

Preparando GitHub

Primeramente, iremos a GitHub y crearemos un nuevo repositorio público

Con el repositorio ya creado, podemos dirigirnos a nuestra máquina ubuntu para subir los archivos.

Una vez en nuestra máquina Linux, podemos utilizar el comando `git config --list` para ver el nombre de usuario y el email que tengamos guardados. Si estos no son los esperados, podemos cambiarlos con los comandos

```
git config --global user.name nombreusuario
git config --global user.email emailusuario
```

Ahora crearemos una nueva carpeta con el mismo nombre que nuestro repositorio (por comodidad) y ejecutaremos el comando `git init`.

```
usuario@pps:~$ mkdir DevSecOps
usuario@pps:~$ cd DevSecOps/
usuario@pps:~/DevSecOps$ git init
Iniciado repositorio Git vacío en /home/usuario/DevSecOps/.git/
```

Una vez hecho esto, ejecutaremos el comando

```
git remote add origin https://github.com/usuario/proyecto.git
git remote add origin https://github.com/Fernandolv123/DevSecOps.git en mi caso
```

Con el proyecto git ya preparado, podemos añadir algún fichero de ejemplo. Dado que no he creado el fichero 'README', lo crearé para subirlo.

```
usuario@pps:~/DevSecOps$ echo "Proyecto 2 PPS" > README.txt
usuario@pps:~/DevSecOps$ cat README.txt
Proyecto 2 PPS
```

Una vez creado el fichero, haciendo uso del comando `"git status"`, nos lo marcará como archivo sin seguimiento.

```
usuario@pps:~/DevSecOps$ git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
  README.txt

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
```

Para cambiar esto, es tan fácil como hacer `"git add"` y volver al comando anterior para comprobar que esté en seguimiento.

```
usuario@pps:~/DevSecOps$ git add README.txt
usuario@pps:~/DevSecOps$ git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
  (usa "git rm --cached <archivo>..." para sacar del área de stage)
  nuevos archivos: README.txt
```

Para realizar el commit, utilizaremos el comando `"git commit"` con el parámetro `-m` para añadir un comentario.

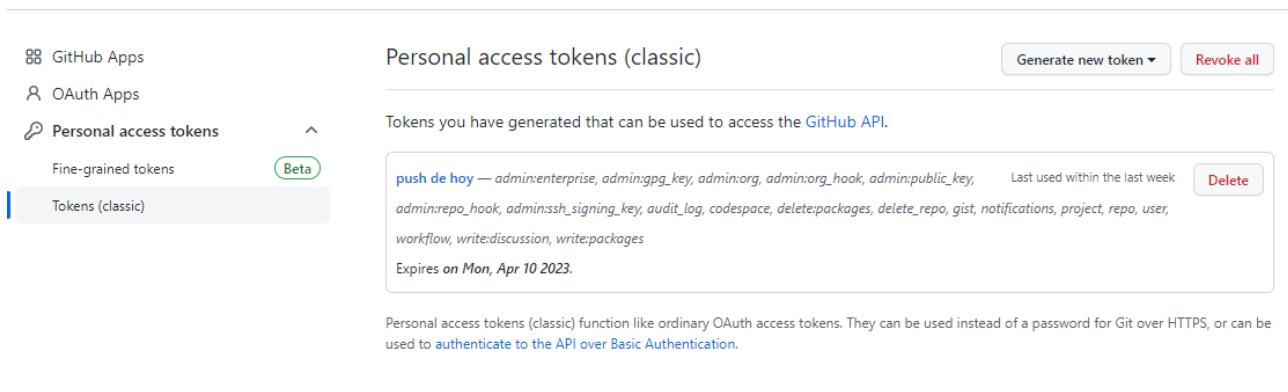
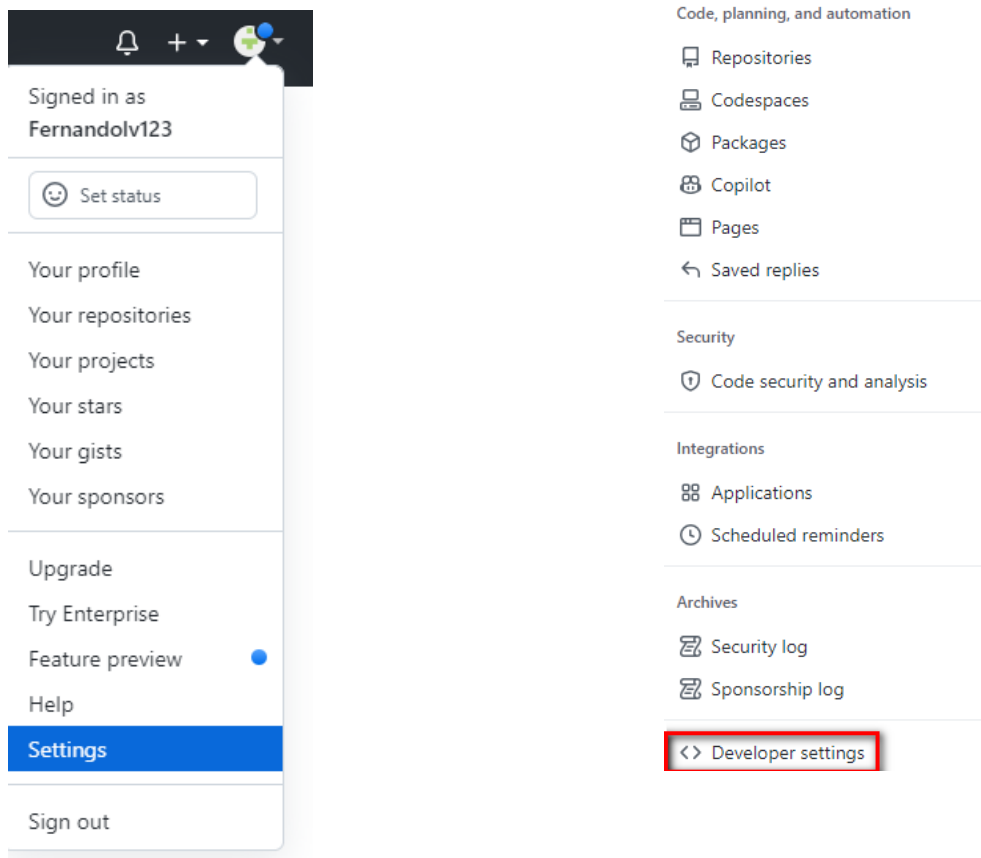
```
usuario@pps:~/DevSecOps$ git commit -m "Fichero README añadido"
[master (commit-raíz) 1420dce] Fichero README añadido
1 file changed, 1 insertion(+)
create mode 100644 README.txt
```

