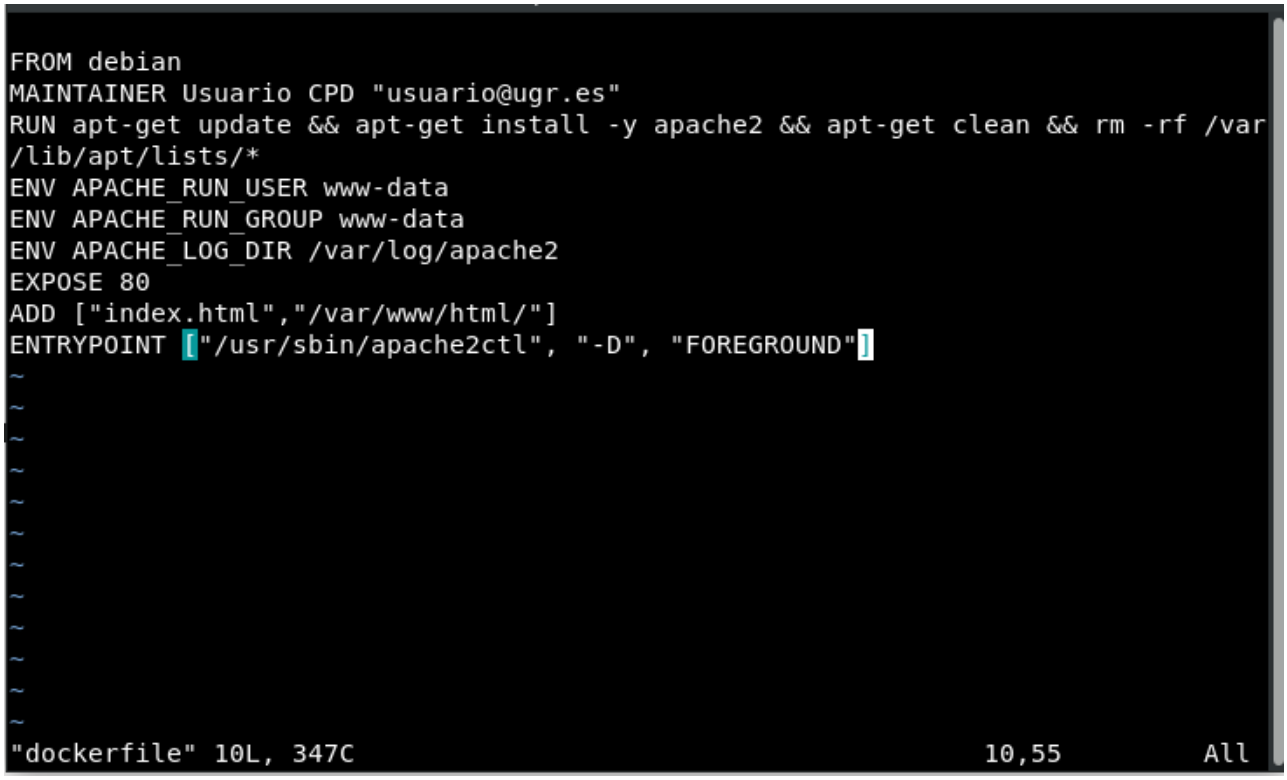
Práctica 4
CPD

Por: Rubén Calvo Villazán

- Crear una imagen personalizada de docker con Apache que cuando se acceda a http://localhost:8888 aparezca vuestro nombre (editando el index.html) (según apartado I). A partir de los pasos explicados en el apartado II subir la imagen a hub.docker.com e indicar el nombre de la imagen creada.

