



Despliegue de aplicaciones web



Tema 3. Docker

1. Qué es Docker

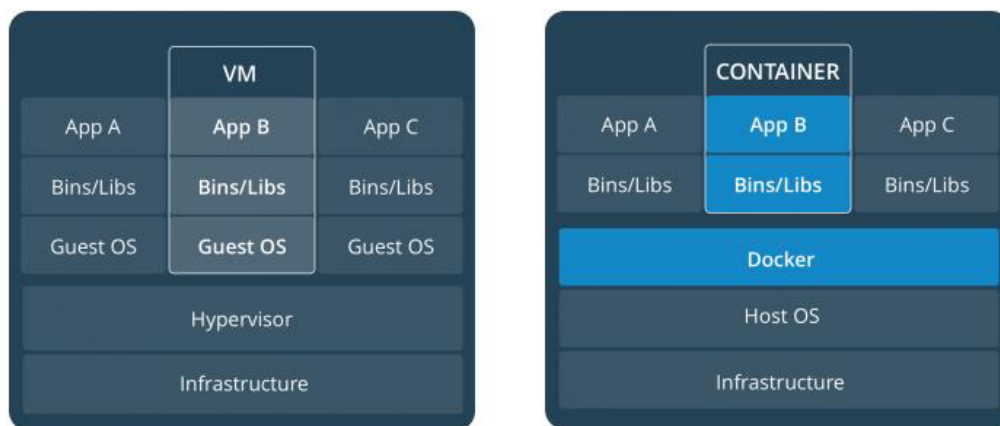
Docker es una plataforma de software que facilita la creación, implementación y ejecución de aplicaciones mediante el uso de contenedores.

Esta plataforma permite crear aplicaciones o servicios que sean independientes y portables. Es decir, sin importar el sistema operativo utilizado o el hardware de cada máquina, si es posible instalar Docker se podrá desplegar en el equipo. Esto evita, por ejemplo, tener que instalar dependencias en el host o servidor, o hacer uso de máquinas virtuales.

Otra situación típica es tener que configurar diferentes entornos completos(entorno de desarrollo, entorno de pruebas, entorno de producción, etc.). Esto no será necesario haciendo uso de Docker, bastará con tenerlo instalado.

Por ejemplo si se desarrolla una aplicación en PHP, utilizando una versión concreta, ésta requerirá dicha versión específica, ciertos módulos determinados y Mysql. El contenedor Docker se puede crear ya con todos esos requerimientos, con las versiones específicas y la aplicación. Y en el servidor no habrá que configurar nada, simplemente desplegar con Docker.

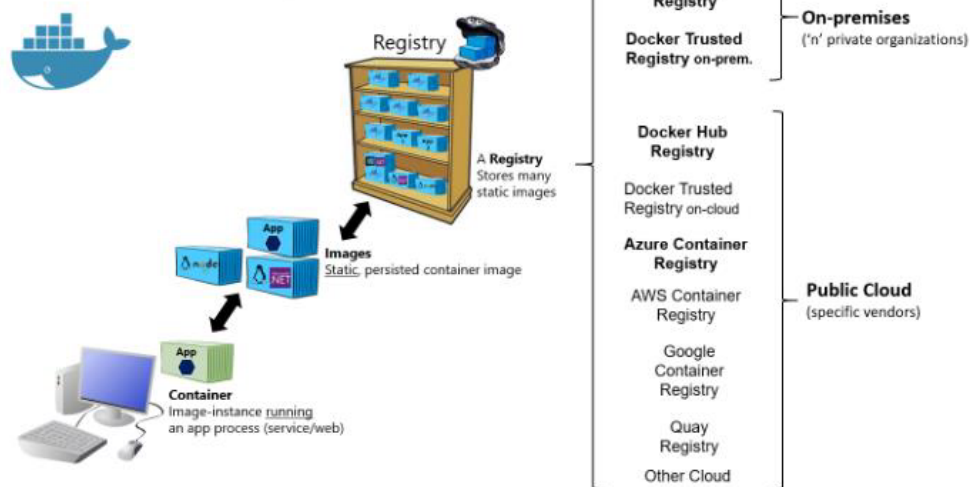
Trabajar con contenedores es más eficiente que hacerlo con máquinas virtuales ya que, como se ejecutan sobre el kernel de la máquina, son mucho más ligeros que las MV. Así será posible ejecutar muchos más contenedores para el mismo equipo que si éstos fueran MV.