(Si durante este proceso cometemos un error, podemos ayudarnos del comando `git reset` para sacar el fichero del commit).

Finalmente, tras este proceso, debemos subir nuestro commit a github con `"git remote add"` seguido del comando `"git push"`.

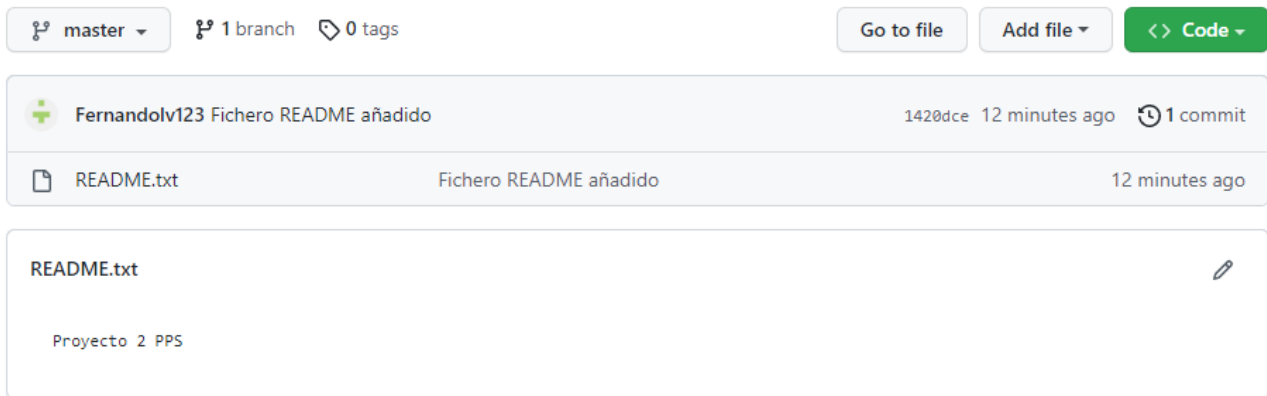
```
usuario@pps:~/DevSecOps$ git remote add DevSecOps http://github.com/Fernandolv123/DevSecOps.git
usuario@pps:~/DevSecOps$ git push DevSecOps
Username for 'https://github.com': Fernandolv123
Password for 'https://Fernandolv123@github.com':
warning: redirigiendo a https://github.com/Fernandolv123/DevSecOps.git/
Enumerando objetos: 3, listo.
Contando objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (3/3), 256 bytes | 128.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
To http://github.com/Fernandolv123/DevSecOps.git
 * [new branch]      master -> master
usuario@pps:~/DevSecOps$
```

Debemos utilizar un token en lugar de contraseña por motivos de seguridad. Para generar un token nos dirigiremos a la página de git hub y, sobre nuestro perfil, nos dirigiremos a “Settings → Developer settings → Personal access tokens → Tokens (classic)”



Cuando creamos un nuevo token, podemos especificarle los permisos deseados.

