**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

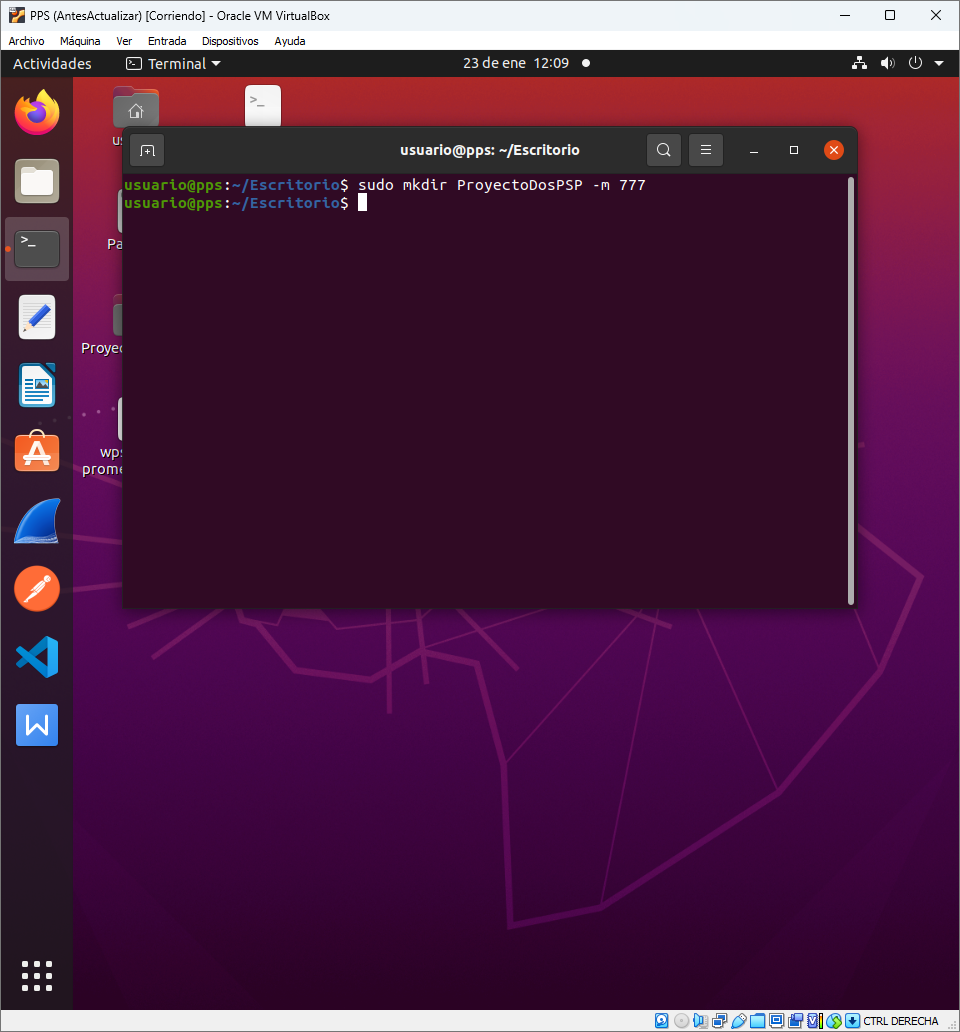
razonables no han logrado su propósito.

