

Unidad 2 / Escenario 3

Lectura fundamental

Máquinas virtuales y Dockers para construcción de ambientes

Contenido

1 Máquinas virtuales

2 Docker

3 ¿Máquinas virtuales o contenedores?

Palabras clave: contenedor, Imagen, pull, commit.

1. Máquinas virtuales

En 1972 debido a la necesidad de simular el comportamiento de las aplicaciones de acuerdo a un sistema operativo específico surge la necesidad de configurar varios equipos en uno, por esta razón se estableció la idea de crear máquinas virtuales.

1.1. ¿Qué es una máquina virtual?

Una máquina virtual es una aplicación o archivo de computador que proporciona las mismas funciones que un equipo real, con las limitaciones de estar alojado dentro de otro host, es decir permite la instalación y administración de sistemas operativos mediante la simulación de los mismos dentro de una máquina física.

Como vemos en la figura 1, cada máquina virtual (VM) tiene un sistema operativo específico en el cual correrá una aplicación de acuerdo con las especificaciones del sistema operativo que requiera, este a su vez estará soportado por el hipervisor, software que ejecuta directamente al hardware de la computadora que aloja la máquina virtual, al igual que la infraestructura de la computadora en sí. (Docker, NR)

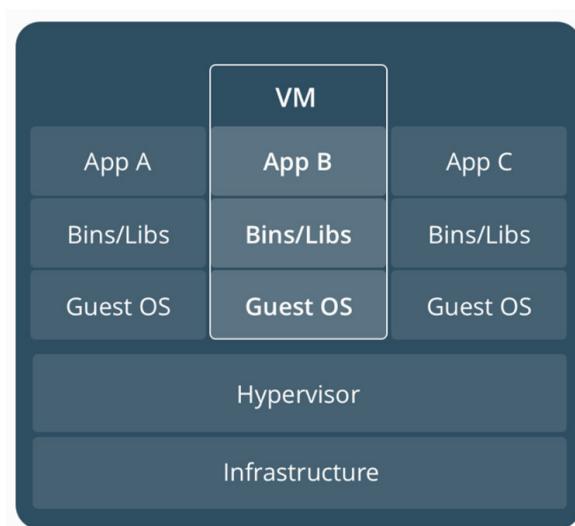


Figura 1. Estructura máquina virtual

Fuente: elaboración propia

1.2. Características

Dentro de las principales características que podemos observar de las máquinas virtuales, encontramos: (Valdés, 2016)

- **Particionamiento:** gracias a las máquinas virtuales es posible realizar particiones de nuestro equipo físico para ejecutar varias máquinas virtuales a la vez, con la desventaja de tener límites de recursos y altos consumos de los mismos.

¿Sabía qué...?



IBM fue el primero que lanzó una máquina virtual con su sistema VM/370 con el fin de simular varios sistemas operativos en una sola máquina.

- **Aislamiento:** las máquinas virtuales aislan las fallas del sistema y ofrecen seguridad a nivel de hardware.
- **Encapsulación:** debido a que garantiza el guardado del estado completo de un ordenador virtual en archivos, la gestión de máquinas virtuales permite que se muevan y copien máquinas virtuales con la misma facilidad que cualquier archivo.
- **Independencia:** al ser una máquina virtual un sistema aislado y encapsulado es posible realizar migraciones a cualquier servidor físico, liberando de responsabilidades y generando libertad.