Finalmente, si nos dirigimos a nuestro repositorio en GitHub, podremos ver nuestro fichero añadido



Docker

PHP

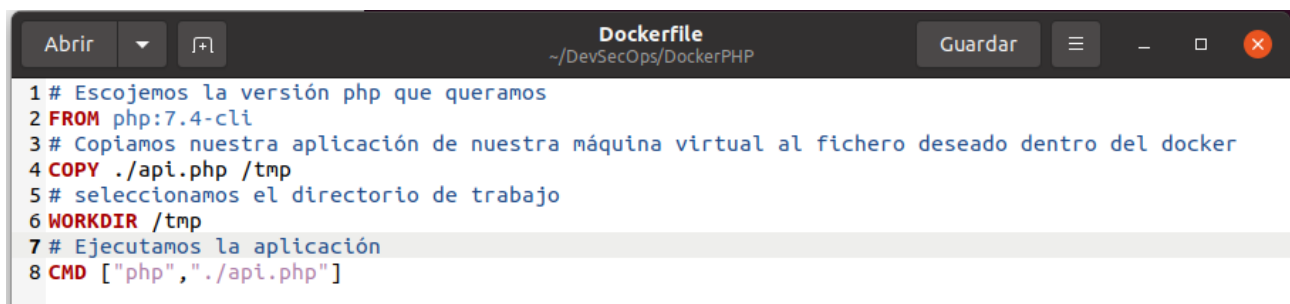
Primero de todo, crearemos una nueva carpeta

```
usuario@pps:~/DevSecOps$ ls
README.txt
usuario@pps:~/DevSecOps$ mkdir DockerPHP
usuario@pps:~/DevSecOps$ cd DockerPHP/
usuario@pps:~/DevSecOps/DockerPHP$
```

creamos una simple aplicación “Hello World” en php



y el fichero Dockerfile



Una vez tenemos ambas cosas, ejecutaremos el comando docker build para construir el docker a partir de la imagen

```
usuario@pps:~/DevSecOps/DockerPHP$ docker build -t dockerphp .
Sending build context to Docker daemon 3.072kB
Step 1/4 : FROM php:7.4-cli
7.4-cli: Pulling from library/php
a603fa5e3b41: Pull complete
c428f1a49423: Pull complete
156740b07ef8: Pull complete
fb5a4c8af82f: Pull complete
972155ae644b: Pull complete
a8e3b94fe6c1: Pull complete
c77004105467: Pull complete
d3e4898bfd25: Pull complete
30f377be4678: Pull complete
Digest: sha256:620a6b9f4d4feef2210026172570465e9d0c1de79766418d3affd09190a7fda5
Status: Downloaded newer image for php:7.4-cli
---> 7bbbb12d1498
Step 2/4 : COPY ./api.php /tmp
---> 0c213b38447f
Step 3/4 : WORKDIR /tmp
---> Running in 5b73dc2a4210
Removing intermediate container 5b73dc2a4210
---> f030e61aaf5c
Step 4/4 : CMD ["php", "./api.php"]
---> Running in 67545563044b
Removing intermediate container 67545563044b
---> 9cbac5763d3e
Successfully built 9cbac5763d3e
Successfully tagged dockerphp:latest
```

La primera vez que lo ejecutamos, nos descargará la versión necesitada por el Dockerfile.

Finalmente, utilizaremos el comando docker run para crear un contenedor

```
usuario@pps:~/DevSecOps/DockerPHP$ docker run --name contenedorphp7 dockerphp
Hello World
```

podemos utilizar parámetros como -d (para ejecutarlo como demonio) o -ti (para hacerlo interactivo).

Como podemos ver en la captura anterior, nuestra aplicación ha funcionado perfectamente.

Podemos utilizar el comando docker ps -a para ver todos nuestros contenedores creados y podemos eliminarlos con docker rm nombrecontenedor (aunque esto está fuera de la práctica).

