**Práctica Contenedores Docker**

**Por: prof. Oscar H. Mondragón**

1. **Objetivo**

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

1. **Herramientas a Utilizar**

* Vagrant
* VirtualBox
* Docker Community Edition
* Vagrant Box Ubuntu

1. **Desarrollo de la Practica**

**PARTE 1. Configuración de Vagrant**

Esta práctica la desarrollaremos usando un Box de Ubuntu 22.04 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-22.04"

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-22.04"

servidorUbuntu.vm.network :private\_network, ip: "192.168.100.3"

servidorUbuntu.vm.hostname = "servidorUbuntu"

end

end

**PARTE 2. Instalación de Docker en Ubuntu 22.04**

1. Desinstalar versiones anteriores de Docker

for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done

1. Configurar el repositorio

Agregue la clave GPG\* oficial de Docker

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

Agregue el repositorio a Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

Instale la última versión de Docker:

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Nota: Puede agregar su usuario a el grupo Docker, de esta manera no necesita anteponer sudo cuando ejecute Docker. Para esto, puede ejecutar el siguiente comando:

$ sudo usermod -aG docker $USER

Para que tome efecto debe salir de la máquina (exit) en ingresar de nuevo con vagrant ssh.

Verifique que Docker Engine quedo 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.

1. 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

1. 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**

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

Por ejemplo, si quisiéramos saber que imágenes de contenedores con que apache existen:

$ sudo docker search apache

NAME DESCRIPTION STARS OFFICIAL

tomcat Apache Tomcat is an open source implementati… 3643 [OK]

maven Apache Maven is a software project managemen… 1552 [OK]

httpd The Apache HTTP Server Project 4667 [OK]

zookeeper Apache ZooKeeper is an open-source server wh… 1413 [OK]

solr Apache Solr is the popular, blazing-fast, op… 982 [OK]

flink Apache Flink® is a powerful open-source dist… 415 [OK]

cassandra Apache Cassandra is an open-source distribut… 1522 [OK]

groovy Apache Groovy is a multi-faceted language fo… 148 [OK]

tomee Apache TomEE is an all-Apache Java EE certif… 114 [OK]

storm Apache Storm is a free and open source distr… 195 [OK]

spark Apache Spark - A unified analytics engine fo… 37 [OK]