Para la generación y administración de máquinas virtuales encontramos distintas herramientas, las cuales son intuitivas y fáciles de utilizar, dentro de las más destacadas están:

- VirtualBox (s.a.) <https://www.virtualbox.org/wiki/Virtualbox>
- VMWare (s.a.) vmware.com
- QEMU (s.a) qemu.org
- Parallels (s.a.) parallels.com

1.3. Tipos de máquinas virtuales

En el mercado podemos encontrar dos tipos de máquinas virtuales, las cuales se presentan a continuación

1.3.1. Máquinas virtuales de sistema

Este tipo de máquina virtual es muy útil para probar distintos sistemas operativos y aplicaciones sin tener que crear nuevas particiones y sectores de arranque para instalarlas; mediante la virtualización es posible usar aplicaciones para simular varios sistemas operativos, es decir puedes tener una computadora Windows y al mismo tiempo tener instalada alguna versión de Linux para probar otras aplicaciones que no puedes instalar en Windows o viceversa. (Kin, 2011)

1.3.2. Máquinas virtuales de proceso

Las máquinas virtuales de proceso están diseñadas para ejecutar un único proceso dentro de un sistema operativo, el cual se ejecuta cuando se ha indicado, ya sea de forma manual o automática, de igual forma se detiene cuando el proceso es terminado. El objetivo es proporcionar un entorno de ejecución independiente del sistema operativo ocultando los detalles de la plataforma subyacente y permitiendo que un programa se ejecute de la misma forma en cualquier plataforma; actualmente una de las máquinas virtuales de proceso más conocida es la de JAVA. (Ángel, 2011)

2. Docker

Docker es un proyecto lanzado en el 2013 que de forma flexible ayuda al despliegue de aplicaciones dentro de contenedores de software, dentro de sus características más importantes se incluye el hecho de que no requiere de utilizar un sistema operativo independiente, lo que lo hace más ligero y rápido, para esto, cuenta con funcionalidades Kernel de Linux como se observa en la figura 3; además de esto Docker posee una manera característica para hacer uso de los elementos de tu equipo, sin necesidad de mayor configuración.

Este software es característico, gracias a la manera como utiliza los recursos del Host, o de tu equipo, minimiza la duplicidad de datos permitiendo la instalación y configuración de forma rápida y sencilla de paquetes de datos llamados contenedores de software.

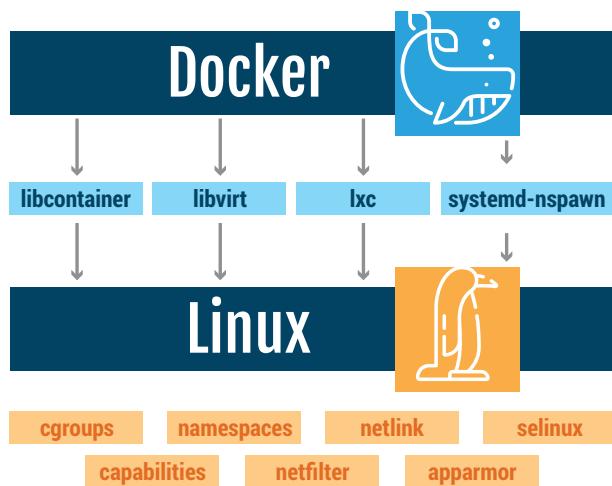


Figura 2. Organización máquina virtual

Fuente: elaboración propia

2.1. ¿Qué son los contenedores de software?

Un contenedor hace referencia a un paquete de software ejecutable y liviano que incluye todo lo que necesitas para correr el sistema que deseas administrar, es decir, siempre que estás desarrollando, es común encontrar el despliegue de aplicaciones para ser probados en diferentes dispositivos, anteriormente era necesario establecer la configuración de forma manual en algunas ocasiones utilizando máquinas virtuales, para manejar diferentes versiones de herramientas necesarias dentro del software.

