Python Web App Deployment Server by Code Creator

AWS Marketplace Customer Instructions

This server provides a ready to use Python web application deployment environment with Flask, FastAPI, Gunicorn, Uvicorn, Nginx, Docker, and Docker Compose. It includes working starter applications, helper commands, and first boot automation that detects the public IP address of your instance.

1. Quick access summary

After the instance launches, **wait about 2 to 5 minutes** for first boot setup to complete. Then connect to the instance and open the web URLs below.

Item	Value
Operating system user	ubuntu
Main landing page	http://YOUR_PUBLIC_IP
Flask starter app	http://YOUR_PUBLIC_IP/flask/
FastAPI starter app	http://YOUR_PUBLIC_IP/fastapi/
FastAPI API documentation	http://YOUR_PUBLIC_IP/fastapi/docs
Main project folder	/opt/codecreator-python-apps

2. Required AWS security group ports

Open only the ports you need. For normal use, SSH and HTTP are enough.

Type	Protocol	Port	Purpose
SSH	TCP	22	Connect to the instance using your Amazon private key and the ubuntu user
HTTP	TCP	80	Open the landing page, Flask starter app, and FastAPI starter app

3. Connect to your instance

Connect to your instance using your Amazon private key and the **ubuntu** user. Replace YOUR_KEY.pem and YOUR_PUBLIC_IP with your own key file and instance public IP address.

```
ssh -i YOUR_KEY.pem ubuntu@YOUR_PUBLIC_IP
```

When you log in, the server displays a message of the day with the most useful commands. You can also read the first login file directly:

```
cat /home/ubuntu/FIRST_LOGIN.txt
```

4. First boot setup

The AMI includes a one time first boot setup service. On a new instance launch, it detects the instance public IP address, writes the customer first login file, prepares the runtime environment, and starts the Docker Compose stack.

- Wait about 2 to 5 minutes after launch before testing the web page.
- The first boot notes are written to /home/ubuntu/FIRST_LOGIN.txt.
- The service is designed to run once for each fresh instance launched from the AMI.

5. Open the web interfaces

Use your instance public IP address in a browser.

http://YOUR_PUBLIC_IP

http://YOUR_PUBLIC_IP/flask/

http://YOUR_PUBLIC_IP/fastapi/

http://YOUR_PUBLIC_IP/fastapi/docs

The main landing page provides links to both starter applications and their health checks.

6. Useful helper commands

Run these commands from the instance as the ubuntu user.

- Show URLs:
  codecreator-python-url
- Show service status and health checks:
  codecreator-python-status
- Show recent logs for all services:
  codecreator-python-logs
- Show Flask logs:
  codecreator-python-logs flask
- Show FastAPI logs:
  codecreator-python-logs fastapi
- Show Nginx logs:
  codecreator-python-logs nginx
- Restart and rebuild the stack after changes:
  codecreator-python-restart

7. Replace the starter applications with your own code

The instance includes two starter patterns. You can use one or both depending on your project.

- Use the Flask folder for traditional Python web applications that run through Gunicorn.
- Use the FastAPI folder for APIs, AI helper services, webhook receivers, automation backends, and modern Python services that run through Uvicorn.

Flask application folder:
cd /opt/codecreator-python-apps/flask-app

FastAPI application folder:
cd /opt/codecreator-python-apps/fastapi-app

After editing app files or requirements, restart and rebuild the stack:
codecreator-python-restart

8. Add Python packages

Each app has its own requirements file. Add Python packages to the correct requirements.txt file, then rebuild the stack.

nano /opt/codecreator-python-apps/flask-app/requirements.txt

nano /opt/codecreator-python-apps/fastapi-app/requirements.txt

codecreator-python-restart

9. Environment variables and API keys

No API keys are required to use the included starter apps. If your own application needs API keys or other secrets, add them carefully and avoid committing them into source code.

For simple testing, you can add variables to the runtime environment file:

sudo nano /opt/codecreator-python-apps/runtime/customer.env

Then restart the stack:

codecreator-python-restart

For production use, follow your organization security policy for secret storage and rotation.

10. Basic troubleshooting

Start with the status command. It checks Docker, containers, the landing page, Flask health, FastAPI health, and FastAPI docs.

codecreator-python-status

If a service is unhealthy or not responding, check logs:

codecreator-python-logs

codecreator-python-logs flask

codecreator-python-logs fastapi

codecreator-python-logs nginx

Restart the stack after making changes:

codecreator-python-restart

If the web page does not load from your browser, confirm your AWS security group allows inbound TCP 80 and that the instance has a public IP address.

11. Important file locations

Path	Purpose
/opt/codecreator-python-apps	Main project directory
/opt/codecreator-python-apps/flask-app	Flask starter application
/opt/codecreator-python-apps/fastapi-app	FastAPI starter application
/opt/codecreator-python-apps/nginx/default.conf	Nginx routing configuration
/opt/codecreator-python-apps/docker-compose.yml	Docker Compose stack definition
/opt/codecreator-python-apps/runtime/customer.env	Runtime environment values
/home/ubuntu/FIRST_LOGIN.txt	First login notes generated on first boot

12. Updating the server

You may apply normal Ubuntu security updates. Review changes carefully before updating application dependencies in production.

sudo apt update

sudo apt -y upgrade

After major updates, confirm the stack is healthy:

codecreator-python-status

13. Helpful official resources

These resources may help you customize the starter applications:

- Flask documentation: https://flask.palletsprojects.com/
- FastAPI documentation: https://fastapi.tiangolo.com/
- Gunicorn documentation: https://gunicorn.org/
- Uvicorn documentation: https://uvicorn.dev/
- Nginx documentation: https://nginx.org/en/docs/
- Docker documentation: https://docs.docker.com/

14. Support note

This AMI provides a ready to use deployment foundation. Your own application code, custom API integrations, third party service accounts, API keys, database design, and production security requirements are your responsibility.