Ejecutamos los siguientes comandos:
$> docker build -f dockerfile <ruta>
$> docker build -t my-apache .
$> docker run -dit --name my-apache -p 8080:80 my-apache

Contenido del Dockerfile:



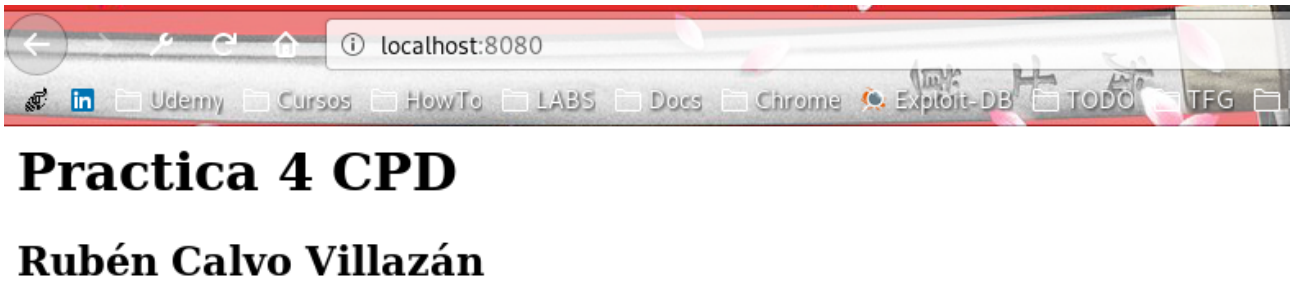
Construcción del servicio apache:

```
root@valkyrie:P4# docker build -t my-apache .
Sending build context to Docker daemon  3.072kB
Step 1/9 : FROM debian
 ---> be2868bebaba
Step 2/9 : MAINTAINER Usuario CPD "usuario@ugr.es"
 ---> Using cache
 ---> bd5fd1f4eae4
Step 3/9 : RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> 63fb069f5e08
Step 4/9 : ENV APACHE_RUN_USER www-data
 ---> Using cache
 ---> ba961dee7bde
Step 5/9 : ENV APACHE_RUN_GROUP www-data
 ---> Using cache
 ---> d613e24140a5
Step 6/9 : ENV APACHE_LOG_DIR /var/log/apache2
 ---> Using cache
 ---> 4e43a99a20db
Step 7/9 : EXPOSE 80
 ---> Using cache
 ---> a2c03d346fcb
Step 8/9 : ADD ["index.html","/var/www/html/"]
 ---> Using cache
 ---> 021db40e3e65
Step 9/9 : ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
 ---> Using cache
 ---> ced17ef1f7da
Successfully built ced17ef1f7da
Successfully tagged my-apache:latest
```

Ejecutamos el servicio apache:

```
root@valkyrie:P4# docker run -dit --name my-apache -p 8080:80 my-apache
8ace2885823eda8bf366dee3d4f6b32d3de07f366c23b85eb13bf5d0b9fd2432
root@valkyrie:P4#
```

Especificamos el puerto 8080 aunque se podría haber especificado cualquiera.
Si accedemos a través del navegador:



Practica 4 CPD

Rubén Calvo Villazán

-Contenido del fichero Dockerfile personalizado del apartado II y ficheros utilizados.

Contenido del dockerfile:

```
FROM debian
MAINTAINER Usuario CPD "usuario@ugr.es"
RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var
/lib/apt/lists/*
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
EXPOSE 80
ADD ["index.html","/var/www/html/"]
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
~
~
~
~
~
~
~
~
~
~
~
~
"dockerfile" 10L, 347C                                        10,55         All
```

Contenido del index.html:

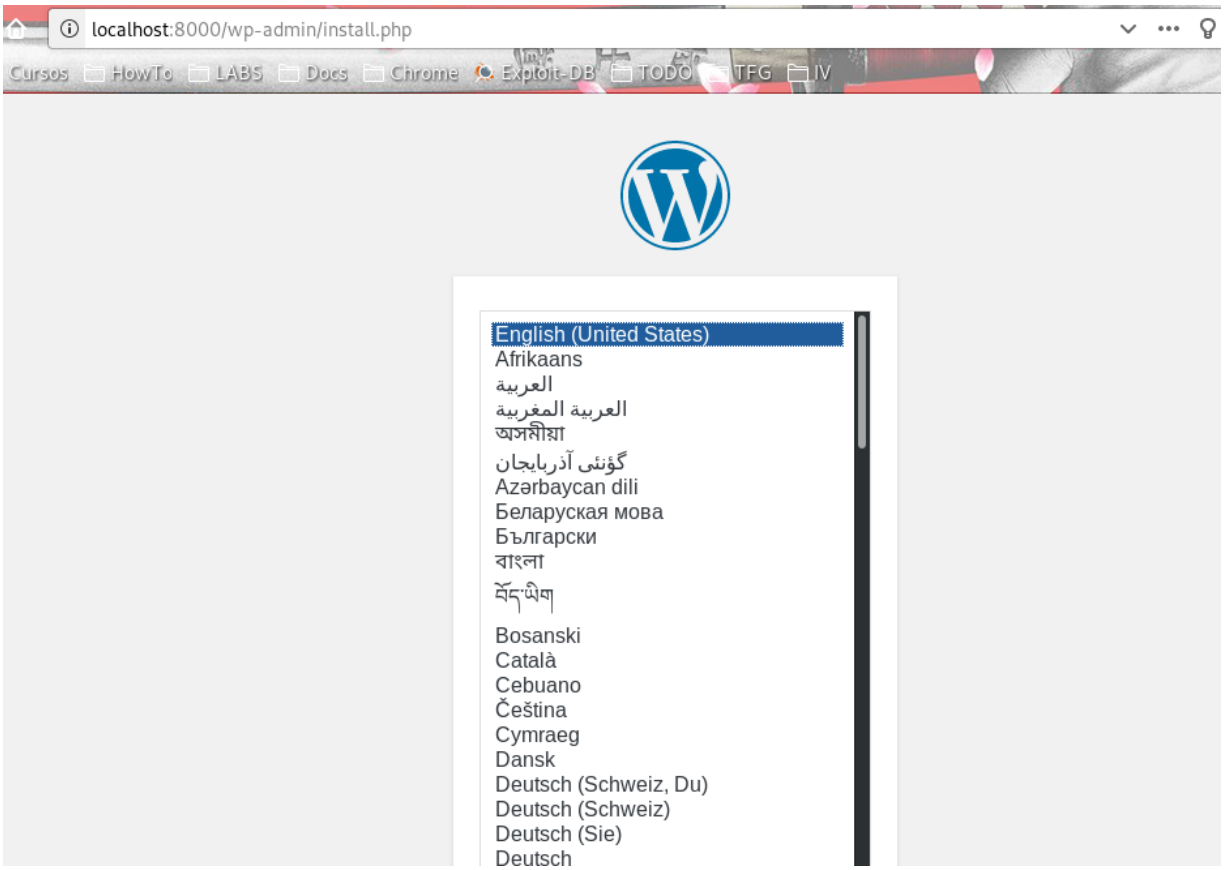
```
root@valkyrie:P4-3# cat ../P4/index.html
<html>
         <meta charset="UTF-8">
        <head>
                <title> P4 CPD </title>
        </head>
        <body>
                <h1> Practica 4 CPD </h1>
                <h2> Rubén Calvo Villazán </h2>
        </body>
</html>
```

-Según el apartado III, una vez desplegado el servidor Wordpress, editar la página principal para que aparezca el nombre del usuario y realizar una captura de pantalla.

Tras tener el docker-compose.yml en el directorio, ejecutamos:
$> docker-compose up -d

Si docker-compose no está instalado:
$> pip install docker-compose

Accedemos al navegador a localhost:8000



Información necesaria

Por favor, debes facilitarnos los siguientes datos. No te preocupes, siempre podrás cambiar estos ajustes más tarde.