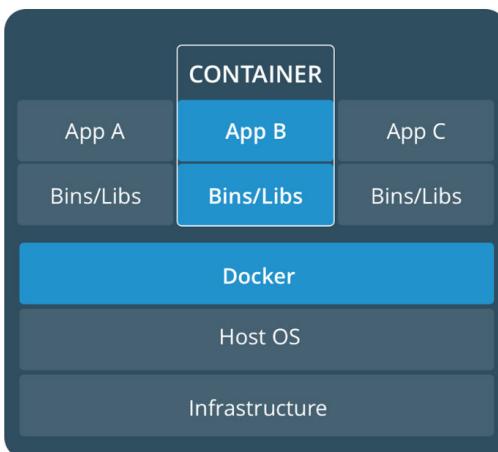


Figura 3. Contenedor software ejecutable

Fuente: elaboración propia

Como podemos observar en la figura 3, la flexibilidad de los contenedores es uno de los parámetros a resaltar, debido a que en un contenedor puede ser utilizado para administrar aplicaciones, corriendo en sistemas como Debian, Ubuntu o Alpine, para despliegue dentro de diferentes desarrolladores, sin importar el sistema operativo o la configuración que su equipo tenga, a todos correr bajo el kernel de Linux podemos tener aplicaciones diferentes en una misma maquina a través de diferentes contenedores. (Docker, s.f.)

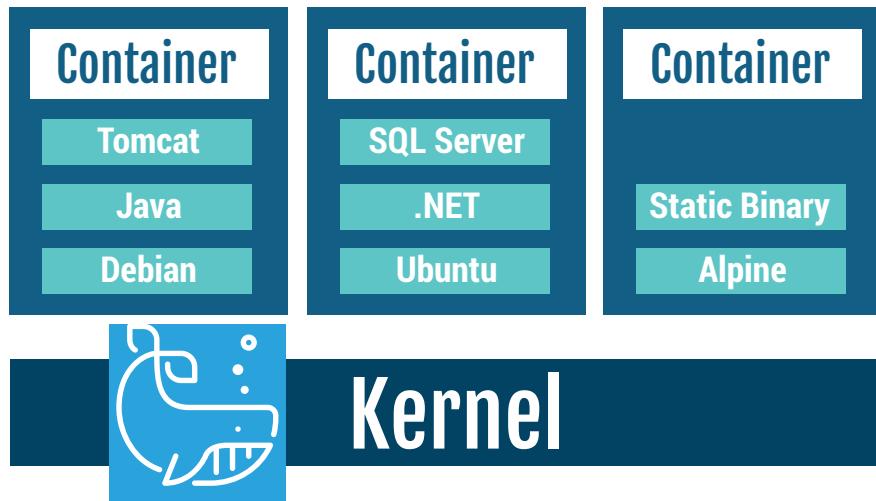


Figura 4. Usos de contenedores

Fuente: elaboración propia

2.2. Características

Los contenedores presentan tres características especiales que los han llevado a posicionarse como mejor herramienta para el despliegue de aplicaciones, estas son:

2.2.1. Ligero

Los contenedores Dockers al compartir los kernel del sistema que los aloja, los hace muchísimo más rápidos que las demás soluciones debido a que no es necesaria la instalación de sistemas operativos, disminuyen el consumo de RAM y de máquina, al igual que el espacio que ocupa en el computador.

2.2.2. Estándar

Docker se basa en estándares abiertos y se ejecutan en todas las principales distribuciones de Linux, Microsoft Windows y en cualquier infraestructura incluyendo VMs, bare-metal y en la nube. (Docker, 2017)

2.2.3. Seguridad

Los contenedores Docker aíslan aplicaciones entre sí y de la infraestructura subyacente, esto aumenta la seguridad en el desarrollo de software, debido a que permite un aislamiento entre todas las aplicaciones al igual que un aislamiento del host. (Fuente, 2015)

2.3. Orquestación de contenedores

Luego de conocer el término contenedores y después de haber usado estos mismos, es normal encontrar en el mundo del desarrollo de software un gran número de contenedores en cada proyecto, debido a que cada instancia por gestionar son alojadas en múltiples contenedores, para tareas de operatividad, gestión de bases de datos y ambientes de distribución.