Ahora crearé este pdf y haré el commit a GitHub con el mismo proceso seguido en el apartado anterior

Commit realizado

```
usuario@pps:~/DevSecOps$ git add DockerPHP/
usuario@pps:~/DevSecOps$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
nuevos archivos: DockerPHP/Dockerfile
nuevos archivos: DockerPHP/api.php
nuevos archivos: Fernando_A_Lorenzo_Vazquez_PPS_Ejercicio_2.pdf

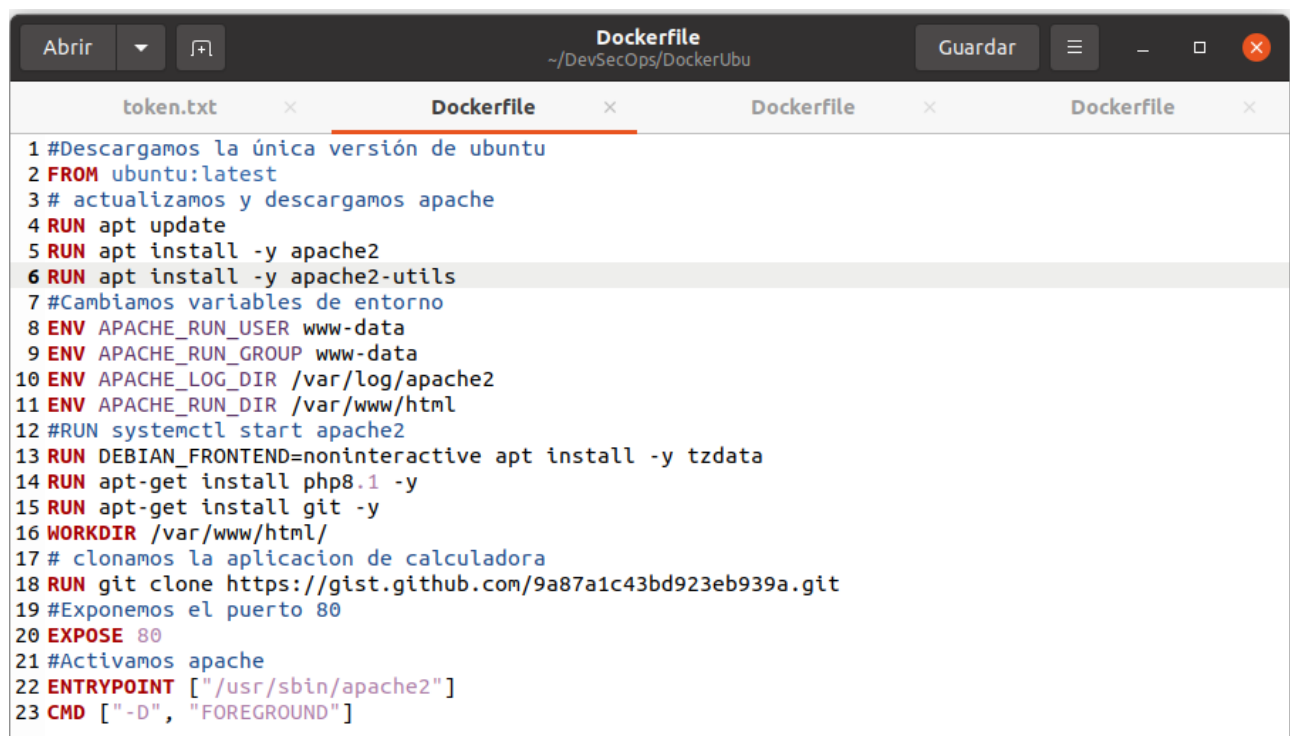
usuario@pps:~/DevSecOps$ git commit
Abortando commit debido que el mensaje está en blanco.
usuario@pps:~/DevSecOps$ git commit -m "Apartado 1 de Docker terminado"
[master 301deaf] Apartado 1 de Docker terminado
 3 files changed, 14 insertions(+)
 create mode 100644 DockerPHP/Dockerfile
 create mode 100644 DockerPHP/api.php
 create mode 100644 Fernando_A_Lorenzo_Vazquez_PPS_Ejercicio_2.pdf
usuario@pps:~/DevSecOps$ git push DevSecOps
Username for 'https://github.com': Fernandolv123
Password for 'https://Fernandolv123@github.com':
warning: redirigiendo a https://github.com/Fernandolv123/DevSecOps.git/
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Compresión delta usando hasta 2 hilos
Comprimiendo objetos: 100% (5/5), listo.
Escribiendo objetos: 100% (6/6), 360.38 KiB | 9.24 MiB/s, listo.
Total 6 (delta 0), reusado 0 (delta 0)
To http://github.com/Fernandolv123/DevSecOps.git
 1420dce..301deaf master -> master
usuario@pps:~/DevSecOps$ ls
```

