**Deploy python flask app on a cloud platform**

**Ex: 7**

**DATE:**

**Aim:** To Deploy a python application on cloud using third party cloud service applications.

**Description**

PythonAnywhere is an online integrated development environment (IDE) and web hosting service (Platform as a service) based on the Python programming language. It provides in-browser access to server-based Python and Bash command-line interfaces, along with a code editor with syntax highlighting. Program files can be transferred to and from the service using the user's browser. Web applications hosted by the service can be written using any WSGI-based application framework. PythonAnywhere was created by Resolver Systems, who also produced Resolver One, a Python-based Spreadsheet program. On 16 October 2012 the product was acquired by a new company, PythonAnywhere LLP, who took on the existing development team. In June, 2022, PythonAnywhere was acquired by Anaconda, Inc. The development team uses PythonAnywhere to develop PythonAnywhere, and say that its collaboration features help because they use the extreme programming methodology.

**Steps to deploy**

**Sign up for PythonAnywhere -** If you don't already have an account on PythonAnywhere, you need to sign up for one. PythonAnywhere offers a free tier that you can use to deploy small-scale projects.

**Set up a virtual environment (optional) -** It's a good practice to create a virtual environment for your Flask app to manage dependencies and isolate your project. You can create a virtual environment using the following command in a PythonAnywhere terminal:

**mkvirtualenv --python=/usr/bin/python3.8 myenv**

Replace **"myenv"** with your desired virtual environment name and adjust the Python version if needed.

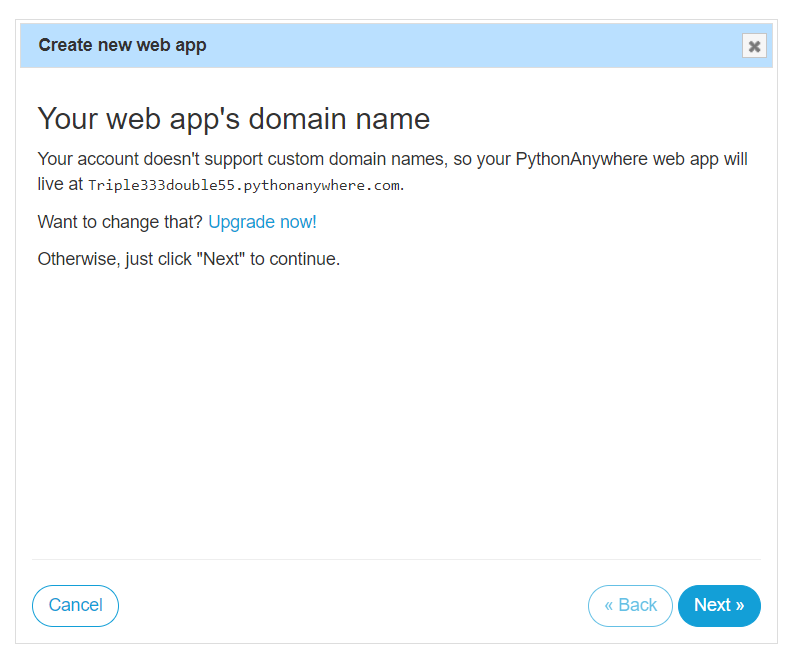
**Clone your Flask app repository -** If your Flask app is in a version-controlled repository (e.g., Git), you can clone it into your PythonAnywhere account. Use the git clone command to fetch your code.

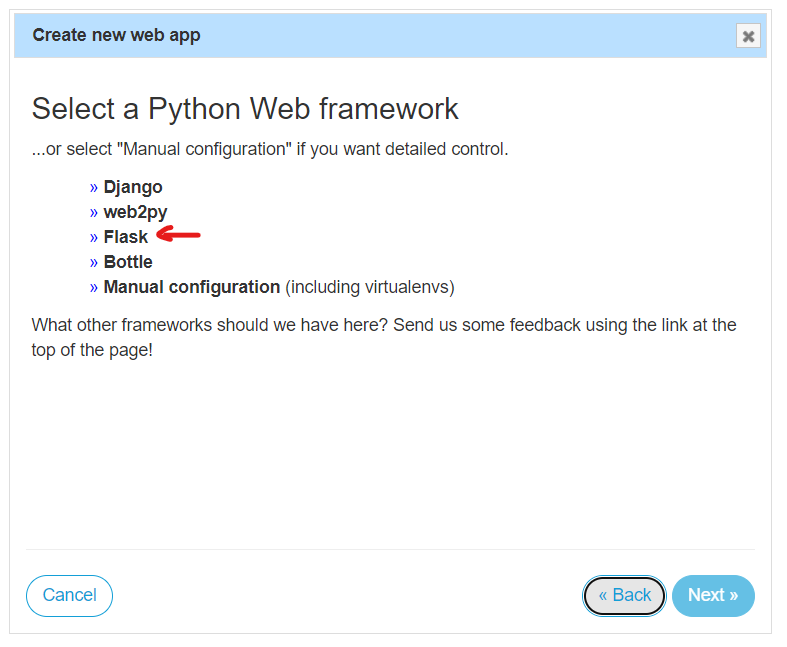
**Set up your Flask app on PythonAnywhere -** Navigate to the "Web" tab on your PythonAnywhere dashboard.