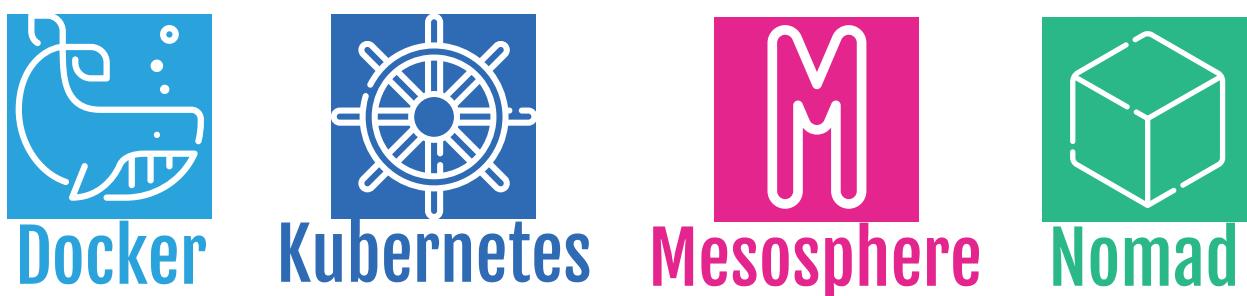


Figura 5. Soluciones para Orquestación de Contenedores

Fuente: elaboración propia

Para la administración de numerosos contenedores, es necesario utilizar herramientas de orquestación de los mismos. Dentro de las más conocidas son destacadas las soluciones que se acaban de presentar (Figura 5), cada uno de ellos comparten las siguientes características:

- **Service Discovery:** cada orquestador debe permitir la visualización y comunicación entre varios contenedores sin importar en qué máquina estén ubicados, por lo que los contenedores deben estar conectados sin importar que cada contenedor esté ubicado en una instancia diferente.
- Scheduling: el orquestador, de manera automática genera la distribución del host y el levantamiento de cada contenedor administrando su inicio y su parada.
- **Health Monitoring:** un orquestador tiene la capacidad de auto recuperar en caso de tener fallas, pausas, etc. la “salud de los contenedores”, es decir, en caso de que un contenedor falle, el orquestador debe buscar la solución y encargarse de reactivarlo.

¿Sabía qué...?



Docker cambió la manera de virtualizar tus máquinas, simplificando la manera de hacerlo y reduciendo considerablemente la cantidad de recursos necesarios.

- **Horizontal Scaling:** la capacidad de escalar de manera horizontal es una característica de estos orquestadores de contenedores, debido a que permiten la creación de un buen número de contenedores de forma automática y asociarlos entre sí.

A partir de la versión de Docker 2.1 el modo Swarm fue integrado dentro de la plataforma, este es un sistema nativo de encapsulamiento que permite agrupar dentro de un host de Docker, varios host de Docker.

Debido a que Swarm hace parte del API estándar de Docker, cualquier herramienta que de por sí pueda ser aplicada en contenedores Docker pueden ser utilizadas en Swarm como es el caso de (Docker, s.f)

- **Dokku** (s.a, dokku.io, s.f.)
- **Docker Compose** (Docker, s.f.)
- **Docker Machine** (Docker, s.f.)
- **Jenkins** (s.a, jenkins.io, s.f.)

3. ¿Máquinas virtuales o contenedores?

En conclusión las máquinas virtuales y los contenedores tienen grandes diferencias; la más destacada es, como ya se ha dicho, el hecho de que las primeras tienen un sistema operativo propio, como vimos en la distribución de los niveles de la máquina (Figura 2) el nivel OS no será compartida, cada aplicación está alojada dentro de una de las máquinas y cada máquina tiene un sistema operativo, lo que la hace mucho más robusta. Esto no ocurre con el uso de contenedores, ya que el sistema operativo puede ser compartido dentro de varios contenedores (Figura 3). Esto, en el caso de las máquinas virtuales, da como consecuencia un uso excesivo de recursos tecnológicos, además que dilata el tiempo de configuración de la máquina, debido a que la instalación y configuración de los SO (Sistemas operativos) tienen una duración bastante considerable, por este motivo los contenedores son una forma bastante eficaz para el despliegue y distribución de aplicaciones y software, facilitando la operación y manipulación de los mismos por parte del equipo de desarrollo.