apachepinot/pinot Pinot is a real-time distributed OLAP datast… 9

bitnami/apache Bitnami Apache Docker Image 93

apache/airflow Apache Airflow 503

apachepinot/pinot-presto This image is a presto image built-in with p… 1

apache/superset Apache Superset 245

apachepinot/pinot-superset 1

apachepinot/pinot-base-build The base image for Apache Pinot build 0

apachepinot/pinot-base-runtime The base image for Apache Pinot runtime 0

apache/tika Apache Tika Server - the content analysis to… 35

apachepinot/thirdeye Docker Image for Apache Thrideye 1

apachepulsar/pulsar Apache Pulsar - Distributed pub/sub messagin… 89

apache/nifi-registry Unofficial convenience binaries for Apache N… 42

apache/apisix Apache APISIX: Cloud-Native API Gateway 89

apachepulsar/pulsar-all Apache Pulsar - Distributed pub/sub messagin… 22

1. Descargar imágenes

$ docker pull httpd

Using default tag: latest

latest: Pulling from library/httpd

f546e941f15b: Pull complete

657739c97e0b: Pull complete

4f4fb700ef54: Pull complete

598e131c66a1: Pull complete

4788cb06a95a: Pull complete

48aa2460acc6: Pull complete

Digest: sha256:104f07de17ee186c8f37b9f561e04fbfe4cf080d78c6e5f3802fd08fd118c3da

Status: Downloaded newer image for httpd:latest

docker.io/library/httpd:latest

1. Ver imágenes descargadas

$ sudo docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

httpd latest 50a1bd9b297f 6 weeks ago 194MB

hello-world latest ee301c921b8a 10 months ago 9.14kB

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

$ sudo docker run -d --name web1 -p 8800:80 httpd

71b95a509905ae37646359b28bd520093a6d4e35076cbf512a9fbd092de92eb6

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

1. Verificar qué contenedores están corriendo actualmente

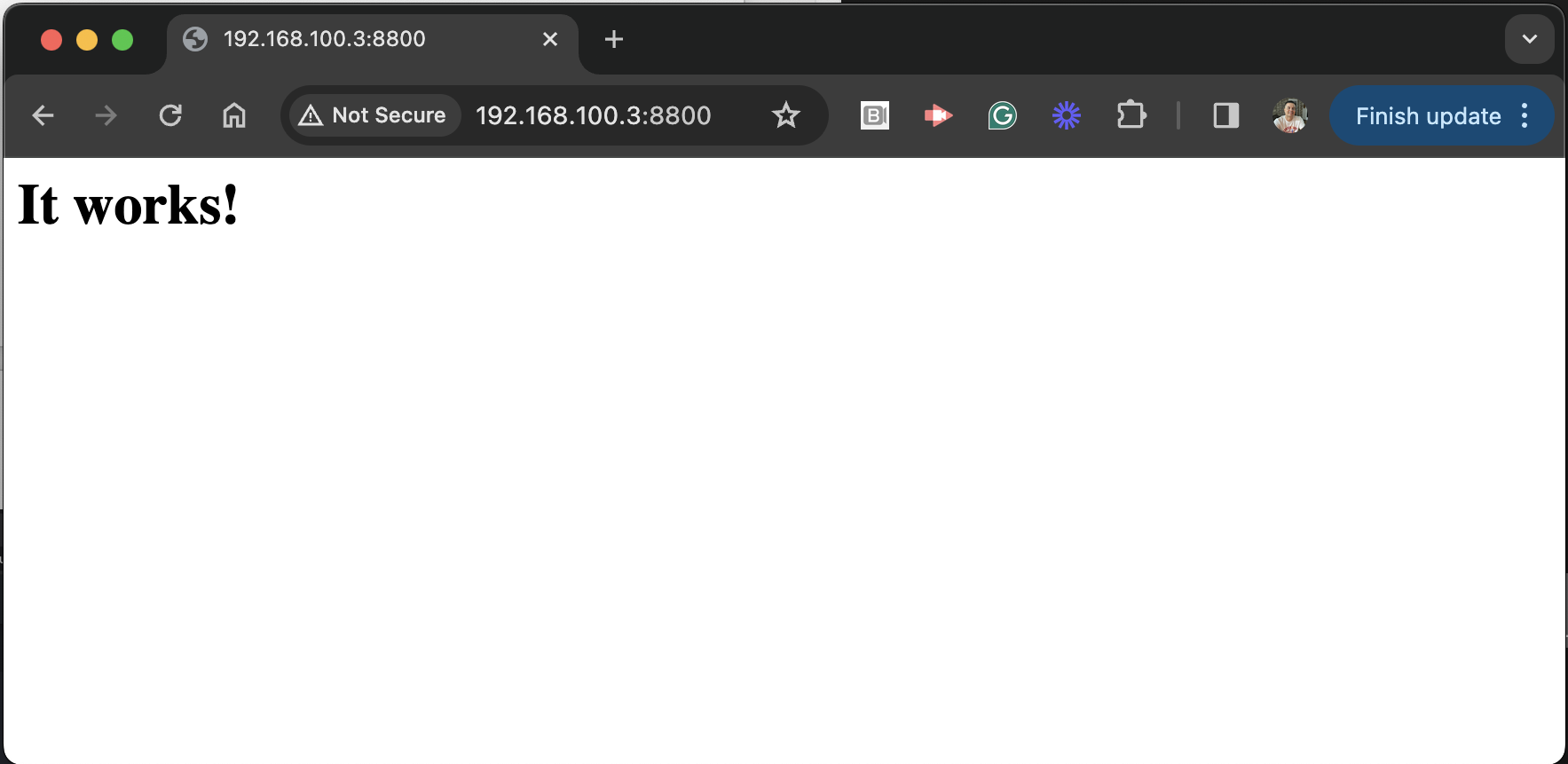
$ sudo docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

71b95a509905 httpd "httpd-foreground" 6 seconds ago Up 5 seconds 0.0.0.0:8800->80/tcp, :::8800->80/tcp web1

1. 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.