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

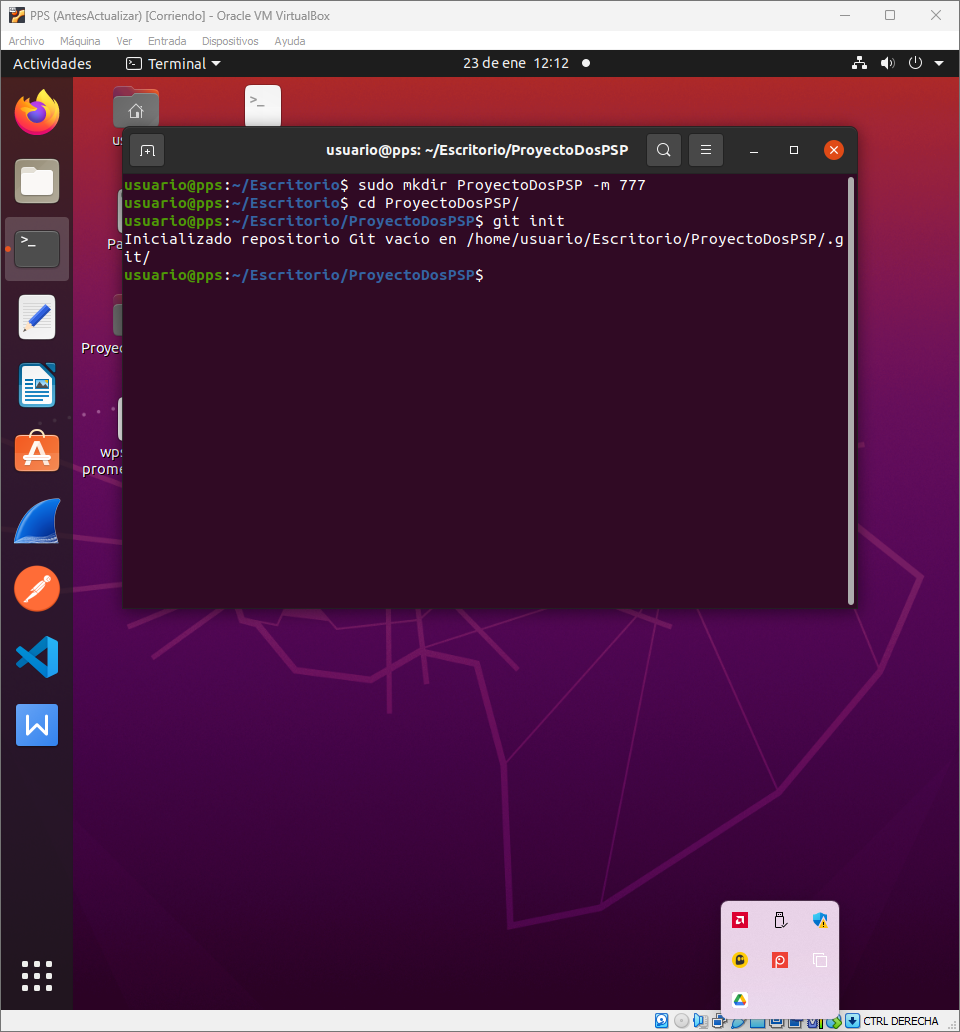
**Docker**

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

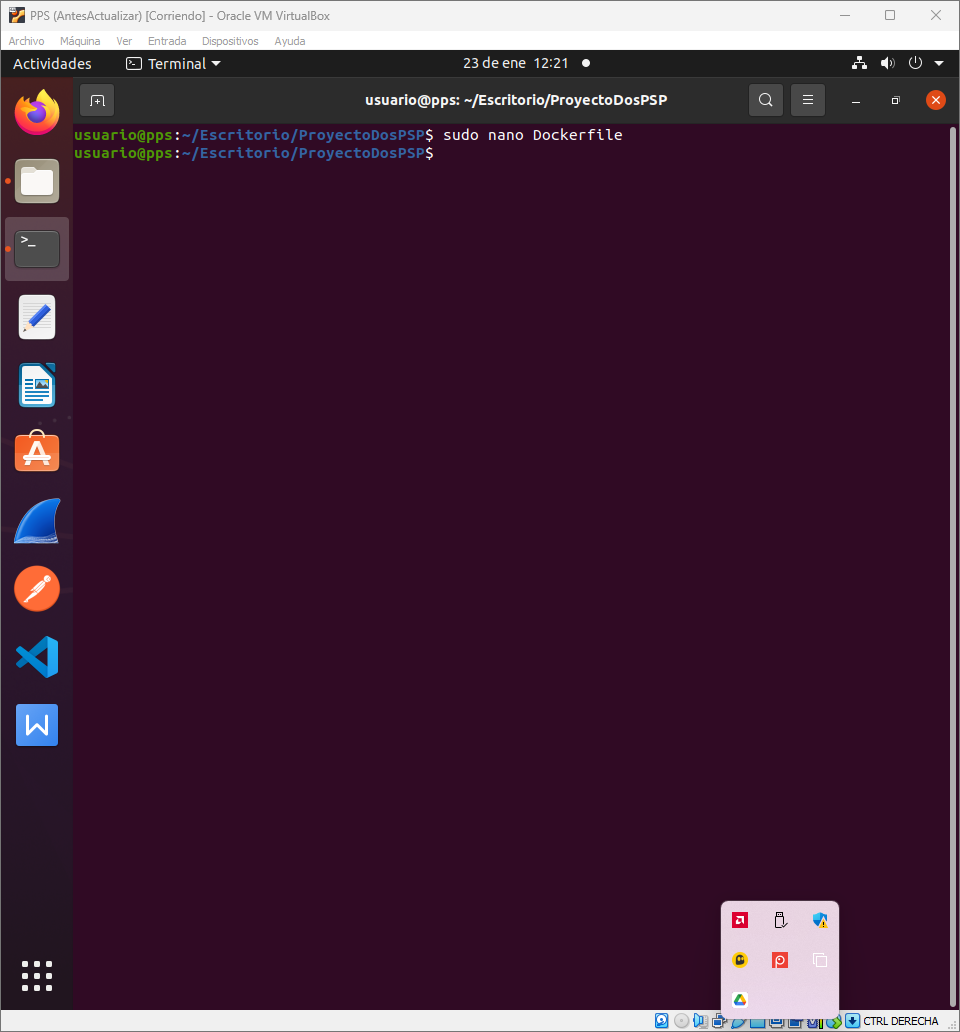
Primero voy crear la carpeta en donde voy iniciar el repositorio de GitHub local, esto la haré con el comando *“sudo mkdir ProyectoDosPSP -m 777”*, la opción -m sirve para cambiar los permisos del directorio creado.