3.1. Configuración Docker a través de comandos

3.1.1. Listar Elementos

Habiendo instalado Docker, implementarlo no tiene gran dificultad, mediante el uso de terminal de comandos es posible administrar no solo los contenedores, de igual forma las imágenes, es importante que cada comando sea digitado en seguida de la palabra docker como comando identificador para la administración de Docker.

3.1.1.1 Listar Imágenes

Mediante el comando docker images, es posible observar qué imágenes están disponibles de manera local, mostrando información como nombre, tag, id, fecha de creación y espacio en disco, como se puede observar en la figura 5.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
php	latest	74deaab6a609	27 hours ago	370MB
eduardoliveros/web	latest	31d97a406c4e	10 days ago	230MB
mytop	latest	1b5971a20796	10 days ago	421MB
mytop	pruebatag	1b5971a20796	10 days ago	421MB
<none>	<none>	f13ac0616b7a	10 days ago	120MB
eduardoliveros/ubuntu	latest	ccc7a11d65b1	13 days ago	120MB
ubuntu	latest	ccc7a11d65b1	13 days ago	120MB
ubuntu	14.04	c69811d4e993	13 days ago	188MB
eduardoliveros/get-started	part1	ecbe05d75fde	2 weeks ago	194MB
friendlyhello	latest	ecbe05d75fde	2 weeks ago	194MB
myjenk	latest	ecbe05d75fde	2 weeks ago	194MB
php	7.1-apache	a47f065418d5	2 weeks ago	391MB
php	<none>	84c6a05223c5	2 weeks ago	370MB
nginx	latest	b8efb18f159b	4 weeks ago	107MB
python	2.7-slim	451c85955bc2	4 weeks ago	182MB
ubuntu	<none>	14f60031763d	5 weeks ago	120MB
hello-world	latest	1815c82652c0	2 months ago	1.84kB

Figura 5. Pantallazo de cómo listar imágenes

Fuente: elaboración propia

3.1.1.2 Listar contenedores

Mediante el comando docker ps es posible observar qué contenedores están disponibles de manera local, mostrando información como id, imagen referente, comando con el que se llamó, fecha de creación, estado, puerto en el que trabaja y nombre, este nombre puede ser asignado, de no hacerlo se generará automáticamente, como se puede observar en la figura 6.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d3dc12ed3043	php:7.1-apache	"docker-php-entryp..."	27 hours ago	Up 27 hours	0.0.0.0:4000->80/tcp	cranky_saha
fe2d8ac1d968	php:7.1-apache	"docker-php-entryp..."	27 hours ago	Exited (0) 27 hours ago		stoic_bell
4e85fa0d1e7c	php:latest	"docker-php-entryp..."	27 hours ago	Exited (1) 27 hours ago		laughing_khorana
7161bb5c5cc8	ubuntu	"bash"	27 hours ago	Exited (0) 27 hours ago		nostalgic_panini
3de6df2f79e97	ubuntu	"/bin/bash"	27 hours ago	Exited (0) 27 hours ago		condescending_goodal
1						2017-08-14.097
50d0bfc7be1d	ubuntu	"/bin/bash"	27 hours ago	Exited (0) 27 hours ago		pensive_franklin
672f1a535849	nginx	"nginx -g 'daemon ..."	27 hours ago	Exited (0) 27 hours ago		ubuntu
77c0c565cb27	ubuntu	"bash"	27 hours ago	Exited (0) 27 hours ago		affectionate_joliot

Figura 6. Pantallazo de cómo listar contenedores

Fuente: elaboración propia

3.2. Administración de Imágenes

3.2.1. Crear Imágenes

Las imágenes se construyen con una única finalidad, por ejemplo las imágenes que parten de ubuntu son construidas generalmente con la finalidad de usar bash; para poder administrar los archivos del servidor, las imágenes de proceso como nginx o apache se construyen para ejecutar operaciones o procesos.