Título del sitio P4-CPD

Nombre de usuario rubcv

Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @.

Contraseña ••••••••••••••••••• Mostrar

Fuerte

Importante: Necesitas esta contraseña para acceder. Por favor, guárdala en un lugar seguro.

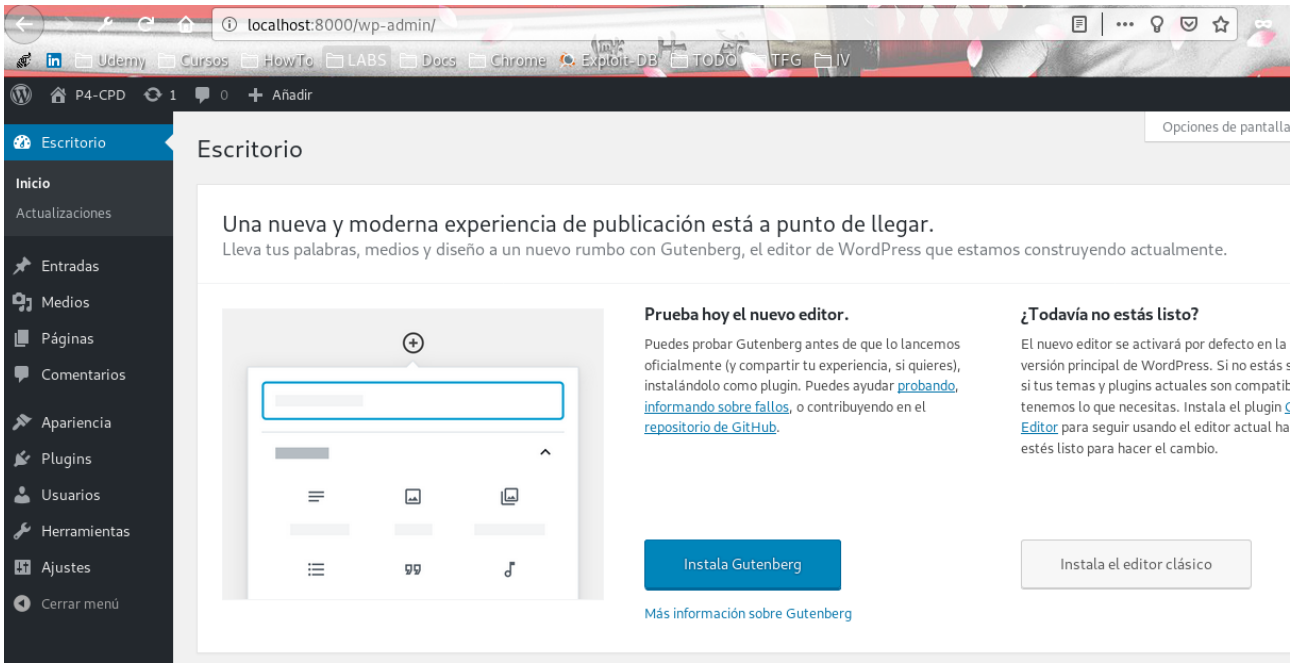
Tu correo electrónico spamdesareth@gmail.com

Comprueba bien tu dirección de correo electrónico antes de continuar.

