Práctica 2: DevSecOps

Alesander Martinez Seijo

2023

Contenido

[1. Crea un Dockerfile que partiendo de una imagen PHP genera una imagen que: 2](#_Toc125794445)

[1. Copia una aplicación en PHP a un directorio del contenedor. Esta aplicación se debe copiar directamente desde un directorio del anfitrión. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL). 8](#_Toc125794446)

[2. Crea un Dockerfile que partiendo de una imagen Ubuntu genera una imagen que: 17](#_Toc125794447)

[a) Instala Apache, de forma que se exponga el puerto 80. 17](#_Toc125794448)

[b) Instala PHP. 17](#_Toc125794449)

[c) Copia una aplicación web en PHP al directorio de Apache que expone las páginas web. 17](#_Toc125794450)

[3. Crea un contenedor para cada una de esas imágenes y verifica que funciona. Para y borra dicho contenedor. 21](#_Toc125794451)

**Puesta en Producción Segura**

Práctica 2: DevSecOps

**Objetivo**

El objetivo de esta práctica es que el alumno comprenda los fundamentos de las DevOps seguras, también conocidas como DevSecOps.

**Corrección de la práctica**

El alumno deberá entregar una memoria con las soluciones a cada apartado y los pasos detallados que ha seguido en el proceso. Se valorará:

• La calidad de la memoria. Se deberán explicar y justificar los procedimientos y resultados, así como de dar soporte a lo descrito mediante capturas de pantalla.

• La corrección del proceso descrito.

• La corrección de los comandos empleados y de las opciones.

• La correcta justificación de los pasos seguidos.

• La propuesta de soluciones alternativas, documentando incluso aquellas que siendo

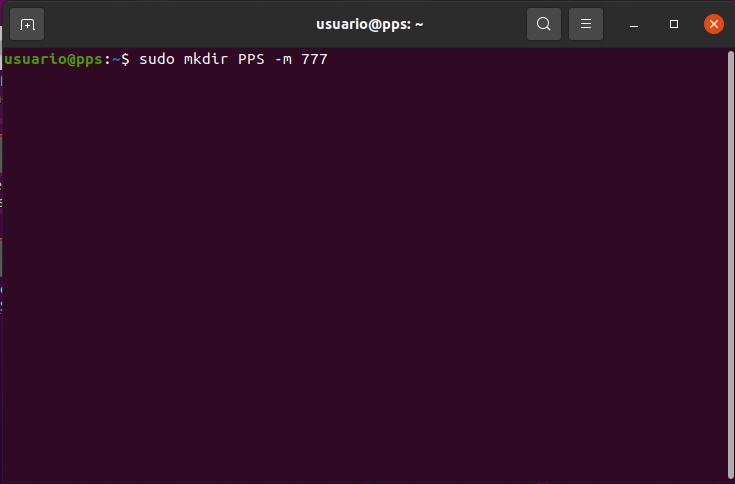
razonables no han logrado su propósito.

Forma de entrega: Vía Moodle. Se habilitará una tarea a tal efecto.

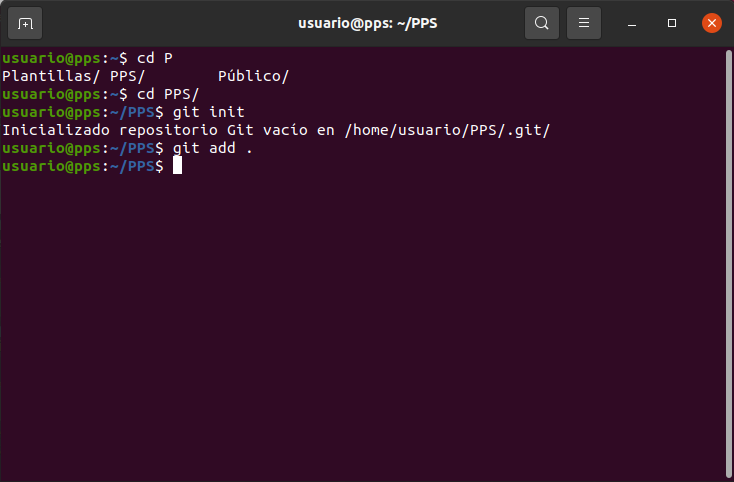
**Docker**

# Crea un Dockerfile que partiendo de una imagen PHP genera una imagen que:

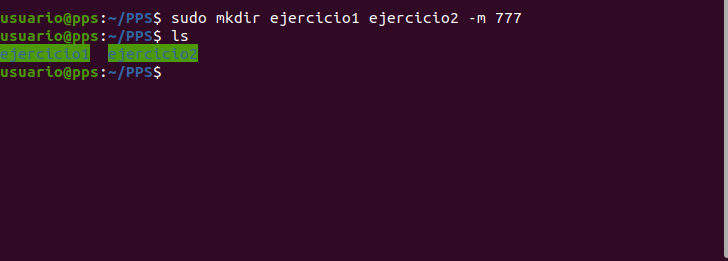
Primero voy crear la carpeta en donde voy iniciar el repositorio de GitHub local, esto la haré con el comando “***sudo mkdir PSP -m 777”***, la opción -m sirve para cambiar los permisos del directorio creado.



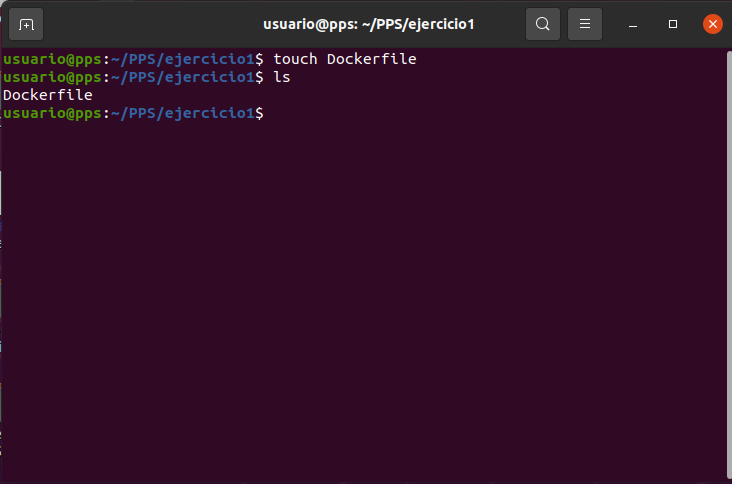
Ahora iniciaré el repositorio de GitHub localmente, esto la haré ejecutando el comando “***git init”*** *y “****git add”***,el comando inicializa el repositorio el local en la carpeta donde estoy.



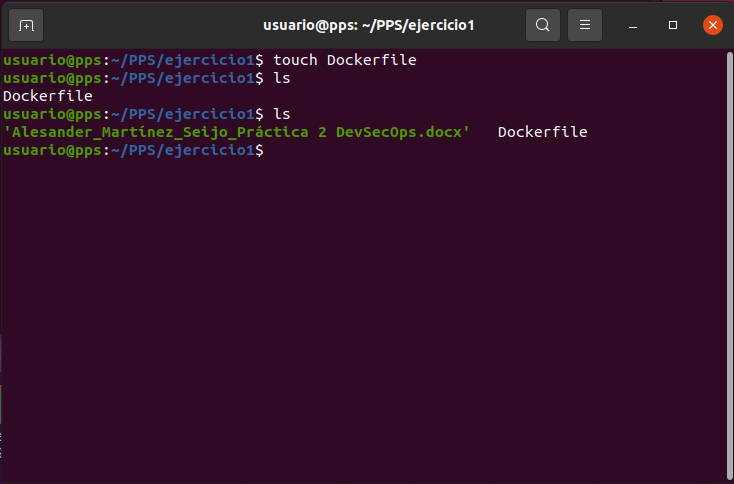
Ahora creo dos carpetas una para el ejercicio uno y otro para el dos.



