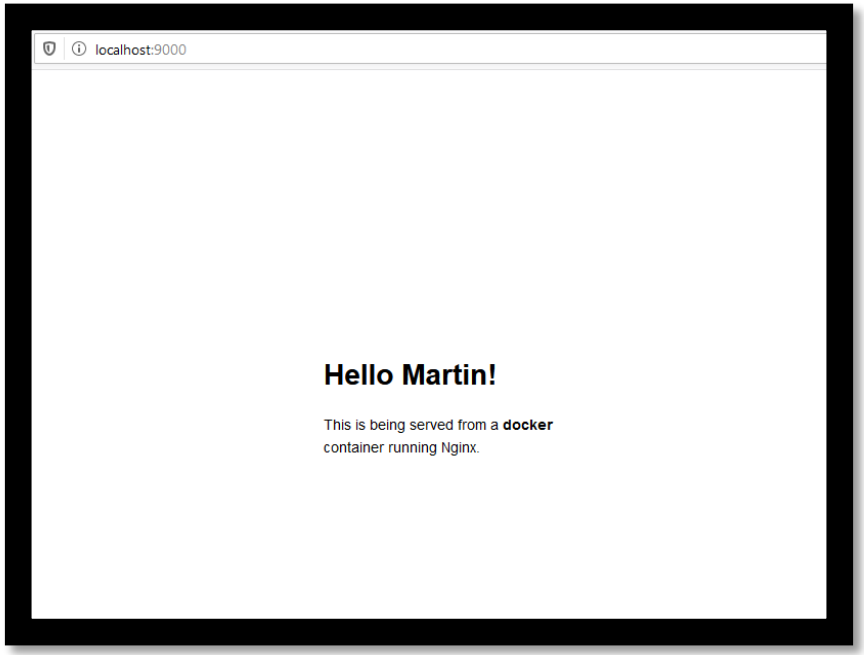
-p 9000:80 seqvence/static-site

```
C:\Users\MESACOCINA>docker run --name static-site -e AUTHOR="Martin" -d -p 9000:80 seqvence/static-site
4ff2b3bba2b549e7fc7bc564a849d26fe60b6bc65943369d9a98ac03611d5988
```

Conectamos el puerto local 9000 con el puerto 80 del contenedor

Docker en red. Ejemplos

- **Probamos el funcionamiento**



Manejando los contenedores. Ejemplos

- **Parar y borrar un contenedor**

docker stop static-site

docker rm static-site

```
C:\Users\MESACOCINA>docker stop static-site
static-site

C:\Users\MESACOCINA>docker rm static-site
static-site
```

Manejando los contenedores. Ejemplos

- **Parar y borrar un contenedor activo**

docker rm -f static-site

```
C:\Users\MESACOCINA>docker rm -f static-site
static-site
```

Manejando los contenedores. Ejemplos

- **Borrar todos los contenedores**

docker rm $(docker ps -aq)

```
PS C:\Users\MESACOCINA> docker rm $(docker ps -aq)
15bae13c8271
012ae82c6af8
c7f29f830136
e9eb0fcc9636
eef774fe4e17
bc8059bef139
11748f38c55a
074a4b9f39ea
```

Manejando las imágenes

Tipos de imágenes

- **Imágenes base**
  - Son imágenes sin una imagen padre
  - Ejemplo: Ubuntu, Debian...
- **Imágenes hijas**
  - Imágenes que proporcionan software adicional
  - Ejemplo: Apache, MySQL...

Manejando las imágenes

Imágenes oficiales VS Imágenes de usuario

- **Imágenes oficiales**
  - Imágenes creadas por entidades de confianza
- **Imágenes de usuario**
  - Imágenes creadas por cualquier usuario normal

Manejando las imágenes. Ejemplos

- **Listamos todas las imágenes**

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
martingarciafigueira/cheers2019	latest	437e09e3639f	7 days ago	4.01MB
<none>	<none>	24a6ab06eb95	7 days ago	356MB
alpine	latest	e7d92cdc71fe	4 weeks ago	5.59MB
ubuntu	latest	ccc6e87d482b	4 weeks ago	64.2MB
martingarciafigueira/ubuntu_apache	latest	ccc6e87d482b	4 weeks ago	64.2MB
golang	1.11-alpine	e116d2efa2ab	6 months ago	312MB
hello-world	latest	fce289e99eb9	13 months ago	1.84kB
seqvence/static-site	latest	f589ccde7957	3 years ago	191MB

El atributo Tag es la versión

Manejando las imágenes. Ejemplos

- **Borrar una imagen**

docker rmi ubuntu

```
C:\Users\MESACOCINA>docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:04d48df82c938587820d7b6006f5071dbbffceb7ca01d2814f81857c631d44df
Deleted: sha256:72300a873c2ca11c70d0c8642177ce76ff69ae04d61a5813ef58d40ff66e3e7c
Deleted: sha256:d3991ad41f89923dac46b632e2b9869067e94fcdffa3ef56cd2d35b26dd9bce7
Deleted: sha256:2e533c5c9cc8936671e2012d79fc6ec6a3c8ed432aa81164289056c71ed5f539
Deleted: sha256:282c79e973cf51d330b99d2a90e6d25863388f66b1433ae5163ded929ea7e64b
Deleted: sha256:cc4590d6a7187ce8879dd8ea931ffaa18bc52a1c1df702c9d538b2f0c927709d
```

Manejando las imágenes. Ejemplos

- **Descargar una versión concreta**

docker pull ubuntu:18.04

```
C:\Users\MESACOCINA>docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
Digest: sha256:8d31dad0c58f552e890d68bbfb735588b6b820a46e459672d96e585871acc110
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
```