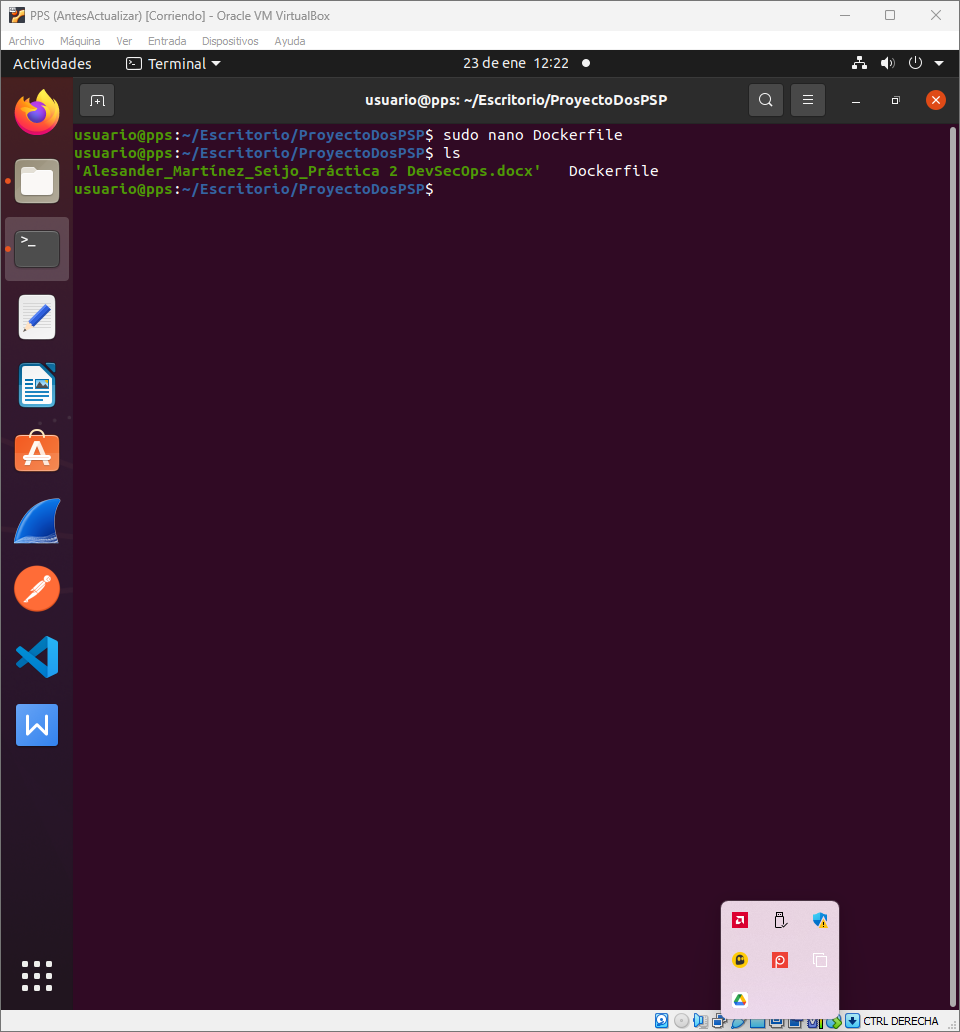


Ahora iniciaré el repositorio de GitHub localmente, esto la haré ejecutando el comando *“git init” Y “git add”*,el comando inicializa el repositorio el local el la carpeta donde estoy.

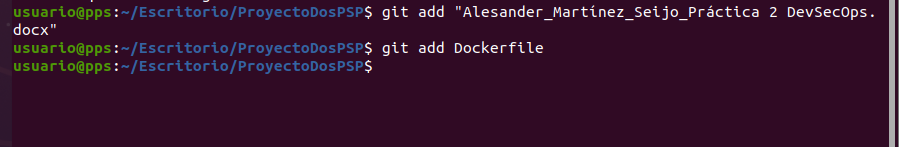


Ahora creo el Dockerfile y añado el archivo de Word.



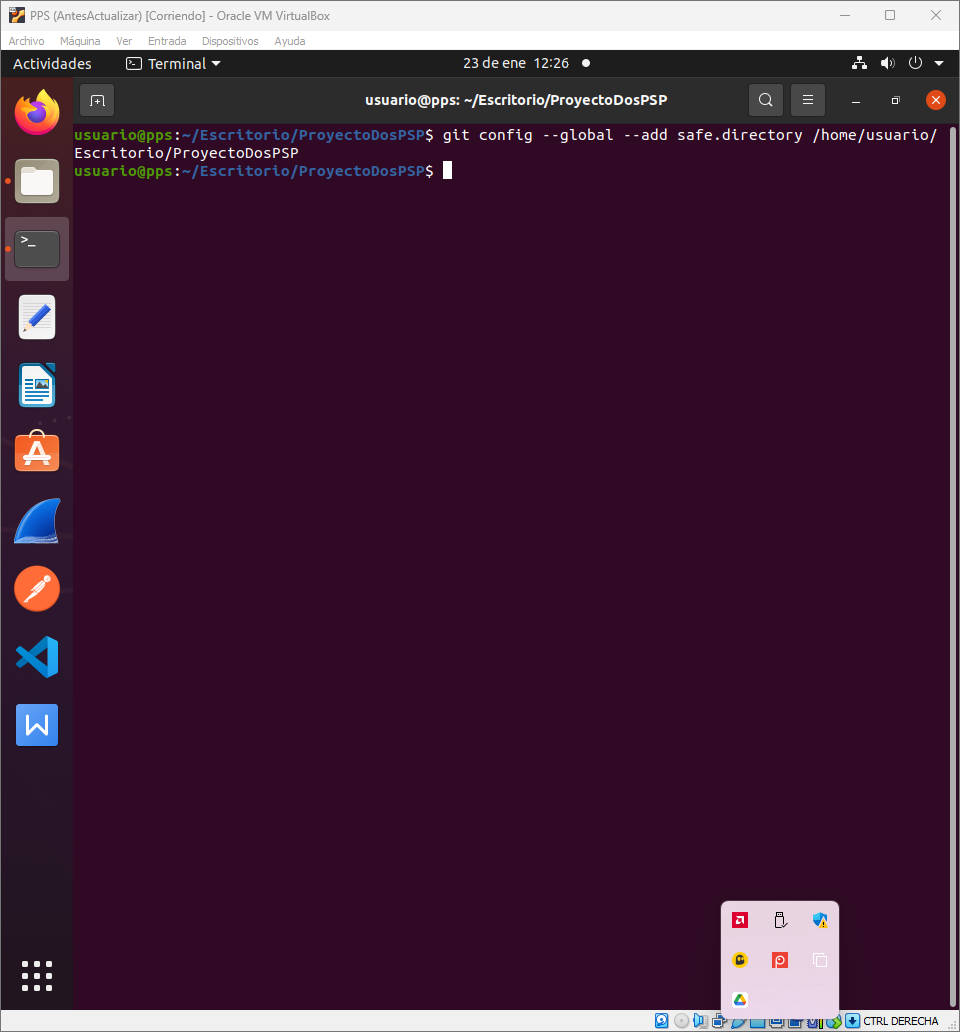


Y hago el *“git add”*, añadiendo los archivos.