1. 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

71b95a509905 httpd "httpd-foreground" 12 minutes ago Up 12 minutes 0.0.0.0:8800->80/tcp, :::8800->80/tcp web1

aa5f5098b957 hello-world "/hello" 28 minutes ago Exited (0) 28 minutes ago

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

$ sudo docker container stop 71b95a509905

71b95a509905

$ sudo docker container rm 71b95a509905

71b95a509905

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]

1. 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

1. 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 sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-Linux-\* &&\

sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-Linux-\*

RUN yum upgrade -y

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"]

1. 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

1. Se crea el contenedor

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

ebb43775ccb53452db74bb9128346feb7634f758bedd56427966944c4b84acd3

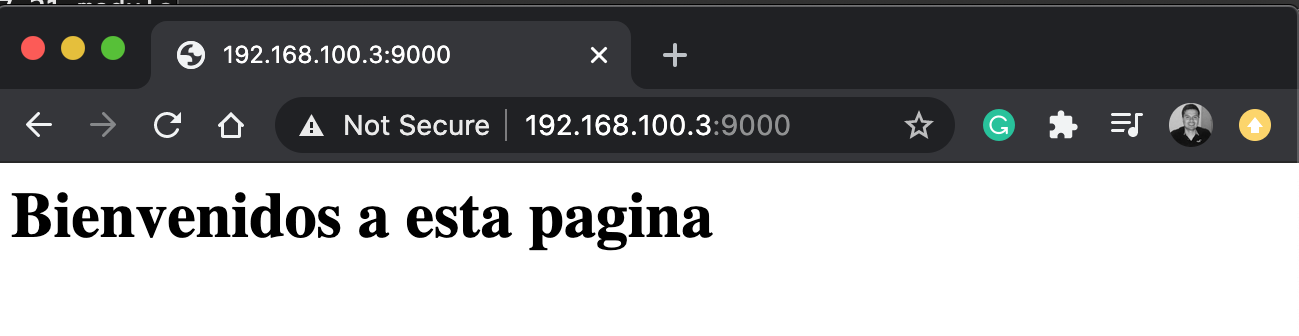
1. Se miran los contenedores creados y ejecutándose

$ 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

1. Verificar el servicio funcionando



**PARTE 5. SUBIR CONTENEDOR AL REGISTRY (DockerHub)**

En esta parte subiremos nuestra imagen a un repositorio de Docker público. Específicamente a Docker Hub (<https://hub.docker.com/>).

Para esto, dirigirse a <https://hub.docker.com> y crear una cuenta.

Luego de crear la cuenta, loggearse a través del terminal

$ sudo docker login

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.

Username: youruser

Password:

WARNING! Your password will be stored unencrypted in /root/.docker/config.json.

Configure a credential helper to remove this warning. See

https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

Debe crear una imagen que incluya el nombre de su repositorio en Docker hub y luego un tag para crear una versión nueva.

En mi caso voy a utilizar un tag para crear una nueva versión de omondragon/centosweb, a la cual llamaremos omondragon/centosweb:v1

$ sudo docker tag omondragon/centosweb omondragon/centosweb:v1

Verifico la nueva imagen creada con el tag suministrado

vagrant@servidorUbuntu:~/test\_docker$ sudo docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

omondragon/centosweb latest 7e37e79f5672 13 minutes ago 583MB

**omondragon/centosweb v1 7e37e79f5672 13 minutes ago 583MB**

omondragon/haproxy latest 440f9872cbfc 4 weeks ago 99.4MB

omondragon/web latest 058fdec1d5c0 4 weeks ago 227MB

hello-world latest feb5d9fea6a5 17 months ago 13.3kB

En su caso debe crear una imagen nueva con el usuario que usa para loggearse en Docker hub. Por ejemplo, si su usuario se llama **youruser**:

$ sudo docker build -t youruser/centosweb .

