Lab 10: Deployment

Objective

The objective of this lab is to publish online the social network developed during the course, using the PythonAnywhere platform. This way, the application will be accessible from any web-enabled device connected to the Internet.

Activity description

Below are the detailed instructions to complete the deployment:

1. Create a free account on the platform:
   https://www.pythonanywhere.com/registration/register/beginner/

2. Compress the **Lab 9** project — you can also use the solution on GitHub — into a `.zip` file and upload it through the Files page, accessible from your personal Dashboard. Click on "**Files**" (**Figure 1**) and you will find an "**Upload a file**" button that allows you to upload the `.zip` file (**Figure 2**).

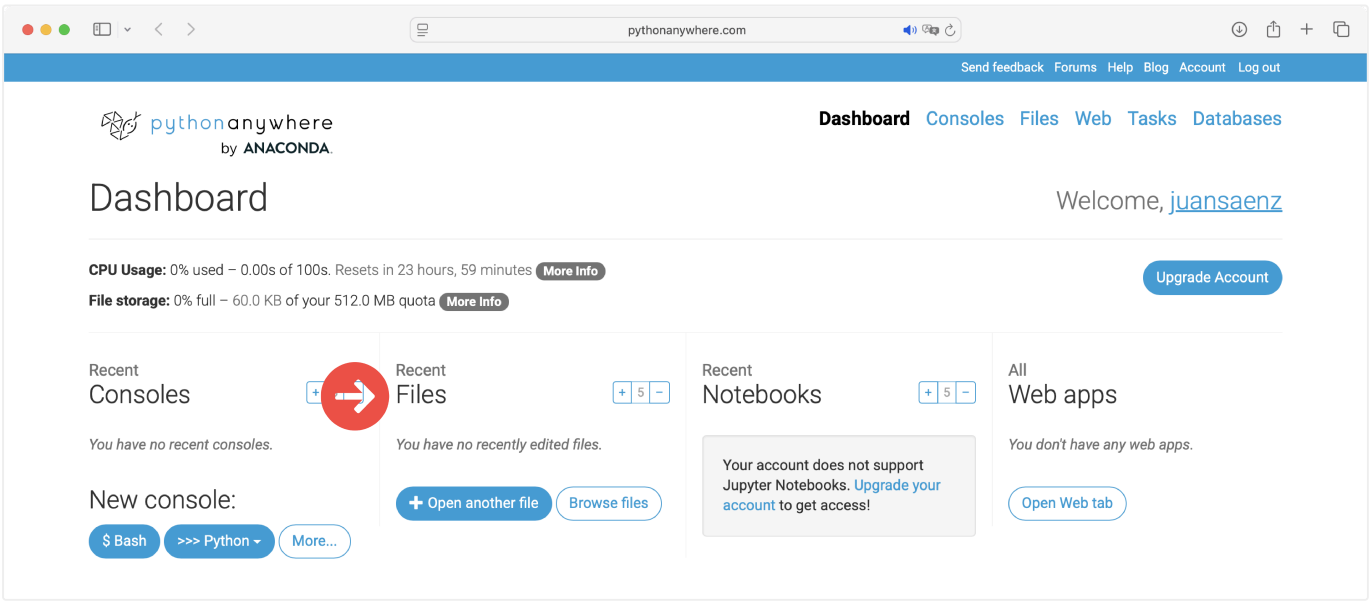

Figure 1: PythonAnywhere Dashboard.

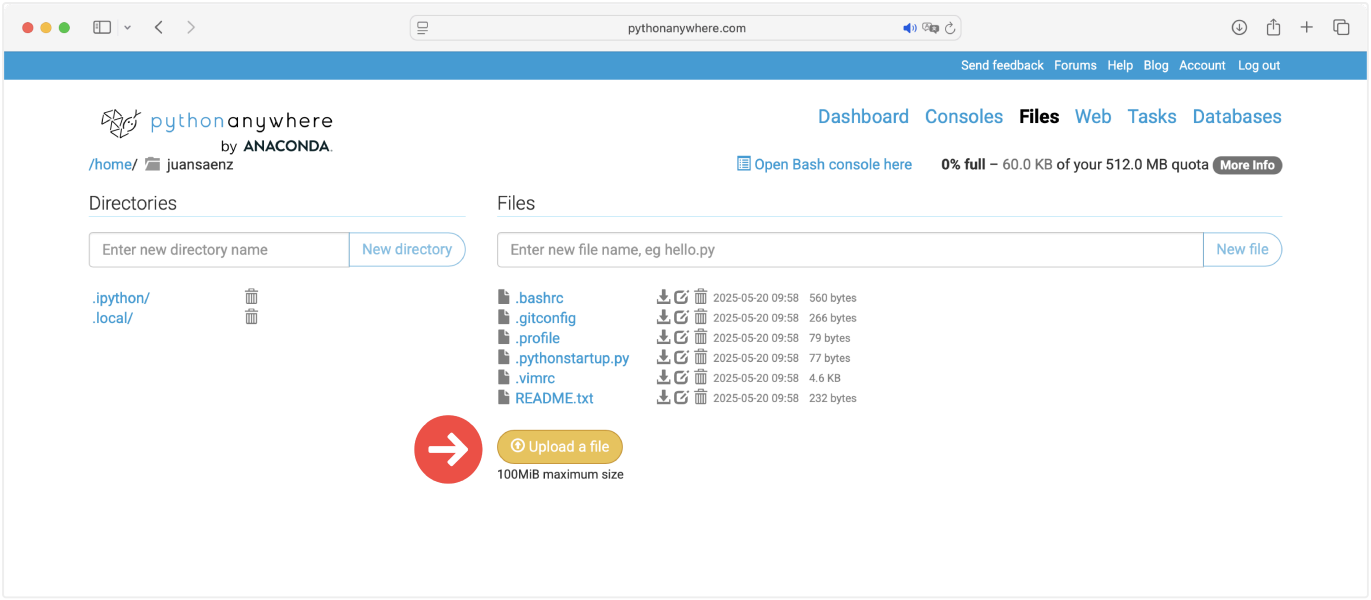

Figure 2: File upload screen.

1. Open a new **Bash** console from your personal Dashboard (**Figure 3**), and run the command `unzip [file name]` to extract the project you just uploaded.

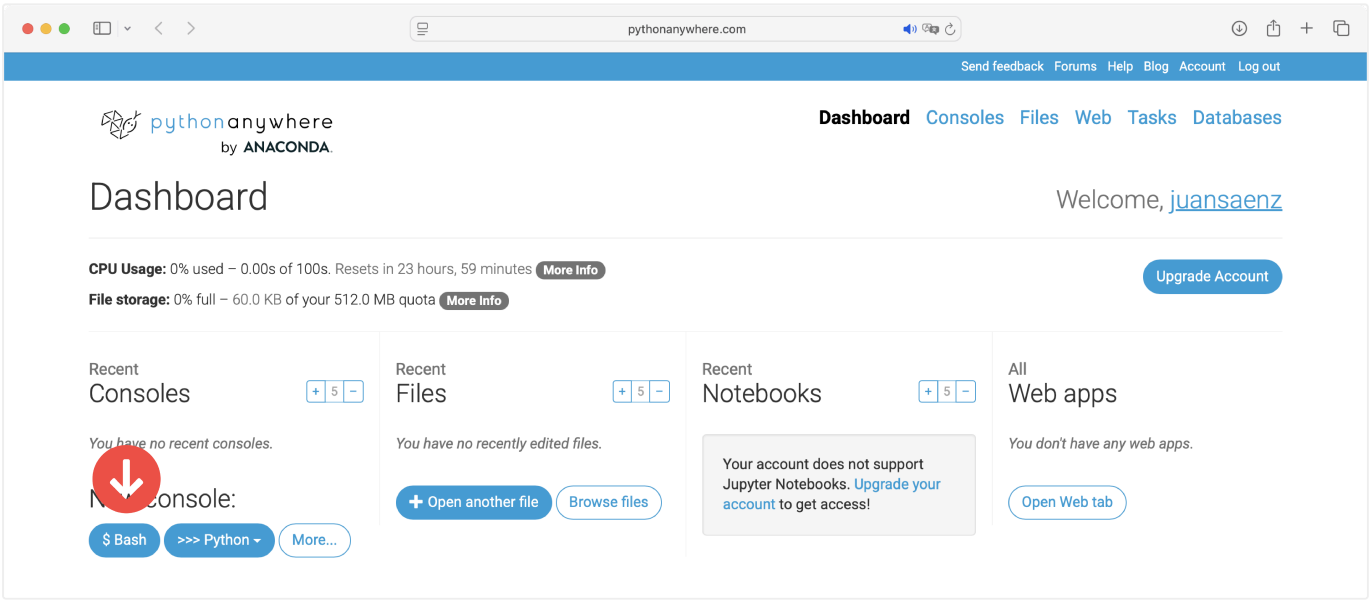

Figure 3: Creating a new Bash console.

4. From the same console, create a new virtual environment by running the following commands:

```
mkvirtualenv --python=/usr/bin/python3.10 my-virtualenv
pip install flask
```

The prompt displayed in the console should now change from `$` to `(my-virtualenv)$`: this is how you know that the `virtualenv` is active.

NOTE: Every time you want to work on the project from the console, you must make sure that the virtualenv is active. You can reactivate it later with `workon my-virtualenv`.

5. Using the console, install all the dependencies used by your social network with the `pip` command. Typically: `Flask`, `flask-login`, `werkzeug`, and `Pillow`.

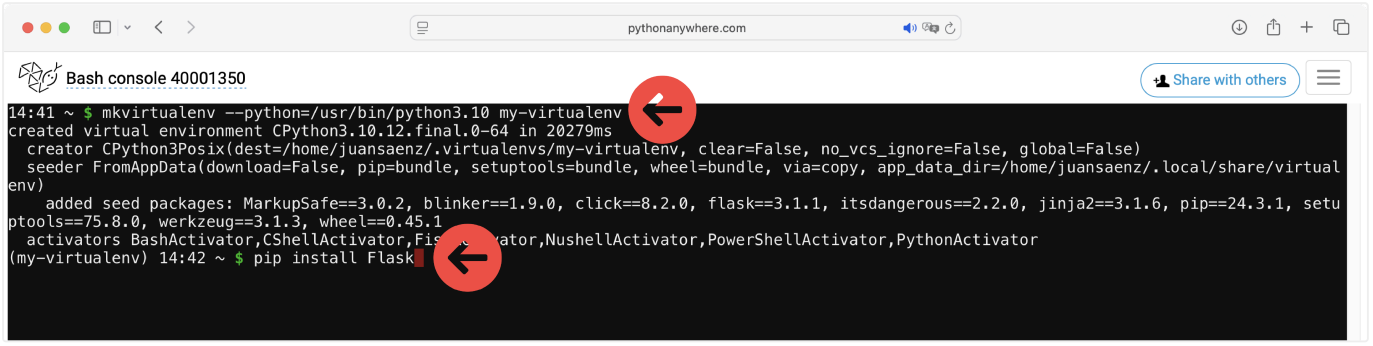

Figure 4: Creating the virtual environment from the console and installing the dependencies.

6. From your personal dashboard, open the "**Web apps**" section, which can be accessed directly from this link: https://www.pythonanywhere.com/web_app_setup.

7. Click on "**Add a new web app**" and follow the steps, making sure to select "**Manual configuration**" when asked to choose the desired Python framework.

NOTE: Make sure you choose the same Python version used in the virtual environment created in step 4, that is, Python 3.10.

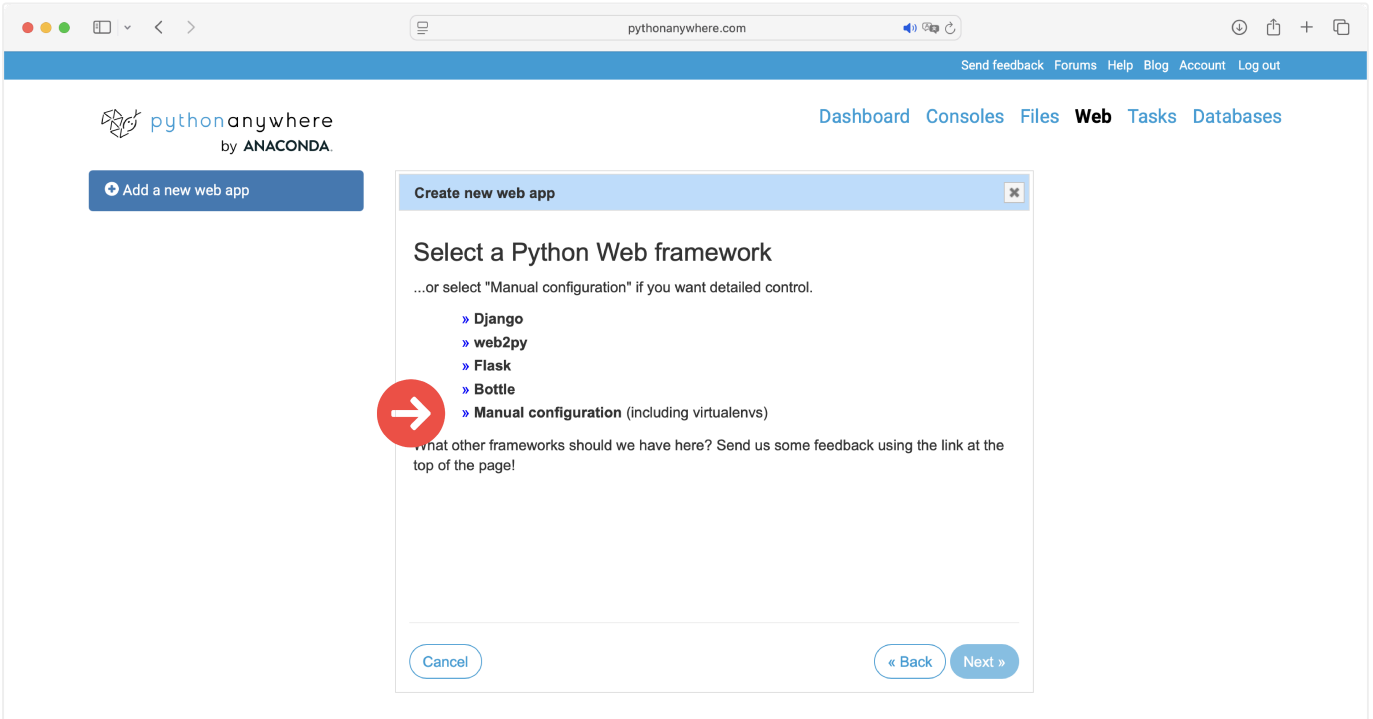

Figure 5: Menu for creating a new Web app.

8. On the page that opens, go to the "**Code**" section and enter the path to the source code uploaded to the platform. Example: `/home/[username]/mysite`, where `mysite` is the name of the project folder that was compressed.
   If there is no "container" folder containing the project files, the path will simply be `/home/[username]/`.

9. In the "**Virtualenv**" section, enter the name of the virtual environment created in step 4: `my-virtualenv`, if you followed the previous instructions exactly.

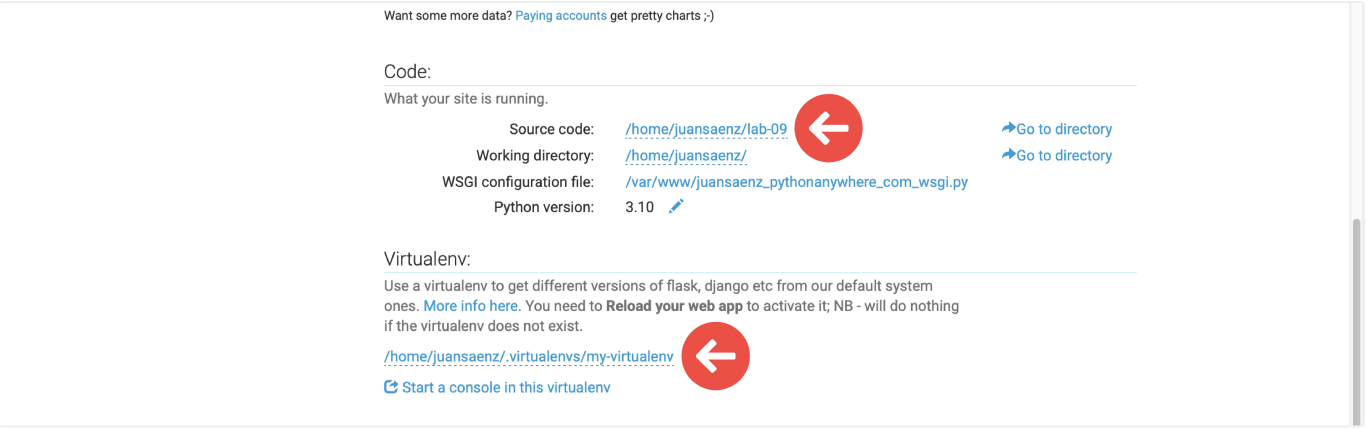

Figure 6: Setting the code and virtual environment.

10. From the same "**Code**" section, open the WSGI configuration file by clicking the corresponding link. In the WSGI file that opens, go to the Flask section and uncomment it, making it look similar to this:

```
# ++++++++++ FLASK ++++++++++
# Flask works like any other WSGI-compatible framework, we just need
# to import the application.  Often Flask apps are called "app" so we
# may need to rename it during the import:
#
#
import sys
#
## The "/home/juansaenz" below specifies your home
## directory -- the rest should be the directory you uploaded your Flask
## code to underneath the home directory.  So if you just ran
## "git clone git@github.com/myusername/myproject.git"
## ...or uploaded files to the directory "myproject", then you should
## specify "/home/juansaenz/myproject"
path = '/home/juansaenz/lab-09'
if path not in sys.path:
    sys.path.append(path)
#
from app import app as application  # noqa
#
```

Figure 7: WSGI configuration file.

11. Modify line 111 of the WSGI file (**Figure 7**). Note that the first term `app` corresponds to the name of your project's main file, for example `app.py`, while the second term `app` refers to the name of the application variable, as in `app = Flask(__name__)`.

12. Since the project is now being run from a different path, you need to update the paths of the static folder and the database inside your `app.py` file and DAO files so that they can be accessed correctly. To do this, go back to the "**Web apps**" page and, in the Code section, open the source code by clicking "**Go to directory**".

13. Modify the paths inside the files by adding `mysite/` at the beginning, where `mysite` always represents the project's main folder. For example, in the Lab 9 solution, the prefix `lab-9/` was added in the functions responsible for managing images, as shown in **Figure 8** on line 92. Similarly, in the DAO files, it was necessary to adapt the database path because the SQLite file was located in the `db` folder, as shown in **Figure 9** on line 6.

```
82          size = POST_IMG_WIDTH, new_height
83          img.thumbnail(size, Image.Resampling.LANCZOS)
84
85          # Extracting file extension from the image filename
86          ext = post_image.filename.split(".")[-1]
87          # Getting the current timestamp in seconds
88          secondi = int(datetime.now().timestamp())
89
90          # Saving the image with a unique filename in the 'static' directory
91          img.save(
92              "lab-09/static/@" + current_user.nickname.lower() + "-" + str(secondi) + "." + ext
93          )
94
95          # Updating the 'immagine_post' field in the post dictionary with the image filename
96          post["immagine_post"] = (
97              "@" + current_user.nickname.lower() + "-" + str(secondi) + "." + ext
98          )
99
100     post["id_utente"] = int(current_user.id)
101     success = posts_dao.add_post(post)
102
```

Figure 8: Example of modifying the static folder path in app.py.

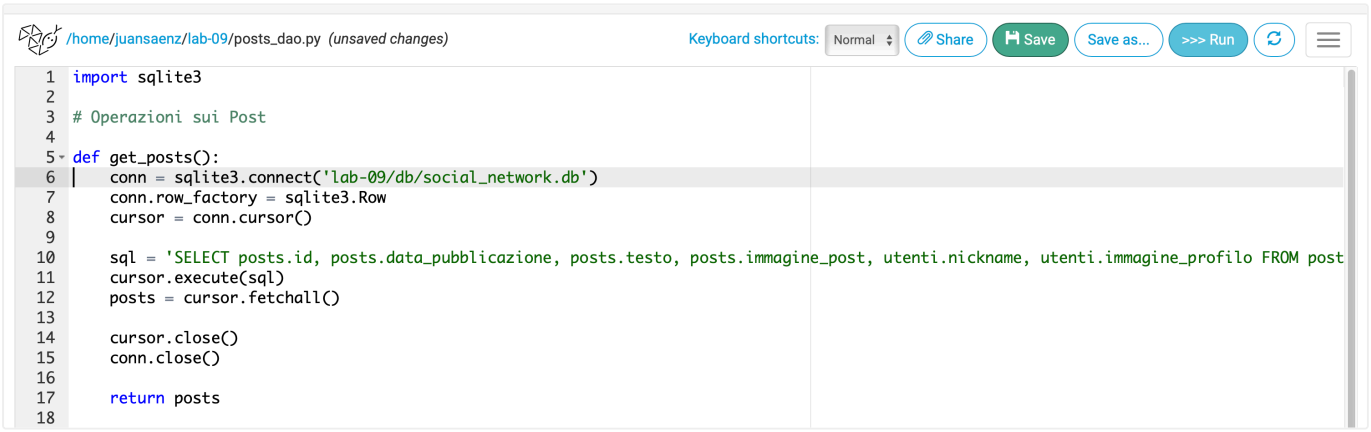

Figure 9: Example of modifying the db folder path in posts_dao.py.

14. If all the steps have been completed correctly, the social network is now accessible at `http://[username].pythonanywhere.com/`, for example http://juansaenz.pythonanywhere.com.

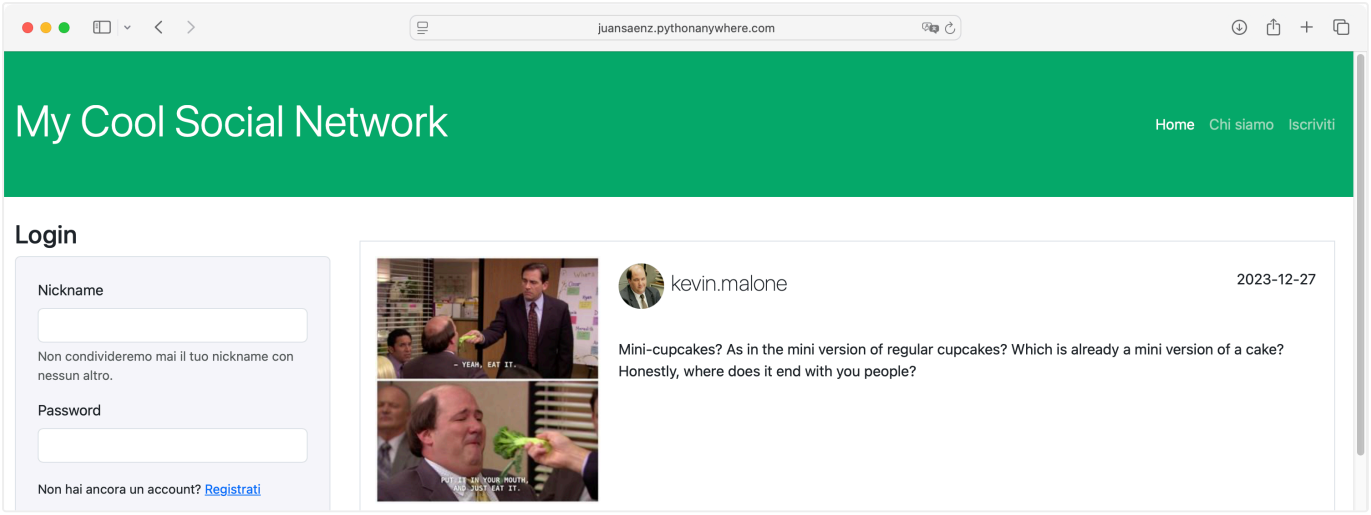

Figure 10: Successful deployment of the social network on PythonAnywhere.

Notes

1. If there are errors when accessing the site, you should check the **Error log**, available from the application page in the **Log files** section, as shown in **Figure 11**.

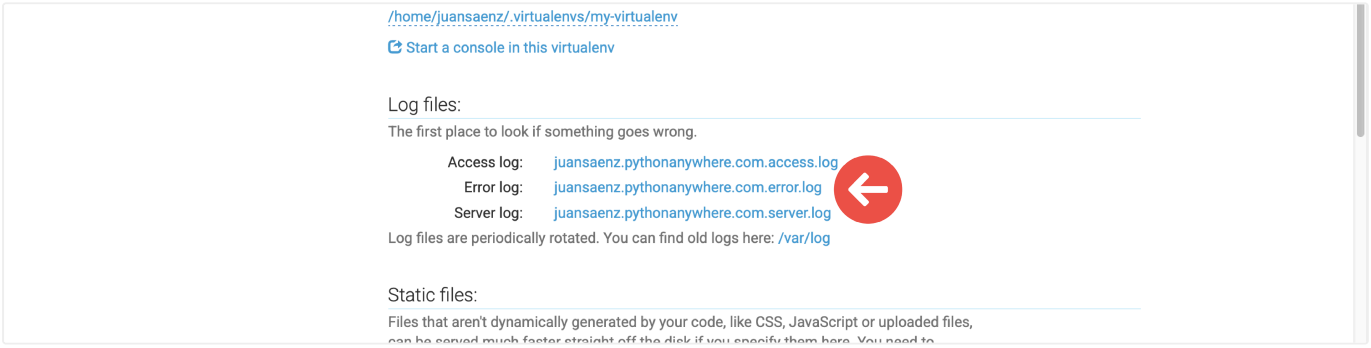

Figure 11: Error log.

2. Similarly, if you make changes to the code, you must save them and then click the button to reload the application (**Figure 12**).

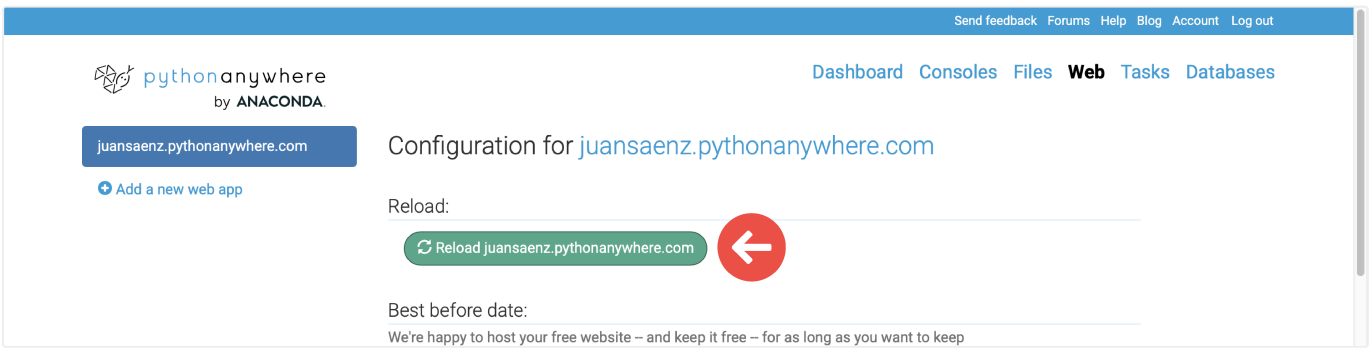

Figure 12: Reloading the application.