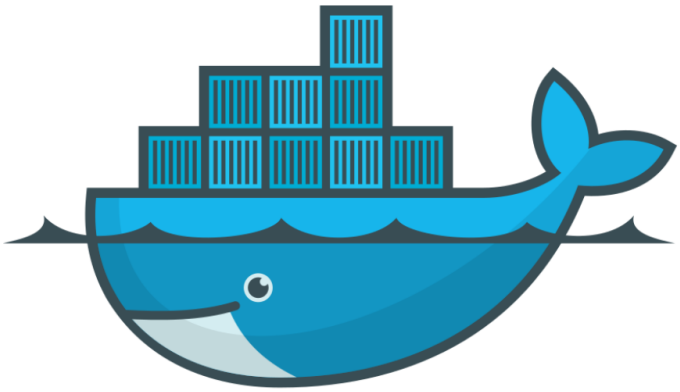
PROGRAMACIÓN WEB 2

INFORME

DOCKER



Profesor(a):
Carlo Jose Luis Corrales Delgado

Estudiantes:
Mamamni Anahua, Victor Narciso
Mamani Huarsaya, Jorge Luis

30 de abril, 2024

Preparando el espacio de trabajo

Empezamos descargando la imagen ubuntu 20.04 de Docker Hub

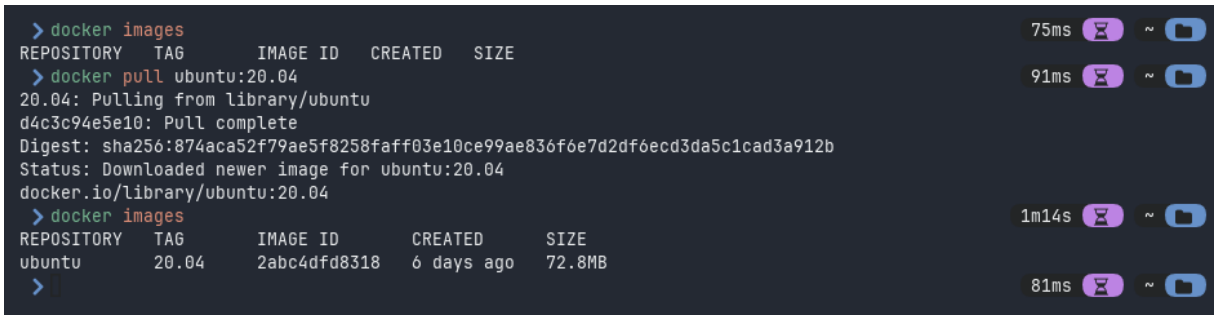


Figure 1: Instalando la imagen ubuntu:20.04

Despues creamos un contenedor a partir de esta imagen y lo iniciamos inmediatamente. Podemos lograr estas acciones por separado usando los comandos docker create y seguidamente zshdocker start. Sin embargo al no ejecutar ningún programa dentro del contenedor, este se detendrá inmediatamente.

Para el primer inicio del contenedor, podemos usar el comando docker run, junto con otros opciones y argumentos:

- Podemos establecer un nombre especifico al contenedor, si no hacemos esto, todas las operaciones que hagamos con el contenedor, tendremos que utilizar su hash generado. Al igual que Git con repositorios remotos. La opción es –name
- Aprovechamos realizanso el **Port Mapping** de los puertos de la maquina host a los puertos del contenedor usando la opcion -p
- Interactuar con el contenedor con la opción -it para que habra una terminal iterativa y dentro de la terminal ejecutamos el inteprete de comandos de bash, el ejcutable es /bin/bash