$ sudo docker tag youruser/centosweb youruser/centosweb:v1

En adelante cambie omondragon por su usuario de dockerHub

Ahora subiremos la imagen al repositorio mediante el comando docker push

$ sudo docker push omondragon/centosweb:v1

The push refers to repository [docker.io/omondragon/centosweb]

5683a22e1dfe: Pushed

84d435426403: Pushed

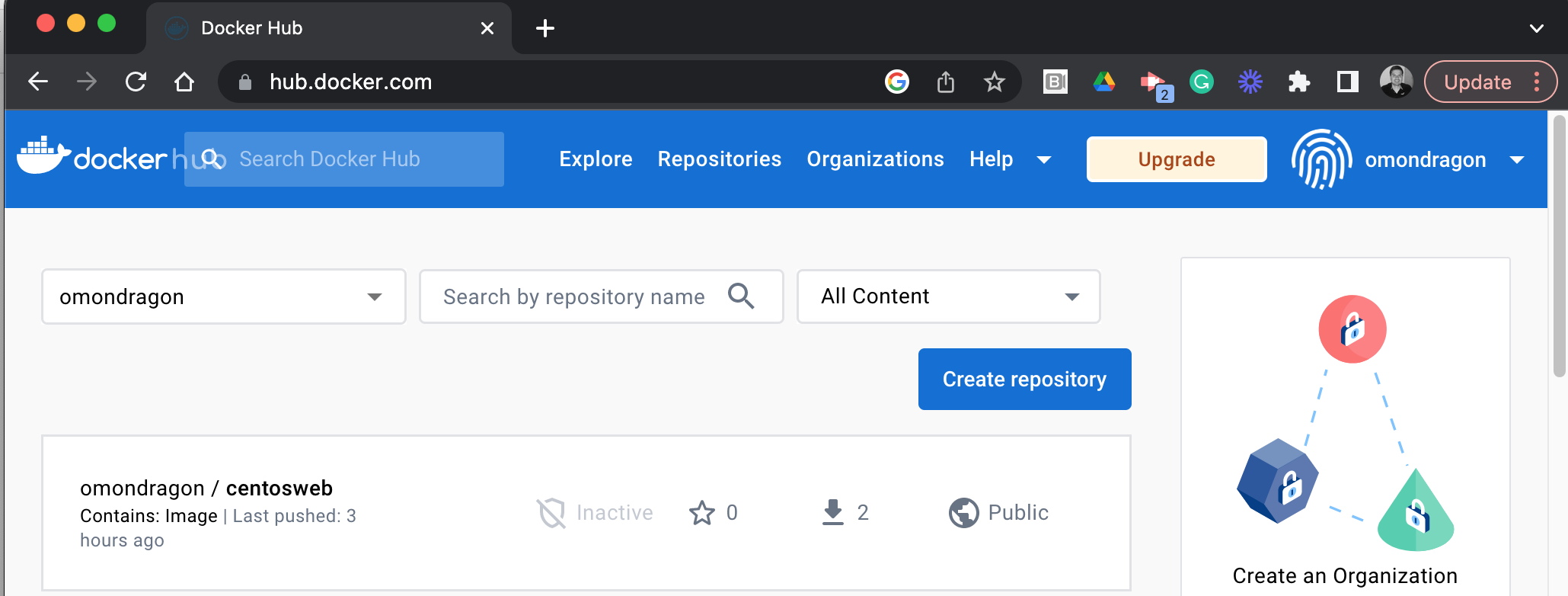
9b4d0428265a: Pushed

5652288c41fe: Pushed

74ddd0ec08fa: Mounted from library/centos

v1: digest: sha256:1cecacafeb01ee5902833866dfbded06672534679cbbdfe9bacbff8d6aa93859 size: 1369

Ahora la imagen estará disponible en el repositorio de dockerhub:



Ahora que la imagen está disponible y pública, puede ser usada desde cualquier otro equipo.