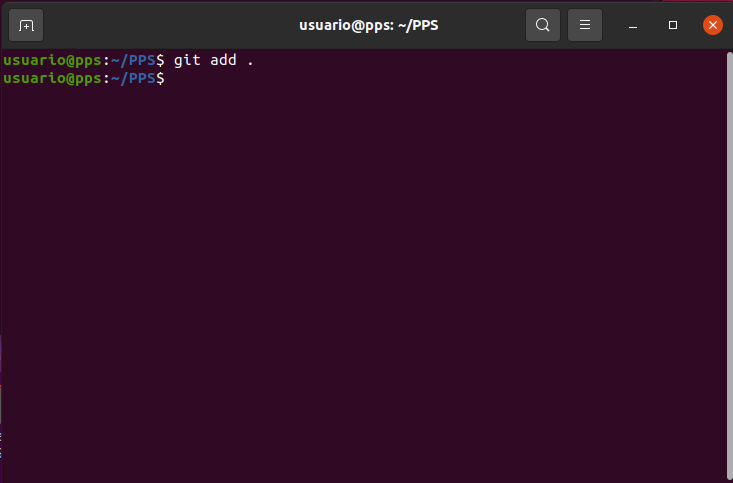
Dentro de la carpeta del ejercicio1 creo el Dockerfile.



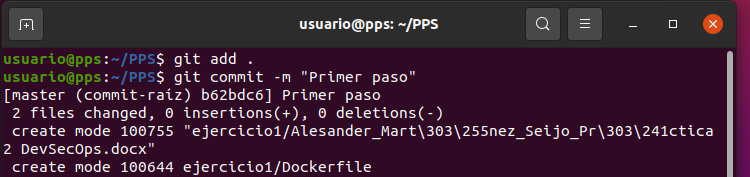
Y añado el documento de Word.



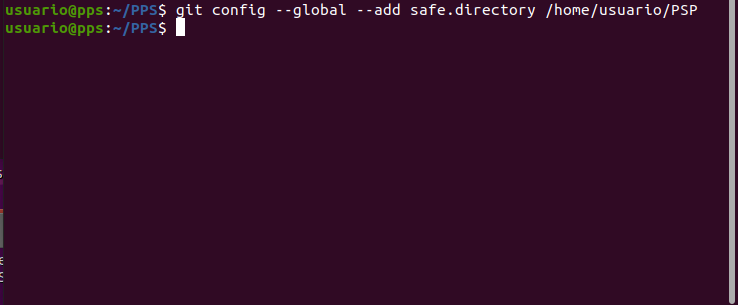
Y hago el “***git add”***, añadiendo los archivos al repositorio local.



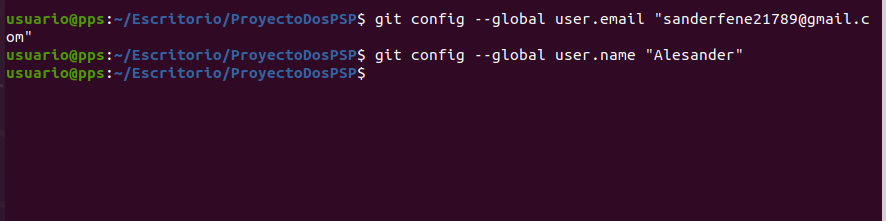
Ahora voy usar el comando de git “***git commit -m “Primer paso”***, esto lo que hará es guardar los cambios en el repositorio local.



Antes de ejecutar este comando ejecuto otro: *“****git config --global --add safe.directory /home/usuario/PSP”***, esto lo hago porque no soy el usuario propietario de la carpeta y para que no de error hago esto y ya podría ejecutar los comandos de git sin problema.

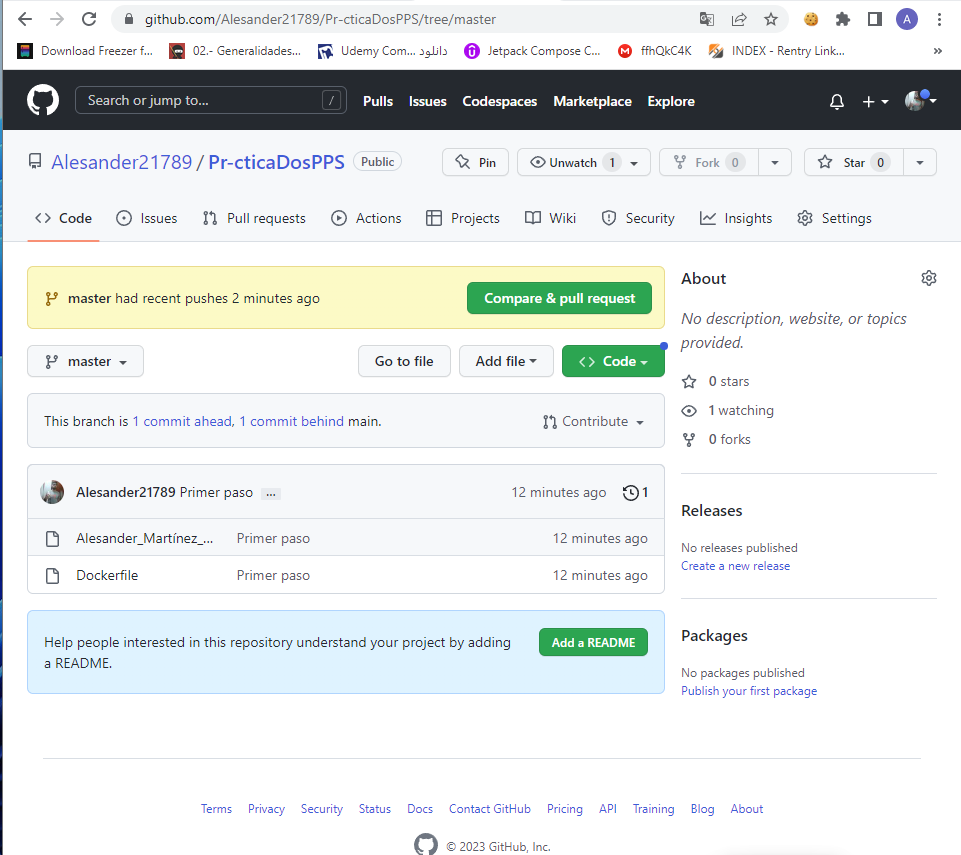


Tengo también que añadir mi correo y mi usuario.



Y ahora ejecutaré el comando de git: “***git push -u origin master”***, con esto subiré los cambios al repositorio remoto.



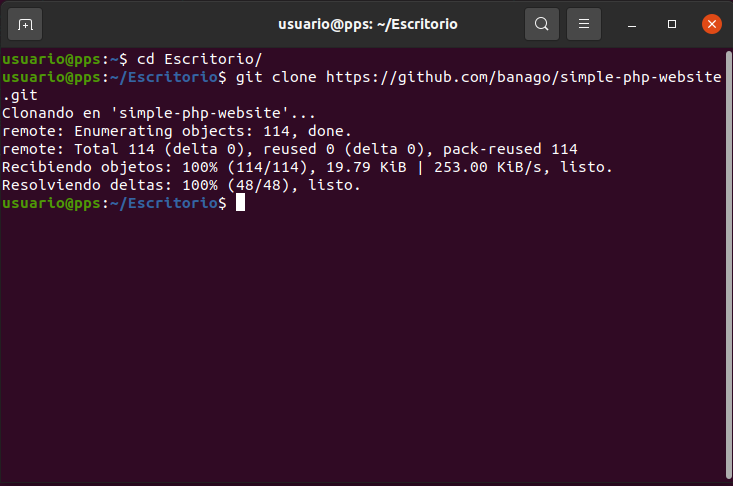


## Copia una aplicación en PHP a un directorio del contenedor. Esta aplicación se debe copiar directamente desde un directorio del anfitrión. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).

