Universidad de El Salvador

Facultad Multidisciplinaria de Occidente
Departamento de Ingeniería y Arquitectura

Asociación de Estudiantes de Ingeniería y Arquitectura
Curso: Programación 2.5

Guía de Trabajo
Docker y Docker Compose

Objetivo:

Que el/la alumn@ conozca sobre el despliegue de aplicaciones dentro de contenedores de software.

Docker

Docker es una plataforma para desarrolladores y administradores de sistemas para desarrollar, implementar y ejecutar aplicaciones con contenedores. El uso de contenedores Linux para implementar aplicaciones se denomina contenedorización. Los contenedores no son nuevos, pero su uso para implementar aplicaciones fácilmente sí lo es.

La contenedorización es cada vez más popular porque los contenedores son:

- Flexible: incluso las aplicaciones más complejas pueden contenerse.
- Ligero: los contenedores aprovechan y comparten el kernel de host.
- Intercambiable: puede implementar actualizaciones y actualizaciones sobre la marcha.
- Portátil: puedes construir localmente, implementar en la nube y ejecutar en cualquier lugar.
- Escalable: puede aumentar y distribuir automáticamente réplicas de contenedores.
- Apilable: puede apilar servicios verticalmente y sobre la marcha.

Imágenes y contenedores

Un contenedor se inicia ejecutando una imagen. Una imagen es un paquete ejecutable que incluye todo lo necesario para ejecutar una aplicación: el código, un tiempo de ejecución, bibliotecas, variables de entorno y archivos de configuración.

Un contenedor es una instancia de tiempo de ejecución de una imagen: la imagen se convierte en memoria cuando se ejecuta (es decir, una imagen con estado o un proceso de usuario).

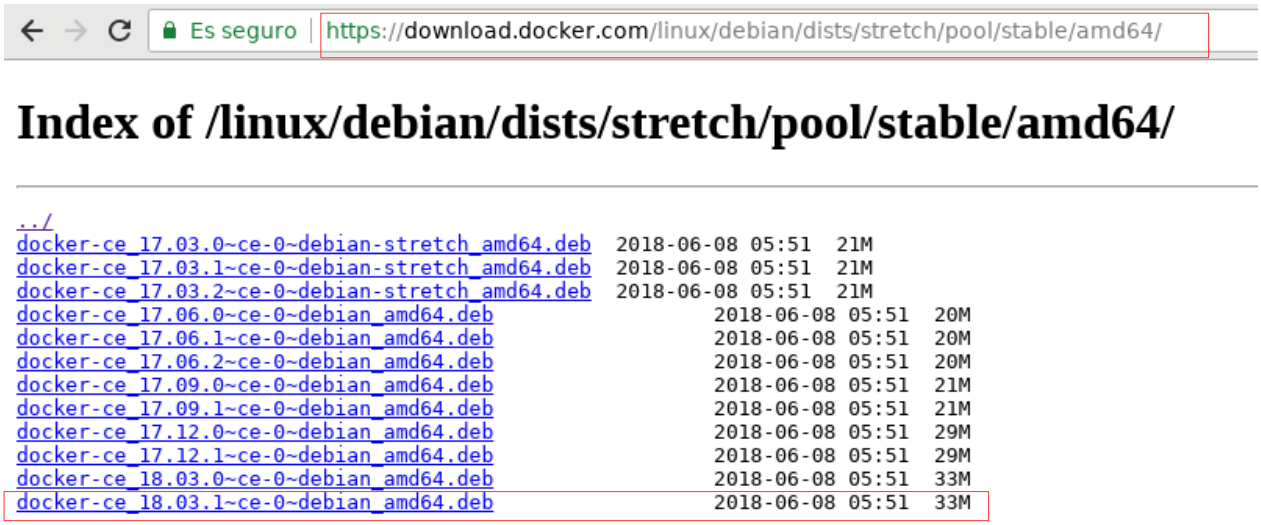
Instalación:

La instalación y la guía en general se realizara en base al sistema operativo Debian Stretch , tomarlo en cuenta si utilizará otra distribución de Linux.

