Deploying a Python Flask Web App to Azure Using GitHub

Goal:

The aim of this project was to create a simple Python Flask web application, deploy it to Azure App Service, and integrate it with GitHub for continuous deployment.

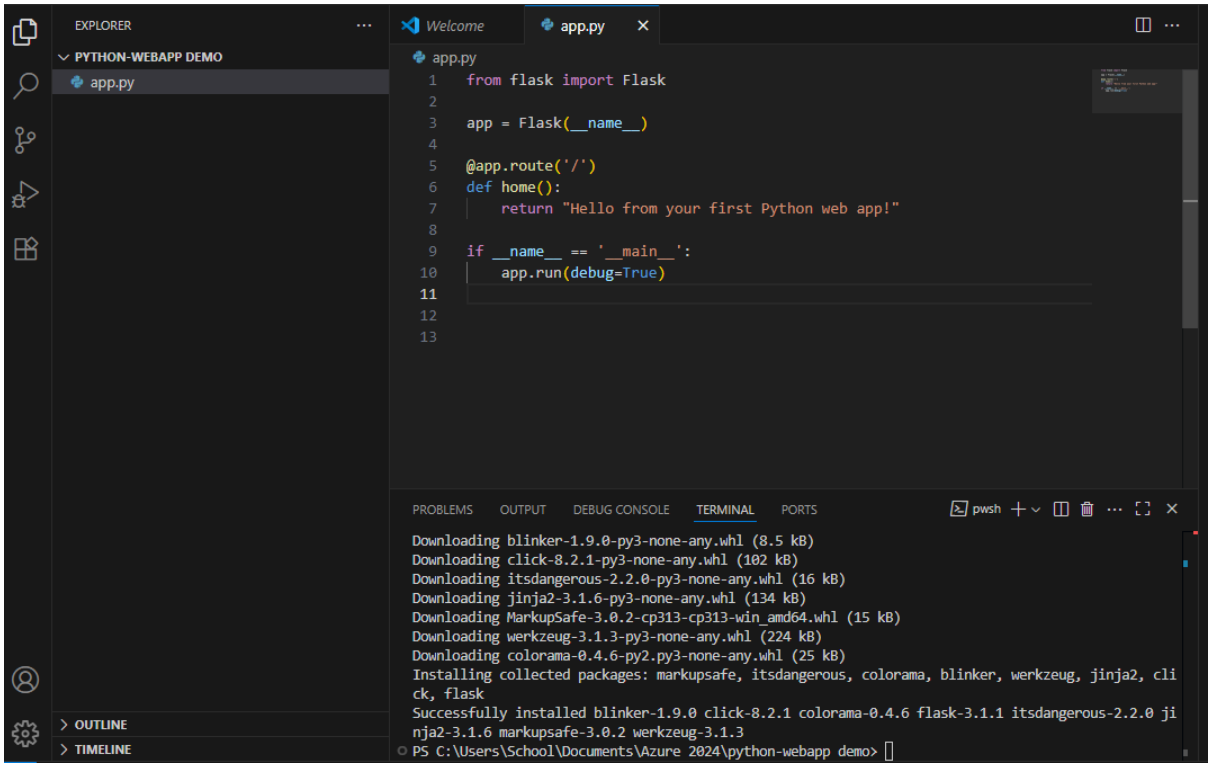
Step 1 - Creating the Flask App Locally

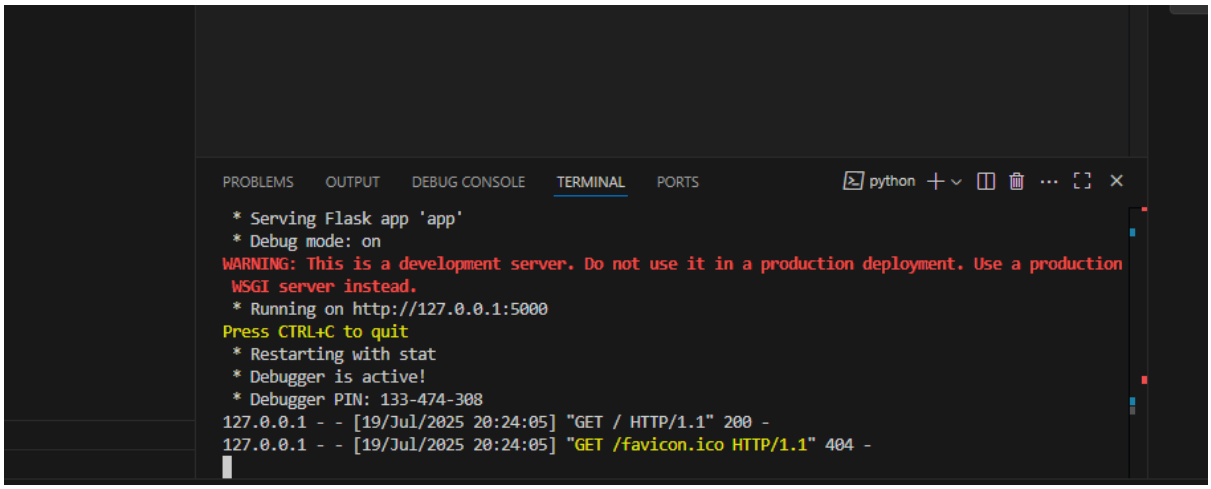
I started by setting up a basic Flask application with two routes:

