Práctica Contenedores Docker

Elaborado por: Prof. Oscar Mondragón

1. Objetivo

Comprender la instalación y uso de la plataforma de contenedores Docker.

2. Herramientas a utilizar

- Vagrant
- Docker Community Edition
- Vagrant Box Ubuntu

3. Desarrollo de la práctica

PARTE 1. Configuración de Vagrant

Esta práctica la desarrollaremos usando un Box de Ubuntu 20.04 en Vagrant. El Vagrantfile que usaremos es el siguiente (con el que venimos trabajando):

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  if Vagrant.has_plugin? "vagrant-vbguest"
    config.vbguest.no_install  = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote   = true
  end

  config.vm.define :clienteUbuntu do |clienteUbuntu|
    clienteUbuntu.vm.box = "bento/ubuntu-20.04"
    clienteUbuntu.vm.network :private_network, ip: "192.168.100.2"
    clienteUbuntu.vm.hostname = "clienteUbuntu"
  end

  config.vm.define :servidorUbuntu do |servidorUbuntu|
    servidorUbuntu.vm.box = "bento/ubuntu-20.04"
    servidorUbuntu.vm.network :private_network, ip: "192.168.100.3"
    servidorUbuntu.vm.hostname = "servidorUbuntu"
  end
end
```

PARTE 2. Instalación de Docker en Ubuntu 20.04

1. Desinstalar versiones anteriores de Docker

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

2. Configurar el repositorio

Actualizar el paquete apt e instale paquetes para permitir que apt use un repositorio a través de HTTPS

```
$ sudo apt-get update

$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

3. Agregar la clave GPG* oficial de Docker

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

* GPG es usado para cifrar y firmar digitalmente. GPG utiliza criptografía de clave pública para que los usuarios puedan comunicarse de un modo seguro.

Verifique que tiene la clave con el fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, buscando los últimos 8 caracteres del fingerprint

```
$ sudo apt-key fingerprint 0EBFCD88

pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid           [ unknown] Docker Release (CE deb) <docker@docker.com>
sub   rsa4096 2017-02-22 [S]
```

4. Agregar un repositorio stable

```
$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

5. Actualizar el paquete apt e instalar la última versión de Docker Engine

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. Verificar que Docker Engine quedó instalado correctamente corriendo la imagen de hello-world

```
sudo docker run hello-world
```

Este comando descarga una imagen de prueba y la ejecuta en un contenedor. Cuando se ejecuta el contenedor, imprime un mensaje informativo y sale.

7. Verificar que Docker esté corriendo

```
$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-04-15 16:13:34 UTC; 14min ago
```

8. Ver información de Docker

Todos los comandos de docker inician con la palabra docker. Para ver la información de docker se hace lo siguiente:

```
$ sudo docker info | more

Server:
 Containers: 3
  Running: 0
  Paused: 0
  Stopped: 3
 Images: 2
 Server Version: 19.03.8
```

Aquí se puede ver información relacionada con las imágenes que se han descargado y de los contenedores que se han creado.

PARTE 3. Descargar una imagen Docker existente y correr sus servicios

9. Verificar qué imágenes de contenedores existen en los repositorios de docker.

Por ejemplo si quisieramos saber que imágenes de contenedores centos + ssh + apache existen:

```
$ sudo docker search centos-ssh-apache
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
jdeathe/centos-ssh-apache-php       Apache PHP - CentOS.                            30                   [OK]
jdeathe/centos-ssh-apache-php-fcgi  Apache PHP-CGI (FastCGI) - CentOS.              3                    [OK]
zhangyuan82/centos-ssh-apache-php                                                   0
amirabbasi/centos-ssh-apache-php
```

10. Descargar imágenes

```
$ sudo docker pull jdeathe/centos-ssh-apache-php
```

11. Ver imágenes descargadas

```
$ sudo docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
ubuntu                              latest              4e5021d210f6        3 weeks ago         64.2MB
jdeathe/centos-ssh-apache-php       latest              c3dbfa3577c9        6 months ago        295MB
hello-world                         latest              fce289e99eb9        15 months ago       1.84kB
```

12. Ejecutar un contenedor basado en una de las imágenes descargadas