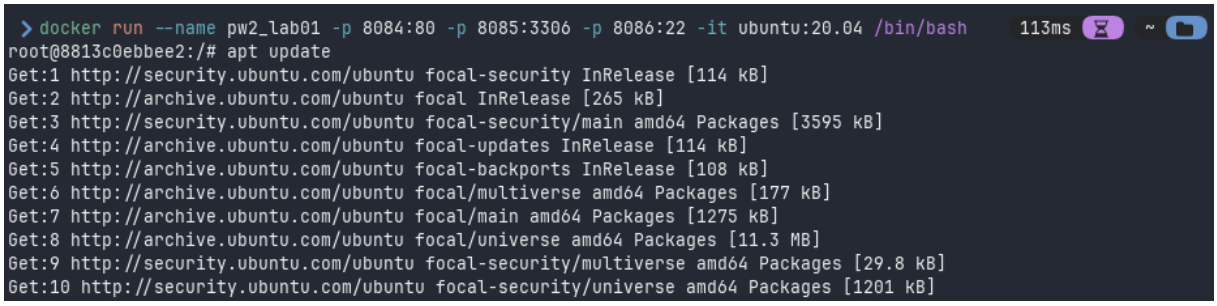


Figure 2: Creando y ejecutando el contenedor

Ahora procedemos con la instalación de un editor de texto necesario para editar código (neovim), un servidor web que implemente el protocolo HTTP y HTTPS (Apache), unos lenguajes de scripting (perl, python), un sistema de gestion de base de datos (MariaDB) y finalmente un programa servidor que implemente el protocolo SSH (openssh)

```
root@8813c0ebbee2:/# apt install neovim apache2 perl python mariadb-server openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-is-python2' instead of 'python'
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils ca-certificates dbus distro-info-data dmsetup file galera-3 gawk
  gir1.2-glib-2.0 iproute2 krb5-locales libaio1 libapparmor1 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libargon2-1 libasn1-8-heimdal libatm1 libbrotli1 libbsd0 libcap2 libcap2-bin libcbor0.6
  libcgi-fast-perl libcgi-pm-perl libconfig-inifiles-perl libcryptsetup12 libcurl4 libdbd-mysql-perl libdbi-perl
  libdbus-1-3 libdevmapper1.02.1 libedit2 libelf1 libencode-locale-perl libexpat1 libfcgi-perl libfido2-1
  libgdbm-compat4 libgdbm6 libgirepository-1.0-1 libglib2.0-0 libglib2.0-data libgssapi-krb5-2 libgssapi3-heimdal
  libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libhx509-5-heimdal libicu66 libicu66 libio-html-perl
  libip4tc2 libjansson4 libjson-c4 libk5crypto3 libkeyutils1 libkmod2 libkrb5-26-heimdal libkrb5-3 libkrb5support0
  libldap-2.4-2 libldap-common liblua5.2-0 libluajit-5.1-2 libluajit-5.1-common liblwp-mediatypes-perl
  libmagic-mgc libmagic1 libmnl0 libmpdec2 libmpfr6 libmsgpackc2 libmysqlclient21 libnghttp2-14 libnss-systemd
  libpam-cap libpam-systemd libperl5.30 libpopt0 libpsl5 libpython2-stdlib libpython2.7-minimal
```

Figure 3: Instalando los programas para crear y administrar nuestro servidor web

Podemos verificar el estado de los programas servidor que hemos instalado. Estos servicios son conocidos como **Demonios** ya que son bucles infinitos que siempre estan escuchando. Los servidores siempre se deben de activar, incluso podemos configurar su habilitación automatica cuando arranque el sistema operativo.

```
root@8813c0ebbee2:/# /etc/init.d/apache2 status
 * apache2 is not running
root@8813c0ebbee2:/# /etc/init.d/mysql status
 * MariaDB is stopped.
root@8813c0ebbee2:/# /etc/init.d/ssh status
 * sshd is not running
root@8813c0ebbee2:/#
```

Figure 4: Estado de los programas servidor

Por el momento solo vamos a iniciar el servidor web Apache y el sistema de gestion de base de datos MariaDB.

```
root@8813c0ebbee2:/# /etc/init.d/apache2 start
 * Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the '
ServerName' directive globally to suppress this message
 *
root@8813c0ebbee2:/# /etc/init.d/mysql start
 * Starting MariaDB database server mysqld                                                    [ OK ]
root@8813c0ebbee2:/#
```

Figure 5: Comenzar los demonios

Podemos probar el funcionamiento del servidor web de nuestro contenedor en nuestro navegador de nuestra maquina host poniendo la dirección IP del contenedor, o simplemente instalando y usando el programa curl que es una herramienta en la terminal.

```
root@8813c0ebbee2:/# curl 172.17.0.2

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.
dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2016-11-16
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }

  body, html {
    padding: 3px 3px 3px 3px;
```