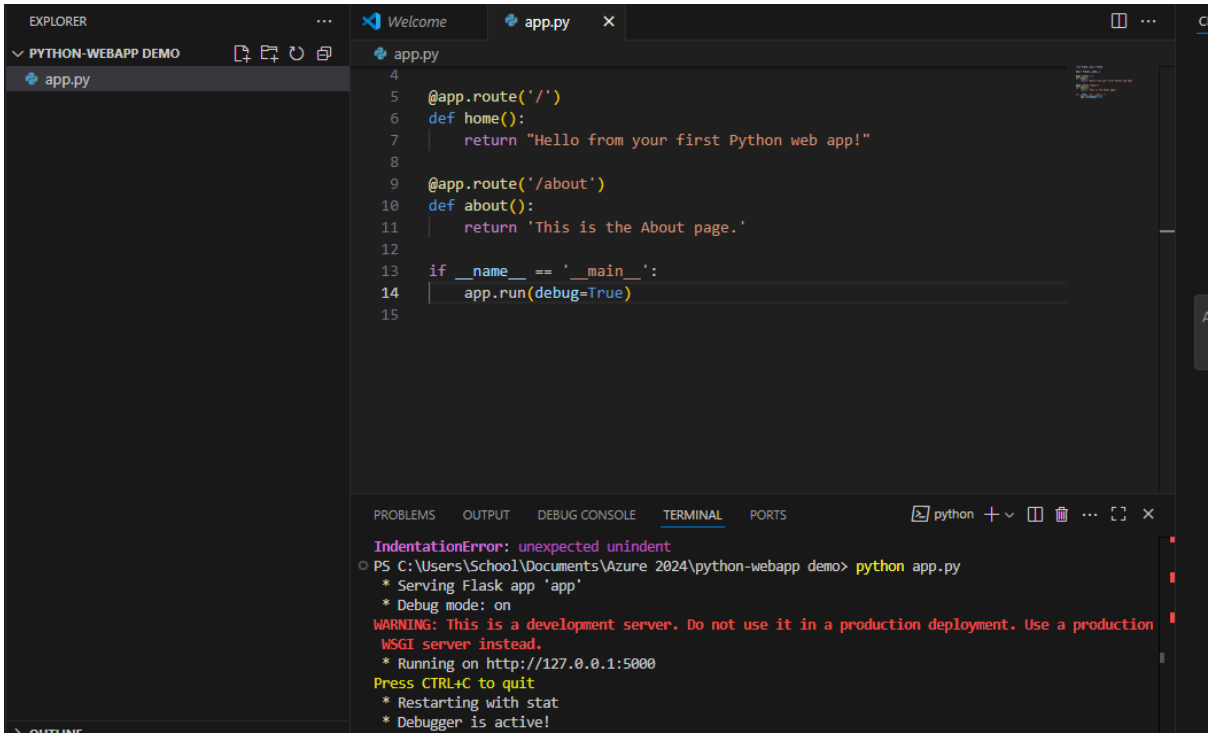
- Home route, displaying “Hello from your first Python web app”.
- About route, displaying “This is the about page”.