Manejando las imágenes. Ejemplos

- **Descargamos la última versión**

docker pull ubuntu

```
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:8d31dad0c58f552e890d68bbfb735588b6b820a46e459672d96e585871acc110
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

Manejando las imágenes. Ejemplo completo

Crear la primera imagen

- Crearemos una aplicación web para mostrar imágenes aleatorias de gatitos usando Python
- Crearemos una carpeta llamada **app-gatitos-montecastelo**
- Descargaremos el fichero **MiPrimeraImagen.zip** del Classroom

Manejando las imágenes. Ejemplo completo

Crear la primera imagen

- Tenemos todos los archivos necesarios para la aplicación web
- Sin embargo, necesitamos Python y Flask para ejecutar la aplicación
- Para ejecutar la aplicación web, crearemos **UNA NUEVA IMAGEN** con dependencias (Python y Flask) y nuestro código fuente
- Tras esto, crearemos **UN NUEVO CONTENEDOR** para ejecutar la aplicación

Manejando las imágenes. Ejemplo completo

Dockerfile

- Es un fichero utilizado para describir una nueva imagen
- Nos especifica:
  - La imagen base
  - Comandos que se van a ejecutar en esta imagen y cuando se lanza
  - Ficheros que se pueden incluir en la imagen
  - Puertos abiertos

Manejando las imágenes. Ejemplo completo

Dockerfile

```
# Imagen base
FROM alpine:3.5

# Instalamos Python y pip
RUN apk add --update py2-pip

# Hacemos un upgrade de pip
RUN pip install --upgrade pip

# Instalamos los módulos Python que la aplicación necesita
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# Copiamos los ficheros necesarios para que la aplicación arranque
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# Establecemos el puerto en que la aplicación va a correr
EXPOSE 5000

# Lanzamos la aplicación
CMD ["python", "/usr/src/app/app.py"]
```

Manejando las imágenes. Ejemplo completo

Construimos la imagen

- Nos posicionamos en la carpeta con el Dockerfile y ejecutamos:

docker build -t miprimeraimagen .

- Las acciones que hemos llevado a cabo son:
  - Crear un nuevo contenedor con una imagen base
  - Ejecutar comandos y copiar ficheros de aplicación
  - Crear un nuevo contenedor con el resultado

Manejando las imágenes. Ejemplo completo

Ejecutamos la nueva imagen

docker run -p 9000:5000 miprimeraimagen

```
C:\dockerPruebas\MiPrimeraImagen>docker run -p 9000:5000 miprimeraimagen
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
172.17.0.1 - - [19/Feb/2020 18:41:17] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2020 18:41:21] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2020 18:41:22] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2020 18:41:22] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [19/Feb/2020 18:41:23] "GET / HTTP/1.1" 200 -
```

Accedemos a la ruta http://localhost:9000/ en el navegador

Actividad 8

Comprobaremos que se muestra la aplicación correctamente

Manejando las imágenes. Ejemplo completo

Construimos la imagen (otra vez)

- Cambiamos el HTML en templates/index.html:
- Creamos la imagen otra vez

docker build -t miprimeraimagen .

- Comprobamos que la imagen se crea muy rápidamente porque solamente se copian los ficheros cambiados

Manejando las imágenes. Ejemplo completo

Publicamos la imagen

- Podemos publicar nuestras imágenes en **Docker Hub**
- Los repositorios públicos son gratuitos
- Podemos tener un repositorio privado (Solamente podemos tener uno gratuito)

Manejando las imágenes. Ejemplo completo

Publicamos la imagen

docker build -t [NOMBRE_USUARIO]/nombreAplicacion .

```
C:\dockerPruebas\MiPrimeraImagen>docker build -t martingarciafigueira/appprueba .
Sending build context to Docker daemon  8.192kB
Step 1/9 : FROM alpine:3.5
 ---> f80194ae2e0c
Step 2/9 : RUN apk add --update py2-pip
 ---> Using cache
 ---> 1bde7835da75
Step 3/9 : RUN pip install --upgrade pip
 ---> Using cache
 ---> 081fd9cea317
Step 4/9 : COPY requirements.txt /usr/src/app/
 ---> Using cache
 ---> 23c4f69f8498
Step 5/9 : RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
 ---> Using cache
 ---> d98068b53c2d
Step 6/9 : COPY app.py /usr/src/app/
 ---> Using cache
 ---> a3c487b5b1f5
Step 7/9 : COPY templates/index.html /usr/src/app/templates/
 ---> Using cache
 ---> b3e3e5c4151b
Step 8/9 : EXPOSE 5000
 ---> Using cache
 ---> 5a7de3712b87
Step 9/9 : CMD ["python", "/usr/src/app/app.py"]
 ---> Using cache
 ---> 475055b6fdf7
Successfully built 475055b6fdf7
Successfully tagged martingarciafigueira/appprueba:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and director
ies added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions f
or sensitive files and directories.
```

Creamos la imagen en nuestro repositorio

Manejando las imágenes. Ejemplo completo

Publicamos la imagen

docker login

docker push martingarciafigueira/appprueba

```
C:\dockerPruebas\MiPrimeraImagen>docker push martingarciafigueira/appprueba
The push refers to repository [docker.io/martingarciafigueira/appprueba]
994593b0c45f: Pushed
b36026224e3b: Pushed
8737d79e4600: Pushed
c655baab28e0: Pushed
e78755dc0a80: Pushed
f0d4ecf9264d: Pushed
f566c57e6f2d: Mounted from library/alpine
latest: digest: sha256:99170ec5e15299c644978828cd4ca48d470e3250367e86af5ae8e1ec6b4bcde2 size: 1783
```

Publicamos la imagen creada

Tema 6 : Docker

Ciclo Superior DAW

Asignatura: Despliegue de aplicaciones web

Curso 20/21