```
$ sudo docker run -d --name web1 -p 8800:80 jdeathe/centos-ssh-apache-php
420e480dd15b5defd1c8faf7f59b910d35c9462e9d63520b6492027b51731a34
```

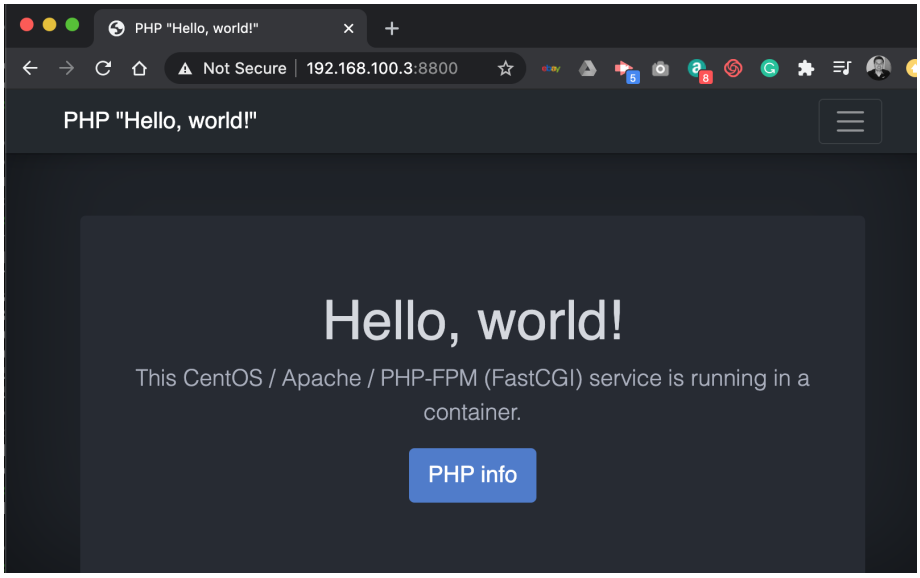
La opción –d permite correr el contenedor en background. La opción –p permite hacer un reenvío desde el puerto 80 del contenedor al 8800 del host.

13. Verificar qué contenedores están corriendo actualmente

```
$ sudo docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS
PORTS                                      NAMES
420e480dd15b        jdeathe/centos-ssh-apache-php   "/usr/bin/supervisor…"   About a minute ago   Up About a
minute (healthy)   22/tcp, 443/tcp, 8443/tcp, 0.0.0.0:8800->80/tcp   web1
```

14. Acceder los servicios del container

La imagen descargada tiene activado el servicio web, desde el browser accedemos a la dirección IP de la máquina virtual de Ubuntu (en este caso 192.168.100.3) por el puerto 8800 que fue el que se definió.



15. Remover un contenedor

Para eliminar una o más imágenes de Docker, use el comando docker container rm seguido de la ID de los contenedores que desea eliminar.

Puede obtener una lista de todos los contenedores pasando el indicador -a al comando docker container ls:

```
$ sudo docker container ls -a
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS
PORTS                                      NAMES
420e480dd15b        jdeathe/centos-ssh-apache-php   "/usr/bin/supervisor…"   4 hours ago         Up 4 hours
(healthy)   22/tcp, 443/tcp, 8443/tcp, 0.0.0.0:8800->80/tcp   web1
61b1ebd88717        ubuntu                     "bash"                   5 hours ago         Exited (0) 5 hours ago
xenodochial_volhard
c68e24de01fc        hello-world                "/hello"                 5 hours ago         Exited (0) 5 hours ago
```

Una vez que conozca el ID DE CONTENEDOR de los contenedores que desea eliminar, páselo al comando docker container rm. Si el container está corriendo, se debe detener primero. Por ejemplo:

```
$ sudo docker container stop 420e480dd15b
```

```
$ sudo docker container rm 420e480dd15b
```

Una vez eliminado puede verificar que el container ya no existe mediante el comando:

```
$ sudo docker container ls -a
```

PARTE 4. IMAGEN DOCKER PROPIA

En esta parte buscamos crear una imagen propia, con un servicio propio instalado. Para esto, descargaremos la imagen oficial de Centos.

```
$ sudo docker search centos
NAME                      DESCRIPTION                    STARS     OFFICIAL   AUTOMATED
centos                    The official build of CentOS.  5940      [OK]
ansible/centos7-ansible   Ansible on Centos7             128                  [OK]
```

16. Descargar la imagen

```
$ sudo docker pull centos
Using default tag: latest
latest: Pulling from library/centos
8a29a15cefae: Pull complete
Digest: sha256:fe8d824220415eed5477b63addf40fb06c3b049404242b31982106ac204f6700
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
```

17. Creamos un archivo Dockerfile dentro de un directorio test_docker

```
$ mkdir test_docker
$ cd test_docker
$ vim Dockerfile
```

Dockerfile

```
FROM centos
LABEL maintainer="Oscar Mondragon"
RUN yum install httpd -y
RUN echo "<h1> Bienvenidos a esta pagina </h1>" > /var/www/html/index.html
EXPOSE 80
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