Screenshots:

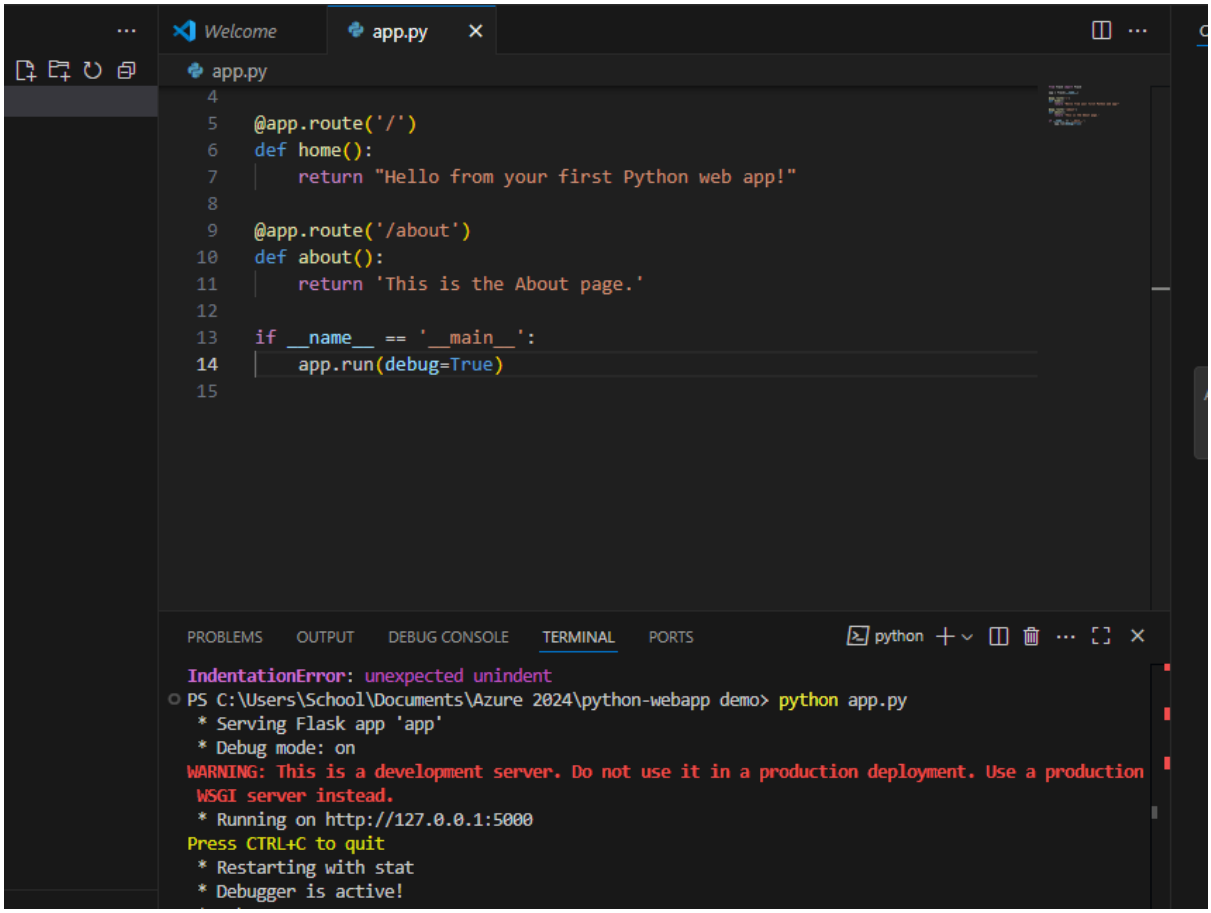
1. Downloading and installing dependencies (Blinker 1.0 and Flask setup).
2. Running the app in debug mode locally.
3. Accessing the Home route in the browser.
4. Accessing the About page.
5. Updating routes to render HTML templates (home.html).


EXPLORER
PYTHON-WEBAPP DEMO
app.py
Welcome
app.py
app.py
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def home():
7 return "Hello from your first Python web app!"
8
9 if __name__ == '__main__':
10 app.run(debug=True)
11
12
13
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
pwsh
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading click-8.2.1-py3-none-any.whl (102 kB)
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
Downloading MarkupSafe-3.0.2-cp313-cp313-win_amd64.whl (15 kB)
Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: markupsafe, itsdangerous, colorama, blinker, werkzeug, jinja2, cli
ck, flask
Successfully installed blinker-1.9.0 click-8.2.1 colorama-0.4.6 flask-3.1.1 itsdangerous-2.2.0 ji
nja2-3.1.6 markupsafe-3.0.2 werkzeug-3.1.3
PS C:\Users\School\Documents\Azure 2024\python-webapp demo>
OUTLINE
TIMELINE