Hay algunos conceptos que hay que tener claros para entender los contenedores:

- Imagen: Se trata de un paquete ejecutable que incluye todo lo necesario para ejecutar un software.
- Contenedor: instancia en ejecución de una imagen.
- Registro: Aplicación para gestionar almacenamiento y envío de imágenes de contenedores.

Basic taxonomy in Docker



1.1. Qué es un contenedor



En un contenedor lo importante es la forma modular que tiene que permite su fácil almacenamiento y transporte.

Un contenedor no es más que un paquete de código completado con las dependencias y herramientas necesarias para poder ejecutar dicho código (librerías del sistema, entorno de ejecución, cualquier tipo de configuración, etc), describiéndolo en un pequeño archivo de configuración llamado **DockerFile**. Por ejemplo, si se tiene una aplicación desarrollada con nodeJS(entorno de ejecución de Javascript del lado del servidor), el contenedor almacenará el código de dicha aplicación, el entorno de ejecución de nodeJS y cualquier otra herramienta que pueda ser necesaria en la ejecución del código.

Por lo tanto, un contenedor con el mismo código, mismas herramientas, mismo entorno de ejecución, etc., tendrá siempre el mismo comportamiento y permitirá obtener siempre el mismo resultado, independientemente del lugar en que se ejecuten.

Una gran ventaja del uso de contenedores es que pueden ser versionados, reutilizados y replicados fácilmente por administradores de sistemas u otros codificadores sin necesidad de conocer el funcionamiento interno de la aplicación.



1.2. Qué es una imagen

Una imagen es una especie de representación estática de una aplicación o servicio, de su configuración y todas sus dependencias, se puede asimilar a una plantilla de un contenedor. Por ejemplo una imagen podría contener un sistema operativo Ubuntu con un servidor Apache y una aplicación web instalada. Normalmente al crear imágenes se parte de una imagen padre a la que se le van añadiendo cosas (p.Ej.: una imagen padre con Ubuntu y Apache modificada para instalar la aplicación).

Para ejecutar la aplicación o el servicio, se crea una instancia de esa imagen, el contenedor, que se ejecutará en el host de Docker. Las imágenes al ser plantillas van a ser usadas para crear nuevos contenedores, y nunca cambian a no ser que se cree una nueva a partir de un contenedor.

En resumen, los contenedores son instancias en ejecución de una imagen y son los que realmente ponen en marcha las aplicaciones y servicios de esa imagen. A partir de una misma imagen, es posible ejecutar diferentes contenedores. Esto permite tener copias de una aplicación ejecutándose en varios contenedores y utilizar balanceadores de carga para distribuir los accesos a ella y ofrecer servicios con más garantías y con menos carga.

Como ya se ha dicho, las imágenes son inmutables, por ello si en un contenedor creado a partir de una imagen concreta se hace algún cambio, al parar dicho contenedor los cambios se perderían. Si se quieren mantener esos cambios sería necesario crear otra imagen a partir del contenedor para que contenga los cambios. Por lo tanto otra ventaja de Docker, es el versionado de los contenedores.

1.3. Qué es un registro

Las imágenes se almacenan en registros, tanto públicos como privados, que actúan como bibliotecas de imágenes. Docker ofrece su propio registro Docker Hub, pero hay otros proveedores que también ofrecen los suyos como Azure, Amazon Web Service o Google. Además, no es extraño que las empresas tengan un registro privado para sus propias imágenes.

Anteriormente se ha hablado del versionado de contenedores. Simplemente con mantener el registro de imágenes junto con el entorno de producción(en local o en la nube), se puede volver de forma sencilla a una versión anterior del contenedor, así la latencia de red sería mínima y la puesta en marcha de un contenedor lo más rápida posible.



2. Instalación

EJERCICIO 1

- Se va a instalar docker en el servidor Ubuntu, siguiendo el **Paso 1** del tutorial accesible a través del enlace “Instalación de Docker” que encontrarás en el Aula Virtual.
- Una vez completado ese primer paso, Docker quedará instalado. Para probar que Docker se ha instalado correctamente, se puede ejecutar un contenedor llamado hello-world que se ejecuta sobre tu máquina y muestra un mensaje. Dicho contenedor se ejecutará mediante el siguiente comando:
`docker run hello-world`
- Visualiza qué versión de docker se ha instalado ejecutando:
`docker --version`



3. Imágenes

Lo que se ha hecho en el segundo punto del ejercicio 1 es ejecutar un contenedor. Como se ha dicho con anterioridad, un contenedor es una instancia de una imagen y, como no se ha descargado la imagen antes de lanzar la ejecución, Docker si la ha descargado antes de proceder a la ejecución en sí.