Una de las formas en las que se pueden crear imágenes de forma local es a través de imágenes existentes de <https://hub.docker.com/explore/>. En este espacio es posible encontrar imágenes pre establecidas con un buen número de productos, lenguajes y funcionalidades, de esta forma solo es realizar un proceso de “pull” del nombre de la imagen y podremos utilizarlo de forma local, como es el caso de la figura 7, del cual tendremos una imagen para el manejo de notificaciones en dispositivos iOS.

```
[Eduardos-MacBook-Pro:docker-apns-send eduardo$ docker pull cloudspace/docker-ios-notification
Using default tag: latest
latest: Pulling from cloudspace/docker-ios-notification
a3ed95caeb02: Already exists
068056c10a94: Already exists
Digest: sha256:d2807f0f962df6eadf3edb075087479d48c3b445f36e611b4a25eefdc0310868
Status: Image is up to date for cloudspace/docker-ios-notification:latest
```

Figura 7. Pantallazo del manejo de notificaciones con comando Pull

Fuente: elaboración propia

Una forma de crear imágenes es a partir de un contenedor local, como se muestra en la figura 8, el comando docker commit d3dc12ed3043 ubuntudep crea una imagen de nombre ubuntudep a partir del contenedor de Id d3dc12ed3043, el cual era un contenedor basado en Ubuntu, al que se le instalaron dependencias, de esta forma, la imagen es creada de forma local y puede ser distribuida posteriormente.

Eduardos-MacBook-Pro:~ eduardo\$ docker commit d3dc12ed3043 ubuntudep				
sha256:697c9a3ac08bd2e2a3b8aa371f2d7c7b7fe6356db59d7ae122c290f988169489				
Eduardos-MacBook-Pro:~ eduardo\$ docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntudep	latest	697c9a3ac08b	13 seconds ago	465MB
php	latest	74deaab6a609	28 hours ago	370MB
eduardoliveros/web	latest	31d97a406c4e	10 days ago	230MB
myhtop	latest	1b5971a20796	10 days ago	421MB
myhtop	pruebatag	1b5971a20796	10 days ago	421MB
<none>	<none>	f13ac0616b7a	10 days ago	120MB
eduardoliveros/ubuntu	latest	ccc7a11d65b1	13 days ago	120MB
ubuntu	latest	ccc7a11d65b1	13 days ago	120MB
ubuntu	14.04	c69811d4e993	13 days ago	188MB
eduardoliveros/get-started	part1	ecbe05d75fde	2 weeks ago	194MB
friendlyhello	latest	ecbe05d75fde	2 weeks ago	194MB
myjenk	latest	ecbe05d75fde	2 weeks ago	194MB
php	7.1-apache	a47f065418d5	2 weeks ago	391MB
php	<none>	84c6a05223c5	2 weeks ago	370MB

Figura 8. Pantallazo de cómo crear imágenes con Commit Imágenes

Fuente: elaboración propia

3.2.2. Eliminar Imágenes

El comando docker rmi elimina las imágenes de forma local, es necesario incluir el identificador para que se identifique qué imagen se quiere remover.

```
Eduardos-MacBook-Pro:phpserver eduardo$ docker rmi 1815c82652c0
Untagged: hello-world:latest
Untagged: hello-world@sha256:f3b3b28a45160805bb16542c9531888519430e9e6d6ffc09d72261b0d26ff74f
Deleted: sha256:1815c82652c03bfd8644afda26fb184f2ed891d921b20a0703b46768f9755c57
Deleted: sha256:45761469c965421a92a69cc50e92c01e0cf94fe026cdd1233445ea00e96289a
```