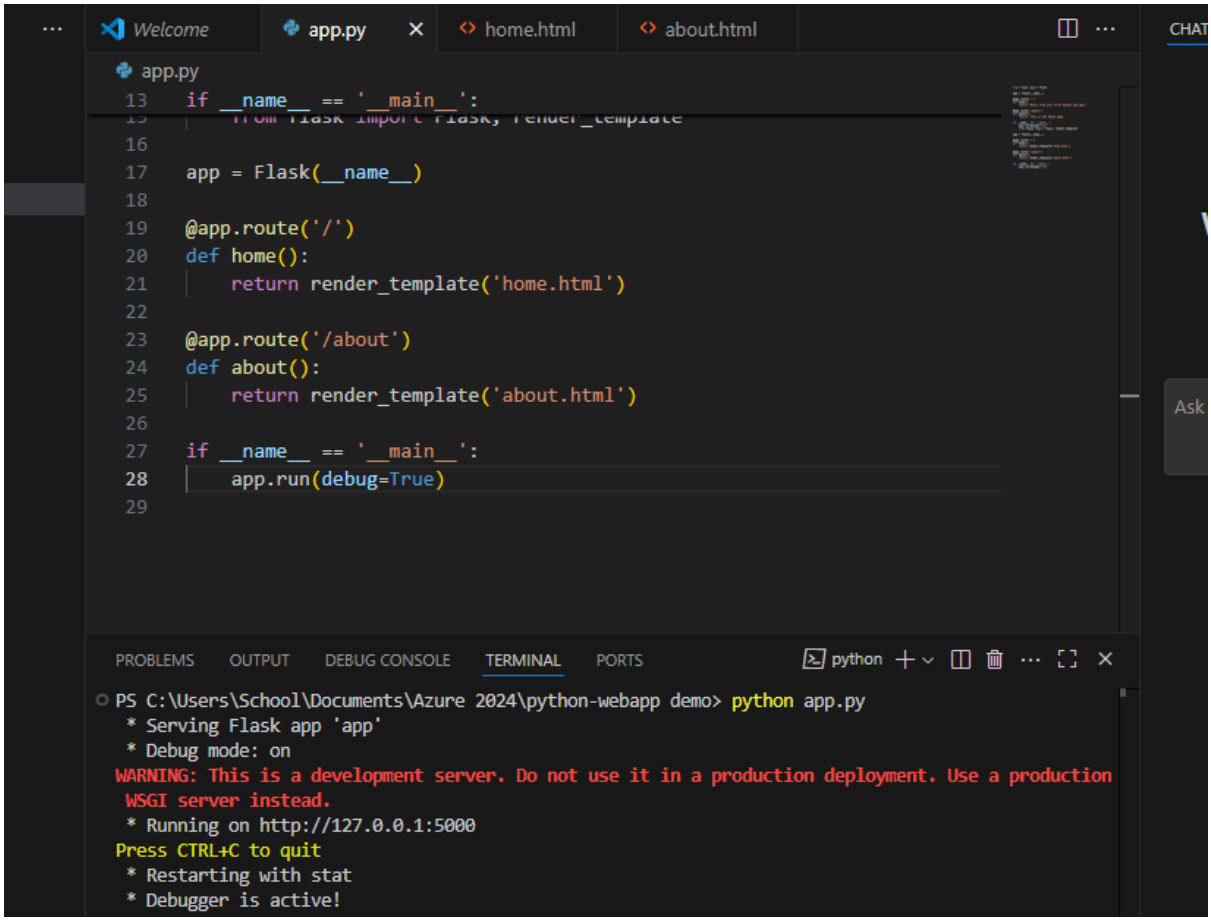

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
python
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production
WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 133-474-308
127.0.0.1 - - [19/Jul/2025 20:24:05] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Jul/2025 20:24:05] "GET /favicon.ico HTTP/1.1" 404 -



Hello from your first Python web app!



This is the About page.



Step 2 - Setting Up Azure Resources

I logged into Azure and created the following resources:

- Resource Group for the project.
- App Service Plan to host the app.
- Web App for deployment.