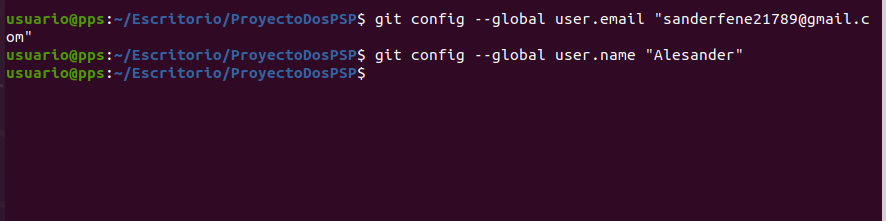


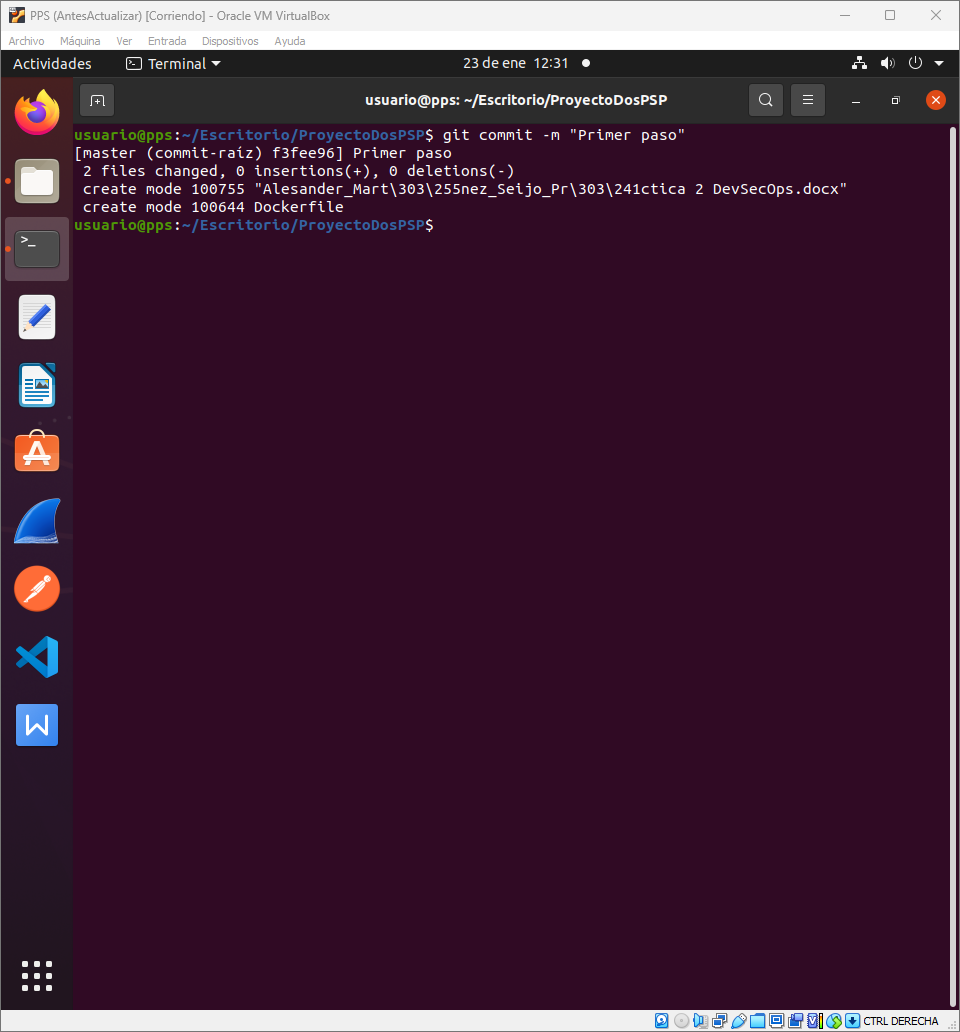
Ahora voy suar el comando de git *“git commit -m “Primer paso”*, esto lo que hará es guardar los cambios en el repositorio local.

Antes de ejecutar este comando ejecuto otro: *“git config --global --add safe.directory /home/usuario/Escritorio/ProyectoDosPSP”*, 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.