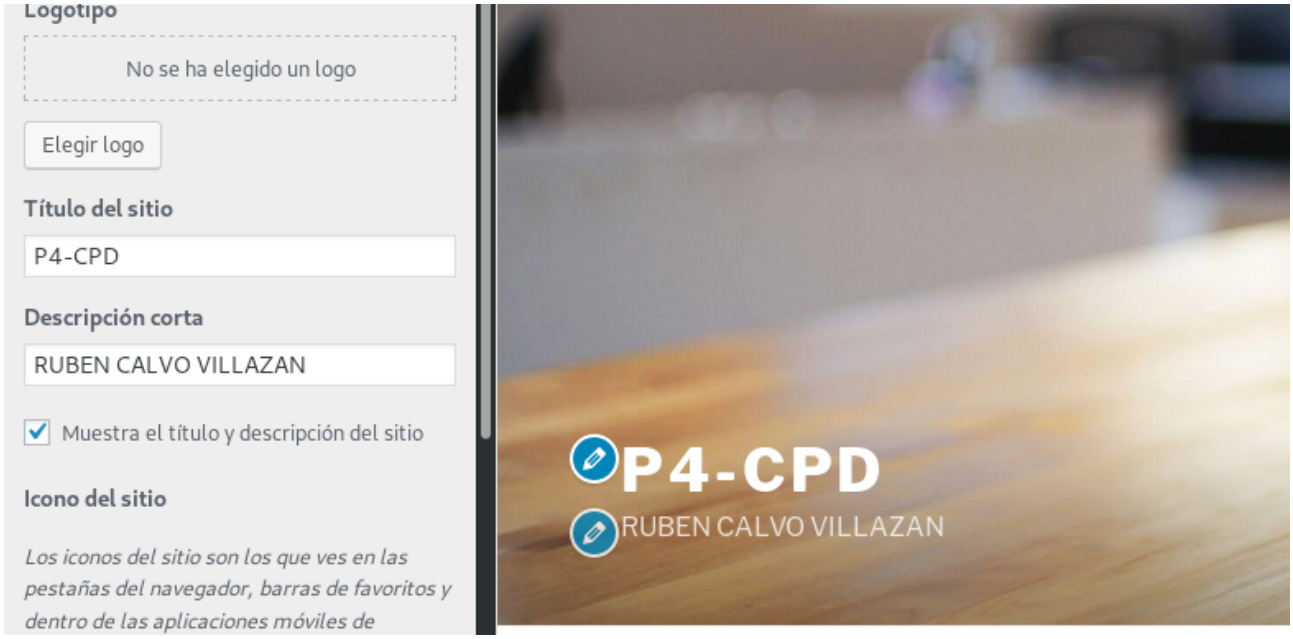
Visibilidad en los motores de búsqueda Disuade a los motores de búsqueda de indexar este sitio

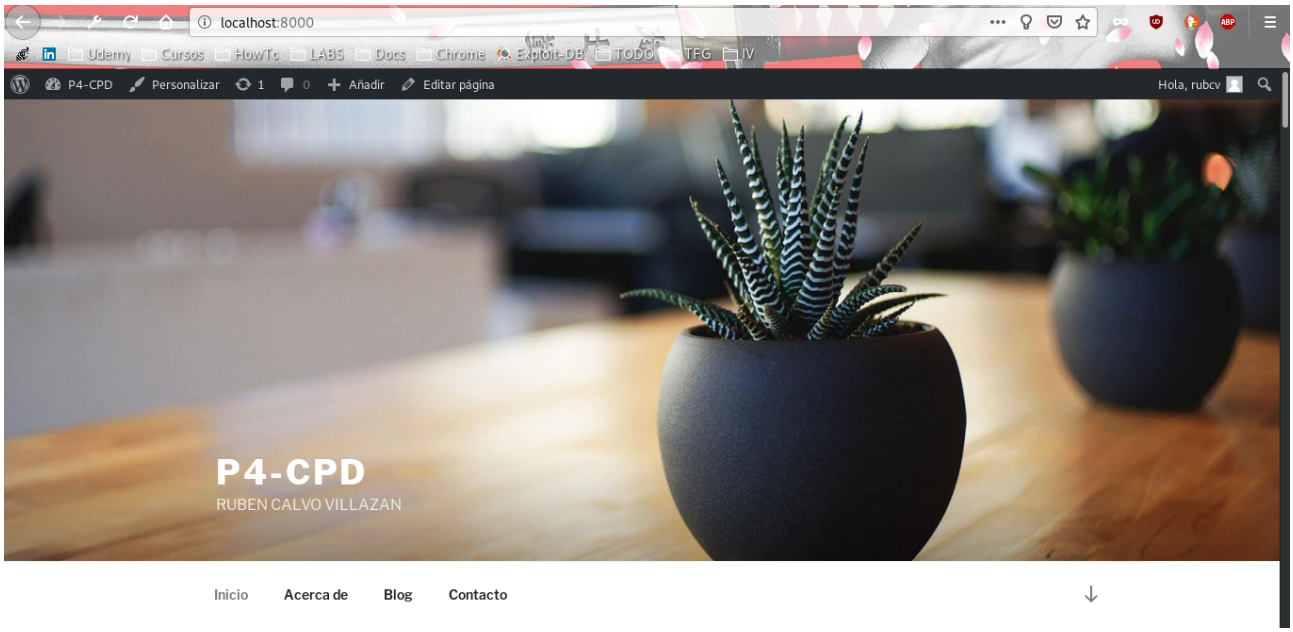
Depende de los motores de búsqueda atender esta petición o no.

