

El objetivo de esta tarea es la virtualización de un servicio, en este caso vamos a virtualizar un servidor moodle mediante el uso de contenedores.

He elegido moodle porque es una plataforma que todos conocéis así que no hace falta que os explique en qué consiste este servicio, podría haber escogido cualquiera de los servicios que hemos visto a lo largo del curso.

### Instalación de Docker en Linux

Docker funciona en la mayoría de las distribuciones Linux. Dependiendo de la distribución los pasos a seguir para la instalación de los paquetes son distintos. En todos los casos habrá que actualizar los repositorios, instalar los certificados de confianza de Docker, añadir el repositorio oficial, actualizarlo y por fin instalar el paquete.

Por todo ello lo más cómodo es ejecutar la instalación automática que se encuentra en <https://get.docker.com> y que ejecuta todos esos comandos por nosotros.

Para ello ejecutamos el siguiente comando:

```
# curl -fsSL https://get.docker.com/ | sh
```

### Primeros pasos con Docker

Una vez instalado docker en nuestro equipo ya estamos en condiciones de ejecutar la primera acción.

Para trabajar con docker vamos a abrir el Shell de comandos y a cada instrucción de docker le antepondremos el comando docker. Para poder ejecutar las instrucciones de Docker necesitamos tener privilegios de administrador, por ello también antepondremos el comando sudo a cada comando de docker.

Si deseamos ejecutar Docker con un usuario sin privilegios debemos añadir dicho usuario al grupo docker. En nuestro caso se trata del usuario ser y lo hacemos con el lanzamiento del siguiente comando:

```
usermod -aG docker ser
```

Una vez añadido, reiniciamos el demonio de Docker con

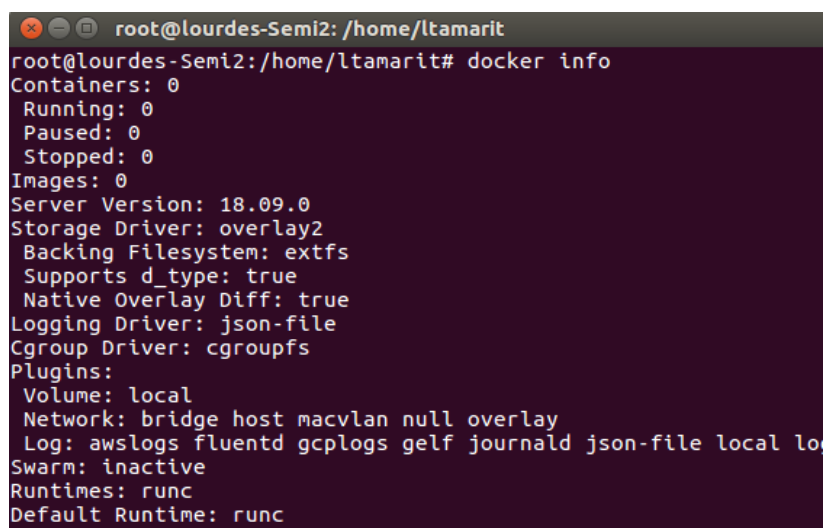
```
/etc/init.d/docker restart
```

para que los cambios surtan efecto en nuestro sistema, o bien cerramos sesión y volvemos a entrar

Para ver que está todo funcionando podemos empezar por comprobar la información del servidor docker mediante la acción info:

```
docker info
```

La salida del comando nos muestra, entre otras cosas, la versión del servidor.



```
root@lourdes-Semi2: /home/ltamarit
root@lourdes-Semi2:/home/ltamarit# docker info
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 18.09.0
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 lo
Swarm: inactive
Runtimes: runc
Default Runtime: runc
```

Podemos visualizar la lista de los comandos de docker mediante la ayuda:

```
docker help
```

Para trabajar con contenedores es necesario conocer el concepto de imágenes. Un contenedor depende de una imagen para funcionar. Cuando se crea un contenedor se basa en el contenido de una imagen, como si fuera una plantilla base. Los cambios y tareas que se realicen en el contenedor solo se verán reflejados en él mismo, y no en la imagen original.

Varios contenedores ejecutándose simultáneamente pueden estar basados en la misma imagen. Un ejemplo podría ser una imagen de Debian. Podríamos crear un contenedor sobre esa imagen para poder virtualizar ese sistema operativo y probarlo y añadir servicios. Podríamos crear otro contenedor distinto sobre esa misma imagen de manera que tendríamos otro Debian nuevo sobre el que poder añadir programas y servicios distintos.