Docker 2

Primeramente, crearemos un nuevo fichero donde trabajaremos y crearemos el dockerfile que editaremos

```
usuario@pps:~/DevSecOps$ mkdir DockerUbu
usuario@pps:~/DevSecOps$ cd DockerUbu/
usuario@pps:~/DevSecOps/DockerUbu$ ls
usuario@pps:~/DevSecOps/DockerUbu$ touch Dockerfile
```

La aplicación que descargare con github será una calculadora, ubicada en:
<https://gist.github.com/anampl/9a87a1c43bd923eb939a>



```
Abrir  Dockerfile  Guardar
~/DevSecOps/DockerUbu

token.txt  Dockerfile  Dockerfile  Dockerfile

1 #Descargamos la única versión de ubuntu
2 FROM ubuntu:latest
3 # actualizamos y descargamos apache
4 RUN apt update
5 RUN apt install -y apache2
6 RUN apt install -y apache2-utils
7 #Cambiamos variables de entorno
8 ENV APACHE_RUN_USER www-data
9 ENV APACHE_RUN_GROUP www-data
10 ENV APACHE_LOG_DIR /var/log/apache2
11 ENV APACHE_RUN_DIR /var/www/html
12 #RUN systemctl start apache2
13 RUN DEBIAN_FRONTEND=noninteractive apt install -y tzdata
14 RUN apt-get install php8.1 -y
15 RUN apt-get install git -y
16 WORKDIR /var/www/html/
17 # clonamos la aplicacion de calculadora
18 RUN git clone https://gist.github.com/9a87a1c43bd923eb939a.git
19 #Exponemos el puerto 80
20 EXPOSE 80
21 #Activamos apache
22 ENTRYPOINT ["/usr/sbin/apache2"]
23 CMD ["-D", "FOREGROUND"]
```

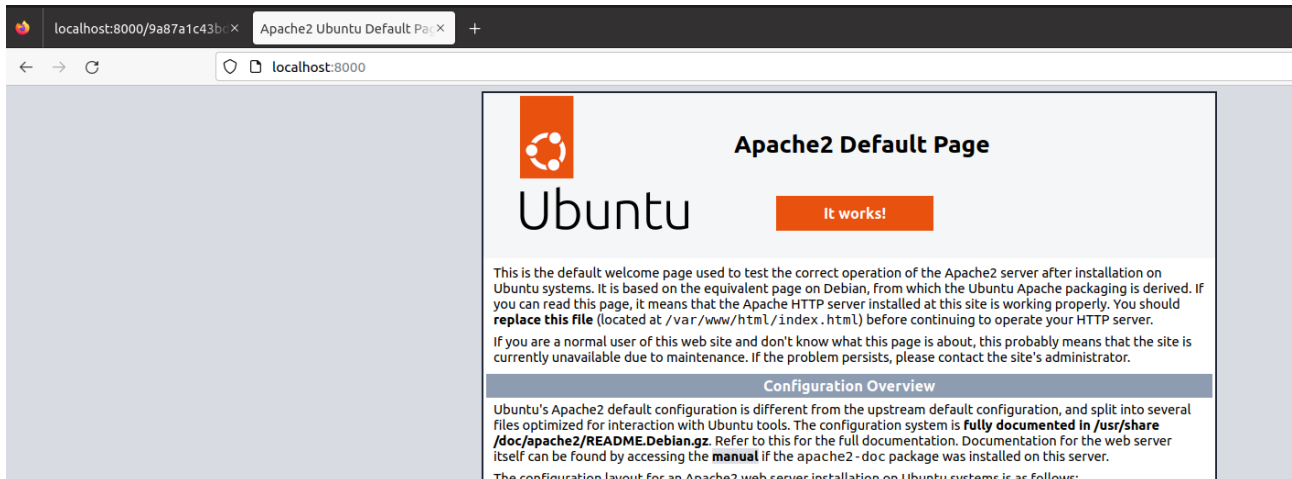
Con el dockerfile creado, solamente tenemos que seguir los pasos que seguimos en el apartado anterior

```
usuario@pps:~/DevSecOps/DockerUbu$ docker build -t dockerubuapache .
Sending build context to Docker daemon 2.56kB
Step 1/16 : FROM ubuntu:latest
```