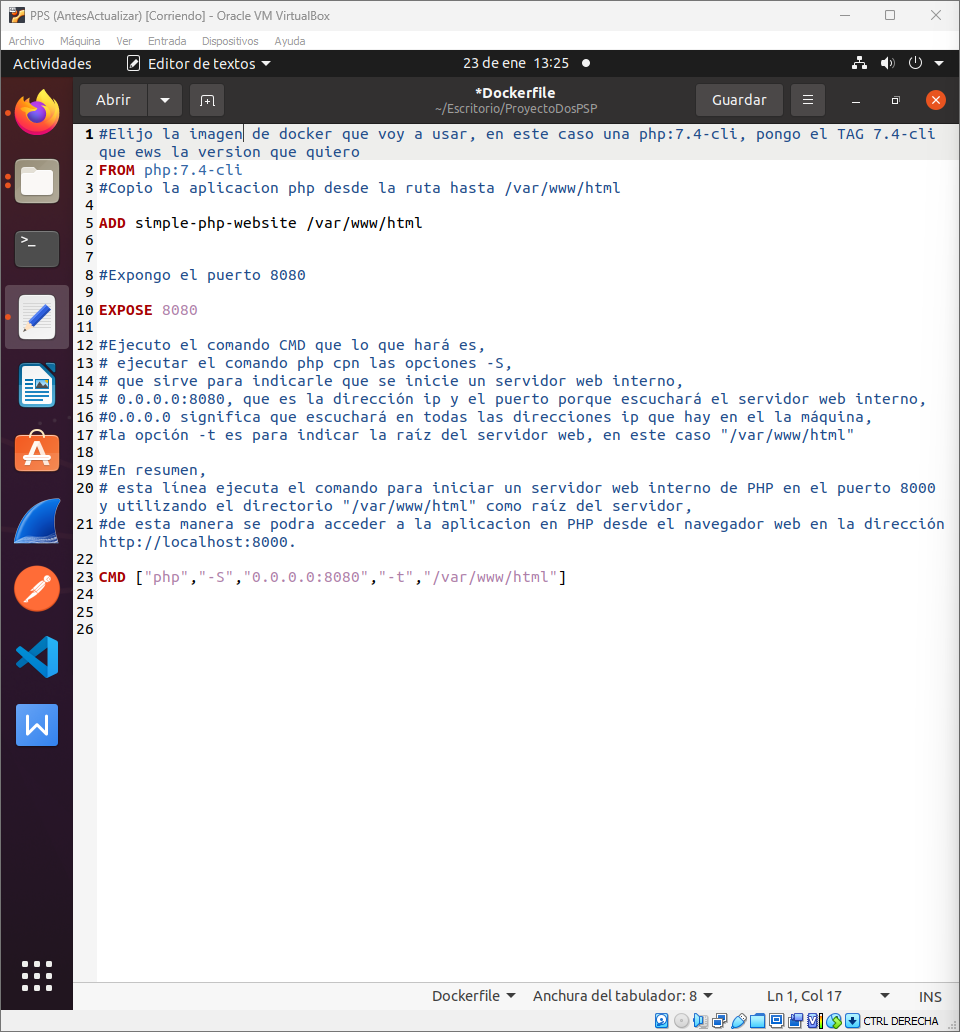
Ahora me descargaré una aplicación web sencilla:



Y la descargaré haciendo un git clone:



Este sería el Dockerfile para este apartado:



#Elijo la imagen de docker que voy a usar, en este caso una php:7.4-cli, pongo el TAG 7.4-cli que es la versión que quiero

FROM php:7.4-cli

#Copio la aplicación php desde la ruta hasta /var/www/html

ADD simple-php-website /var/www/html

#Expongo el puerto 8080

EXPOSE 8080

#Ejecuto el comando CMD que lo que hará es,

# ejecutar el comando php con las opciones -S,

# que sirve para indicarle que se inicie un servidor web interno,

# 0.0.0.0:8080, que es la dirección ip y el puerto porque escuchará el servidor web interno,

#0.0.0.0 significa que escuchará en todas las direcciones ip que hay en el la máquina,

#la opción -t es para indicar la raíz del servidor web, en este caso "/var/www/html"

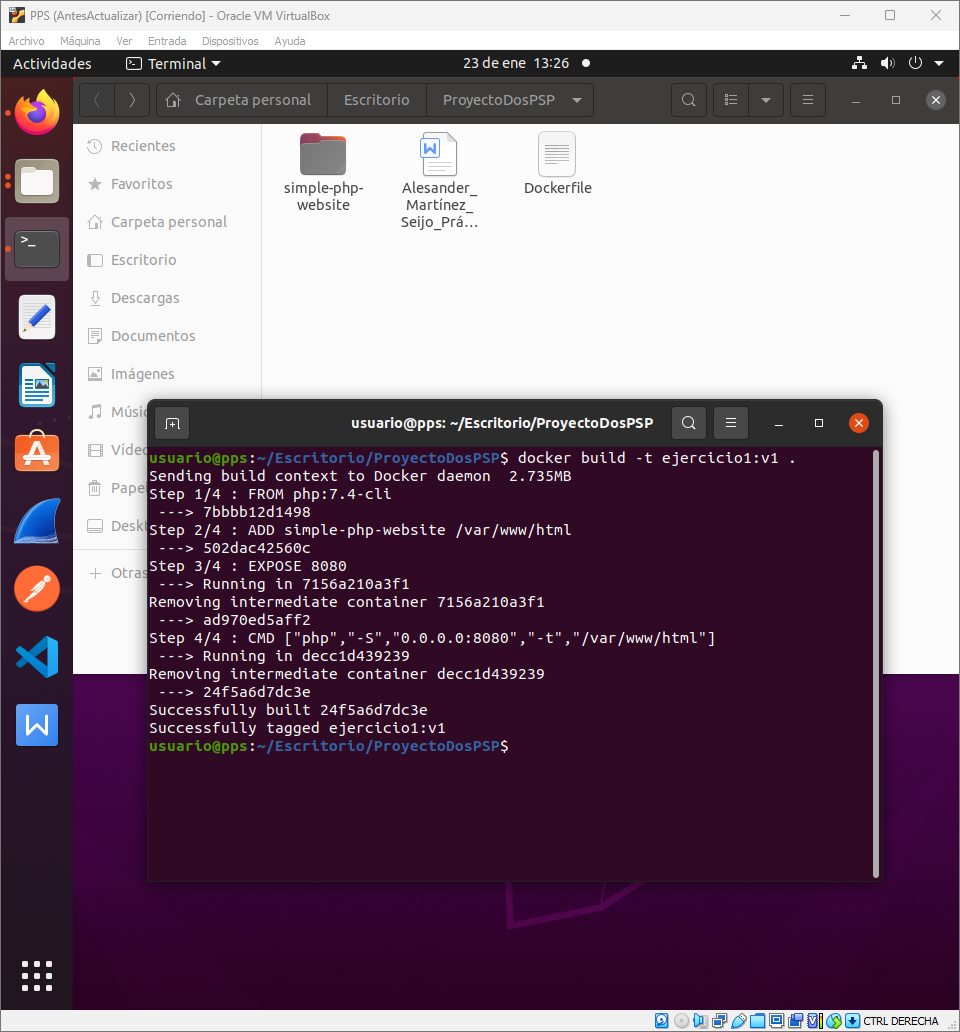
#En resumen,

# esta línea ejecuta el comando para iniciar un servidor web interno de PHP en el puerto 8000 y utilizando el directorio "/var/www/html" como raíz del servidor,

#de esta manera se podrá acceder a la aplicación en PHP desde el navegador web en la dirección http://localhost:8000.