Podríamos incluso descargar una imagen en la que ya está instalado, configurado y listo para utilizar un servidor con un Apache, o un Moodle o cualquier otro programa o servicio que se nos ocurra sin importarnos siquiera sobre qué sistema operativo corre.

Existen multitud de imágenes listas para ser descargadas en el Hub de docker (<https://hub.docker.com/>). Este es un repositorio donde es posible buscar imágenes, ver cómo utilizarlas y obtener información de ellas. Podemos acceder a este repositorio desde la web y copiar el comando necesario para descargar la imagen o directamente buscar la imagen y descargarla desde el Shell de docker.

En primer lugar podemos comprobar qué imágenes tenemos descargadas:

```
docker images
```

Obviamente si acabamos de instalar docker no nos aparecerá ninguna imagen descargada en nuestro equipo.

Es interesante comprobar la información que nos muestra, como el ID de imagen y el tamaño de la misma.

Para buscar imágenes del repositorio oficial lo podemos hacer de 2 maneras. La primera consiste en acceder a la web del Hub de Docker (<https://hub.docker.com/>) y ahí buscar la imagen deseada. Nos dirá el nombre de la misma y el comando (pull) a ejecutar para la descarga.

La otra opción es la de buscarla directamente desde la línea de mandatos mediante el comando search. En este tutorial vamos a buscar las imágenes que aparezcan con el nombre moodle:

```
docker search --limit 5 moodle
```

Le estamos indicando que sólo nos muestre los 5 primeros resultados de la búsqueda.

```
root@lourdes-Semi2:/home/ltamarit# docker search --limit 5 moodle
NAME                DESCRIPTION                                     STARS
jauer/moodle        Moodle LMS container for use with a external... 49
jhardison/moodle     Includes latest Ubuntu 18.04, PHP7, and Mood... 30
bitnami/moodle       Bitnami Docker Image for Moodle                 21
up2university/moodle Moodle image for Up2U                           4
up2university/moodle-mysql MySQL image for Up2U Moodle                     2
root@lourdes-Semi2:/home/ltamarit#
```

Una vez localizada la imagen que deseamos, sólo falta descargarla mediante el comando pull. En nuestro caso el nombre es *jhardison/moodle*:

```
docker pull jhardison/moodle
```

Comprobamos que la tenemos descargada:

```
docker images
```

```
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
jhardison/moodle     latest       5ec97634d27f     4 days ago      639MB
root@lourdes-Semi2:/home/ltamarit#
```

## Contenedores en Docker

Una vez descargada la imagen es hora de crear nuestro contenedor.

Para crear un contenedor tenemos 2 comandos. El comando create crea un contenedor a partir de una imagen pero no lo ejecuta. El comando run crea y ejecuta un contenedor.

La sintaxis del comando create es la siguiente:

```
docker create --[opciones] imagen [comando]
```

En caso de que la imagen no estuviera descargada, el comando create la descargará. Al contenedor creado se le asignará un identificador de 64 caracteres (se suelen mostrar 12) que nos mostrará por pantalla y que será necesario para futuras operaciones que hagamos con él.

Como ejemplo:

```
docker run -ti debian cat/etc/debian_version
```

Este comando creará un contenedor y lo ejecutará a partir de una imagen llamada debian. Si la imagen no está descargada, automáticamente docker la descargará del repositorio.

Una vez creado el contenedor se ejecutará dentro del mismo la orden cat/etc/debian\_version. Cuando este comando termina su ejecución, visualiza la versión de la distribución y el contenedor se detiene. Las opciones -ti indican que se ha de iniciar el contenedor con la posibilidad de acceder al terminal (-t) y que se ha de iniciar el contenedor en modo interactivo (-i).

En nuestro caso vamos a crear dos contenedores, para ello miramos las instrucciones en el docker hub (Apartado Usage) y en nuestro caso vamos a modificar un parámetro para adecuar el contenedor a nuestro escenario:

- `docker run -d --name DB -p 3306:3306 -e MYSQL_DATABASE=moodle -e MYSQL_ROOT_PASSWORD=moodle -e MYSQL_USER=moodle -e MYSQL_PASSWORD=moodle mysql:5`
- `docker run -d -P --name moodle --link DB:DB -e MOODLE_URL=http://moodle.iesc.org:8080 -p 8080:80 jhardison/moodle`

Con el primer comando se descarga la imagen moodle ya que de esta no hemos hecho un pull y con esa imagen crea un contenedor

Con el segundo comando crea un contenedor a partir de la imagen que nos descargamos previamente llamada [jhardison/moodle](#).