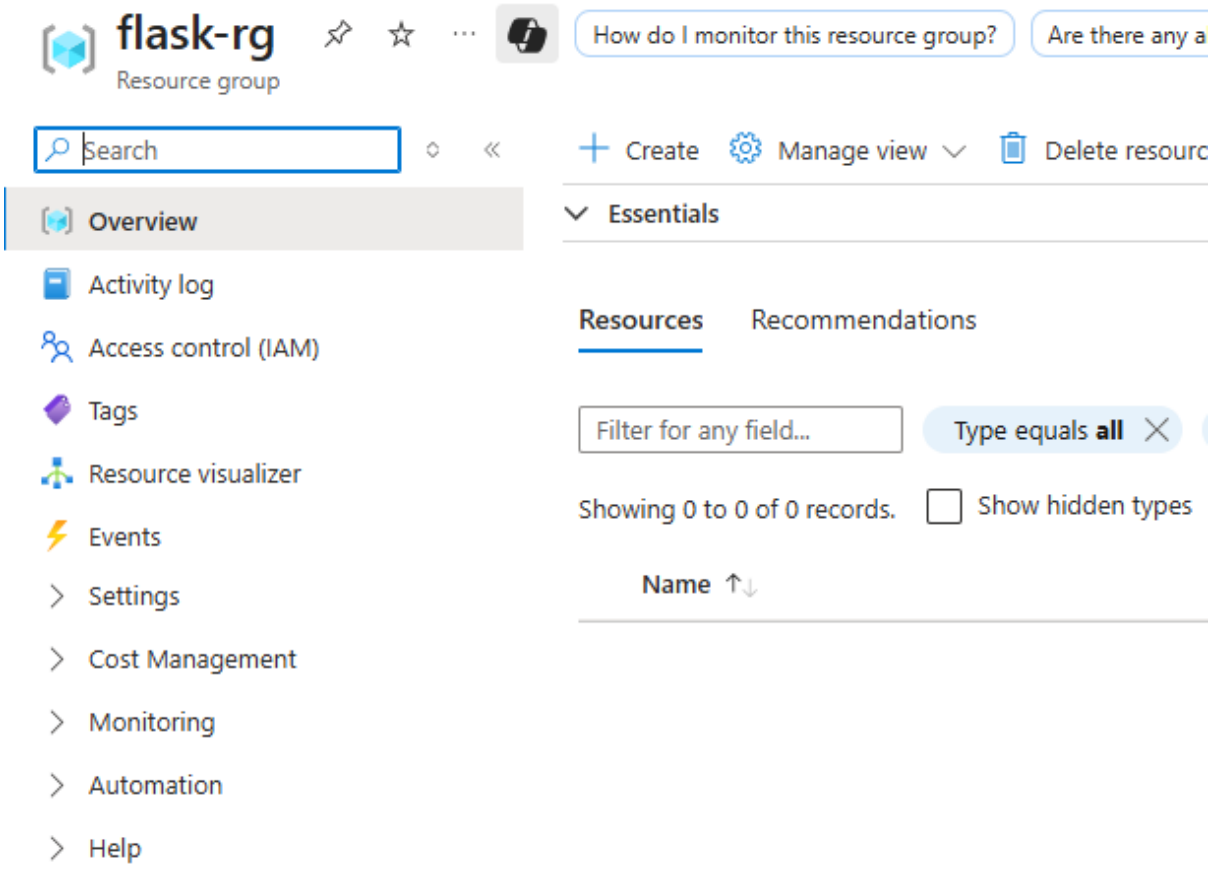
Screenshots:

6. Azure Resource Group creation.

7. App Service Plan setup.

8. Web App created.

9. Default Azure Web App “running and waiting for content” page.


flask-rg
Resource group
How do I monitor this resource group?
Are there any a
Search
Create
Manage view
Delete resourc
Overview
Activity log
Access control (IAM)
Tags
Resource visualizer
Events
Settings
Cost Management
Monitoring
Automation
Help
Essentials
Resources
Recommendations
Filter for any field...
Type equals all
Showing 0 to 0 of 0 records.
Show hidden types
Name

Create App Service Plan

Basics Tags **Review + create**

Summary

App Service Plan
by Microsoft

Details

Subscription	Azure subscription 1
Resource Group	flask-rg
Name	flask-appserviceplan
Operating System	Linux
Region	South Africa North
SKU	Free
ACU	Shared infrastructure
Memory	1 GB memory