Instalar WordPress



Editamos la página principal:



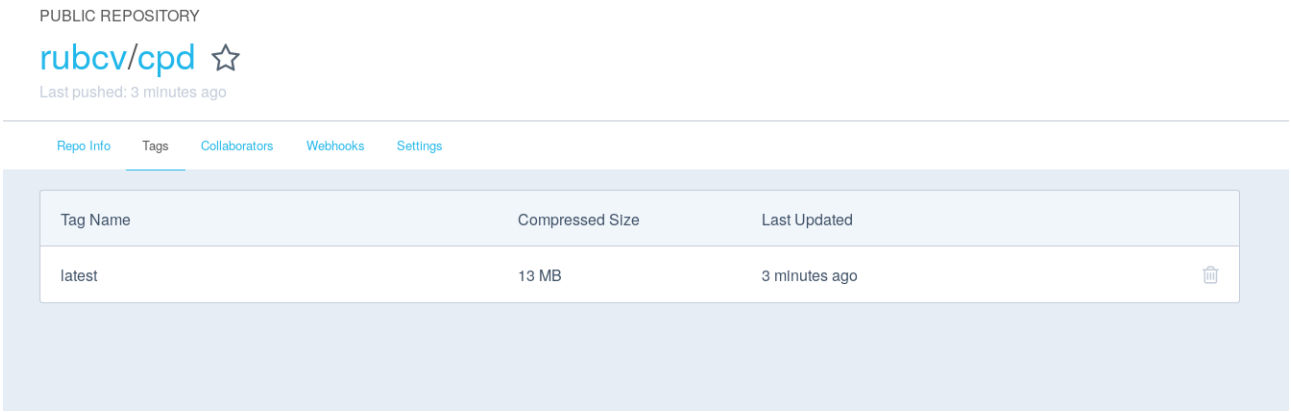


-Siguiendo los pasos del apartado IV, realice un programa en C, Python, que permita evaluar el tiempo de ejecución de un algoritmo (ej. el cálculo de las N primeras cifras de PI), y comprobar cómo podemos aumentar o reducir la CPU dedicada y por tanto el tiempo de ejecución. Prepare un contenedor con dicho experimento, súbalo a hub.docker.com e indique en el documento los pasos que realiza para el experimento y los tiempos de ejecución obtenidos.