* Click the "Add a new web app" button.
* Select "Manual configuration."
* Choose the Python version you want to use (e.g., Python 3.8).
* Click "Next" to create the web app.

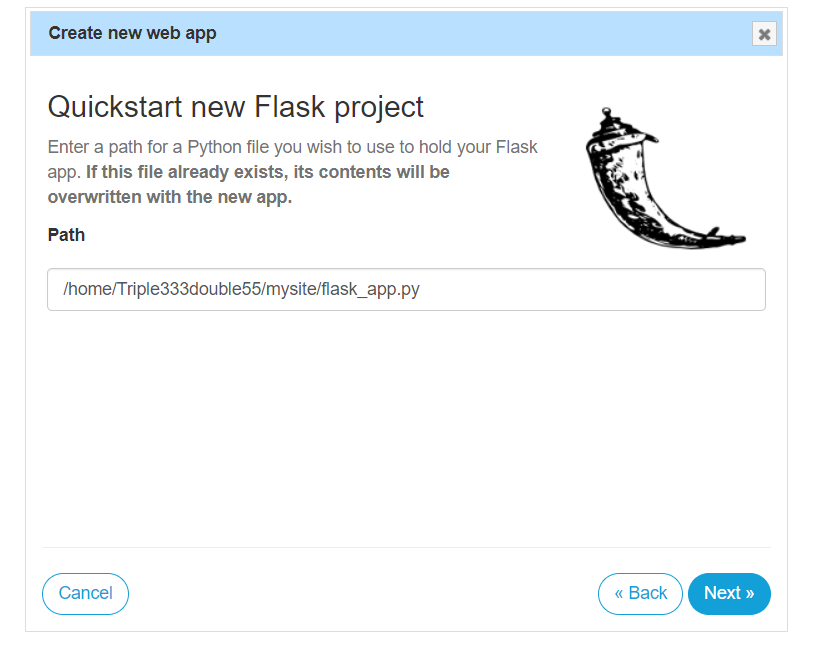
A close-up of a white background

Description automatically generated









**Configure the web app -** After creating the web app, you'll be redirected to the "Web" tab where you can configure it.

* In the "Code" section, set the source code directory to your Flask app's project directory.
* In the "Virtualenv" section, select the virtual environment you created earlier (if applicable).
* In the "WSGI configuration file" section, set the path to your Flask app's WSGI file. The default value is usually fine: /var/www/your\_username\_pythonanywhere\_com\_wsgi.py.
* Click "Save" to apply the changes.

**Install dependencies -** In the PythonAnywhere terminal, navigate to your project directory and install your app's dependencies using pip:

**pip install -r requirements.txt**

**Configure your database (if applicable) -** If your Flask app uses a database, you'll need to configure your database connection settings in your Flask app's configuration. Make sure your database is accessible from PythonAnywhere.

**Restart the web app -** To start your Flask app, return to the "Web" tab in your PythonAnywhere dashboard, and click the "Reload" button next to your web app.

**Access your Flask app -** You can access your deployed Flask app by clicking on the "Open Web" button in the "Web" tab. PythonAnywhere provides you with a URL for your app (e.g., [**https://your\_username.pythonanywhere.com**](https://your_username.pythonanywhere.com)).

A screenshot of a computer

Description automatically generated

Your Flask app should now be up and running on PythonAnywhere. You can make updates to your app, commit changes to your repository, and reload the web app as needed to deploy new versions. Remember that PythonAnywhere's free tier has limitations, so consider upgrading to a paid plan if your project requires more resources or advanced features.

**Code**

A red arrow pointing to a box

Description automatically generated

A screenshot of a computer

Description automatically generated

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello from Flask!'

@app.route('/sample')

def sample\_route():

return 'This is a new route added!!!'

**Output**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**