Figura 9. Pantallazo de cómo eliminar imágenes

Fuente: elaboración propia

3.3. Administracion de contenedores

3.3.1. Crear contenedores

El comando run permite ejecutar una imagen creada y a partir de esta crear un contenedor como se muestra en la figura 10 en el comando Docker run –it ubuntudep bash es utilizado para correr un contenedor mediante la imagen previamente creada ubuntudep usando el intérprete bash, fuera de crear el contenedor basado en Ubuntu, tenemos conexión con el mismo para modificar sus archivos directamente.

```
[Eduardos-MacBook-Pro:docker-apns-send eduardo$ docker run -it ubuntudep
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Se
t the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Se
t the 'ServerName' directive globally to suppress this message
[Thu Aug 24 19:30:38.341550 2017] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.10 (Debian) PHP/7.1.8 conf
igured -- resuming normal operations
[Thu Aug 24 19:30:38.341620 2017] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
^C[Thu Aug 24 19:30:59.136934 2017] [mpm_prefork:notice] [pid 1] AH00169: caught SIGTERM, shutting down
[Eduardos-MacBook-Pro:docker-apns-send eduardo$ docker run -it ubuntudep bash
root@fa9e9429ab1d:/var/www/html# ]
```

Figura 10. Pantallazo de Comando run

Fuente: elaboración propia

3.4. Eliminar contenedores

El comando docker rm elimina los contenedores de forma local, es necesario incluir el identificador para que se identifique que contenedor se quiere remover.

```
[Eduardos-MacBook-Pro:phpserver eduardo$ docker rm 77c0c565cb27
77c0c565cb27]
```

Figura 11. Pantallazo de cómo eliminar contenedores

Fuente: elaboración propia

3.5. Ejecutar contenedores

En caso de ejecutar un contenedor ya creado y utilizado localmente, podemos usar el comando exec como es el caso de la figura 12, el comando sudo docker exec -i -t d3dc12ed3043 /bin/bash permite ejecutar el contenedor junto con bash para poder modificar sus archivos y por ejemplo actualizar sus dependencias.

```
Eduardos-MacBook-Pro:~ eduardo$ sudo docker exec -i -t d3dc12ed3043 /bin/bash
Password: 
root@d3dc12ed3043:/var/www/html# ls
root@d3dc12ed3043:/var/www/html# apt-get update
```

Figura 12. Pantallazo de cómo ejecutar un contenedor con Docker exec

Fuente: elaboración propia

3.6. DockerFile

Este archivo contiene todas las instrucciones iniciales que el usuario desea que docker ejecute, con el fin de crear una imagen, dentro de este documento podemos establecer la imagen de la que se quiere partir, los puertos en los que será montado el proceso, configuraciones de directorio, etc.

En la página oficial de documentación de docker para crear dockerfiles, <https://docs.docker.com/engine/reference/builder/#expose>, se pueden ver en detalle cada uno de los comandos de configuración de este archivo. La figura 13 muestra una configuración básica para un servidor php que trabajará dentro del puerto 4000 en localhost.

```
FROM php
ENV APP_DIR /var/www/app
ENV APPLICATION_ENV development
RUN mkdir -p $APP_DIR
WORKDIR $APP_DIR
EXPOSE 80
VOLUME $APP_DIR
CMD ["php", "-S", "0.0.0.0:4000", "-t", "./public", "./public/index.php"]
```