Por defecto, Docker descarga las imágenes de un registro propio, Docker Hub. Este registro es gratuito y cualquiera puede hacerse una cuenta y almacenar sus imágenes en el.

A continuación se van a ver una serie de comandos necesarios para poder gestionar imágenes.

Listar imágenes

Con el siguiente comando se puede ver un listado de las imágenes existentes:

```
docker image ls
```

Y el resultado será, en forma de tabla, algo similar a la siguiente imagen, en la que se pueden ver las imágenes(en este caso solo una) existentes:

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
sonia@ubuntuserverSonia:~$ sudo su
[sudo] password for sonia:
root@ubuntuserverSonia:/home/sonia# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest             bf756fb1ae65       10 months ago     13.3kB
root@ubuntuserverSonia:/home/sonia#
```

Buscar imágenes

Con el comando **docker search** se pueden hacer búsquedas de imágenes en Docker Hub. En el siguiente ejemplo se ha realizado una búsqueda de imágenes que tengan un nombre relacionado con JavaScript:

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker search JavaScript
NAME                DESCRIPTION                                     STA
node                Node.js is a JavaScript-based platform for s... 941
ghost               Ghost is a free and open source blogging pla... 128
couchdb             CouchDB is a database that uses JSON for doc... 379
convergenclabs/javascript-examples  Provides several examples that demonstrate t... 3
hoopla/javascript-build  Javascript build 1
jfactory/javascript-slave  Dockerfile: https://github.com/codenvy/docke... 0
codenvy/javascript_html  Dockerfile: https://github.com/codenvy/docke... 0
cyberdojofoundation/javascript-node  Run a server for JavaScript Obfuscator with ... 0
entelectchallenge/javascript  0
hexletboy/javascript_numbers_exercise  0
1337kavin/javascript-obfuscator  0
cyberdojofoundation/javascript-node_cucumber  0
javascriptjenkins/sbapi  0
elderhq/javascript-circle-image  0
cyberdojofoundation/javascript-node_mocha_chai_sinon  0
cyberdojofoundation/javascript-node_assert  0
```



Historial de una imagen

Con el siguiente comando se puede ver cómo se construyó una imagen concreta:

```
docker image history nombrelimagen
```

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker image history hello-world
IMAGE          CREATED          CREATED BY          SIZE
bf756fb1ae65   11 months ago   /bin/sh -c #(nop)  CMD ["/hello"]      0B
<missing>      11 months ago   /bin/sh -c #(nop)  COPY file:7bf12aab75c3867a... 13.3kB
root@ubuntuserverSonia:/home/sonia#
```

Eliminar imágenes en desuso

El comando prune permite borrar todas las imágenes que no se estén utilizando, siempre y cuando no haya ningún contenedor de la misma. Con -a borrará todas las no utilizadas:

```
docker image prune -a
```

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    bf756fb1ae65   11 months ago  13.3kB
root@ubuntuserverSonia:/home/sonia# docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
root@ubuntuserverSonia:/home/sonia# docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    bf756fb1ae65   11 months ago  13.3kB
root@ubuntuserverSonia:/home/sonia#
```

Borrar imágenes

Para borrar una o más imágenes se puede hacer uso del comando rm:

```
docker image rm nombrelimagen
```

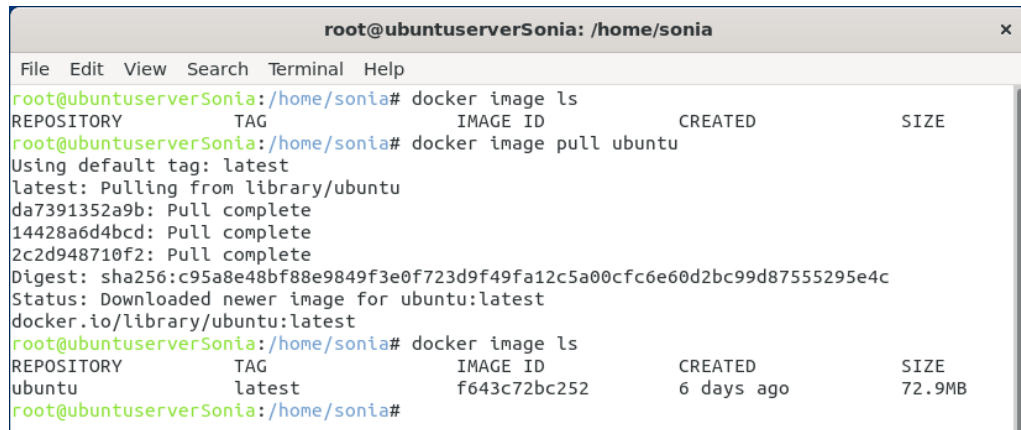
En la siguiente imagen se puede ver como se ha intentado borrar la imagen, no se ha permitido por existir contenedores de la misma, y después se ha forzado con la opción -f:

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    bf756fb1ae65   11 months ago  13.3kB
root@ubuntuserverSonia:/home/sonia# docker image rm hello-world
Error response from daemon: conflict: unable to remove repository reference "hello-world"
(must force) - container 4393059d306d is using its referenced image bf756fb1ae65
root@ubuntuserverSonia:/home/sonia# docker image rm -f hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:e7c70bb24b462baa86c102610182e3efcb12a04854e8c582838d92970a09f323
Deleted: sha256:bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b
root@ubuntuserverSonia:/home/sonia# docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
root@ubuntuserverSonia:/home/sonia#
```

Descargar imágenes de un registro

Con el siguiente comando se pueden descargar imágenes de un repositorio:

```
docker image pull nombreImagen
```



```
root@ubuntuServerSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuServerSonia:/home/sonia# docker image ls
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
root@ubuntuServerSonia:/home/sonia# docker image pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
Digest: sha256:c95a8e48bf88e9849f3e0f723d9f49fa12c5a00cfc6e60d2bc99d87555295e4c
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
root@ubuntuServerSonia:/home/sonia# docker image ls
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
ubuntu              latest       f643c72bc252     6 days ago      72.9MB
root@ubuntuServerSonia:/home/sonia#
```

De las imágenes, de la misma forma que en un repositorio para control de versiones, pueden existir distintas etiquetas. Si en la descarga no se indica la etiqueta docker descarga la mas actual. Para indicar la etiqueta, se ha de incluir ésta tras el nombre de la imagen a descargar siendo separados ambos datos por .:

```
docker image pull nombreImagen:nombreEtiqueta
```

Subir imágenes a un registro

Para subir imágenes a un repositorio se hace uso del comando push. Para esto se ha de tener un usuario del registro elegido y hay que logearse primero:

```
docker image push nombreImagen
```

Opcionalmente se puede indicar un nombre de etiqueta:

```
docker image push nombreImagen:nombreEtiqueta
```

Esto se verá mas adelante en la unidad.

Equivalencias

De los comandos vistos hasta el momento, a continuación se incluyen algunas equivalencias:

- **docker images** tiene el mismo resultado que **docker image ls**
- **docker rmi imagen** es la versión corta de **docker image rm imagen**
- **docker pull imagen** hace lo mismo que **docker image pull imagen**
- **docker push imagen** es equivalente a **docker image push imagen**

Información en detalle de una imagen

Con el comando inspect se puede ver información detallada de una o mas imágenes. tales como etiquetas, comandos, versiones, etc.

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker image inspect hello-world
[
  {
    "Id": "sha256:bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b",
    "RepoTags": [
      "hello-world:latest"
    ],
    "RepoDigests": [
      "hello-world@sha256:e7c70bb24b462baa86c102610182e3efcb12a04854e8c582838d92970a09f323"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2020-01-03T01:21:37.263809283Z",
    "Container": "71237a2659e6419aee44fc0b51ffbd12859d1a50ba202e02c2586ed999def583",
    "ContainerConfig": {
      "Hostname": "71237a2659e6",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      "Cmd": [
        "/bin/sh",

```

Crear una imagen

El comando build permite crear imágenes propias a través de un fichero llamado DockerFile. Este comando se verá más adelante en el tema.

docker image build nombreImagen path

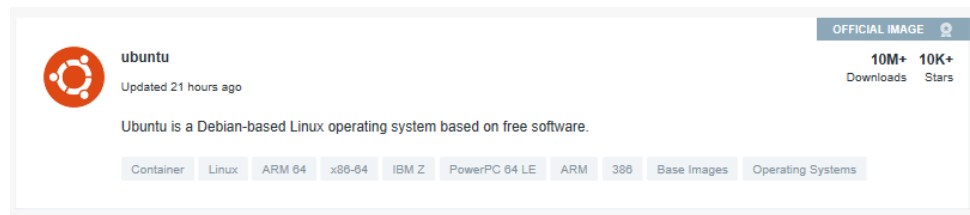
ó

docker image build nombreImagen:nombreEtiqueta path

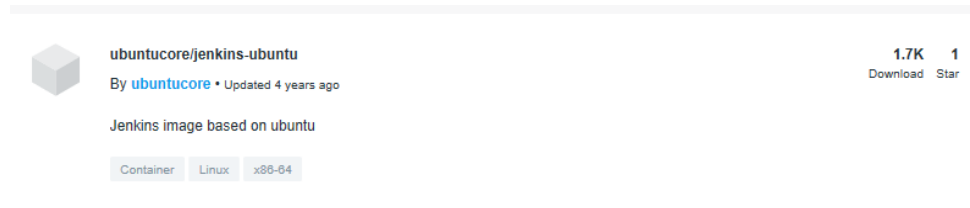
EJERCICIO 2

Este ejercicio va a servir para ir familiarizándote con Docker.