Create Web App

Basics Database Deployment Networking Monitor + secure Tags **Review + create**

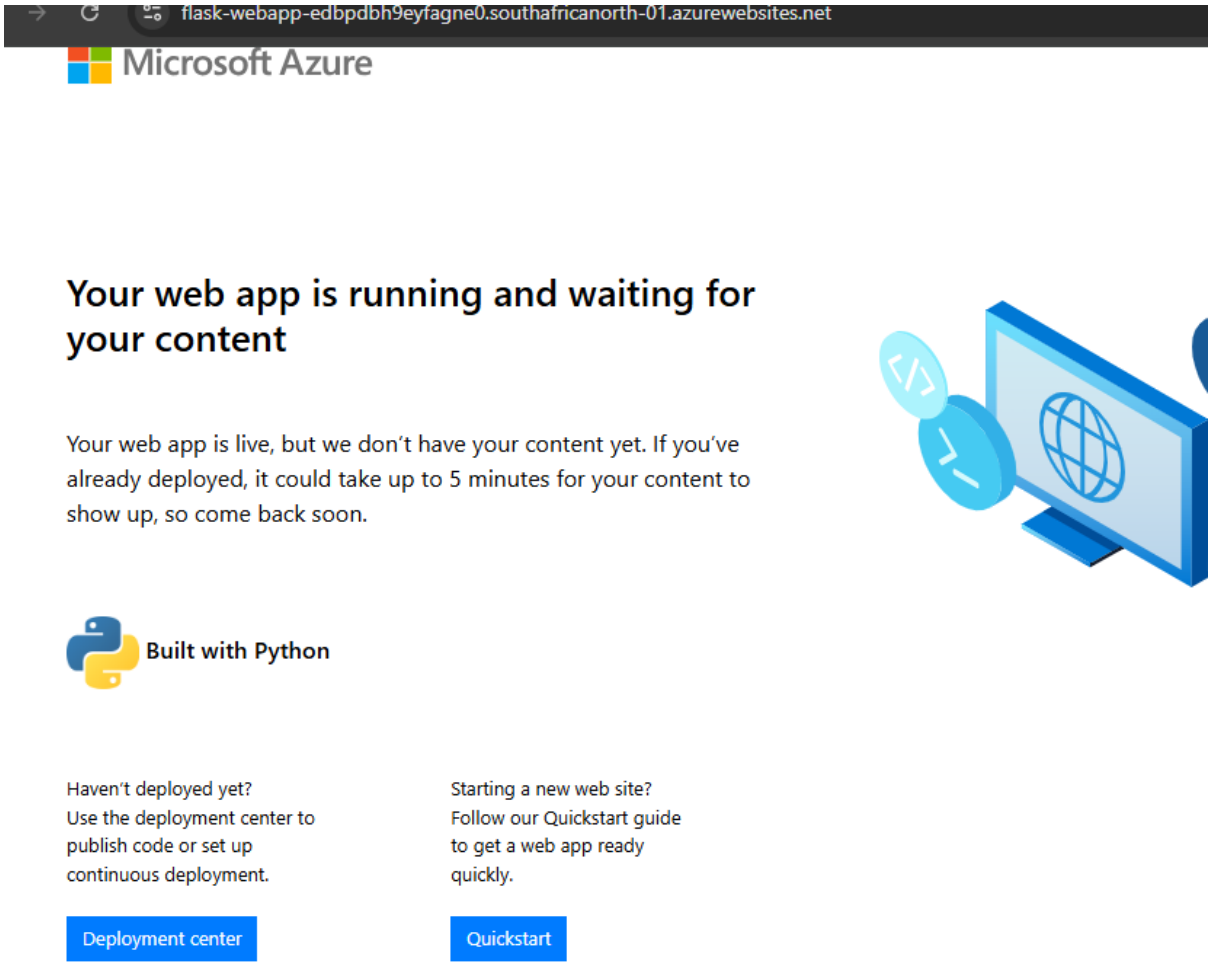
Summary

Web App
by Microsoft

Basic authentication for this app is currently disabled and may impact deployments. Click to learn more.

Details

Subscription	a4fa4fb6-daa1-43d2-9d72-42a1674464b0
Resource Group	flask-rg
Name	flask-webapp
Secure unique default hostname	Enabled
Publish	Code
Runtime stack	Python 3.11



Step 3 - Configuring GitHub for Deployment

I initialized Git in the project folder and set my global configuration:

- Username and email for Git commits.
- Initial commit for the Flask app.

I then connected the local repository to GitHub:

- Added the remote URL for the repository.
- Pushed the project files to GitHub.