18. Se ejecuta el Dockerfile para generar la imagen

```
$ sudo docker build -t omondragon/centosweb .

Sending build context to Docker daemon  24.06kB
Step 1/6 : FROM centos
 ---> 470671670cac
Step 2/6 : LABEL maintainer="Oscar Mondragon"
 ---> Running in d38b810b6f5e
Removing intermediate container d38b810b6f5e
 ---> 39739df38015
Step 3/6 : RUN yum install httpd -y
 ---> Running in 79d25b73ae3a
CentOS-8 - AppStream                            56 kB/s | 6.8 MB     02:05
CentOS-8 - Base                                 78 kB/s | 6.0 MB     01:18
CentOS-8 - Extras                              2.6 kB/s | 5.5 kB     00:02
Dependencies resolved.

Complete!
Removing intermediate container 79d25b73ae3a
 ---> 1f89aa6e565f
Step 4/6 : RUN echo "<h1> Bienvenidos a esta pagina </h1>" > /var/www/html/index.html
 ---> Running in a3cb3ccccce4
Removing intermediate container a3cb3ccccce4
 ---> 326c6c36f505
Step 5/6 : EXPOSE 80
 ---> Running in 0ab9488aa822
Removing intermediate container 0ab9488aa822
 ---> a6fc9b53924d
Step 6/6 : CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
 ---> Running in 4d4a07eda583
Removing intermediate container 4d4a07eda583
 ---> 633c655a62fb
Successfully built 633c655a62fb
Successfully tagged omondragon/centosweb:latest
```

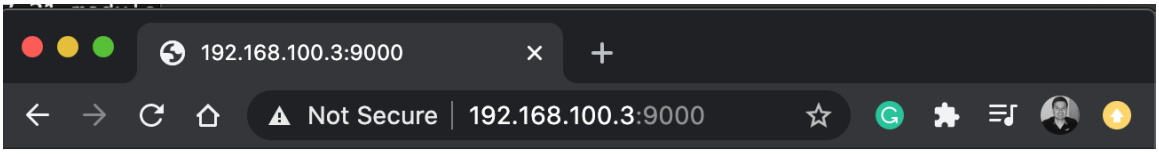
19. Se crea el contenedor

```
$ sudo docker run --name webprueba -d -p 9000:80 omondragon/centosweb

ebb43775ccb53452db74bb9128346feb7634f758bedd56427966944c4b84acd3
```

20. Se miran los contenedores creados y ejecutándose

```
$ sudo docker ps

CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS
NAMES
ebb43775ccb5        omondragon/centosweb   "/usr/sbin/httpd -D …"   About a minute ago   Up About a minute
0.0.0.0:9000->80/tcp   webprueba
```

21. Verificar el servicio funcionando



Bienvenidos a esta pagina

PARTE 5. COPIAR ARCHIVOS DESDE DIRECTORIO DEL HOST

En esta parte buscamos crear una imagen propia con la opción de copiar los archivos de una carpeta del host local a una carpeta del contenedor.

22. Crear el directorio en el host

Crear un directorio test_docker2. Dentro de test_docker2 se crea un directorio llamado voldocker en el host local en donde se van a almacenar los archivos a copiar.

```
vagrant@machine1:~/test_docker2$ ls
voldocker
```

Dentro del directorio voldocker crear una página llamada index.html con un vínculo a otra llamada pagina1.html

```
vagrant@machine1:~/test_docker2/voldocker$ ls
index.html  pagina1.html
```

index.html

```
<h1>Prueba directorios</h1>
<a href="pagina1.html">Ir a pagina1</a>
```

23. Se crea el archivo Dockerfile en el directorio test_docker2 con base en el cuál se creará la imagen.

```
vagrant@machine1:~/test_docker2$ ls
Dockerfile  voldocker
```

Dockerfile

```
FROM centos
LABEL maintainer="Oscar Mondragon"
RUN yum install httpd -y
COPY voldocker/ /var/www/html/
RUN chmod -R 777 /var/www/html
EXPOSE 80
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

24. Se crea la imagen con el nombre que quieran darle. En este caso se le dio el nombre omondragon/testdir

```
vagrant@machine1:~/test_docker2$ sudo docker build -t omondragon/testdir .