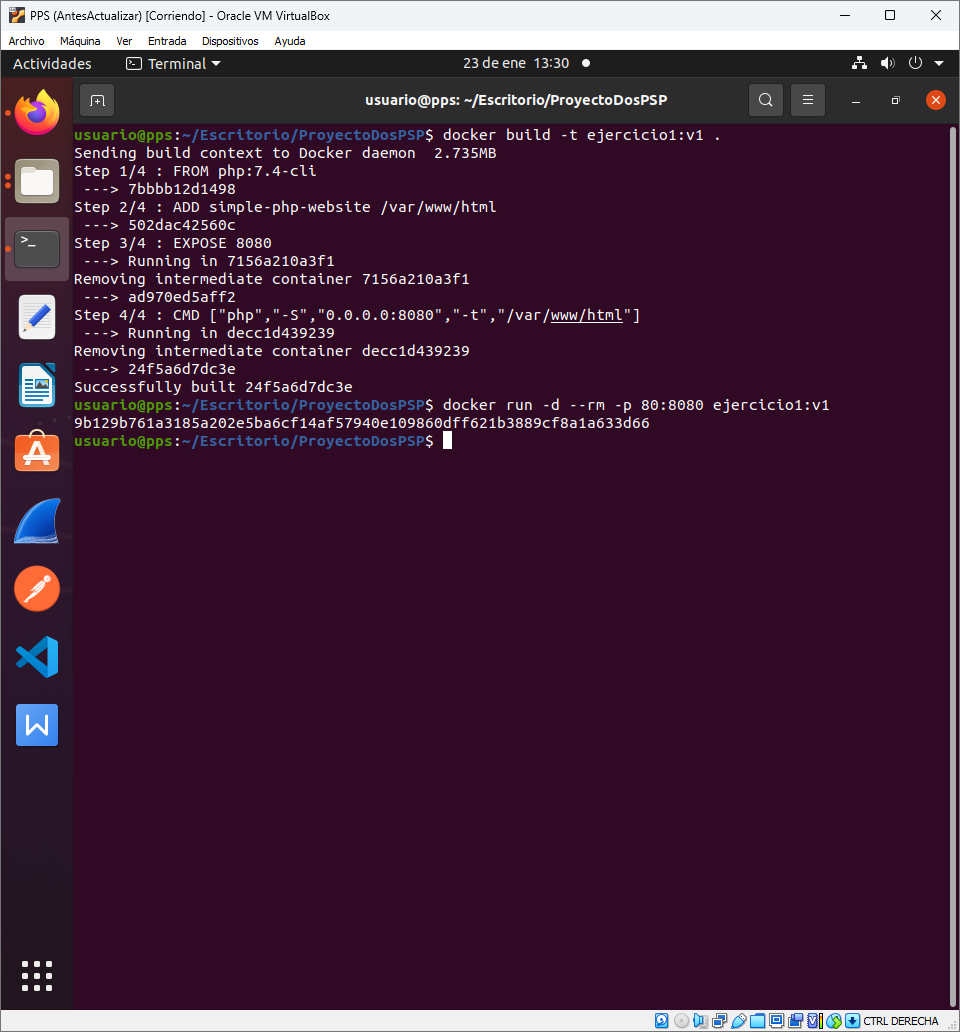
CMD ["php","-S","0.0.0.0:8080","-t","/var/www/html"]

Para crear el contenedor a partir del Dockerfile, usaré el comando “***docker build -t ejercicio1:v1 .”***.

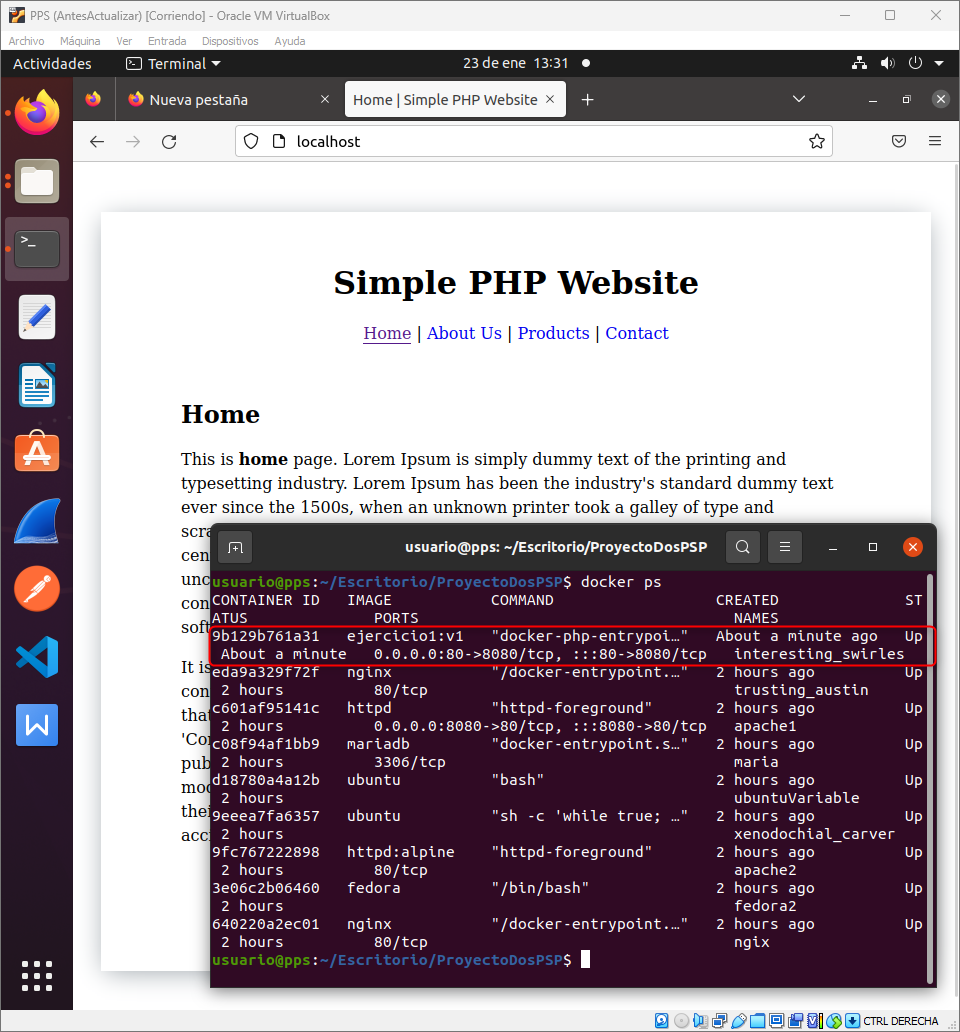


Ahora para crear una imagen con este contenedor, ejecutaré el comando: “***docker run -d –rm -p 80:8080 ejcercicio1:v1”***.

La opción run es para crear un contenedor y lanzarlo, la -d es para que se ejecute en modo demonio, es decir en segundo plano, --rm es para que cuando se cierre se elimine, -p es para decir que quiero que el puerto 8080 de la imagen de Docker se mapee al puerto 80 de la máquina real y por último el nombre de la imagen.



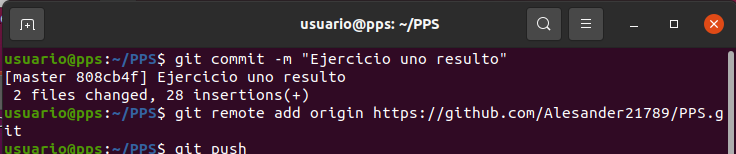


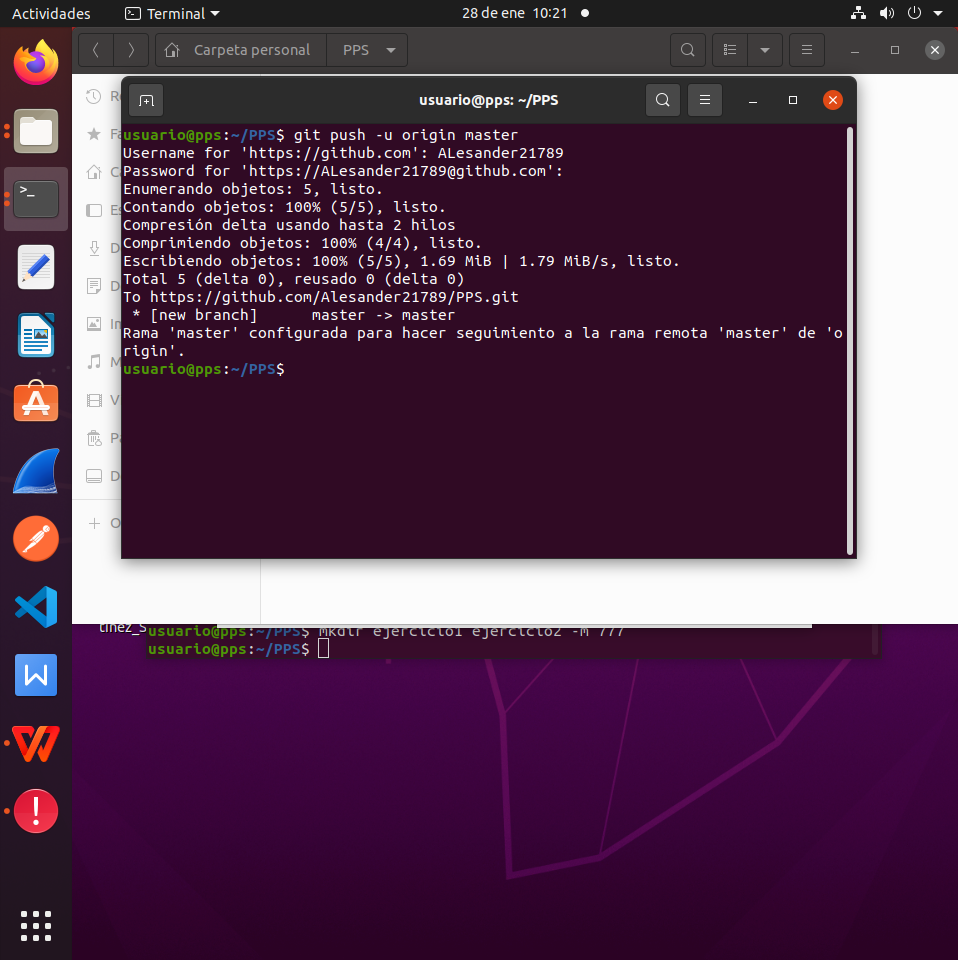


Por último, en la carpeta donde tengo el proyecto ejecutaré los siguientes comandos:

Ahora ***“git commit -m “Ejercicio uno resuelto”***, para guardar los cambios en el repositorio local.

Y ahora ejecutaré el comando de git: “***git push -u origin master”***, con esto subiré los cambios al repositorio remoto.





## Crea un Dockerfile que partiendo de una imagen Ubuntu genera una imagen que:

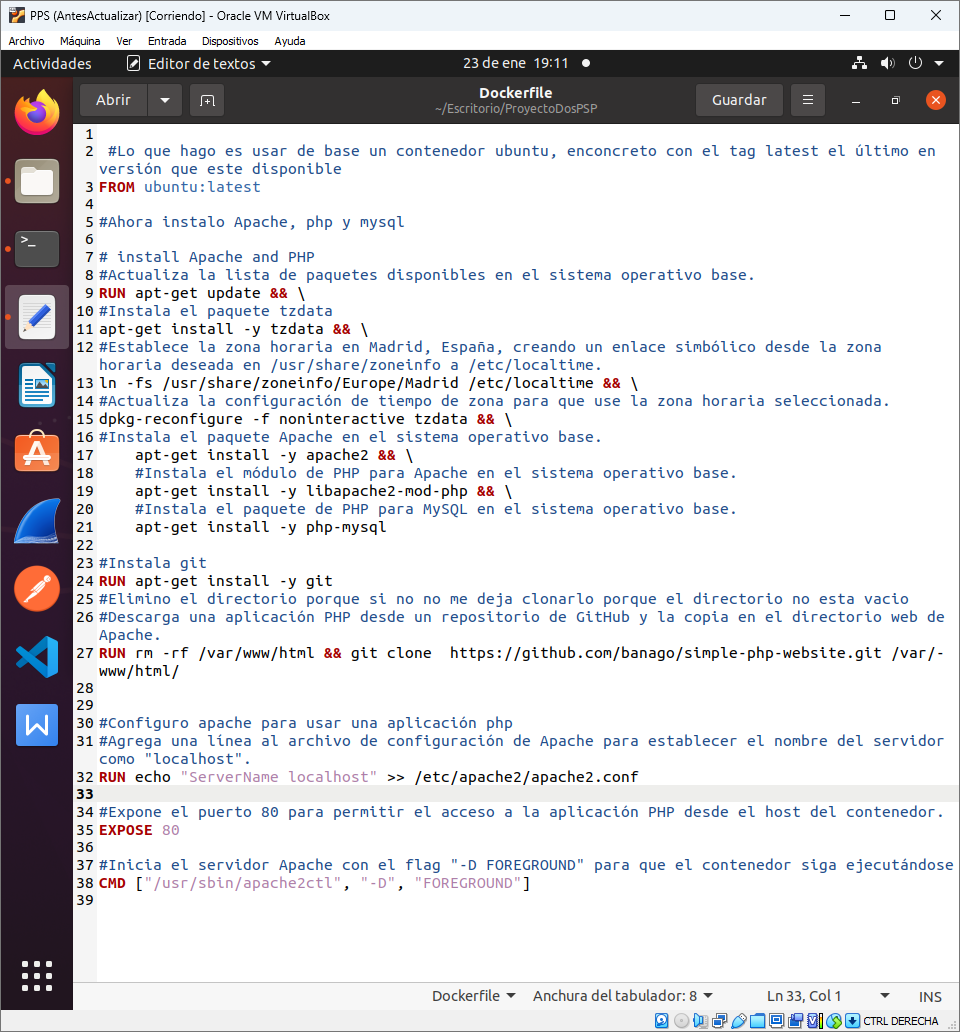
### Instala Apache, de forma que se exponga el puerto 80.

### Instala PHP.

### Copia una aplicación web en PHP al directorio de Apache que expone las páginas web.

Esta aplicación se debe descargar automáticamente mediante algún comando como *git clone* o *curl*. Para facilitar las cosas, debe de ser una aplicación sencilla que no empleebases de datos (ya que si no también habría que instalar un MySQL).

Para hacer todo lo requerido creo este Dockerfile:



#Lo que hago es usar de base un contenedor ubuntu, en concreto con el tag latest el último en versión que esté disponible

FROM ubuntu:latest

#Ahora instalo Apache, php y mysql

# install Apache and PHP

#Actualiza la lista de paquetes disponibles en el sistema operativo base.

RUN apt-get update && \

#Instala el paquete tzdata

apt-get install -y tzdata && \

#Establece la zona horaria en Madrid, España, creando un enlace simbólico desde la zona horaria deseada en /usr/share/zoneinfo a /etc/localtime.

ln -fs /usr/share/zoneinfo/Europe/Madrid /etc/localtime && \

#Actualiza la configuración de tiempo de zona para que use la zona horaria seleccionada.

dpkg-reconfigure -f noninteractive tzdata && \

#Instala el paquete Apache en el sistema operativo base.

apt-get install -y apache2 && \

#Instala el módulo de PHP para Apache en el sistema operativo base.

apt-get install -y libapache2-mod-php && \

#Instala el paquete de PHP para MySQL en el sistema operativo base.

apt-get install -y php-mysql

#Instala git

RUN apt-get install -y git

#Elimino el directorio porque si no me deja clonarlo porque el directorio no está vacío

#Descarga una aplicación PHP desde un repositorio de GitHub y la copia en el directorio web de Apache.

RUN rm -rf /var/www/html && git clone https://github.com/banago/simple-php-website.git /var/www/html/

#Configuro apache para usar una aplicación php

#Agrega una línea al archivo de configuración de Apache para establecer el nombre del servidor como "localhost".

RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf

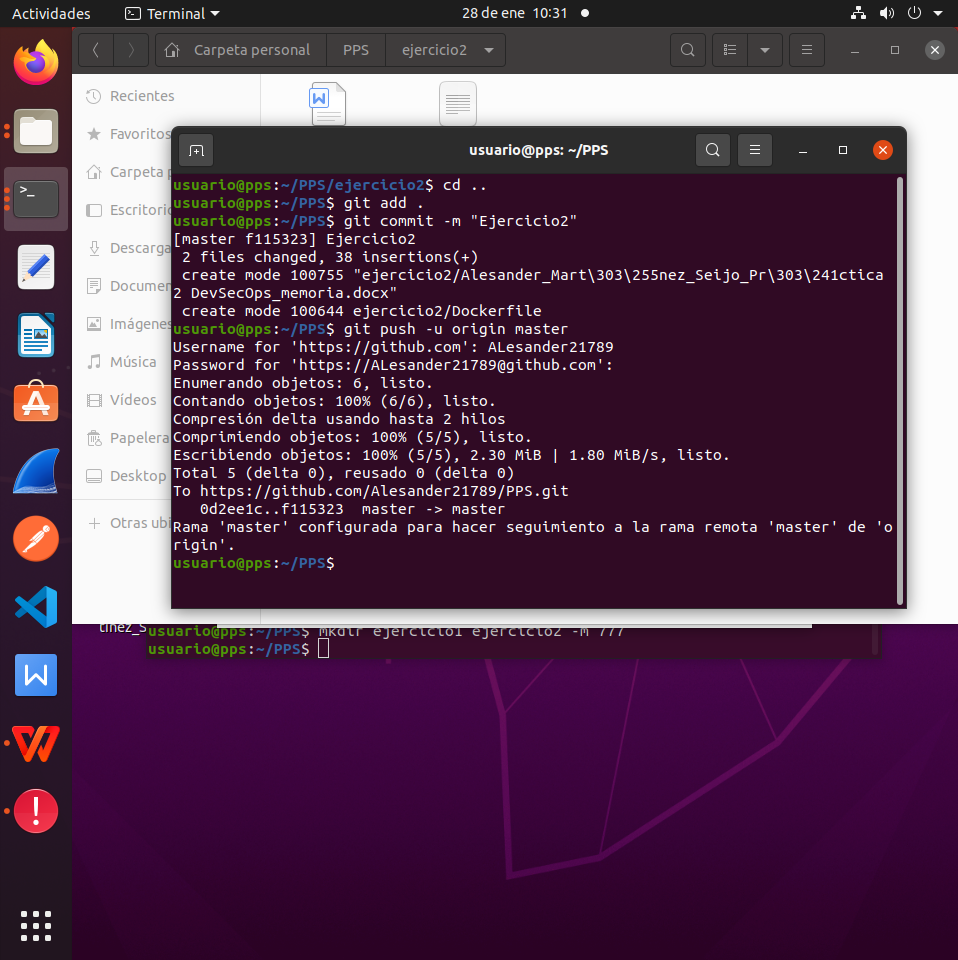
#Expone el puerto 80 para permitir el acceso a la aplicación PHP desde el host del contenedor.

EXPOSE 80

#Inicia el servidor Apache con el flag "-D FOREGROUND" para que el contenedor siga ejecutándose

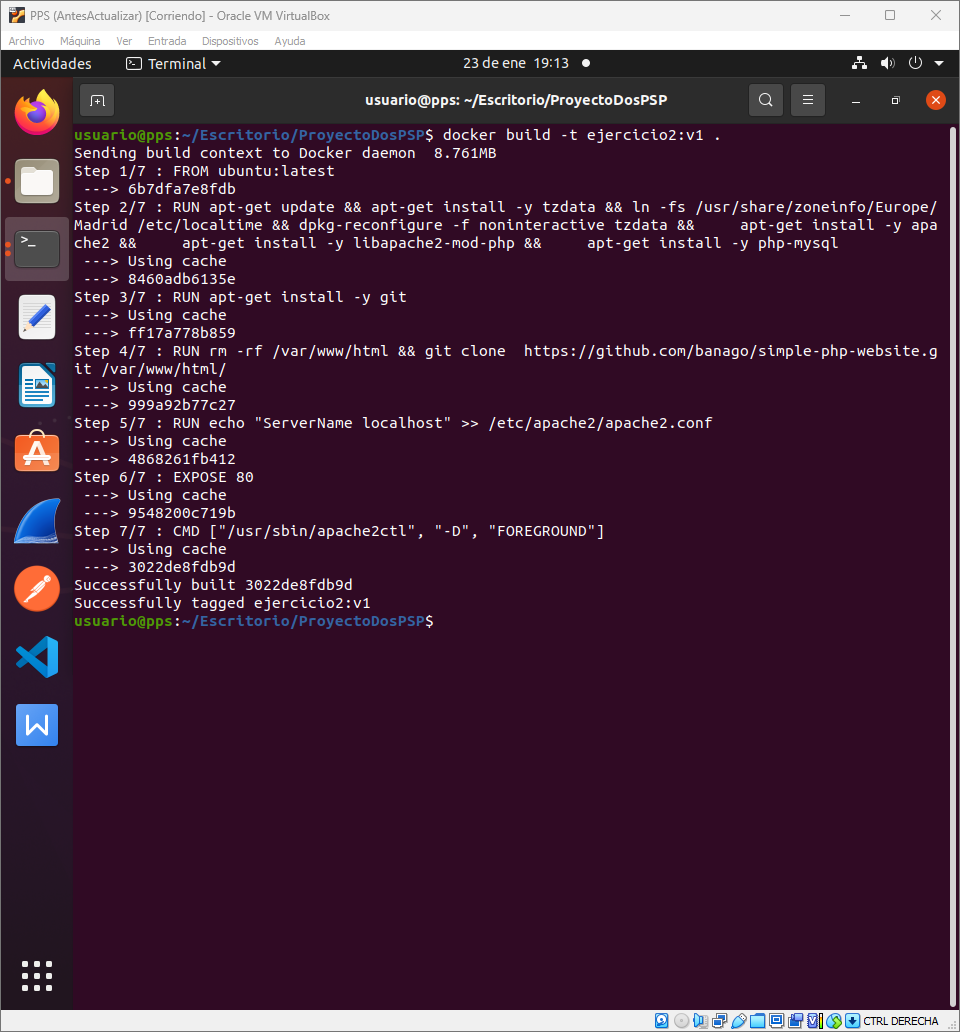
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]

Ahora ejecutaré un commit y un git push :



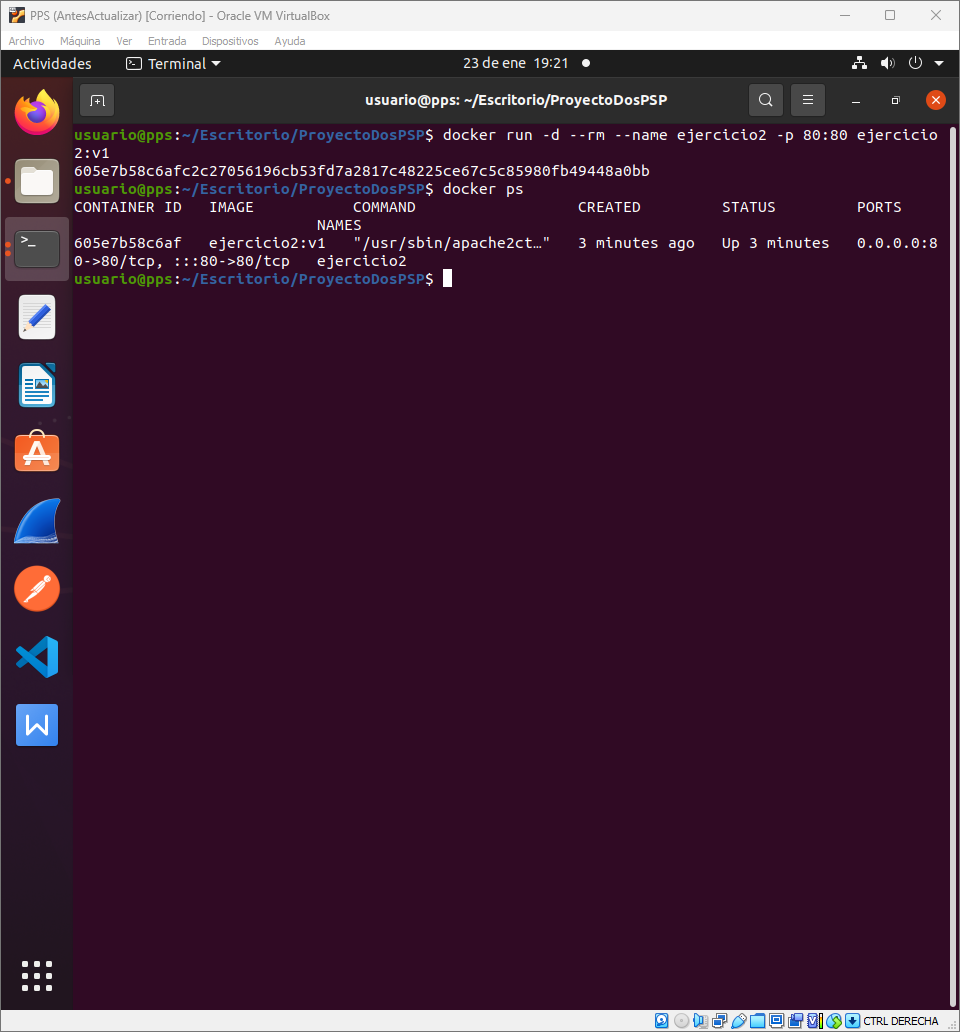
## Crea un contenedor para cada una de esas imágenes y verifica que funciona. Para y borra dicho contenedor.