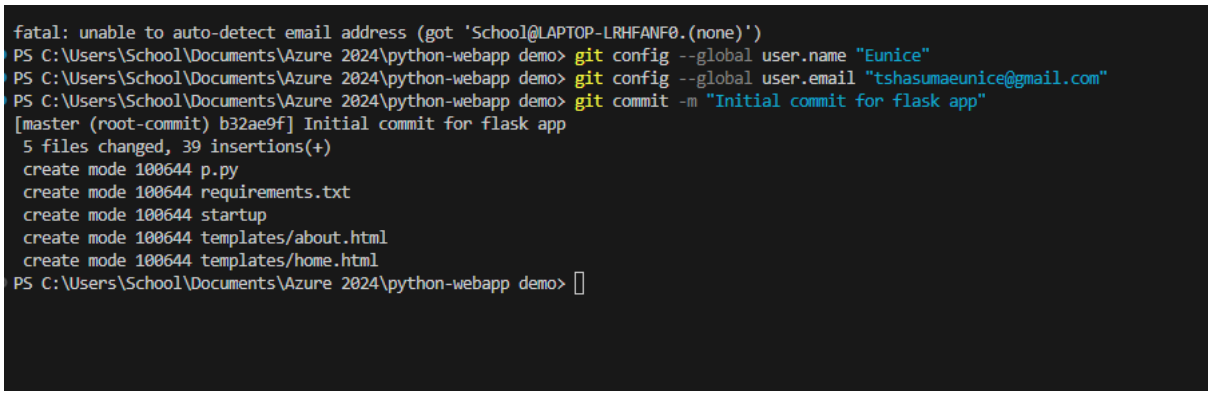
Screenshots:

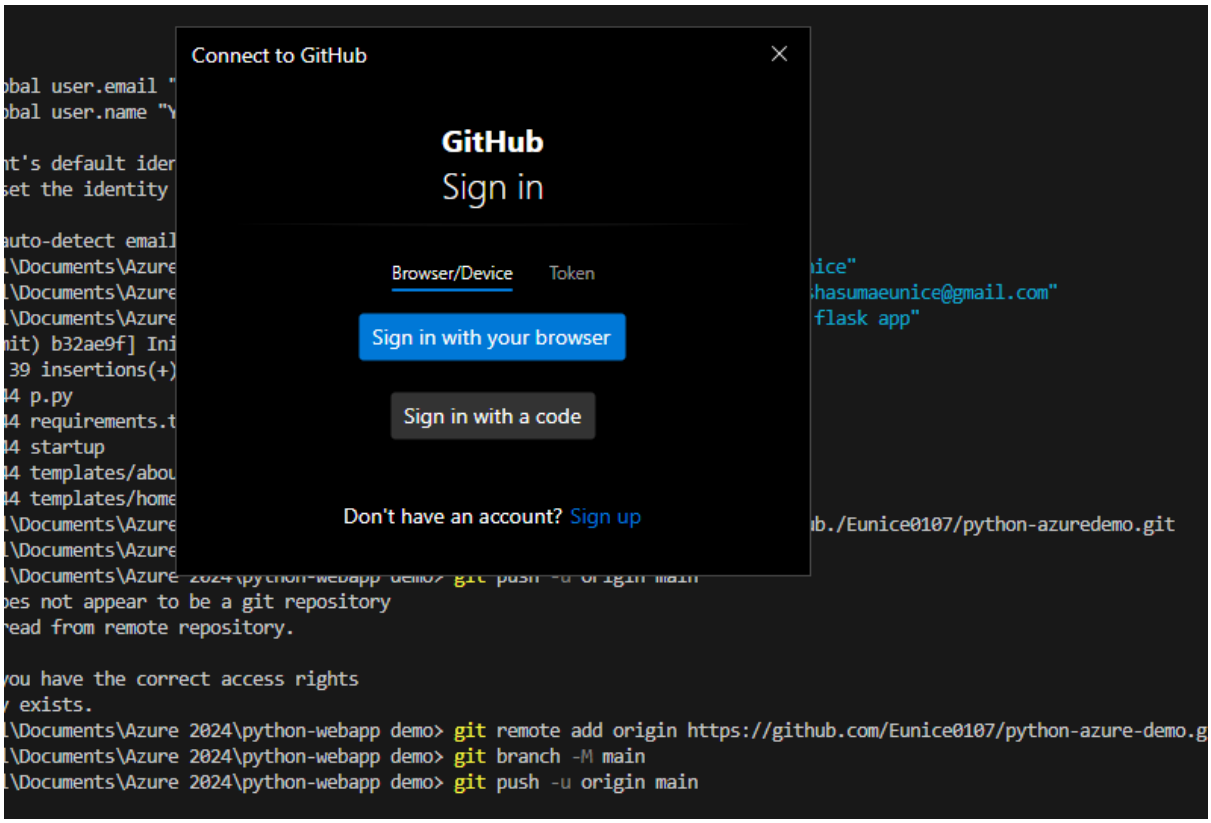
10. Git config setup commands.