Sending build context to Docker daemon  4.608kB
Step 1/7 : FROM centos
 ---> 470671670cac
Step 2/7 : LABEL maintainer="Oscar Mondragon"
 ---> Using cache
 ---> 39739df38015
Step 3/7 : RUN yum install httpd -y
 ---> Using cache
 ---> 1f89aa6e565f
Step 4/7 : COPY voldocker/ /var/www/html/
 ---> Using cache
 ---> a36e1ef4cfd3
Step 5/7 : RUN chmod -R 777 /var/www/html
 ---> Running in f5f699a827d3
Removing intermediate container f5f699a827d3
 ---> 11011e1685e5
Step 6/7 : EXPOSE 80
 ---> Running in 6696abe1c8a3
Removing intermediate container 6696abe1c8a3
 ---> 5ebf8b100c09
Step 7/7 : CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
 ---> Running in 912b1208090e
```

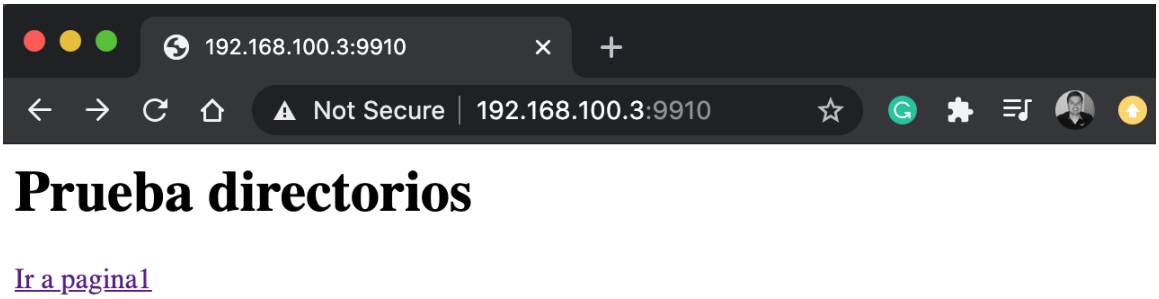
```
Removing intermediate container 912b1208090e
 ---> a505236510b0
Successfully built a505236510b0
Successfully tagged omondragon/testdir:latest
```

25. Probar el nuevo contenedor

Se establece el contenedor basado en la imagen creada. Opcionalmente, se le da un nombre (en este caso se le dio el nombre webcontainer) y se le asigna un puerto (en este caso 9910), el puerto 80 que aparece se refiere al puerto en el cual se está ofreciendo el servicio dentro del container; el puerto 9910 es el puerto a través del cual se accede al servicio de manera externa.

```
vagrant@machine1:~/test_docker2$ sudo docker run -d --name webcontainer -p 9910:80 omondragon/testdir
b05c9ac9adc2d70b32063409b199080ac28d99b852a8b546d3baed675a1e7dc8
```

La prueba se hace desde afuera, accediendo a la dirección IP de la máquina Ubuntu (192.168.100.3) y el puerto especificado para reenvío (9910).



26. Para ver los logs de un contenedor particular puede ejecutar

```
docker logs ContainerName
```

o

```
docker logs ContainerID
```

Ejemplo:

Para obtener el id o el nombre:

```
$ sudo docker container ls -a
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS              PORTS
NAMES
ba59538b9f7b   omondragon/testdir   "/usr/sbin/httpd -D …"   2 minutes ago   Up 2 minutes
0.0.0.0:9910->80/tcp, :::9910->80/tcp   webcontainer
```

Verificar logs:

```
$ sudo docker logs ba59538b9f7b
$ sudo docker logs webcontainer
```

27. Entrar al terminal del contenedor (no es necesario en muchas ocasiones)

```
sudo docker exec -it webcontainer /bin/bash
```

3. Ejercicio

1. **Contenedor para Data Science e IA**. Genere y pruebe un contenedor Docker con Jupiter notebooks y librerías que usualmente se usan en Data Science e IA (tensor flow, preprocess).

Ver:
https://towardsdatascience.com/make-your-data-science-life-easy-with-docker-c3e1fc0dee59

2. **Volúmenes docker**. Investiga cómo funcionan los volúmenes en Docker para compartir directorios entre el anfitrión y un contenedor.

Puede consultar fuentes como:

https://ricardogeek.com/usando-volumenes-en-docker/

Los comandos

```
docker volume list
docker volume inspect <IDVolumen>
```

pueden ser útiles

Implemente un ejemplo usando volúmenes.

3. **Contenedor para IA (TensorFlow + Scikit-learn).**

Clonar el siguiente repositorio y experimente con el contenedor docker que contiene TensorFlow y scikit-learn con Python3.7

https://github.com/asashiho/ml-jupyter-python3

4. **Bibliografía**

- Sitio oficial Docker. https://www.docker.com/
- Instalar Docker en Ubuntu. Inglés. https://docs.docker.com/engine/install/ubuntu/
- Instalar Docker en Ubuntu. Español. https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-18-04-1-es
- Core OS rkt containers. https://coreos.com/rkt/
- Docker en LXD. https://stgraber.org/2016/04/13/lxd-2-0-docker-in-lxd-712/