En la documentación oficial de docker nos proporcionan varias formas para poder instalarlo, en esta guía se instalará desde un paquete.

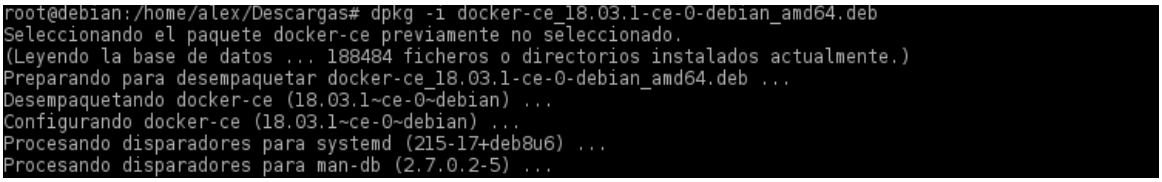
Enlace: https://docs.docker.com/install/linux/docker-ce/debian/#install-docker-ce

Para ello descargaremos el .deb, para esto debemos ir a https://download.docker.com/linux/debian/dists/, se elige la versión de Debian que tenemos, luego buscamos *pool/* y luego *stable/*, elegimos *amd64/* (64 bits) y descargamos el archivo .deb para la versión de Docker CE que desea instalar, en este caso descargaremos la versión 18.03.1.



Una vez descargado el paquete .deb procedemos a instalarlo, para ello, abrimos la consola y nos movemos a la ubicación donde se descargo y como root ejecutamos el comando

dpkg -i docker-ce_18.03.1-ce-0-debian_amd64.deb



Para verificar la instalación podemos ejecutar el comando

docker -v

Para poder usar docker con nuestro usuario normal y no solo con el root, debemos agregar nuestro usuario al grupo docker, esto se hace con el siguiente comando, cambiando “nombreUsuario” por el nombre de nuestro usuario:

```
usermod -aG docker nombreUsuario
```

Una vez hecho esto debemos cerrar sesión y volver a ingresar.

Algunos comandos de docker:

`docker` → Para ver información de los comandos que podemos ejecutar.

`docker -v` → Para ver la versión de docker instalada.

`docker info` → Para ver aún más detalles sobre la instalación de su docker.

`docker ps` → Para ver una lista de sus contenedores, también se puede usar `docker ps -a`.

`docker build` → Para crear una imagen de docker.

`docker run nombreImagen` → Para correr un contenedor a partir de una imagen.

`docker image ls` → Muestra una lista de las imágenes que tenemos en nuestro equipo.

`docker container ls` → Muestra una lista de los contenedores que tenemos corriendo. Para ver todos los contenedores se puede usar el comando `docker container ls --all`

`docker stop nombreContenedor` → Para detener un contenedor que esta corriendo.

`docker rm nombreContenedor` → Para borrar un contenedor.

Dockerfile

Dockerfile define lo que sucede en el ambiente dentro del contenedor. El acceso a recursos como las interfaces de red y unidades de disco está virtualizado dentro de este entorno, que está aislado del resto de su sistema, por lo que necesita mapear los puertos con el mundo exterior, y se debe ser específico acerca de qué archivos desea “copiar” a ese ambiente. Sin embargo, después de hacer eso, puede esperar que la compilación de su aplicación definida en el Dockerfile se comporte exactamente igual donde sea que se ejecute.

El Dockerfile es un archivo de texto con las instrucciones para nuestro contenedor, entre las instrucciones que puede llevar un Dockerfile tenemos:

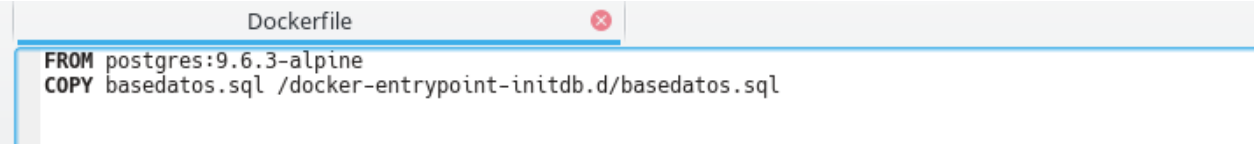
`FROM` → Que indica a partir de que imagen se construirá el contenedor.