Como veis podíamos habernos saltado el paso de hacer el pull de la imagen

jhardison/moodle, pero quería que conocierais los comandos para bajarte una imagen y visualizarla

Opciones importantes:

- Con la opción `--name` asignamos un nombre al contenedor. No es obligatorio, en caso de no especificarlo se le asigna un nombre autogenerado.
- Con la opción `-p` se publican los puertos especificados, de manera que le estamos diciendo que el puerto 80 del contenedor será accesible desde el puerto 8080 de nuestro equipo host.
- Con la opción `MOODLE_URL=http://moodle.iesc.org:8080` le estamos indicando cómo llamaremos al contenedor, fijaros en que he puesto un nombre que pertenece a nuestro dominio. Tendremos que dar de alta este registro en nuestro servidor DNS o en el caso de que no tengamos ninguno funcionando darlo de alta en el archivo hosts como siempre

### ¿Por qué hemos creado dos contenedores?

Moodle es un servicio que necesita una base de datos para almacenar la información, es por ello que el contenedor de moodle hace uso de otro contenedor de base de datos mysql concretamente. En este ejemplo al contenedor de moodle se le ha llamado moodle y al contenedor de mysql se le ha llamado DB

El orden en el que se crean los contenedores es importante, primero debemos crear el de mysql y después el de moodle ya que el segundo hace uso del primero. Si ejecutamos los comandos anteriores al revés nos dará un error.

### Ejecución de contenedores

Una vez creado el contenedor hemos de ponerlo en ejecución. Como lo hemos creado con el comando `run`, el mismo mandato además de crearlo lo ha puesto en ejecución.

Si hubieramos utilizado `create` habríamos de iniciarlo mediante el comando `start`.

Para iniciar el contenedor es necesario indicarle o bien el nombre que le hemos asignado o bien el ID que nos mostró al crearlo. Si optamos por especificar el identificador es suficiente con indicarle los cuatro primeros caracteres y no los 12.

```
root@lourdes-Semi2:/home/ltamarit# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
e21d8df0a76a	jhardison/moodle	"/etc/apache2/foregr..."	2 minutes ago	Up 2 minutes	0.0.0.0:8080->80/tcp, 0.0.0.0:3306->3306/tcp
85dab8dbd818	mysql:5	"docker-entrypoint.s..."	2 minutes ago	Up 2 minutes	0.0.0.0:3306->3306/tcp, 3306->3306/tcp

- `docker start 85da`  
que es lo mismo que `docker start DB`

- `docker start e21d`

que es lo mismo que `docker start moodle`

En nuestro caso podemos acceder directamente porque el contenedor está iniciado:

Desde el navegador `http://localhost:8080`

comenzará la configuración del servidor moodle

## Installation - Moodle 3.6.1 (Build: 20181205)

Moodle 3.6.1 (Build: 20181205)

For information about this version of Moodle, please see the online [Release Notes](#)

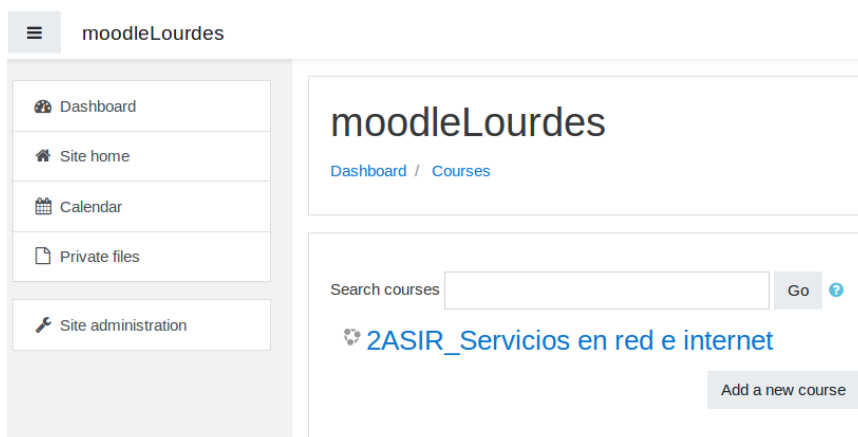
### Server checks

Name	Information	Report	Plugin	Status
unicode		must be installed and enabled		
database	mysql (5.7.24)	version 5.6 is required and you are running 5.7.24		
php		version 7.0.0 is required and you are running 7.2.10.0.0.18.04.1		
pcreunicode		should be installed and enabled for best results		
php_extension	iconv	must be installed and enabled		
php_extension	mbstring	should be installed and enabled for best results		
php_extension	curl	must be installed and enabled		
php_extension	openssl	must be installed and enabled		
php_extension	tokenizer	should be installed and enabled for best results		

Pulsamos en continuar (hay que ser pacientes tarda un rato)

Installation
System
Success
antivirus_clamav
Success
availability_completion
Success
availability_date
Success
availability_grade
Success
availability_group
Success
availability_grouping

Una vez acabemos de configurar nombre de usuario, contraseña, correo.. ya tendremos un servidor de moodle funcionando, creamos ahora un curso de prueba para comprobar que todo va según lo esperado.



Los contenedores se crean en una red por defecto 172.17.0.0/16 asignándoles una IP automática en ese rango. Como no la sabemos podemos inspeccionar el contenedor y ver qué IP privada le ha asignado de manera automática Docker:

```
docker inspect moodle
```

Para ver solo la información que nos interesa (en Linux):

```
docker inspect moodle | grep "IPAddress"
```

Un mandato interesante de docker es el que nos da la posibilidad de ejecutar un comando dentro del sistema operativo que está corriendo en el contenedor que se está ejecutando.

En nuestro ejemplo vamos a ver la versión de Linux (uname -a) que corre bajo nuestro moodle con el comando exec:

```
docker exec moodle uname -a
```

Para comprobar los contenedores que tenemos creados ejecutamos el comando ps:

```
docker ps
```

Este comando nos muestra los contenedores que están en ejecución. Para visualizar también aquellos que están creados pero no ejecutándose tecleamos:

```
docker ps -a
```

Podemos detener los contenedores que están ejecutándose mediante el comando stop:

```
docker stop moodle
```

Y volver a iniciarlos con el comando start.

```
docker start moodle
```

Existe una herramienta interesante que nos ayudará a controlar el espacio de disco que

está siendo utilizado por los elementos de Docker. Es el comando siguiente:

```
docker system df
```

```
root@lourdes-Semi2:/home/ltamarit# docker system df
TYPE                TOTAL                ACTIVE              SIZE                RECLAIMABLE
Images              2                   2                  1.011GB             0B (0%)
Containers          2                   1                  57.45kB             57.45kB (99%)
Local Volumes       4                   2                  495.8MB             224.3MB (45%)
Build Cache         0                   0                   0B                  0B
```

Muestra el espacio usado por Docker, detallando cuánto es ocupado por las imágenes de Docker, cuánto por los contenedores y por los volúmenes. Incluye el número total de elementos y los activos actualmente.

En la imagen se ve que tengo dos contenedores y dos imágenes

### Volver a arrancar el servicio.

Cuando apaguemos el equipo el servicio se parará y no se arranca de forma automática deberemos arrancar los dos contenedores en el orden correcto para poder trabajar con ellos:

```
ltamarit@lourdes-Semi2: ~
ltamarit@lourdes-Semi2:~$ sudo docker start DB
[sudo] password for ltamarit:
DB
ltamarit@lourdes-Semi2:~$ sudo docker start moodle
moodle
ltamarit@lourdes-Semi2:~$
```

### Eliminación de contenedores e imágenes

Para eliminar un contenedor lo primero que tengo que hacer es pararlo

```
docker stop moodle
```

Y para borrarlo utilizo el comando:

```
docker rm moodle
```

Si visualizamos los contenedores veremos que ya no está el nuestro:

```
docker ps -a
```

Con el comando rm hemos eliminado el contenedor pero no la imagen sobre la que se creó. Para eliminar la imagen de nuestro disco utilizamos el mandato rmi:



```
docker rmi jhardison/moodle
```

Comprobamos que ha desaparecido la imagen con:

```
docker images
```

### **Limpieza de elementos en Docker**

Además del borrado manual de elementos, existe otra opción que es limpiar Docker con el comando `prune`. El comando `prune` se aplica a contenedores, volúmenes, redes o imágenes y sirve para eliminar aquellos elementos que no están en uso. Por ejemplo, para eliminar las imágenes que no están siendo utilizadas por ningún contenedor activo ejecutamos:

```
docker image prune -a
```

Para eliminar todos los elementos que no estén en uso utilizamos el siguiente comando:

```
docker system prune
```

Este unifica todas las opciones `prune` de contenedores, volúmenes, redes e imágenes, borrando todas aquellas que no estén en uso.