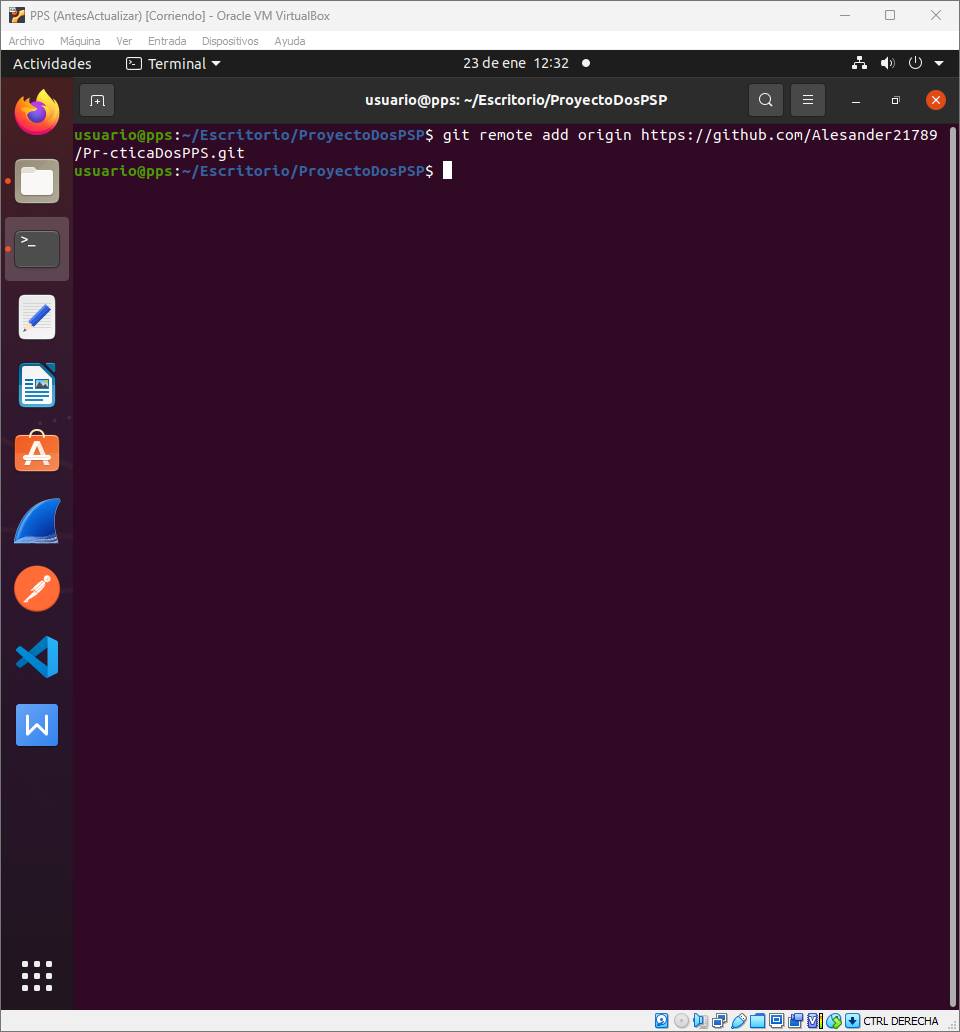


Tengo también que añadir mi correo y mi usuario.



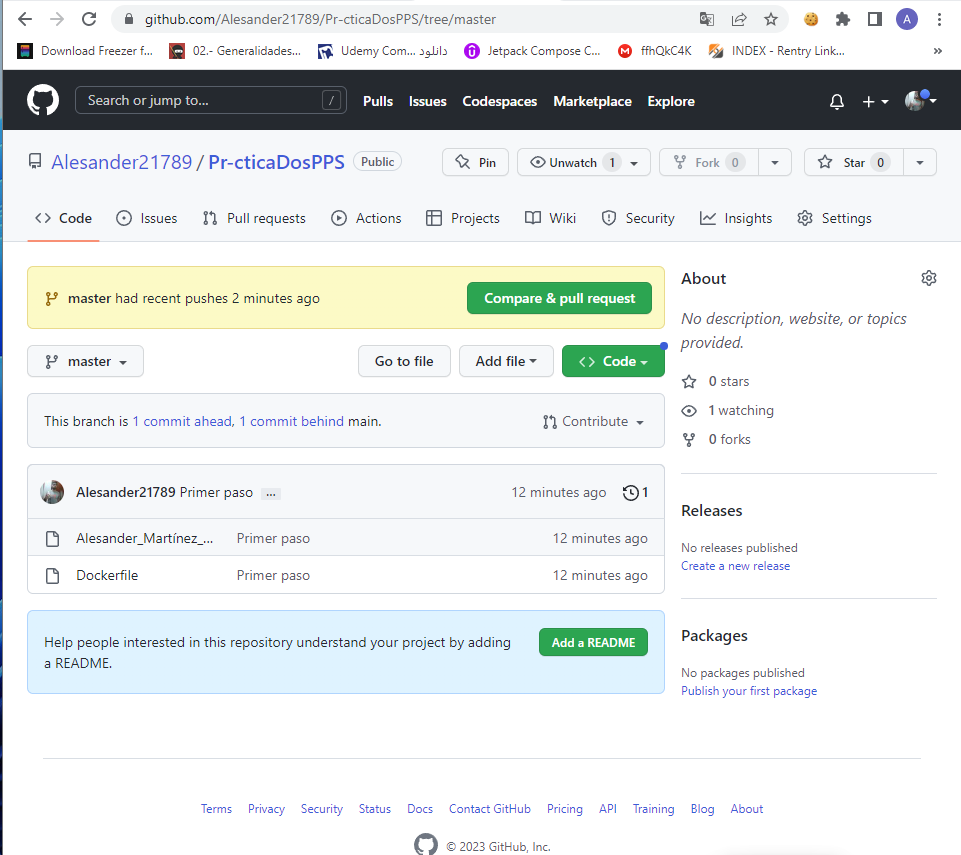


Ahora asociaré el repositorio local con el remoto, esto se hará con el comando *“git remote add origin y la url del repositorio”*.