Creamos la imagen y el contenedor (podemos utilizar el comando `docker ps` para ver los contenedores que estén activos)

```
usuario@pps:~/DevSecOps/DockerUbu$ docker run -d -p 8000:80 --name contenedorubuapache dockerubuapache
3b6bee01644c6cb76f2a994c15970482699de8a970cb24c1f70d86dc88f5f987
usuario@pps:~/DevSecOps/DockerUbu$ docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED        STATUS        PORTS                               NAMES
3b6bee01644c   dockerubuapache   "/usr/sbin/apache2 -..."  4 seconds ago Up 2 seconds  0.0.0.0:8000->80/tcp, :::8000->80/tcp contenedorubuapache
usuario@pps:~/DevSecOps/DockerUbu$
```

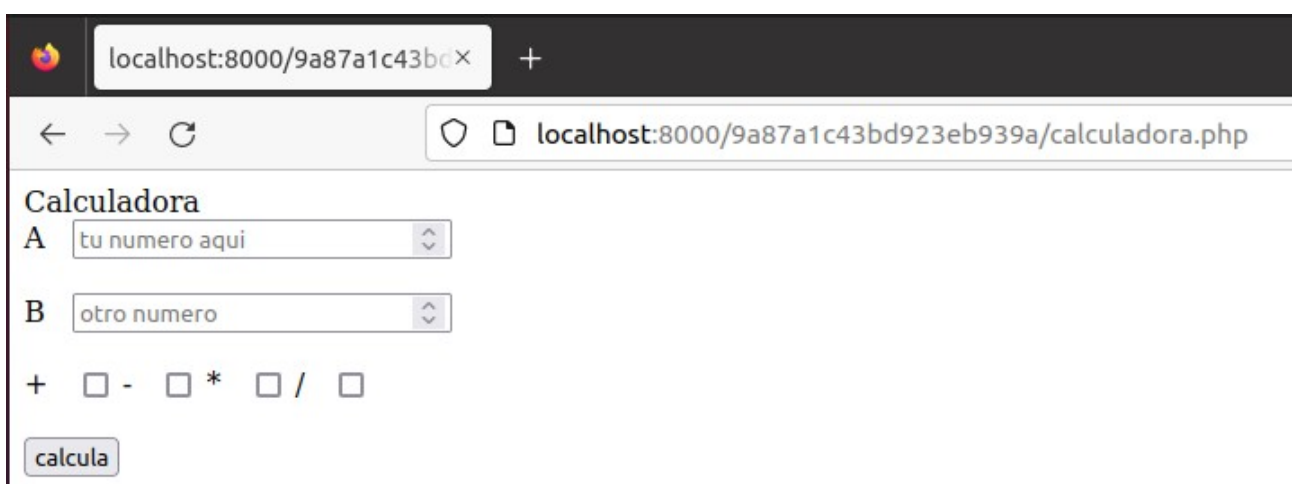

Ahora nos podemos dirigir al puerto que hayamos designado para encontrar nuestro servidor apache



si realizamos el *git clone* a parte, podemos ver la estructura de ficheros

```
usuario@pps:~/pruebacalculadoragithub$ git clone https://gist.github.com/9a87a1c43bd923eb939a.git
Clonando en '9a87a1c43bd923eb939a'...
remote: Enumerating objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3
Desempaquetando objetos: 100% (3/3), 485 bytes | 80.00 KiB/s, listo.
usuario@pps:~/pruebacalculadoragithub$ ls
9a87a1c43bd923eb939a
usuario@pps:~/pruebacalculadoragithub$ cd 9a87a1c43bd923eb939a/
usuario@pps:~/pruebacalculadoragithub/9a87a1c43bd923eb939a$ ls
calculadora.php
```

y ahora podemos dirigirnos a esta url (esto podría cambiarse dentro del propio dockerfile, pero para mostrar que funciona es suficiente)



Ahora realizaré el pdf y lo subiré a github siguiendo los pasos de la primera parte