- Primero vas a probar todos los comandos vistos hasta el momento, salvo push y build. Para ello, descarga la imagen codenvy/javascript_html.
- Ve a Docker Hub y date de alta: <https://hub.docker.com/>
- Busca imágenes que contenga html, java, php(por separado). Verás que hay imágenes de dos tipos:
 - nombre: Las que solo tienen un nombre son imágenes oficiales y las que se recomienda usar:



- nombre/otroTexto: Las que tienen este formato son imágenes creadas por una persona o empresa:



4. Contenedores

A continuación se van a ver algunos comandos relacionados con contenedores.

Ejecutar un contenedor

Con el siguiente comando se puede crear una instancia de una imagen, es decir, ponerla en ejecución. Docker generará un nombre y un ID aleatorios para el contenedor.

```
docker run nombrelimagen
```

El uso de este comando ya se vio al arrancar la imagen hello-world.

Es posible darle un nombre concreto al contenedor, mediante la opción `--name`. El nombre no puede estar asignado ya a otro contenedor:

```
docker run --name nombreContenedor nombrelImagen
```

Otra de las opciones interesantes es ejecutar el contenedor en segundo plano para poder seguir trabajando con la consola de docker. Esto se consigue con la opción `-d`:

```
docker run -d nombrelimagen
```



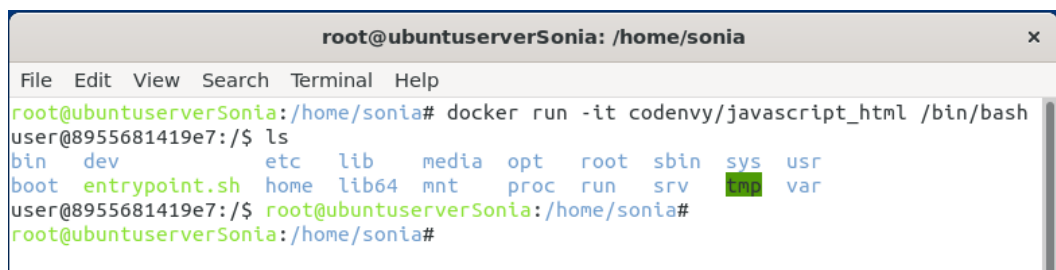
```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker run -d --name jsHTML codenvy/javascript_html
8882d9d3ca3bc57dfc740c0de2d3155af9ab6047be29528212dba64ee581d859
root@ubuntuserverSonia:/home/sonia#
```

Docker también permite ejecutar la consola del contenedor para poder trabajar con ella (la opción `-it` permite tener un terminal interactivo):

```
docker run -it nombrelimagen /bin/bash
```

Para poder salir de la consola del contenedor: `Ctrl+p Ctrl+q`

En el siguiente ejemplo se ha entrado a la consola propia del contenedor y se ha listado el contenido de su carpeta raíz:



```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker run -it codenvy/javascript_html /bin/bash
user@8955681419e7:/$ ls
bin  dev  etc  lib  media  opt  root  sbin  sys  usr
boot  entrypoint.sh  home  lib64  mnt  proc  run  srv  tmp  var
user@8955681419e7:/$ root@ubuntuserverSonia:/home/sonia#
root@ubuntuserverSonia:/home/sonia#
```



Listar contenedores

Para listar todos los contenedores en funcionamiento, se utiliza el comando ps. Si se quieren ver también los que están parados, se utilizará la opción -a:

```
docker ps -a
```

```
root@ubuntuServerSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuServerSonia:/home/sonia# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
8f318592ba99        codenvy/javascript_html   "/entrypoint.sh /bin..."   20 minutes ago     Up 20 minutes      80/tcp, 4200/tcp   vigorc
root@ubuntuServerSonia:/home/sonia# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
8dca2fbc0409        hello-world         "/hello"            19 minutes ago     Exited (0) 19 minutes ago              80/tcp, 4200/tcp
8f318592ba99        codenvy/javascript_html   "/entrypoint.sh /bin..."   20 minutes ago     Up 20 minutes
6c9d280b3941        hello-world         "/hello"            5 days ago         Exited (0) 5 days ago
4393059d306d        hello-world         "/hello"            12 days ago        Exited (0) 12 days ago
root@ubuntuServerSonia:/home/sonia#
```