Figura 13. Pantallazo de cómo crear Dockerfiles

Fuente: elaboración propia

3.7. Crear Imagen y contenedor

El comando docker build - < Dockerfile corre el contenedor con la configuración indicada dentro del archivo docker, creando así de forma local la imagen y el contenedor correspondiente.

```
Eduardos-MacBook-Pro:phpserver eduardo$ docker build - < Dockerfile
Sending build context to Docker daemon 2.048kB
Step 1/8 : FROM php
latest: Pulling from library/php
ad74af05f5a2: Already exists
a1e75557f244: Already exists
6ab4f72a86ad: Already exists
5a4d1f40c3a5: Pull complete
f9fb5f43b39b: Pull complete
648be3b739f0: Pull complete
a384d8cabcd7: Pull complete
Digest: sha256:e90add962fdd8ad341d28bdc2dd3b7a59d9cad2ec9f4e8286fc5bdaf8d4836d
Status: Downloaded newer image for php:latest
    --> 84c6a05223c5
Step 2/8 : ENV APP_DIR /var/www/app
    --> Running in 16b0a1049544
    --> 8e2e9cfb8860
Removing intermediate container 16b0a1049544
Step 3/8 : ENV APPLICATION_ENV development
    --> Running in d61eae57a550
    --> 0ba4e60f5cc4
Removing intermediate container d61eae57a550
Step 4/8 : RUN mkdir -p $APP_DIR
    --> Running in 4aa93113954b
    --> 076de6427897
Removing intermediate container 4aa93113954b
Step 5/8 : WORKDIR $APP_DIR
    --> 2b37036fbcc3
Removing intermediate container 2aac999a47aa
Step 6/8 : EXPOSE 80
    --> Running in 9322827adb07
    --> 5f71215ee809
Removing intermediate container 9322827adb07
Step 7/8 : VOLUME $APP_DIR
    --> Running in 1402cc8263a0
    --> b0c72827affd
Removing intermediate container 1402cc8263a0
Step 8/8 : CMD php -S 0.0.0.0:4000 -t ./public ./public/index.php
    --> Running in 34500977f129
    --> 74deaab6a609
Removing intermediate container 34500977f129
Successfully built 74deaab6a609
```

Figura 14. Pantallazo Log de ejecución archivo Docker

Fuente: elaboración propia

Referencias

- Ángel, E. (2011). *maquikit.blogspot*. Recuperado de: <http://maquikit.blogspot.com.co/2011/08/maquina-virtual-de-proceso.html>
- Docker. (2017). *Docker.com*. Retrieved 07 16, 2017, from Docker: <https://www.docker.com/what-container>
- Docker. (s.f.). *docs.docker*. Recuperado de: <https://docs.docker.com/get-started/part1/>
- Docker. (s.f.). *docs.docker.com*. Recuperado de: [https://docs.docker.com/swarm/overview/.](https://docs.docker.com/swarm/overview/)
- Docker. (NR, NR NR). Recuperado de: <https://docs.docker.com/get-started/part1/>
- Fuente, T. d. (2015). Recuperado de: <https://blyx.com/2015/05/22/buenas-practicas-de-seguridad-en-docker/>
- Kin (2011). *Webadicto*. Recuperado de: <http://webadicto.net/maquina-virtual-de-sistema-caracteristicas-beneficios-desventajas/>
- Kubernetes (s.f) Recuperado de: [Kubernetes.io](https://kubernetes.io).
- Mesosphere (s.) Recuperado de: [Mesosphere.com](https://mesosphere.com).
- Nomadproject (s.f) Recuperado de: [www.nomadproject.io/](https://nomadproject.io)
- Valdés, B. (2016). *osandnet*. Recuperado de: <http://www.osandnet.com/maquina-virtual-caracteristicas-tipos/>

INFORMACIÓN TÉCNICA



FACULTAD DE
**INGENIERÍA, DISEÑO
E INNOVACIÓN**

Módulo: Énfasis Profesional I (Integración continua)

Unidad 2: Integración continua y manejo de contenedores

Escenario 3: Máquinas virtuales y Dockers para
construcción de ambientes

Autor: Eduardo Enrique Oliveros Acosta

Asesor Pedagógico: Amparo Sastoque Romero

Diseñador Gráfico: Paola Andrea Melo

Asistente: Eveling Peñaranda

Este material pertenece al Politécnico Grancolombiano. Por
ende, es de uso exclusivo de las Instituciones adscritas a la Red
Illumino. Prohibida su reproducción total o parcial.