Creo una imagen con el Dockerfile del ejercicio dos, para hacer esto hago uso del comando “***build”*** de docker, sirve para crear una imagen desde un Dockerfile, la agrego la opción -t que es para indicar el nombre de la imagen que en este caso será ejercicio2 y los dos puntos para indicar la versión que será v1, por último, la añado un punto para indicar donde está el Dockerfile.

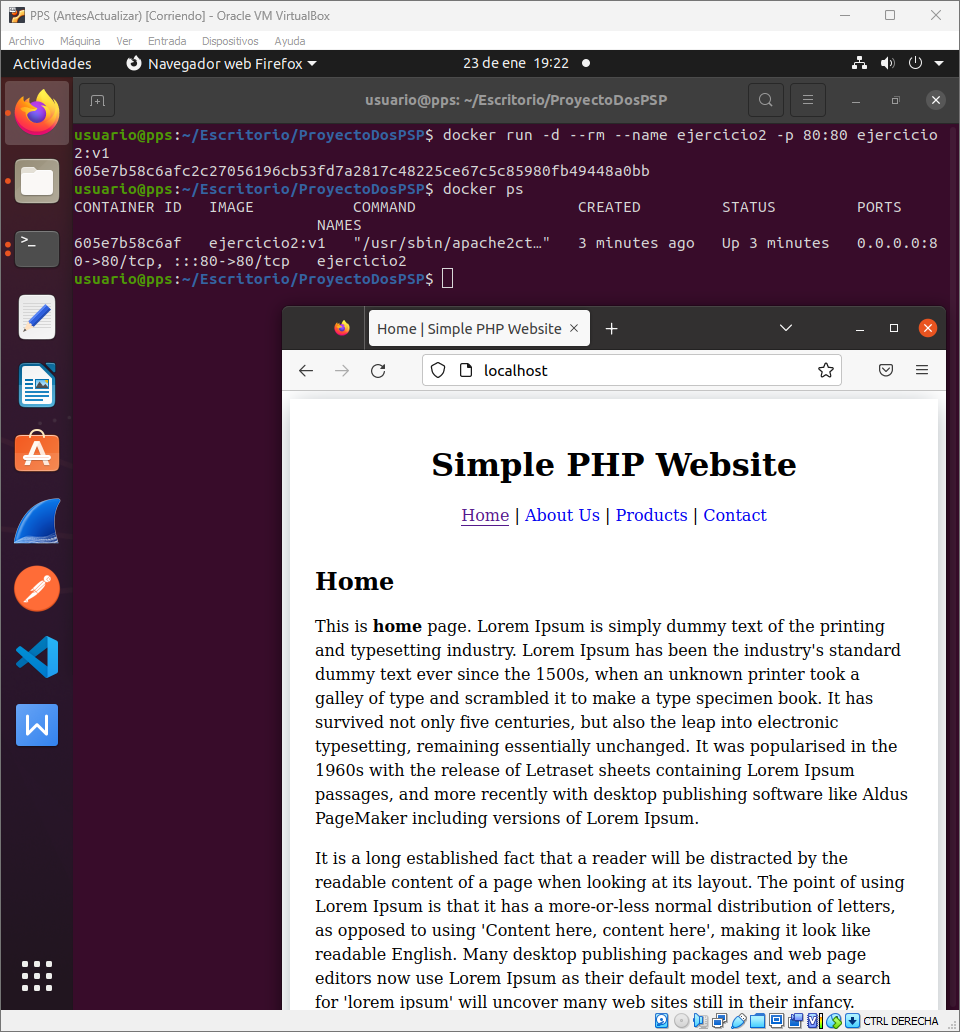


Ahora procederé a crear la imagen, usaré el comando de “***docker run ”*** que sirve para descargar una imagen e inicializarla, en este caso como la imagen que voy a usar está en local no tendrá que bajarse nada, las opciones del comando son; -d para que se inicie como un demonio, es decir en segundo plano, --rm para que cuando se apague se borre, -- name para indicar el nombre de la imagen que en este caso será ejercicio2, -p par mapear el puerto 80 de la máquina real con el 80 del contendor y por último el nombre de la imagen que voy usar que en este caso es ejercicio2 y la versión v1 de esta.

Ahora compruebo si está ejecutándose con el comando Docker ps, que sirve para ver los contenedores en ejecución:



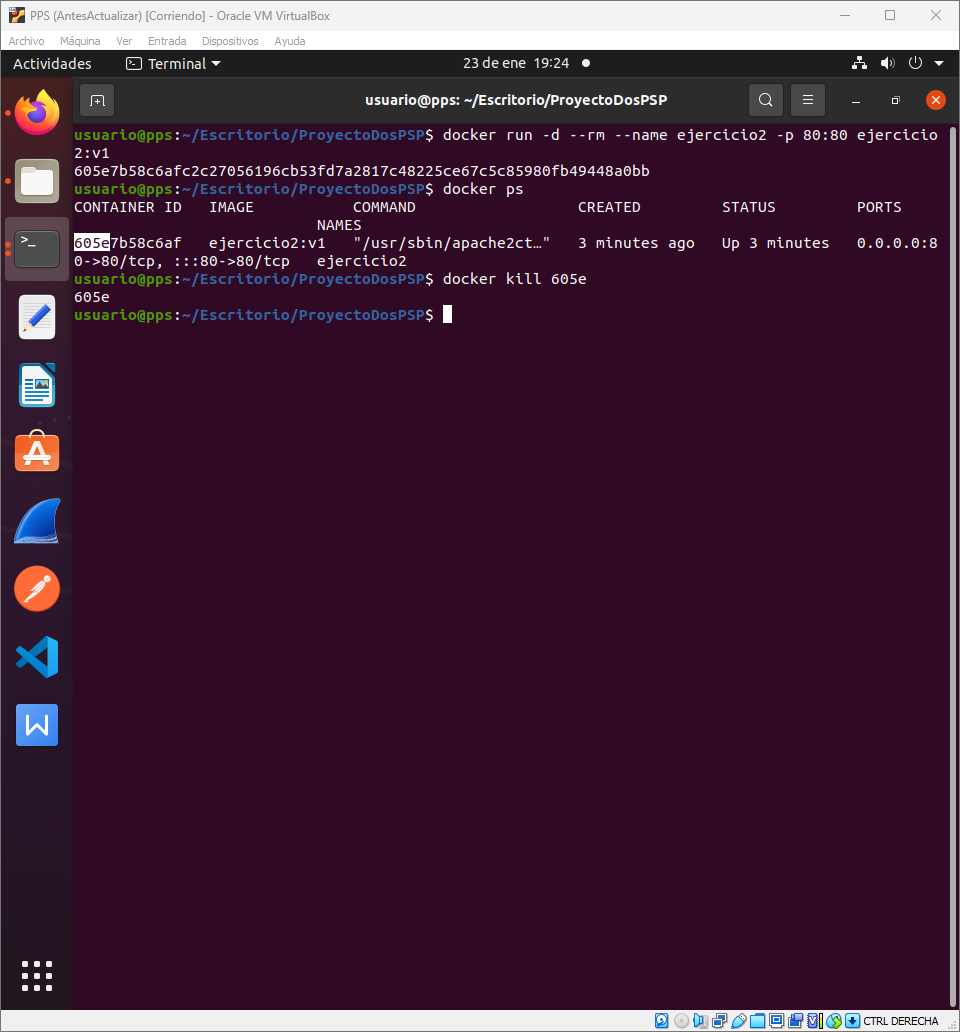
Ahora abro el navegador y pongo localhost:80 para ver si el servidor está funcionando.