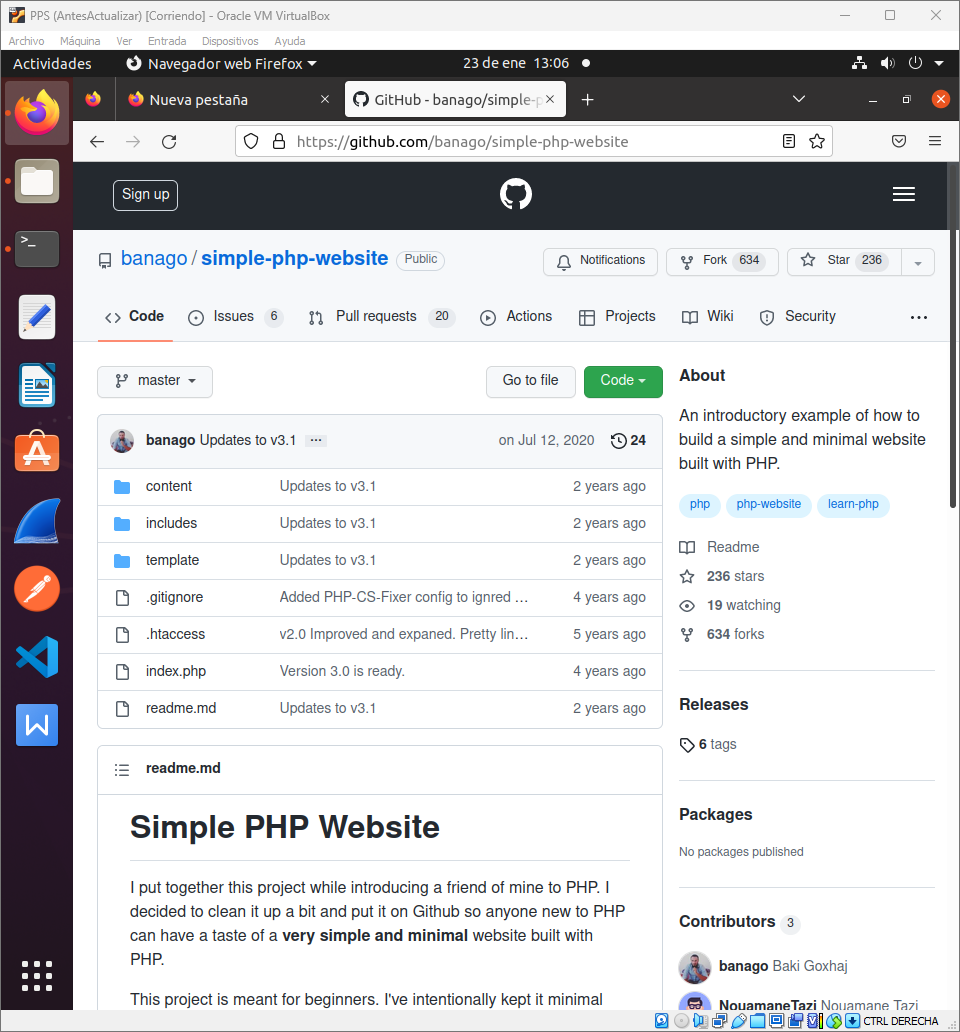
Y ahora ejecutaré el comando de git: *“git push -u origin master”*, con esto subiré los cambios al repositorio remoto.



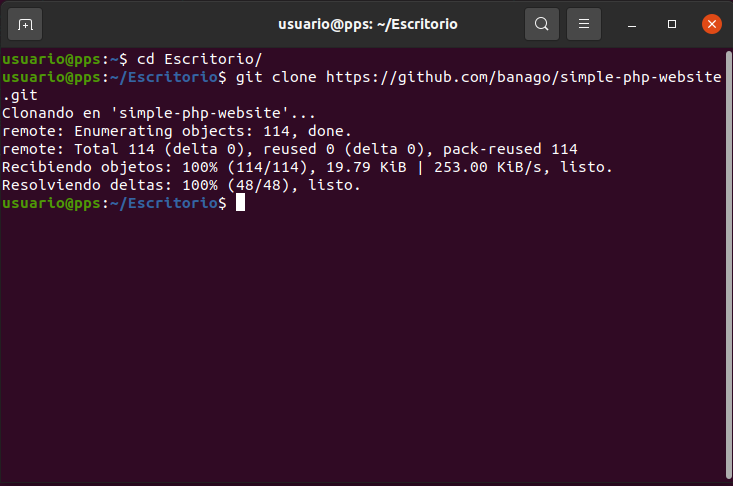


## Copia una aplicación en PHP a un directorio del contenedor. Esta aplicación se debe copiar directamente desde un directorio del anfitrión. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).

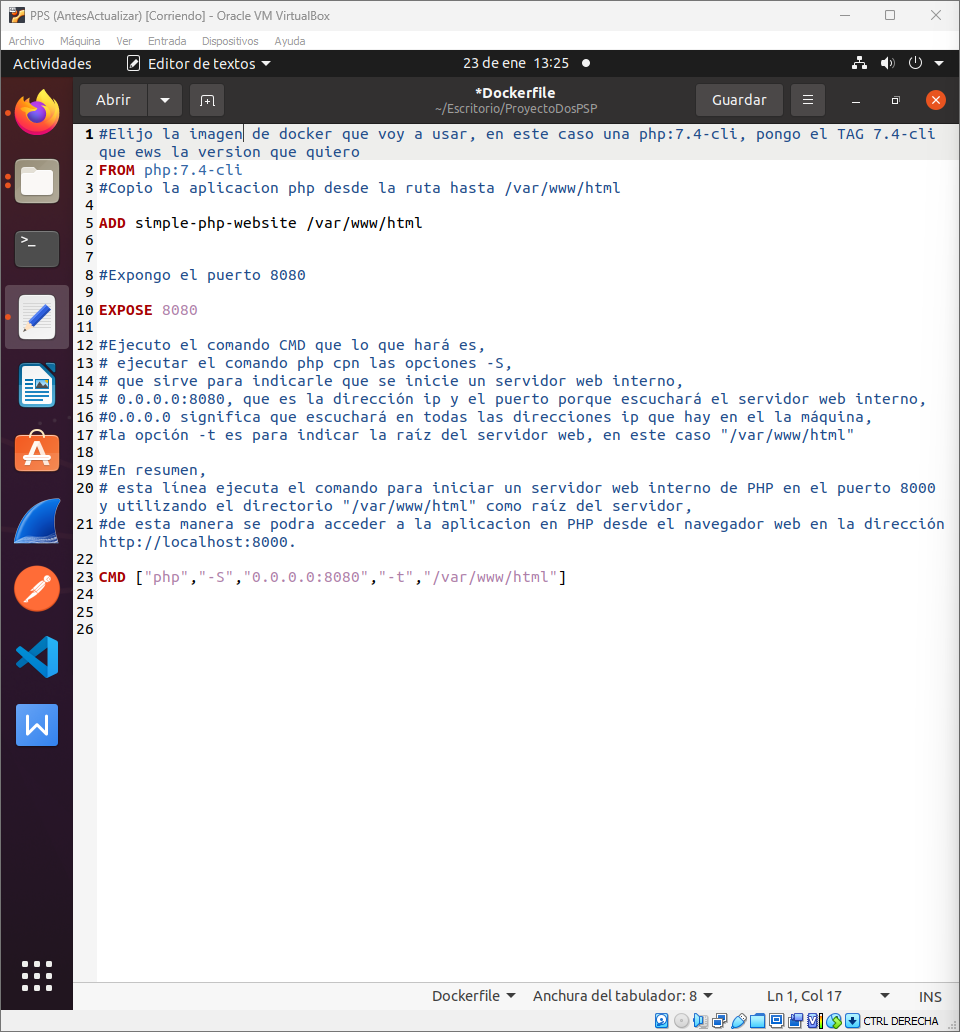
Ahora me descargaré una aplicación web sencilla:



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 ews la version que quiero

FROM php:7.4-cli

#Copio la aplicacion 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 cpn 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"

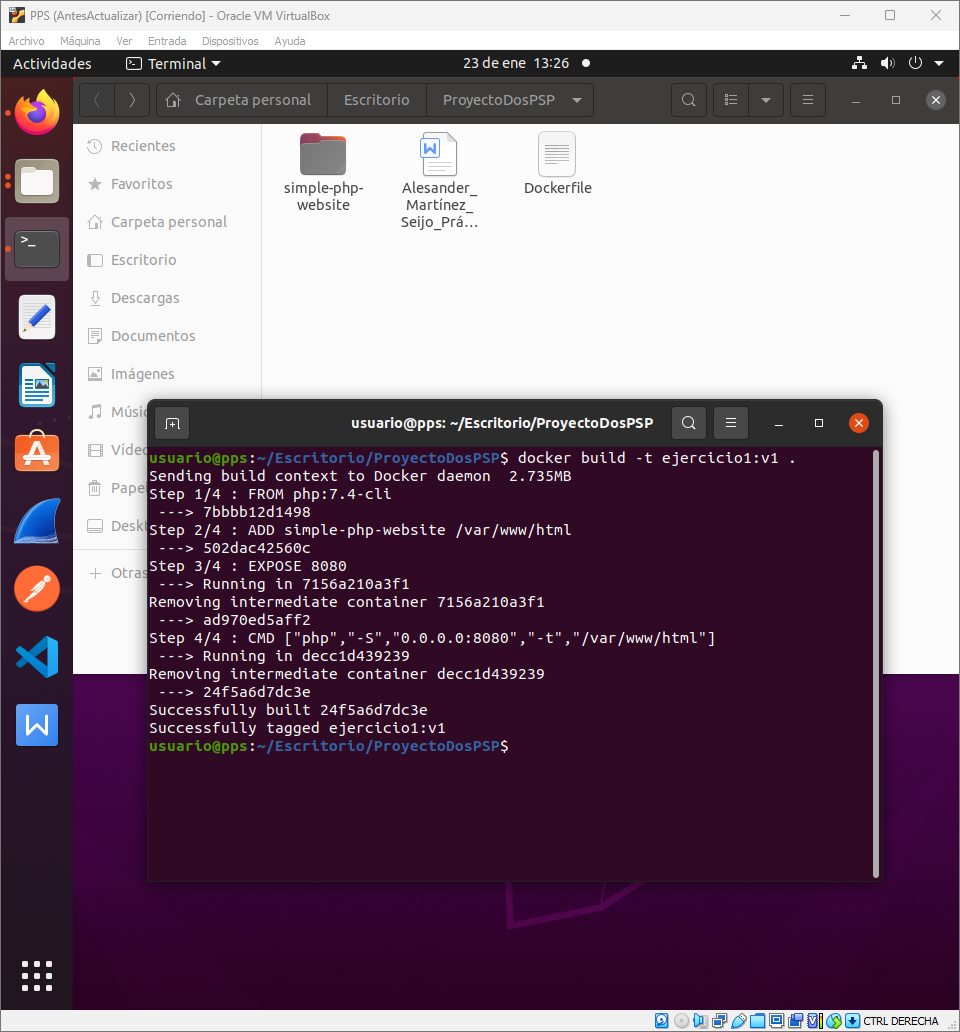
#En resumen,

# 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 podra acceder a la aplicacion en PHP desde el navegador web en la dirección http://localhost:8000.