Figure 6: Archivo *index.html* ubicado en el directorio por defecto */var/www/html*

Habilitando la ejecución de CGIs en Apache

Apache trae consigo dos modulos nativos *mod-cgi* y *mod-cgid* y por defecto Apache solo escuha y ejecuta los scripts ubicados en el directorio especifico */usr/lib/cgi-bin*

Vamos a crear un test en Python para ver el funcionamiento. Una vez que creamos el archivo .py, tenemos que otorgarle el permiso de ejecución, ya que es un script que Apache debe de ejecutar.

```
#!/usr/bin/env python3

print("Content-Type: text/html\n")

print("""
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CGI con Python</title>
</head>
<body>
    <h1>¡Hola desde un script CGI en Python!</h1>
</body>
</html>
""")
~
```

Figure 7: Archivo en */usr/lib/cgi-bin/test.py*

```
root@8813c0ebbee2:/# cd /lib/cgi-bin/
root@8813c0ebbee2:/lib/cgi-bin# nvim test.py
root@8813c0ebbee2:/lib/cgi-bin# chmod 755 test.py
root@8813c0ebbee2:/lib/cgi-bin# ls
test.py
root@8813c0ebbee2:/lib/cgi-bin#
```

Figure 8: Estableciendo permisos de ejecución

Si intentamos ejecutar el script, Apache ni intentará buscarlos porque hasta el momento solo escuha en el directorio */var/www/html*

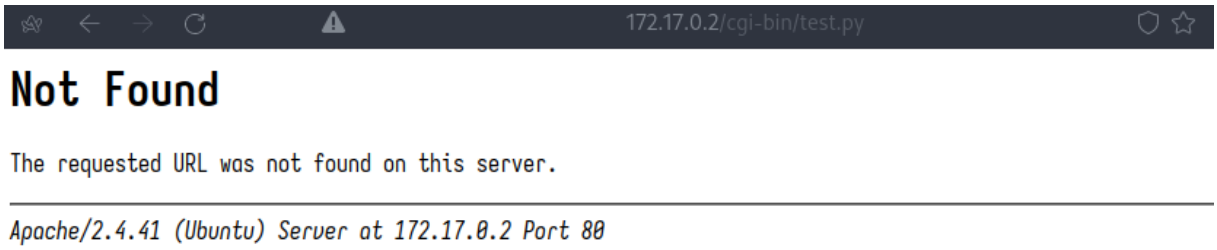


Figure 9: Apache no escucha en este directorio

Pasa que alta hacer alguna configuracion para que Apache pueda realizar esta funcionalidad. Nosotros debemos de habilitar los modulos CGI. Una vez activados los modulos, debemos de reiniciar Apache para que las configuraciones sean leidas.

```
root@8813c0ebbee2:/lib/cgi-bin# a2enmod cgid
Module cgid already enabled
root@8813c0ebbee2:/lib/cgi-bin# /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the '
ServerName' directive globally to suppress this message
                                                                                                          [ OK ]
root@8813c0ebbee2:/lib/cgi-bin# curl 172.17.0.2/cgi-bin/test.py

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CGI con Python</title>
</head>
<body>
    <h1>¡Hola desde un script CGI en Python!</h1>
</body>
</html>

root@8813c0ebbee2:/lib/cgi-bin#
```

Figure 10: Activar los modulos, reiniciar el servidor y probar

Ahora si podemos estar seguros que los CGIs se van a ejecutar, tambien podemos probarlo desde el navegador de la maquina host

Figure 11: Test exitoso

Cambiando a directorios personalizados

Para hacer los cambios, nosotros debemos modificar los archivos de configuración de apache. Apache maneja el concepto de **Virtual Host** que nos permite tener varias aplicaciones web en solo un servidor web fisico. Esto se logra configurando Apache para que responda a diferentes nombres de dominio (o direcciones IP) y sirva contenido específico para cada uno de ellos.