Conectarse a clienteUbuntu, instalar docker y probar desde ahí, por ejemplo:

vagrant@clienteUbuntu:~$ sudo docker run --name webcliente -d -p 9900:80 omondragon/centosweb:v1

Unable to find image 'omondragon/centosweb:v1' locally

v1: Pulling from omondragon/centosweb

a1d0c7532777: Pull complete

a4e38749de96: Pull complete

b2f153aa63f3: Pull complete

2e051a87d5fc: Pull complete

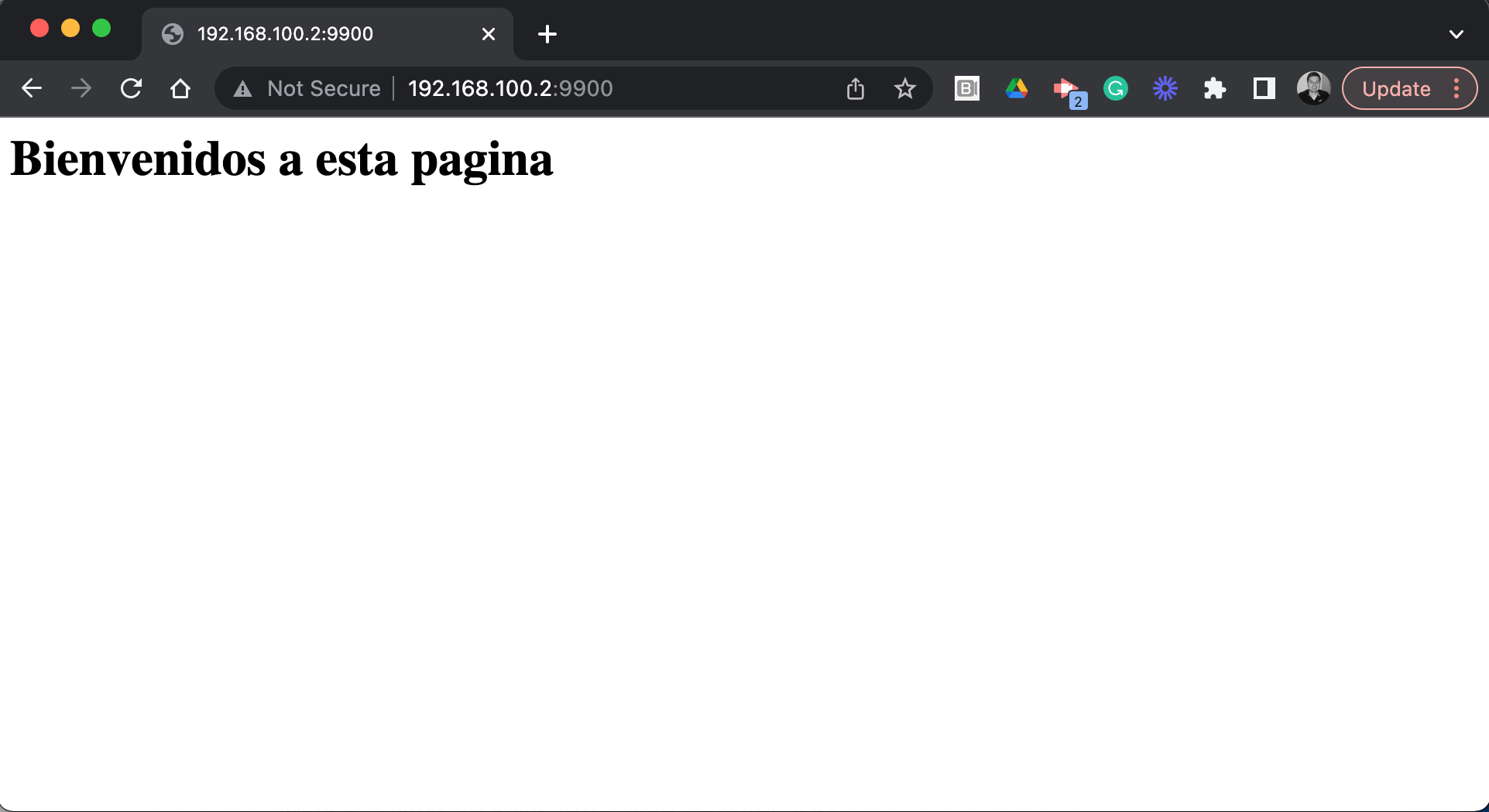
3a044004153d: Pull complete

Digest: sha256:1cecacafeb01ee5902833866dfbded06672534679cbbdfe9bacbff8d6aa93859

Status: Downloaded newer image for omondragon/centosweb:v1

ab08658c08f67d79498b58bea3717e1b8c64433a83fad60c04c596a5904c9a02

En este caso accederemos a <http://192.168.100.2:9900> para verificar



**PARTE 6. 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.

1. Cree el directorio en el host

Cree 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 cree una pagina llamada index.html con un vinculo 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>

1. 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 sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-Linux-\* &&\

sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-Linux-\*

RUN yum upgrade -y

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"]

1. 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

1. Probar el nuevo contenedor

Se crea el contenedor basado en la imagen creada. Se le un nombre (esto es opcional), 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 esta ofreciendo el servicio dentro del container, el puerto 9910 es el puerto a través del cual se accederá 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).



1. 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

Verficar logs:

$ sudo docker logs ba59538b9f7b

$ sudo docker logs webcontainer

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

sudo docker exec -it webcontainer /bin/bash

1. **Ejercicios**
2. **Imagen propia + DockerHub.** Construya una imagen propia de Docker en la cual despliegue un sitio web personalizado y súbalo a su repositorio de docker hub.
3. **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 (tensorflow, preprocess).

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

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

Implemente un ejemplo usando volúmenes.