`RUN` → Para correr comandos en el contenedor.

`ADD` o `COPY` → Para agregar o copiar archivos al contenedor.

Ejemplo:

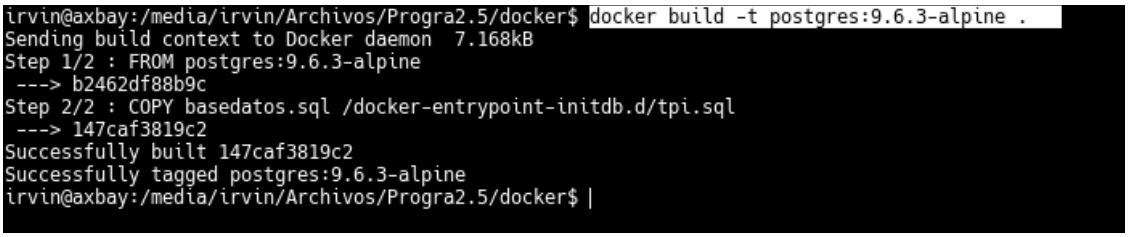
Creamos un archivo de texto con el combre *Dockerfile* en una carpeta vacía, el cual contendrá lo siguiente:



Además en la misma ubicación del Dockerfile tenemos un archivo llamado *basedatos.sql* que es un script de una base de datos.

Ejecutamos el siguiente comando en la ubicación donde esta el *Dockerfile* para construir la imagen de docker

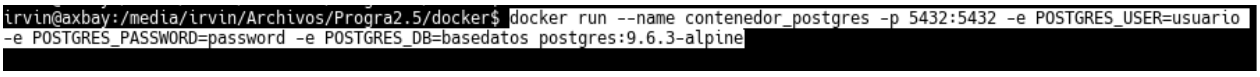
```
docker build -t postgres:9.6.3-alpine .
```



Si verificamos nuestra nueva imagen con el comando `docker image ls` obtendremos

Para correr nuestro contenedor usamos en siguiente comando:

```
docker run --name contenedor_postgres -p 5432:5432 -e POSTGRES_USER=usuario
-e POSTGRES_PASSWORD=password -e POSTGRES_DB=basedatos postgres:9.6.3-alpine
```



Donde:

--name → es para asignarle un nombre al contenedor. Si no se le establece, docker nos establecerá uno.

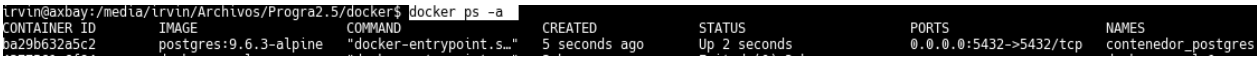
-p → es para asignarle el puerto que estará vinculado entre nuestra maquina y el contenedor.

-e → es para definir variables de entorno. En este caso como el contenedor que estamos utilizando es de una imagen de postgres se le puede definir como variables de entorno un usuario, una contraseña y una base de datos.

Por ultimo lleva el nombre de la imagen a partir de la cual correrá el contenedor.

Podemos ejecutar el siguiente comando para ver el contener corriendo:

docker ps o docker ps -a



Para detener el contenedor usamos el comando:

docker stop contenedor_postgres

Nota: el nombre con el que detuvimos el contenedor es el nombre que le establecimos en el comando para correrlo.

Si lo deseamos podemos borrar el contenedor creado con el comando:

docker rm contenedor_postgres

Así como también podemos borrar la imagen creada con el comando:

docker image rm postgres:9.6.3-alpine

Docker Compose

Compose es una herramienta para definir y ejecutar aplicaciones Docker de contenedores múltiples. Con Compose, se usa un archivo YAML para configurar los servicios de la aplicación. Luego, con un solo comando, se crea e inicia todos los servicios desde su configuración.