Ejemplo con el directorio raiz /aplicaciones_web que almacena la aplicacion web proyecto-final-pweb1

Las bloques de configuracion que vamos a realizar van a definir el nombre de dominio, el directorio raiz y otras opciones de configuracion. Vamos a crear un archivo */etc/apache2/sites-available/aplicaciones_web* y aqui colocamos nuestras configuraciones.

```
root@8813c0ebbee2:/# cd /etc/apache2/sites-available/
root@8813c0ebbee2:/etc/apache2/sites-available# ls
000-default.conf  default-ssl.conf
root@8813c0ebbee2:/etc/apache2/sites-available# cp 000-default.conf aplicaciones_web.conf
root@8813c0ebbee2:/etc/apache2/sites-available# nvim aplicaciones_web.conf
root@8813c0ebbee2:/etc/apache2/sites-available# cd /
root@8813c0ebbee2:/# mkdir -p aplicaciones_web/proyecto-final-pweb1
root@8813c0ebbee2:/#
```

Figure 12: Creación del archivo de configuración

Es muy importante el uso de **ScriptAlias** porque nos permite organizar todos los scripts de la aplicación en un solo directorio especifico, el cual es el unico directorio que Apache busca los scripts y los va a ejecutar.

```
Alias /aplicaciones_web "/aplicaciones_web/proyecto-final-pweb1"

<Directory "aplicaciones_web/proyecto-final-pweb1">
        Options Indexes FollowSymLinks MultiViews
        Allowoverride All
        Require all granted
</Directory>

<VirtualHost *:80>
        # Nombre de dominio para esta aplicación
        ServerName academiapiensa.com

        ServerAdmin webmaster@localhost
        DocumentRoot /aplicaciones_web/proyecto-final-pweb1

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # Ejecutar scripts en otro directorio diferente a /usr/lib/cgi-bin
        ScriptAlias /cgi-bin /aplicaciones_web/proyecto-final-pweb1/cgi-bin
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
~
```

Figure 13: Configuración del Virtual Host

Una vez que hicimos las respectivas configuraciones, ahora necesitamos activar el sitio web y activar el modulo **mod-rewrite** que se encarga de realizar las redirecciones de solicitudes.

```
root@8813c0ebbee2:/etc/apache2/sites-available# a2ensite aplicaciones_web.conf
Enabling site aplicaciones_web.
To activate the new configuration, you need to run:
  service apache2 reload
root@8813c0ebbee2:/etc/apache2/sites-available# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  service apache2 restart
```

Figure 14: Habilitamos el sitio web configurado

Por ultimo habilitamos la configuración que basicamente consiste en crear un enlace simbólico desde el archivo de configuración */etc/apache2/sites-available/aplicaciones_web* al archivo correspondiente en */etc/apache2/sites-enabled/aplicaciones_web*.
Para que las configuraciones se establezcan, reiniciamos el servidor Apache.

```
root@8813c0ebbee2:/etc/apache2/conf-available# a2enconf cgi_enabled.conf
Enabling conf cgi_enabled.
To activate the new configuration, you need to run:
  service apache2 reload
root@8813c0ebbee2:/etc/apache2/conf-available# /etc/init.d/apache2 restart
 * Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the '
ServerName' directive globally to suppress this message
                                                                                                                [ OK ]
root@8813c0ebbee2:/etc/apache2/conf-available#
```

Figure 15: Habilitando las configuración

Agregando un usuario para que pueda controlar el sitio web creado

Podemos agegar a un usuario para que se encargue de administrar este sitio web. Entonces lo primero que haremos es crear el usuario en el sistema y después cambiaremos el propietario y el grupo del directorio raiz */aplicaciones_web* y todos los directorios y archivos dentro de este.

```
root@8813c0ebbee2:/# adduser pw2
Adding user `pw2' ...
Adding new group `pw2' (1000) ...
Adding new user `pw2' (1000) with group `pw2' ...
Creating home directory `/home/pw2' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for pw2
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@8813c0ebbee2:/# chown -R pw2:www-data /aplicaciones_web/
root@8813c0ebbee2:/# cd /aplicaciones_web/
root@8813c0ebbee2:/aplicaciones_web# ls -la
total 12
drwxr-xr-x 3 pw2  www-data 4096 May  3 21:54 .
drwxr-xr-x 1 root root     4096 May  3 21:54 ..
drwxr-xr-x 2 pw2  www-data 4096 May  3 21:54 proyecto-final-pweb1
root@8813c0ebbee2:/aplicaciones_web#
```