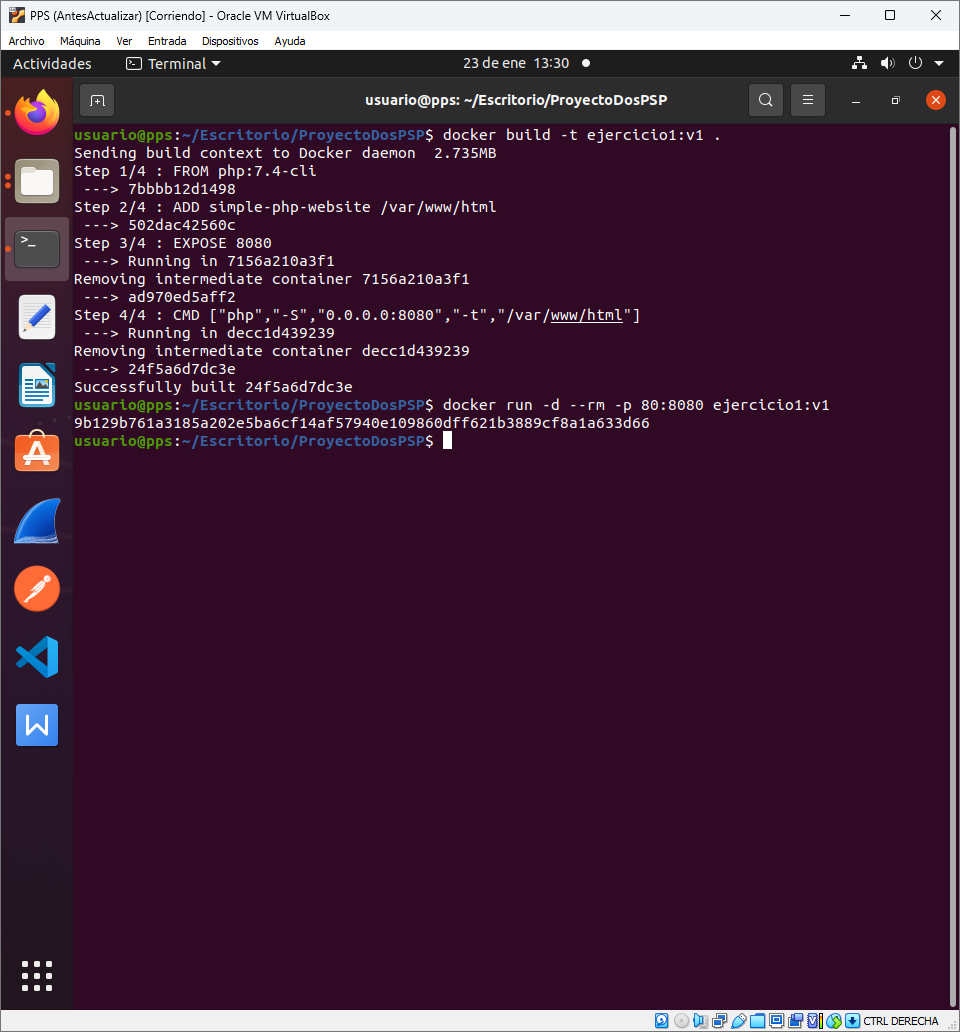
CMD ["php","-S","0.0.0.0:8080","-t","/var/www/html"]

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

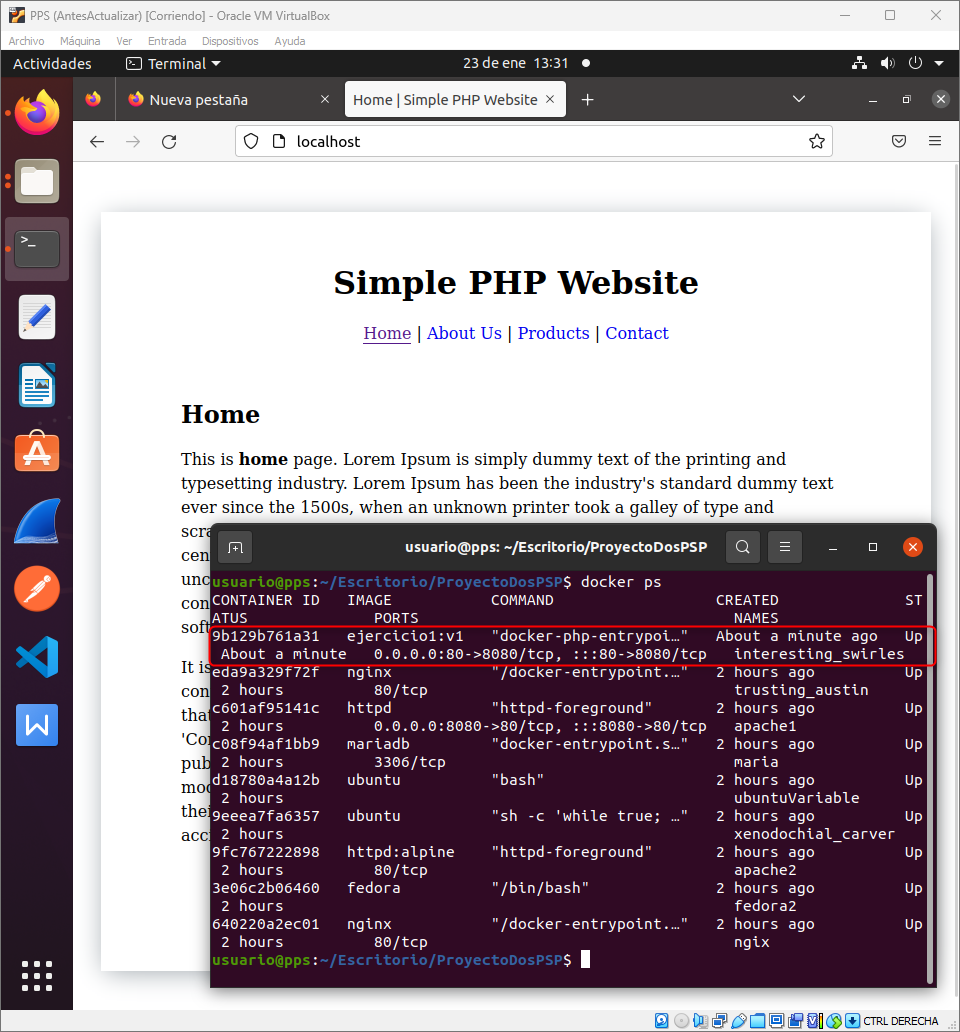


Ahora para crear una imagen con este contenedor, ejecutaré el comando: *“docker run -d –rm -p 80:8080 ejcercicio1:v1”*.

La opción run es para crear la imagen y lanzarla, 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 del contenedor.







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

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

Y *“git push”*, para subir 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).

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

## Emplea un comando para lanzar 20 contenedores de la segunda imagen, cada uno mapeado en un puerto distinto del anfitrión. Cuando veas que funcionan, para y borra dichos contenedores.

**Git**

# Mantén un repositorio de Git público con:

## • Memoria de la práctica.

## • El fichero Dockerfile.

## Haz un *commit* y un *push* del proyecto cada vez que superes uno de los ítems de la sección anterior.