Compose funciona en todos los entornos: producción, puesta en escena, desarrollo y prueba. Puede obtener más información sobre cada caso en Casos de uso común .

Usar Compose es básicamente un proceso de tres pasos:

1. Define\ir el entorno de la aplicación con un Dockerfile para que pueda reproducirse en cualquier lugar.
2. Definir los servicios que componen la aplicación docker-compose.yml para que puedan ejecutarse juntos en un entorno aislado.
3. Ejecutar el comando docker-compose up y Compose inician y ejecutan toda tu aplicación.

Instalación:

Para la instalación de esta herramienta iremos al siguiente enlace https://docs.docker.com/compose/install/ y buscamos las instrucciones de instalación en Linux.

Para la instalación tenemos que tener instalado *curl* en nuestro sistema, sino esta instalado el comando para instalarlo desde consola es apt install curl

```
root@debian:/# apt install curl
```

Una vez hemos hecho esto en la documentación de docker nos proporcionan el comando para instalarlo desde consola, el cual es:

curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

```
root@debian:/# curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/loca
l/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   617    0   617    0     0    207      0 --:--:--  0:00:02 --:--:--   207
 36 10.3M   36 3892k    0     0   120k      0  0:01:28  0:00:32  0:00:56  294k
```

Luego de la instalación se le debe dar permiso de ejecución al binario, esto se hace con el siguiente comando:

chmod +x /usr/local/bin/docker-compose

```
root@debian:/# chmod +x /usr/local/bin/docker-compose
```

Para verificar la instalación usamos el comando:

docker-compose --version

```
root@debian:/# docker-compose --version
docker-compose version 1.21.2, build a133471
root@debian:/#
```

Algunos comandos de docker-compose:

docker-compose up → Inicia y ejecuta toda la aplicación.

docker-compose up --build → Construye los contenedores y luego inicia y ejecuta toda la aplicación.

docker-compose down → Detiene toda la aplicación.

Ejemplo:

Crearemos la siguiente estructura de archivos:

El archivo webapplication.war es una aplicación web y el archivo basedatos.sql es un script de una base de datos.

El contenido de los otros archivos será el siguiente:

payara → Dockerfile

Dockerfile

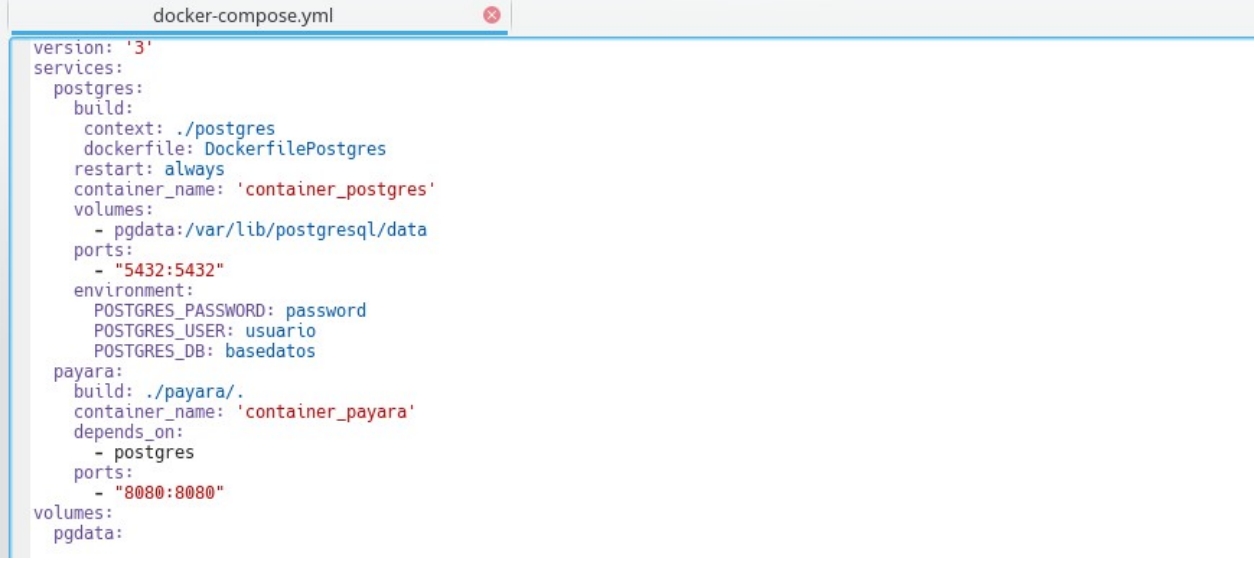
```
FROM payara/micro:4.181
ADD webapplication.war /opt/payara
ENTRYPOINT java -jar /opt/payara/payara-micro.jar --deploy /opt/payara/webapplication.war
```