Figure 16: Agregado un usuario y cambiando propietario

Ahora podemos probar el servicio SSH, para ello lo vamos a activar y nos conectaremos desde la maquina host

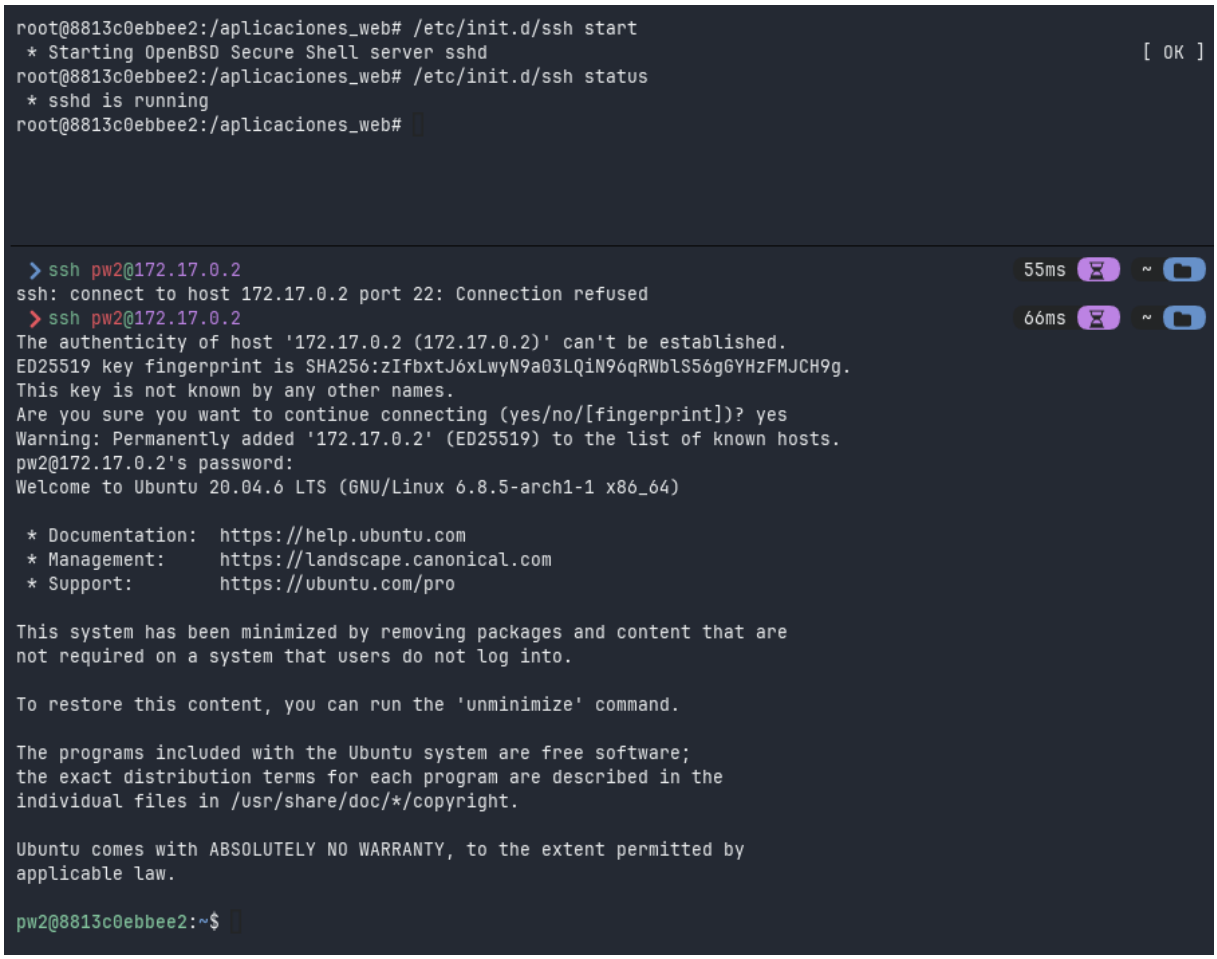


Figure 17: El servicio de SSH esta habilitado