Otra opción disponible para listar contenedores es indicar las columnas que se quieren mostrar en el listado. Se indicará con la opción `--format` seguida de los nombres de los campos que se quieren mostrar:

```
docker ps --format nombresCampos
```

nombresCampos tendrá el siguiente formato(\t es el separador, podrá ser cualquiera):

```
"table {{ .NombreCol1 }}\t{{.NombreCol2}}\t{{.NombreColN }}"
```

```
root@ubuntuServerSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuServerSonia:/home/sonia# docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}" -a
CONTAINER ID        IMAGE               NAMES
6cc2d279054d        hello-world         HELLO
8dca2fbc0409        hello-world         mystifying_easley
8f318592ba99        codenvy/javascript_html   vigorous_joliot
6c9d280b3941        hello-world         dazzling_brahmagupta
4393059d306d        hello-world         heuristic_panini
root@ubuntuServerSonia:/home/sonia#
```

En el siguiente enlace se pueden ver los nombres de cada columna de la tabla:

<https://docs.docker.com/engine/reference/commandline/ps/>

Manejar un contenedor

Para parar un contenedor en ejecución se hace uso del comando stop y es necesario indicar el ID o el nombre del contenedor:

```
docker stop IDoNombreContendor
```

Y para parar todos los contenedores:

```
docker stop $(docker ps -a -q)
```

Si no se puede parar el contenedor con stop, se puede forzar la parada con el comando kill:

```
docker kill IDoNombreContendor
```



Si se quiere volver a arrancar un contenedor parado con anterioridad, se usa el id o el nombre con el comando start:

```
docker start IDoNombreContendor
```

Como ya se ha visto, cuando se para un contenedor no se elimina y es posible volverlo a arrancar. Para eliminarlo definitivamente se usa el comando rm seguido del ID o el nombre del contenedor:

```
docker rm IDoNombreContendor
```

Y si lo que se quiere es eliminar todos los contenedores, tras pararlos con la opción vista con anterioridad, se puede hacer:

```
docker rm $(docker ps -a -q)
```

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}"
CONTAINER ID      IMAGE               NAMES
root@ubuntuserverSonia:/home/sonia# docker run -d --name jshtml codenvy/javascript_html
a7c59ec687e2db2b44af646e226fbd526b8c3f32c023d5ee3b66af966d3e94c9
root@ubuntuserverSonia:/home/sonia# docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}"
CONTAINER ID      IMAGE               NAMES
a7c59ec687e2      codenvy/javascript_html  jshtml
root@ubuntuserverSonia:/home/sonia# docker stop jshtml
jshtml
root@ubuntuserverSonia:/home/sonia# docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}"
CONTAINER ID      IMAGE               NAMES
root@ubuntuserverSonia:/home/sonia# docker start jshtml
jshtml
root@ubuntuserverSonia:/home/sonia# docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}"
CONTAINER ID      IMAGE               NAMES
a7c59ec687e2      codenvy/javascript_html  jshtml
```

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker kill jshtml
jshtml
root@ubuntuserverSonia:/home/sonia# docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}"
CONTAINER ID      IMAGE               NAMES
root@ubuntuserverSonia:/home/sonia# docker rm jshtml
jshtml
root@ubuntuserverSonia:/home/sonia# docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Names}}" -a
CONTAINER ID      IMAGE               NAMES
5b4da20b8a37      codenvy/javascript_html  prueba1
0850825d86dc      codenvy/javascript_html  musing_dhawan
6cc2d279054d      hello-world         HELLO
8dca2fbc0409      hello-world         mystifying_easley
8f318592ba99      codenvy/javascript_html  vigorous_joliot
6c9d280b3941      hello-world         dazzling_brahmagupta
4393059d306d      hello-world         heuristic_panini
root@ubuntuserverSonia:/home/sonia#
```

Ejecutar comandos Linux en un contenedor

En puntos anteriores se ha visto que se podía arrancar el contenedor entrando a la vez en la consola del mismo. En caso de que el contenedor ya esté corriendo, hay dos alternativas:

Por un lado, se puede ejecutar directamente un comando en el contenedor:

`docker exec IDoNombreContenedor comando`

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker exec jsHTML ls
bin
boot
dev
entrypoint.sh
etc
home
lib
lib64
media
mnt
```

Como segunda opción, se puede abrir el terminal del contenedor:

`docker exec -it nombreContenedor /bin/bash`