Futuro:

```
usuario@pps:~/DevSecOps$ git add DockerUbu/
usuario@pps:~/DevSecOps$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    nuevos archivos: DockerUbu/Dockerfile
    modificados:      Fernando_A_Lorenzo_Vazquez_PPS_Ejercicio_2.pdf
```

añadimos los archivos y los subimos

```
usuario@pps:~/DevSecOps$ git commit -m "Ejercicio 2 subido y memoria actualizada"
[master 812743c] Ejercicio 2 subido y memoria actualizada
 2 files changed, 23 insertions(+)
 create mode 100644 DockerUbu/Dockerfile
usuario@pps:~/DevSecOps$ git push DevSecOps
Username for 'https://github.com': Fernandolv123
Password for 'https://Fernandolv123@github.com':
usuario@pps:~/DevSecOps$ git push DevSecOps
Username for 'https://github.com': Fernandolv123
Password for 'https://Fernandolv123@github.com':
warning: redirigiendo a https://github.com/Fernandolv123/DevSecOps.git/
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Compresión delta usando hasta 2 hilos
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (5/5), 735.20 KiB | 13.87 MiB/s, listo.
Total 5 (delta 0), reusado 0 (delta 0)
To http://github.com/Fernandolv123/DevSecOps.git
   301deaf..812743c  master -> master
usuario@pps:~/DevSecOps$
```

Docker 3

Este ejercicio fue hecho junto a los otros dos ya que todos fueron probados antes de subirse

Docker 4

Por rapidez, haremos un bucle que cree 20 contenedores a partir de nuestra imagen creada en el punto 2

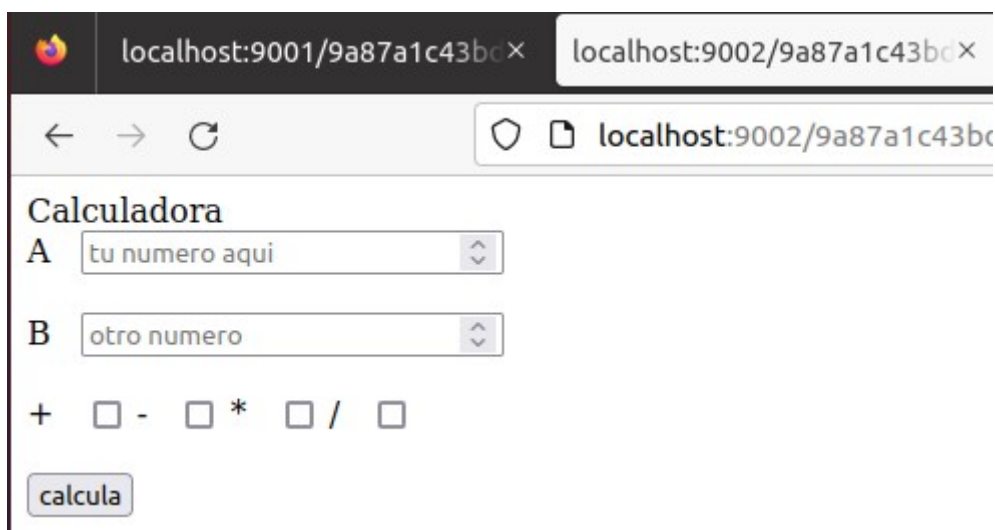
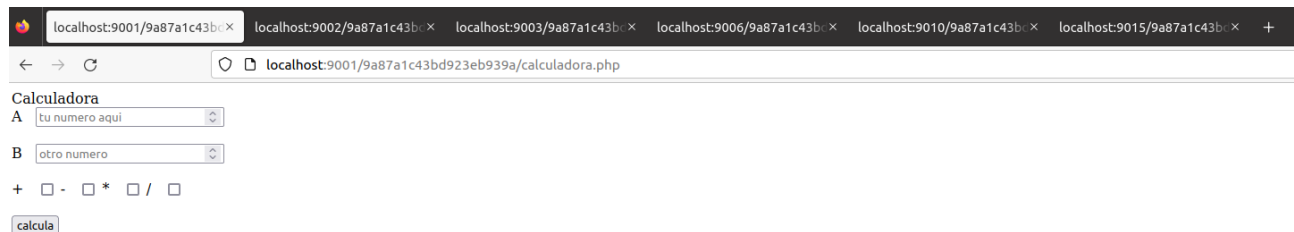
(Realmente solo he abierto 19 contenedores, ya que el bucle empieza en 1 y termina en 20 siendo 19 de diferencia, para crear 20 contenedores se debe cambiar el '1' por un '0' de la siguiente forma:

```
for i in `seq 0 20`; do docker run -dit -p $((9000+$i)):80 --name contenedorej4$i dockerubuapache; done
```

```
usuario@pps:~/DevSecOps$ for i in `seq 1 20`; do docker run -dit -p $((9000+$i)):80 --name contenedorej4$i dockerubuapache; done
bdeea48f32b94764aa3593d004dae9f27d7085c7c0fd4f4cc6c17a35b65baa9c
8d2e098b61107d16ac2c738ac90191a49f2a30eab6c9084428aa649129870c61
8a1e42445892d0b838c76d09ef3730fcda246391f5aea2e2bec3cbb51d077bab
a65122d8971482cf965c034392fc8866550c200446041163dbbd17e2caadc0
bc53989d133e94743b1396fdb9f94a502f7b8079cf536277bd1ce2f3c8c8ac58
bed80518c7829a240e3a0fe34f1bc1857176a376e10dbe4c4781d0bf60db36d
baf1a8eb5de371b89f0e4ad5869fed690f371fb9cc11644f9c00c5b0500fdb0f
725fecf5954e0070960467871d80e29fc1086c1de3a1c24a6006a1bddb089b1
b7b58b499b96eb3fe1b1371e6104242d0706b4281895883b54616dbcd5bf34f
f0419ba77b6c9a7cc7cba71bc1991f5dd70f0615ed0463fb7be3a524f32a6002
b71c6607934f62a63d7139bf2ed5be6d9f043c6e0b2bfa6992151bc678d2632c
1ebf05812c4c57845ef0b06bca1273a1a6ccc64ec27be83f959d185d6f886106
2e98436af5a0becfa99dd5845577f92d3f0a361dd729634432e83c56c257669f
928516ac514b4945a631e820bf0e1cc084ff1fcc0c9d4c306e995162f5e8fa3f
d9507b796fd704b935d7b3dc7c7f287b86e241890e712408ab424576a77e07d7
be7bcc846db6fe81694e162447ae5ef9bcf1714ee4ca056646843fdf78bee90
c51457401b3a44cc2a3c6627d18ed9a0fae4fdac593d45e3a2a9615bc8ebb05e
d57852ab84b795acd9f44ede41e396440105fb099d61de3c24280a7c7388fb4b
610c9621998b2b8f2fa91f8d8b1e1290b89062cefd0ab18ca7cafec340c7ccb
a7ad922e6e068bc5b806bb2c127b2409515358ea342f5b4e54cc5a0e044638f1
```