postgres → DockerfilePostgres

DockerfilePostgres

```
FROM postgres:9.6.3-alpine
COPY basedatos.sql /docker-entrypoint-initdb.d/basedatos.sql
```

docker-compose.yml



Nota: el docker-compose.yml es un archivo de tipo YAML por tanto la estructura de las opciones es muy importante.

Si prestamos atención a los nombre de las opciones que lleva este archivo podemos notar que los nombre son bastante descriptivos.

`version` → Muestra la versión del archivo compose.

`services` → Dentro de ello se establecen los servicios que utilizaremos, en este caso usaremos dos, a los cuales les llamamos postgres y payara.

`build` → Es para especificar una ruta al contexto de compilación del servicio. También se le puede establecer por separado el contexto con `context` y el nombre del Dockerfile con `dockerfile`.

`container_name` → Es para especificar un nombre personalizado al contenedor, en lugar de un nombre generado.

`restart` → Para establecer la forma de reinicio del contenedor.

`volumes` → Es un mecanismo para que se guarden los datos generados y utilizados por los contenedores Docker al detenerse.

`ports` → Para definir los puertos que utilizará el contenedor.

`environment` → Para establecer las variables de entorno del contenedor.

`depends_on` → Para establecer si un servicio depende de otro de los servicio que se estan iniciando.

En la ubicación donde esta el archivo docker-compose.yml ejecutamos el siguiente comando para iniciar y ejecutar la aplicación:

docker-compose up --build

```
irvin@axbay:/media/irvin/Archivos/Progra2.5/docker-compose$ docker-compose up --build
Creating network "dockercompose_default" with the default driver
Building postgres
Step 1/2 : FROM postgres:9.6.3-alpine
 ---> fef3e7810192
Step 2/2 : COPY basedatos.sql /docker-entrypoint-initdb.d/basedatos.sql
 ---> Using cache
 ---> ae785c045acf
Successfully built ae785c045acf
Successfully tagged dockercompose_postgres:latest
Building payara
Step 1/3 : FROM payara/micro:4.181
 ---> 12ff405411b4
Step 2/3 : ADD webapplication.war /opt/payara
 ---> Using cache
 ---> 036d533ed102
Step 3/3 : ENTRYPOINT java -jar /opt/payara/payara-micro.jar --deploy /opt/payara/webapplication.war
 ---> Running in 565d8340bedb
Removing intermediate container 565d8340bedb
 ---> d246c40b958a
Successfully built d246c40b958a
Successfully tagged dockercompose_payara:latest
Creating container_postgres ... done
Creating container_payara   ... done
```

Una vez se levanten los contenedores ya estará accesible tanto la base de datos en postgres como el servidor payara.

Acceder a la url http://localhost:8080/webapplication/ una vez se han levantado los contenedores, para ver payara funcionando y mostrará lo siguiente:

Luego si deseamos detener la aplicación el comando a ejecutar sería el siguiente:

docker-compose down

```
irvin@axbay:/media/irvin/Archivos/Progra2.5/docker-compose$ docker-compose down
Stopping container_payara   ... done
Stopping container_postgres ... done
Removing container_payara   ... done
Removing container_postgres ... done
Removing network dockercompose_default
irvin@axbay:/media/irvin/Archivos/Progra2.5/docker-compose$
```