```
user@8882d9d3ca3b: /
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker exec -it jsHTML /bin/bash
user@8882d9d3ca3b:/$
```

Guardar un contenedor como imagen

Ya se ha dicho que los cambios realizados en un contenedor se pierden a no ser que se haga una nueva imagen a partir del mismo. Tras detener el contenedor al que se le han aplicado ciertos cambios, se puede crear una imagen mediante el comando commit:

`docker commit IDoNombreContenedor nombreImagen:nombreTag`

```
root@ubuntuserverSonia: /home/sonia
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia# docker commit jsHTML jsHTML:last
invalid reference format: repository name must be lowercase
root@ubuntuserverSonia:/home/sonia# docker commit jsHTML jshtml:last
sha256:78f052c639a00164a8a61868a1c76ac24e131b7003acee224ad4c795246a5230
root@ubuntuserverSonia:/home/sonia# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jshtml               latest             78f052c639a0       10 seconds ago     359MB
ubuntu               latest             f643c72bc252       12 days ago        72.9MB
hello-world          latest             bf756fb1ae65       11 months ago      13.3kB
codenvy/javascript_html latest             aefa9e0d8aad       5 years ago        359MB
root@ubuntuserverSonia:/home/sonia# docker commit jsHTML jshtml
sha256:60c7f6ba32348dc735f1866686c87bb8b5f993f6853ebab49dab4d5a28458f2b
root@ubuntuserverSonia:/home/sonia# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jshtml               latest             60c7f6ba3234       2 seconds ago      359MB
jshtml               last               78f052c639a0       About a minute ago 359MB
ubuntu               latest             f643c72bc252       12 days ago        72.9MB
hello-world          latest             bf756fb1ae65       11 months ago      13.3kB
codenvy/javascript_html latest             aefa9e0d8aad       5 years ago        359MB
root@ubuntuserverSonia:/home/sonia#
```



EJERCICIO 3

- Lista los contenedores que tienes (corriendo y parados).
- Elimina todos los contenedores.
- Arranca, en segundo plano, la imagen que descargaste para el ejercicio 2 con el nombre `codEnvJsHtml`.
- Una vez arrancada lista el contenido de su directorio raíz.
- Para el contenedor.
- Arranca la imagen, de nuevo con el nombre `codEnvJsHtml`. ¿Qué pasa? Si hay algún problema, solúcionalo siguiendo las indicaciones que te haya dado docker.
- Reinicia el primer contendor.
- Lista los contenedores. ¿Cuántos hay?



5. Trabajar con Docker, primeros pasos

Una imagen se puede crear a partir de un contenedor haciendo las modificaciones necesarias, o se puede crear a través de un fichero donde se definen las características que se quieren añadir a dicha imagen, fichero llamado **Dockerfile**.

5.1. Imagen para una página estática

A continuación se va a ver cómo crear una imagen para una página estática. En este caso se va a disponer de un servidor Apache:

EJERCICIO 4

Este ejercicio se va a hacer de forma conjunta. Es recomendable que vayas apuntando todos los conceptos, pasos o datos que vayan surgiendo.

➤ Ejecutar la imagen de Apache:

El primer paso es ejecutar la imagen de Apache. Como ya se ha visto anteriormente, si no se ha descargado con anterioridad la imagen, Docker la descargará del registro:

```
docker run -d --name apache2 -p 82:80 bitnami/apache
```

- Con 82:80 se está asociando el puerto 82 del host al 80 del contenedor.
- El no estar logeado en dockerhub puede dar problemas en la descarga. Para hacer login hay que ejecutar le siguiente comando:

```
docker login
```

- ✓ ¿Qué hace la opción -p?
- ✓ ¿Qué opción debería usarse si se quiere que Docker asigne de forma aleatoria un puerto del host al puerto asignado a Apache en el contenedor?

Ejecuta también la imagen httpd:

```
docker run -d --name apache1 -P httpd
```

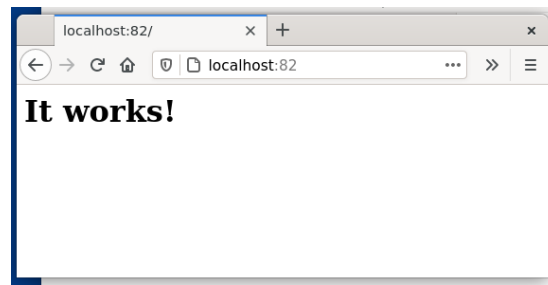
Ahora lista los contenedores que tienes arrancados:

- ✓ ¿Qué diferencia hay entre los contenedores apache2 y apache1?

Lista las imágenes para ver la diferencia de tamaño de las dos últimas imágenes descargadas.

➤ Acceder al servidor:

Una vez arrancado el contenedor apache2, se puede probar el acceso al servidor a través del navegador. En este caso hay que indicar el puerto 82 que es el que se ha establecido para la conexión al arrancarlo.



➤ Incluir una página estática

Arranca un tercer contenedor de la misma imagen pero con algún dato más:

```
docker run -d -p 81:80 --name apache3 -v /home/sonia/Desktop/pruebasDocker/ParadorGredos:/usr/local/apache2/htdocs/ httpd
```

Donde:

- -v: Proporciona un método bind. Esto quiere decir que "pasa" un directorio de nuestra máquina host al contenedor.
- /home/sonia/Desktop/pruebasDocker/ParadorGredos/ es el directorio local donde está la página web que queremos mostrar.
- /usr/local/apache2/htdocs/ directorio del contenedor donde se ubicaría la página.

Una vez arrancado el contenedor(o antes, da igual) incluye tu página web en el directorio local que has indicado al arrancar el contenedor.

En el navegador, con localhost:puerto/nombrePagina.html, se mostrará tu página.

EJERCICIO 5

En el ejercicio anterior se ha creado un contenedor que, a través de un servidor apache, sirve una página estática. Pero esa página, si el contenedor se elimina, desaparece.

Para que cada vez que se cree un contenedor de la imagen usada se muestre la página sin tener que referenciar su ubicación fuera del contenedor, vamos a crear una nueva imagen que ya la contenga:

➤ Crear un directorio para el Dockerfile y la página

Crea un directorio que contendrá el Dockerfile y la página que se va a mostrar:

```
mkdir mi_imagen
```

Mueve el contenido del directorio usado en el ejercicio 4 (en mi caso ParadorGredos) a la nueva carpeta creada.



➤ Crear Dockerfile

En la carpeta creada anteriormente, crea un documento llamado Dockerfile que tendrá el siguiente contenido:

```
FROM httpd
COPY index.html /usr/local/apache2/htdocs/index.html
COPY img /usr/local/apache2/htdocs/img
COPY scripts /usr/local/apache2/htdocs/scripts
COPY style /usr/local/apache2/htdocs/style
```

- ✓ En el contenido del Dockerfile hay dos instrucciones, FROM y COPY. ¿Qué hace cada una de ellas?

➤ Generar imagen

Ya se ha visto que con commit se puede crear una imagen a partir de un contenedor, pero en este caso se quiere generar una imagen a través del fichero Dockerfile. Para ello se utiliza el comando build visto en el punto 3:

```
sudo docker build -t sonia/mi_imagen .
```

- ✓ ¿Qué indica la opción -t?
- ✓ ¿El punto final es obligatorio? ¿Por qué? ¿Se puede sustituir por algún otro dato?

Una vez creada la imagen lista las imágenes que tienes y comprueba los datos de la que acabas de crear.

```
root@ubuntuserverSonia: /home/sonia/Desktop/pruebasDocker/ParadorGredos
File Edit View Search Terminal Help
root@ubuntuserverSonia:/home/sonia/Desktop/pruebasDocker/ParadorGredos# nano Dockerfile
root@ubuntuserverSonia:/home/sonia/Desktop/pruebasDocker/ParadorGredos# ls
Dockerfile  index.html  img         scripts     style
root@ubuntuserverSonia:/home/sonia/Desktop/pruebasDocker/ParadorGredos# sudo docker build -t sonia/mi_imagen .
Sending build context to Docker daemon  22.53kB
Step 1/5 : FROM httpd
--> 0a30f4c29d25
Step 2/5 : COPY index.html /usr/local/apache2/htdocs/index.html
--> a52822ffe767
Step 3/5 : COPY img /usr/local/apache2/htdocs/img
--> 620db951d9f1
Step 4/5 : COPY scripts /usr/local/apache2/htdocs/scripts
--> 46d27630acd9
Step 5/5 : COPY style /usr/local/apache2/htdocs/style
--> 05874ec01615
Successfully built 05874ec01615
Successfully tagged sonia/mi_imagen:latest
root@ubuntuserverSonia:/home/sonia/Desktop/pruebasDocker/ParadorGredos#
```

➤ Arrancar un contenedor

Arranca un contenedor a partir de la imagen que acabas de crear. Ahora ya no es necesario indicarle un directorio externo al contenedor donde esté alojada la página.

Si todo ha ido bien podrás acceder a tu página estática a través del navegador.