Al haber partido de una imagen supuestamente funcional, no deberíamos tener problema con estos contenedores.

A continuación mostraré algunos contenedores empezando por el primero (9001) hasta el último (9020)



localhost:9001/9a87a1c43bc×localhost:9002/9a87a1c43bc×localhost:9003/9a87a1c43bc×

←→↻

localhost:9003/9a87a1c43bd923eb939a/calculadora.php

Calculadora

A

B

+☐-☐*☐/☐

calcula

localhost:9001/9a87a1c43bc×localhost:9002/9a87a1c43bc×localhost:9003/9a87a1c43bc×localhost:9006/9a87a1c43bc×

←→↻

localhost:9006/9a87a1c43bd923eb939a/calculadora.php

Calculadora

A

B

+☐-☐*☐/☐

calcula

localhost:9001/9a87a1c43bc×localhost:9002/9a87a1c43bc×localhost:9003/9a87a1c43bc×localhost:9006/9a87a1c43bc×localhost:9010/9a87a1c43bc×

←→↻

localhost:9010/9a87a1c43bd923eb939a/calculadora.php

Calculadora

A

B

+☐-☐*☐/☐

calcula

localhost:9001/9a87a1c43bc×localhost:9002/9a87a1c43bc×localhost:9003/9a87a1c43bc×localhost:9006/9a87a1c43bc×localhost:9010/9a87a1c43bc×localhost:9015/9a87a1c43bc×

←→↻

localhost:9015/9a87a1c43bd923eb939a/calculadora.php

Calculadora

A

B

+☐-☐*☐/☐

calcula

localhost:9020/9a87a1c43bc×

+

←→↻

localhost:9020/9a87a1c43bd923eb939a/calculadora.php

Calculadora

A

B

+☐-☐*☐/☐

calcula

Ahora que ya hemos comprobado la funcionalidad de los contenedores, podemos eliminarlos una vez más en bucle con:

```
for i in `seq 1 20`; do docker rm --force contenedorej4$i; done
```

```
usuario@pps:~/DevSecOps$ for i in `seq 1 20`; do docker rm --force contenedorej4$i; done
contenedorej41
contenedorej42
contenedorej43
contenedorej44
contenedorej45
contenedorej46
contenedorej47
contenedorej48
contenedorej49
contenedorej410
contenedorej411
contenedorej412
contenedorej413
contenedorej414
contenedorej415
contenedorej416
contenedorej417
contenedorej418
contenedorej419
contenedorej420
```

NOTA: si hubiesemos ejecutado el comando `docker ps` antes de eliminarlos, podríamos haber visto todos los contenedores en ejecución.