Puede consultar fuentes como:

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

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

Clone el siguiente repositório y experimente con el contenedor docker que contiene TensorFlow y scikit-learn with Python3.7

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

NOTA:

Si aparece el siguiente error:

Package 'libav-tools' has no installation candidate

Comente la línea **libav-tools \** en el Dockerfile (línea ~16)

En el Dockerfile se debe de cambiar en la instlacion de librerías para data science en la linea 31

Cambiar Sklearn por **scikit-learn**

**4.Desafíos [Hasta 0.5 Puntos en una nota de practicas]**

Desarrolle uno de los siguientes puntos:

1. **(Vale por 1.0 puntos) CUDA + Python + Docker**. Demuestre en funcionamiento de un contenedor Docker con aceleración por GPUs (necesita maquina con GPU disponible).

Sugerencia: <https://developer.nvidia.com/how-to-cuda-python>

1. **Docker en LXD**. LXD se enfoca en contenedores de sistema. Es decir, podemos correr una distribución completa de Linux en nuestro contenedor. Los contenedores de aplicación (o de software) como Docker y RKT se diferencian de los contenedores de LXD, en que son usados para distribuir aplicaciones y típicamente corren un único proceso dentro de ellos

En este punto, deberá investigar como configurar un contenedor Docker dentro de LXD que corra una aplicación básica, por ejemplo un servidor web.

Ver instrucciones en: <https://stgraber.org/2016/04/13/lxd-2-0-docker-in-lxd-712/>

1. **Docker + Flask**. En el siguiente repositorio encontrara los archivos requeridos para crear un container con una aplicación web Flask en Docker

<https://github.com/omondragon/docker-flask-example>

Clone el repositorio y pruebelo creando y corriendo un container. Tenga en cuenta que Flask esta siendo ejecutado en modo de prueba y expone el puerto 5000.

Si desea profundizar en Flask puede consultar:

<https://flask.palletsprojects.com/en/1.1.x/>

1. **Bibliografía**

* Sitio oficial Docker. <https://www.docker.com/>
* Instalar Docker en Ubuntu. Ingles. <https://docs.docker.com/engine/install/ubuntu/>
* Instalar Docker en Ubuntu. Espanol. <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/>
* Publicar tu imagen en Docker Hub. <https://www.returngis.net/2019/02/publicar-tu-imagen-en-docker-hub/>