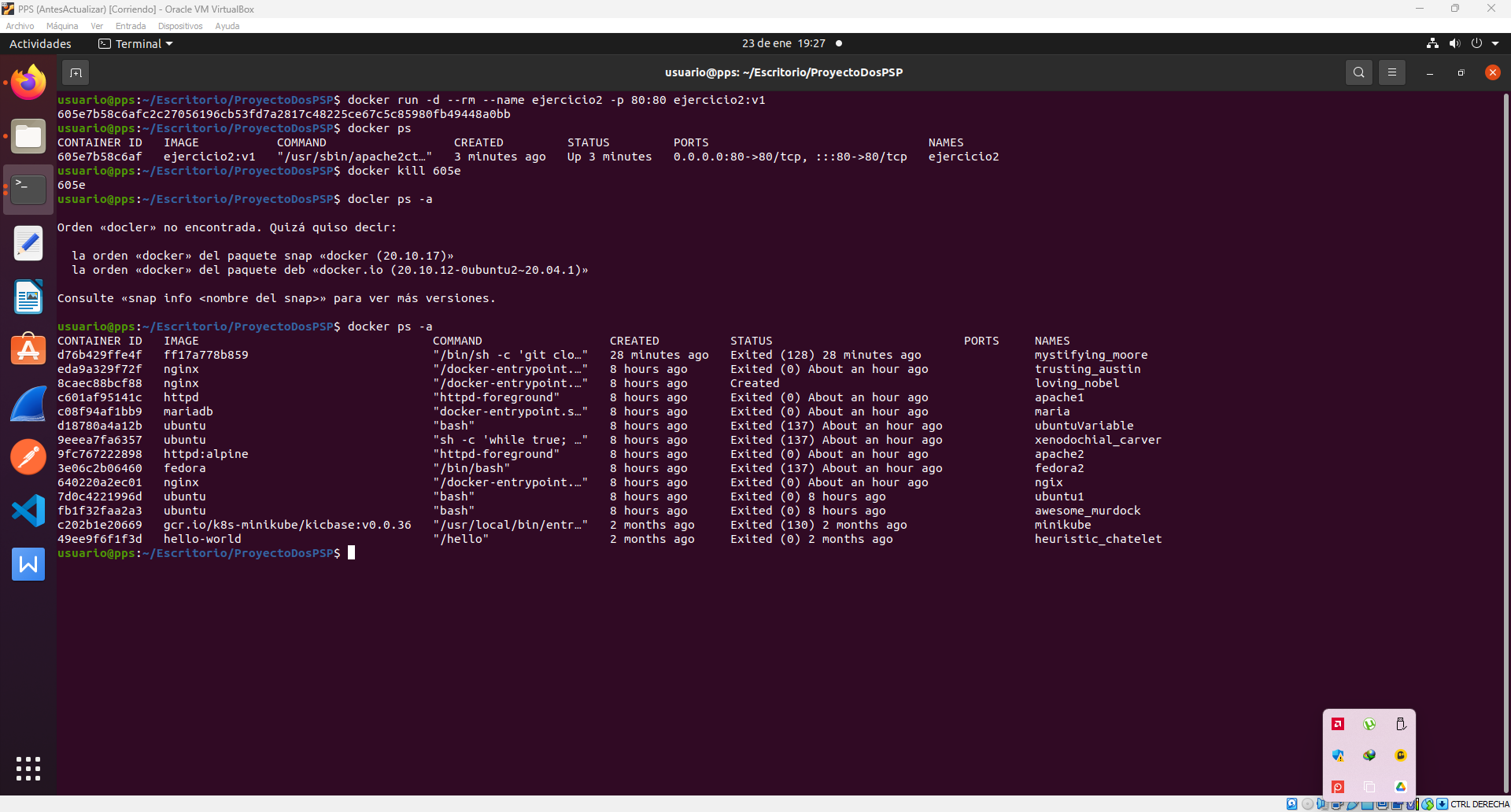
Se puede ver que está funcionando correctamente.

Para eliminar este contenedor tengo dos opciones:

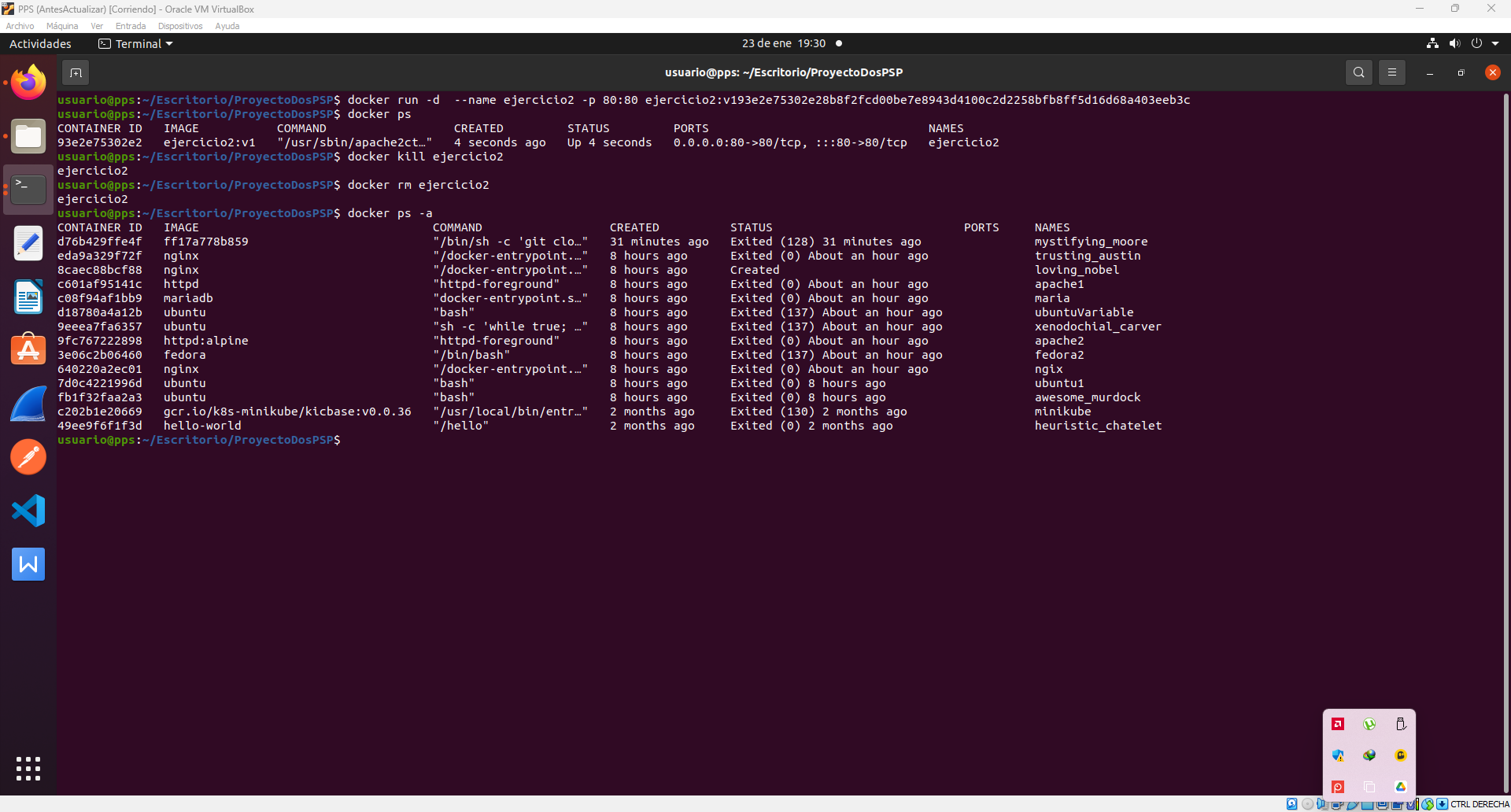
1. Si yo lo cierro por la opción –rm que le puse, se eliminará, para cerrarlo usaré el comando de “***docker kill”*** usando los cuatro primeros números del identificador del contenedor:



Ahora comprobaré que no existe ningún contenedor con el id anterior, esto lo haré con el comando “***dokcer ps -a”***, que me muestra todos los contenedores:



1. La otra manera es sin poner la opción –rm, entonces lo que tendría que hacer es hacer un Docker kill para cerrar el contenedor, porque si está corriendo no me dejará eliminarlo, y por último un docker rm poniendo después el id del contenedor o su nombre, entonces voy a volver a crear otro contenedor sin mesa opción y a probar este método:



La parte correspondiente del Dockerfile del ejercicio uno ya la hice antes.

Ahora lo que tengo que hacer es un git commit y git push.