

PythonAnywhere

En este taller vamos a subir a un servidor una API desarrollada en local. Para ello utilizaremos un servicio hosting de Python dedicado, y gratuito 😊 llamado PythonAnywhere

<https://www.pythonanywhere.com/>

Lo primero que tienes que hacer es registrarte. NO te pedirá tarjeta de crédito ya que tiene un servicio gratuito que sirve para nuestro propósito. En el servicio gratuito se incluye una máquina virtual Linux con los paquetes y configuraciones necesarias para realizar un despliegue de software en Python. Hay un límite uso de CPU que se renueva cada día. Para el uso que le vamos a dar en los talleres de clase, es más que suficiente.

Para montar una app Flask tenemos dos opciones:

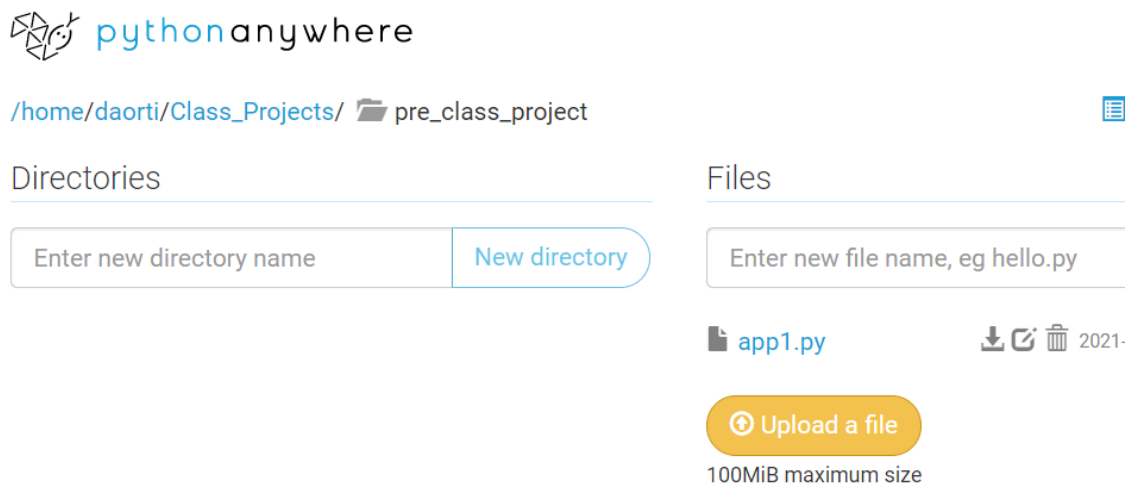
1. Desplegar una web, configurando manualmente el environment, así como sus paquetes y versiones.
2. Desplegar una app con la versión de Flask de PythonAnywhere

Este taller está basado en el [tutorial oficial](#) de subida de una app en Flask de PythonAnywhere. Empezaremos con el despliegue manual y continuaremos con el despliegue sin entorno virtual.

1. Subir los archivos

Lo primero que hacemos es compartir los archivos de la app en Flask a PythonanyWhere. Vamos a Files, creamos una carpeta para el proyecto, y los subimos.

[Aquí](#) se describe cómo compartir archivos en PythonAnywhere.



2. Establecer el entorno virtual

Vamos a crear el entorno virtual donde se desplegará la app de Flask. Para ello, ejecuta el siguiente comando, que creará un entorno virtual con la versión de Python 3.8:

```
mkvirtualenv --python=/usr/bin/python3.8 my-virtualenv
```

En este momento estás trabajando con un entorno virtual creado. Fíjate que lo primero que aparece en la línea de comandos es el nombre sobre el que estás trabajando, por lo que cada “pip install” que hagas se ejecutará únicamente sobre ese entorno, permitiendo manejar varios entornos con distintos paquetes y versiones.

Para familiarizarte con esta pieza de software y evitar futuros problemas, realiza las siguientes operaciones:

1. Desactiva el entorno virtual, por si en un futuro deseas crear otro y cambiarte.
Comando “deactivate”.
2. Ahora escogeremos este entorno virtual para activarlo. Busca dónde está instalado.
Por defecto se instalará en /home/nombreusuario/.virtualenvs
El directorio “.virtualenvs” está oculto. Para ver archivos ocultos tienes que listarlos mediante “ls -a”.
Vamos a la carpeta “home/nombreusuario/.virtualenvs/my-virtualenv/bin” y ejecutamos “source activate”. Otra opción es “workon my-virtualenv”

```
(my-virtualenv) 19:12 ~ $ pwd
/home/daorti
(my-virtualenv) 19:16 ~ $ deactivate
19:16 ~ $ ls -a
. .bashrc .config .local .profile .pythonstartup.py .virtualenvs README.txt
.. .cache .gitconfig .my.cnf .python_history .vimrc Class_Projects
19:16 ~ $ cd .virtualenvs/my-virtualenv/bin
19:17 ~/.virtualenvs/my-virtualenv/bin $ source activate
(my-virtualenv) 19:17 ~/.virtualenvs/my-virtualenv/bin $
```

3. Instalamos Flask

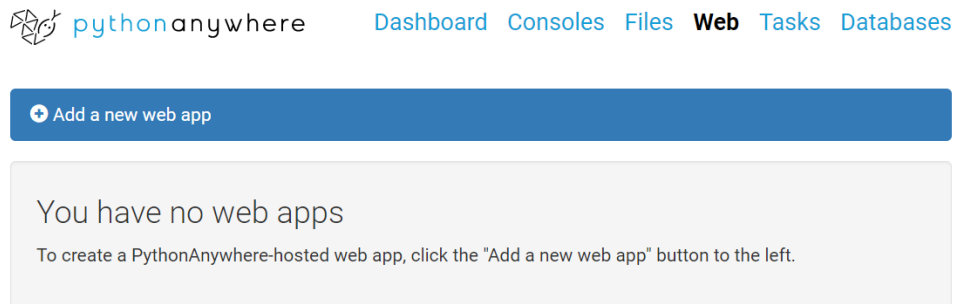
En este momento tenemos un entorno virtual prácticamente vacío, con Python 3.8. Procedemos a instalar Flask:

pip install flask

```
(my-virtualenv) 19:17 ~/.virtualenvs/my-virtualenv/bin $ pip list
Package Version
-----
pip      21.1
setuptools 56.0.0
wheel    0.36.2
(my-virtualenv) 19:20 ~/.virtualenvs/my-virtualenv/bin $ pip install flask
Looking in links: /usr/share/pip-wheels
Collecting flask
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
    | 94 kB 798 kB/s
Collecting Werkzeug>=0.15
  Downloading Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
    | 298 kB 19.2 MB/s
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting click>=5.1
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
    | 82 kB 338 kB/s
Collecting Jinja2>=2.10.1
  Downloading Jinja2-2.11.3-py2.py3-none-any.whl (125 kB)
    | 125 kB 19.7 MB/s
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp38-cp38-manylinux2010_x86_64.whl (32 kB)
Installing collected packages: MarkupSafe, Werkzeug, Jinja2, itsdangerous, c
Successfully installed Jinja2-2.11.3 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7
us-1.1.0
(my-virtualenv) 19:20 ~/.virtualenvs/my-virtualenv/bin $ pip list
Package Version
-----
click      7.1.2
Flask      1.1.2
itsdangerous 1.1.0
Jinja2     2.11.3
MarkupSafe 1.1.1
pip        21.1
setuptools 56.0.0
werkzeug   1.0.1
wheel      0.36.2
(my-virtualenv) 19:20 ~/.virtualenvs/my-virtualenv/bin $
```

4. Montamos la APP web en el Dashboard

Vamos a la pestaña de “Web” -> “Add a new web app”



Después seleccionamos “**Manual configuration**”. A continuación “**Python 3.8**”, siempre y cuando el entorno virtual desplegado sea Python 3.8. Después le damos a **Next**.

¡Ya tendríamos la web desplegada! Ahora falta todo lo demás!!

5. Configuración de la web

Tenemos que realizar algunas configuraciones en el servidor para que el despliegue sea correcto:

1. Tenemos que indicar qué entorno virtual queremos que use la web. Habrá que escribir una ruta con la siguiente pinta:

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

</home/daorti/.virtualenvs/my-virtualenv>

2. Modificar el WSGI: se trata de un archivo de configuración que nos permite utilizar Python a través de Internet, a través del protocolo HTTP.

Code:

What your site is running.

Source code:	Enter the path to your web app source code
Working directory:	/home/daorti/
WSGI configuration file:	/var/www/daorti.pythonanywhere.com/wsgi.py
Python version:	3.8

Vete al final del archivo y descomenta las siguientes líneas. Tienes que establecer la ruta donde se encuentre tu app Flask, siendo “app1” el nombre del archivo donde está

la app y “app” el nombre que le has dado a la app en el código de Flask (“app = Flask(__name__)”):

```
98 #
99 import sys
100 #
101 ## The "/home/daorti" below specifies your home
102 ## directory -- the rest should be the directory you uploaded your Flask
103 ## code to underneath the home directory. So if you just ran
104 ## "git clone git@github.com/myusername/myproject.git"
105 ## ...or uploaded files to the directory "myproject", then you should
106 ## specify "/home/daorti/myproject"
107 path = '/home/daorti/Class_Projects/pre_class_project'
108 if path not in sys.path:
109     sys.path.append(path)
110 #
111 from app1 import app as application # noqa
112 #
```

Guarda el archivo

3. Sustituye el “run()” por el siguiente código, de esta manera no correrá la app cuando se importe, evitando un error 504
if __name__ == '__main__':
 app.run()
4. Por último, hacemos click en Reload, para que recargue la app con todos los cambios.



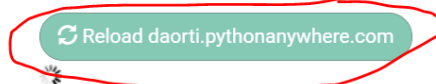
[Dashboard](#)

daorti.pythonanywhere.com

[+ Add a new web app](#)

Configuration for [daorti.pythonanywhere.com](#)

Reload:



¡Si todo ha ido bien ya deberíamos tener la app funcionando!

5. Despliegue en entorno predefinido Flask

Elimina la web anterior y crea una nueva sin necesidad de entorno virtual. Esto tendrá sentido si no te preocupa la versión de Flask instalada en el servidor, ni de las librerías que vayas a usar.

1. Sube la nueva app al servidor



/home/daorti/Class_Projects/ pre_class_project2

Directories

Enter new directory name

New directory

__pycache__/



Files

Enter new file name, eg hello.py

app2.py

20

Upload a file

100MiB maximum size

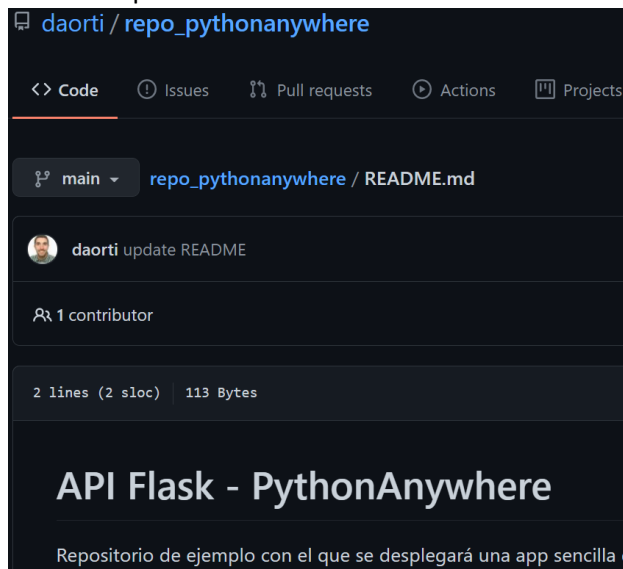
2. Vete a la pestaña “Web” -> “Add a new web app” -> “Next” -> “Flask” -> “Python 3.8”
-> Indicamos donde está la app -> “Next”

¡Ya la tendríamos desplegada!

6. Despliegue desde GitHub

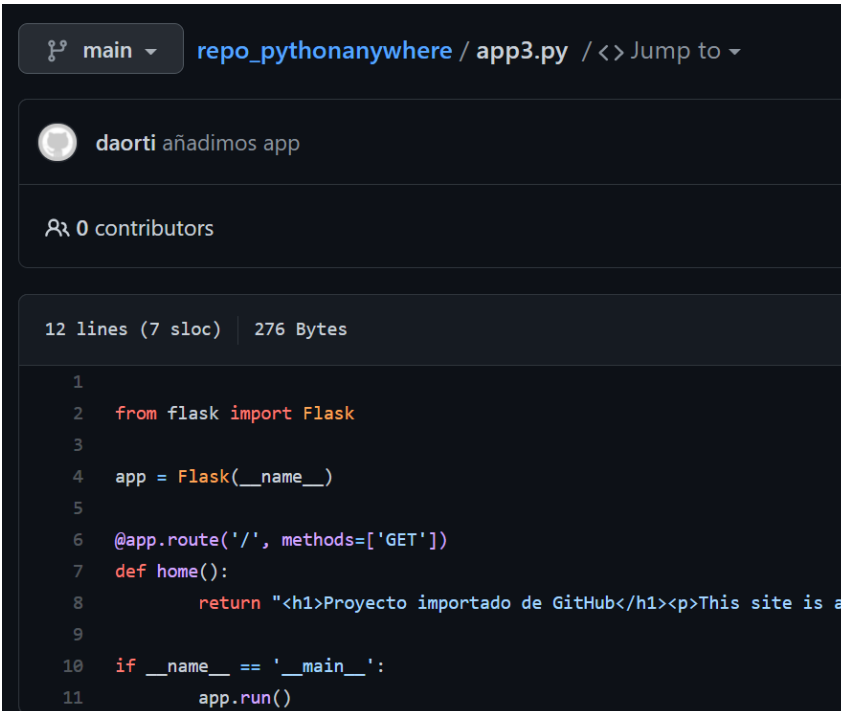
Probemos ahora a tener un proyecto sincronizado con GitHub. La máquina de PythonAnywhere ya viene con Git instalado, por lo que no resultará muy complicado clonar y actualizar el código desde un repositorio remoto de GitHub.

1. Lo primero, eliminar el proyecto del apartado anterior
2. Crea un repo en GitHub



3. Abre un terminal en PythonAnywhere y clona el repositorio.
4. Despliega una app web en PythonAnywhere. Vete a la pestaña de Web, y crea una app nueva, al igual que en el apartado anterior, con el Flask que viene por defecto. Esto desplegará un app con un “Hello Flask”. Solo falta particularizarlo para nuestra app.
5. Todo en terminal. Realiza un copy paste de “app1.py” en el nuevo repositorio que has clonado. Modifica el archivo con “nano” para saber en el despliegue que efectivamente estamos subiendo esta app. Por último, sube los cambios al repo de GitHub y actualiza la app en PythonAnywhere.

```
13:50 ~/Class_Projects/pre_class_github/repo_pythonanywhere (main)$ ls -la
.  ..  .git  README.md  app3.py
```



The screenshot shows a GitHub repository interface. At the top, the repository name 'repo_pythonanywhere' is displayed, along with a 'main' branch selector and a 'Jump to' dropdown. Below this, a commit by user 'daorti' is shown with the message 'añadimos app'. The commit details indicate '12 lines (7 sloc)' and '276 Bytes'. The commit message is expanded to show the code for 'app3.py', which is a Flask application. The code includes imports for Flask, initialization of the app, a route for the home page, and a main block to run the app.

```
1
2  from flask import Flask
3
4  app = Flask(__name__)
5
6  @app.route('/', methods=['GET'])
7  def home():
8      return "<h1>Proyecto importado de GitHub</h1><p>This site is a
9
10 if __name__ == '__main__':
11     app.run()
```

6. Ahora cada vez que queramos realizar un nuevo despliegue desde el código de GitHub simplemente ejecutaremos un “git pull” en el servidor de PythonAnywhere y actualizaremos la app desde el dashboard.