11. GitHub sign-in in the browser.

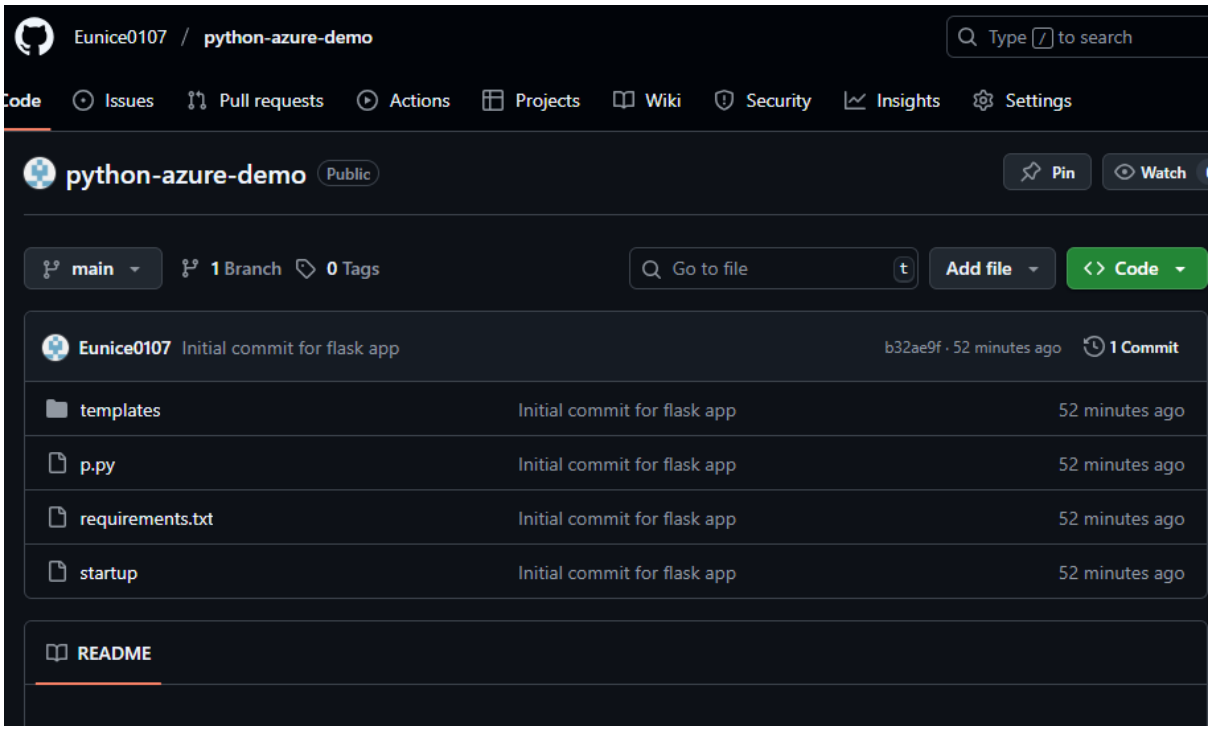
12. Adding the GitHub remote URL.

13. GitHub repository view showing project files (templates/, p.py, requirements.txt, startup.txt).






```
PS C:\Users\School\Documents\Azure 2024\python-webapp demo> git remote add origin https://github.com/Eunice0107/python-azure-demo.git
PS C:\Users\School\Documents\Azure 2024\python-webapp demo> git branch -M main
PS C:\Users\School\Documents\Azure 2024\python-webapp demo> git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 883 bytes | 55.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Eunice0107/python-azure-demo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\School\Documents\Azure 2024\python-webapp demo>
```



Step 4 - Deploying to Azure

Azure was linked to the GitHub repository for automatic deployment.

Once the code was pushed, Azure built and deployed the app.

Screenshots:

14. Azure build and deployment process.

15. Final confirmation - “Hi Python Developer, your app is up and running” in the browser.