El programa en Python:

```
from decimal import Decimal, getcontext
import time


start_time = time.time()

N = 10 # Precision

# Calculo de PI

getcontext().prec = N

print sum(1/Decimal(16)**k *
          (Decimal(4)/(8*k+1) -
           Decimal(2)/(8*k+4) -
           Decimal(1)/(8*k+5) -
           Decimal(1)/(8*k+6)) for k in range(N))

print("--- %s segundos ---" % (time.time() - start_time))
```

Nos creamos el repositorio en Docker hub:



Construimos el fichero dockerfile:

```
root@valkyrie:P4-3# docker build -f dockerfile . --tag cpd
Sending build context to Docker daemon  4.096kB
Step 1/4 : FROM alpine:3.8
 ---> 196d12cf6ab1
Step 2/4 : RUN apk update
 ---> Using cache
 ---> 591c6df64041
Step 3/4 : RUN apk add curl
 ---> Using cache
 ---> 2065aa6c21cd
Step 4/4 : RUN apk add vim
 ---> Using cache
 ---> 35e44e2d2203
Successfully built 35e44e2d2203
Successfully tagged cpd:latest
root@valkyrie:P4-3# docker tag cpd rubcv/cpd
```

Nos logeamos desde terminal:

```
root@valkyrie:P4-3# 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: rubcv
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
```

Creamos la etiqueta con el nombre que queramos, por ejemplo ‘cpd’

Realizamos el push al repositorio de Docker.

```
root@valkyrie:P4-3# docker login
Authenticating with existing credentials...
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
root@valkyrie:P4-3# docker tag cpd rubcv/cpd
root@valkyrie:P4-3# docker push rubcv/cpd
The push refers to repository [docker.io/rubcv/cpd]
657b3396eaaf: Pushed
82ddba453dbd: Pushed
35f5c8458565: Pushed
df64d3292fd6: Layer already exists
latest: digest: sha256:b34515bdd4fb54dd095f23e75d399dd68b51a3e9c586964808663123
5c6d047 size: 1160
```

Para ejecutar el programa creado anteriormente en Python, creamos el contenedor Docker:

```
root@valkyrie:P4-3# docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
473ede7ed136: Pull complete
c46b5fa4d940: Pull complete
93ae3df89c92: Pull complete
6b1eed27cade: Pull complete
Digest: sha256:29934af957c53004d7fb6340139880d23fb1952505a15d69a03af0d1418878cb
Status: Downloaded newer image for ubuntu:latest
root@d68e2e6dad5a:/#
root@d68e2e6dad5a:/#
root@d68e2e6dad5a:/# docker rename d68e2e6dad5a cpd
bash: docker: command not found
root@d68e2e6dad5a:/# exit
exit
root@valkyrie:P4-3# docker rename d68e2e6dad5a cpd
root@valkyrie:P4-3# docker restart cpd
cpd
root@valkyrie:P4-3# docker attach cpd
root@d68e2e6dad5a:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
```

Le cambiamos el nombre para referirnos a él como cpd.
Instalamos Python en el contenedor, etc.

Al ejecutar la orden run de Docker, nos aparece el siguiente error:

```
root@valkyrie:P4-3# docker run -it --cpus=".01" cpd /home/programa.py
docker: Error response from daemon: OCI runtime create failed: container_linux.g
o:348: starting container process caused "exec: \"/home/programa.py\": stat /hom
e/programa.py: no such file or directory": unknown.
```

He probado a poner el programa en /bin/, poner un script que ejecute el programa en /bin/, añadir el directorio a $PATH pero no ha funcionado. Como lo que interesa es ver los tiempos de ejecución aplicando distintas configuraciones de CPU, si ejecutamos comandos (por ejemplo ‘ls’) modificando la cpu:

```
root@valkyrie:P4-3# docker run -it --cpus=".5" cpd ls
bin   etc   lib   mnt   root  sbin  sys   usr
dev   home  media proc  run   srv   tmp   var
```

Con –cpus modificamos con cuantos núcleos queremos la ejecución (en este caso 0.01 hasta 4.00):

Si especificamos 4 núcleos:

```
root@valkyrie:P4-3# docker run -it --cpus="4.0" cpd time ls
bin   etc   lib   mnt   root  sbin  sys   usr
dev   home  media proc  run   srv   tmp   var
real    0m 0.00s
user    0m 0.00s
sys     0m 0.00s
```

Si especificamos 0.01: