

Deploying a Python Application with Nginx Proxy

Introduction

This project provides a comprehensive guide to deploying a Python application in a production environment with a proxy server. It walks through setting up the development and server environment, installing necessary dependencies, configuring the application, and integrating a proxy server (such as Nginx) to efficiently route and manage incoming traffic. By following this guide, the application can achieve improved performance, scalability, and security.

Features

- Production-Ready Deployment – Configured to run smoothly in a production environment.
- Proxy Server Integration – Uses Nginx (or Apache) to efficiently manage and route incoming requests.
- Environment Management – Supports virtual environments and .env configuration for secure and isolated setups.
- Security and Performance – Proxy setup improves request handling, security headers, and overall app performance.

Prerequisites

Before deploying this Python application, ensure the following are installed and configured:

- Python 3.x – The application requires Python 3 or higher.
- Pip – Python package manager to install dependencies.
- Git (optional) – For cloning the repository.
- Virtual Environment (venv) – Recommended for isolating project dependencies.
- Proxy Server – Nginx or Apache for handling and routing incoming requests.

Steps to Deploy

Step 1: Launch EC2 instance and Establishing a secure connection to your EC2 instance

1. Launch instance

The screenshot shows the AWS EC2 Instances page. On the left, a sidebar menu is open under the 'Instances' section, listing various options like Instances, Instance Types, Launch Templates, and Capacity Reservations. The main content area displays a table titled 'Instances (1)'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. One row is visible for an instance named 'python' with the ID 'i-0e7def9ecfba8b922'. The instance is shown as 'Running' with a green checkmark, of type 't3.micro', and its status is 'Initializing'. At the top right of the table, there are buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below the table, a section titled 'Select an instance' is visible.

2. Copy the SSH command

The screenshot shows the 'Connect to instance' page for the instance 'i-0e7def9ecfba8b922'. The top navigation bar includes links for CloudShell, Feedback, and Console Mobile App. The main content area is titled 'Connect' and provides instructions for connecting using a browser-based client. It lists four steps: opening an SSH client, locating the private key file 'server-key.pem', changing its permissions to 400, and connecting to the instance's Public DNS. A callout bubble indicates that the command has been copied. Below this, a terminal window shows the copied command: 'ssh -i "server-key.pem" ec2-user@ec2-3-238-230-120.compute-1.amazonaws.com'. A note at the bottom states: 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.' The bottom of the screen shows a Windows taskbar with various pinned icons and system status information.

3. Paste command in Git bash

```
ec2-user@ip-172-31-64-114:~  
hp@DESKTOP-M40M3TO MINGW64 /c/geeta_workspace/ssh-key  
$ ssh -i "server-key.pem" ec2-user@ec2-3-238-230-120.compute-1.amazonaws.com  
The authenticity of host 'ec2-3-238-230-120.compute-1.amazonaws.com (64.179.100.3) can't be established.  
ED25519 key fingerprint is SHA256:20Xn/qFx0aIor/tiU3eczvK112Xsh2tJEMeq85MSeo.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-3-238-230-120.compute-1.amazonaws.com' (ED25519) to the list of known hosts.  
          _###_      Amazon Linux 2023  
     ~  \###/_\###/  
     ~  \###/  
     ~  #/  
     ~  V-'.-'>  
     ~  /'  
     ~  /'  
     /'  
     /m'  
[ec2-user@ip-172-31-64-114 ~]$ |
```

Step 2: Update Packages and Install Python

update

```
sudo yum update
```

```
~~      \###|  
~~      '#/ __  
~~      V~' '-->  
~~~      /  
~~_.  _/ /  
_/_/ /  
/_m/  
[ec2-user@ip-172-31-64-114 ~]$ sudo yum update  
Amazon Linux 2023 Kernel Livepatch repository  
Dependencies resolved.  
Nothing to do.  
Complete!  
[ec2-user@ip-172-31-64-114 ~]$
```

install python

```
sudo yum install python3 -y
```

```
[ec2-user@ip-172-31-64-114 ~]$ sudo yum install python3 -y
Last metadata expiration check: 0:01:05 ago on Tue Dec 2 09:44:45 2025.
Package python3-3.9.24-1.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-64-114 ~]$ |
```

install pip

```
sudo yum install python3-pip
```

```
[ec2-user@ip-172-31-64-114 ~]$ sudo yum install python3-pip
Last metadata expiration check: 0:02:03 ago on Tue Dec 2 09:44:45 2025.
Dependencies resolved.
=====
 Package           Architecture   Version      Repository    Size
=====
Installing:
 python3-pip        noarch       21.3.1-2.amzn2023.0.14    amazonlinux   1.8 M
Installing weak dependencies:
 libCRYPT-compat    x86_64       4.4.33-7.amzn2023      amazonlinux   92 k
=====
Transaction Summary
=====
Install 2 Packages

Total download size: 1.9 M
Installed size: 11 M
Is this ok [y/N]:
```

Step 3: Upload/Clone Your Application

1. Install git sudo yum install git -y

```
[ec2-user@ip-172-31-64-114:~]
Verifying : python3-pip-21.3.1-2.amzn2023.0.14.noarch
2/2

Installed:
  libCRYPT-compat-4.4.33-7.amzn2023.x86_64          python3-pip-21.3.1-2.amzn2023.0.14.noarch

Complete!
[ec2-user@ip-172-31-64-114 ~]$ sudo yum install git
Last metadata expiration check: 0:03:25 ago on Tue Dec 2 09:44:45 2025.
Dependencies resolved.
=====
 Package           Architecture   Version      Repository    Size
=====
Installing:
 git              x86_64       2.50.1-1.amzn2023.0.1    amazonlinux   53 k
Installing dependencies:
 git-core          x86_64       2.50.1-1.amzn2023.0.1    amazonlinux   4.9 M
 git-core-doc      noarch       2.50.1-1.amzn2023.0.1    amazonlinux   2.8 M
 perl-Error        noarch       1:0.17029-5.amzn2023.0.2  amazonlinux   41 k
 perl-File-Find    noarch       1.37-477.amzn2023.0.7   amazonlinux   25 k
 perl-Git          noarch       2.50.1-1.amzn2023.0.1    amazonlinux   41 k
 perl-TermReadKey x86_64       2.38-9.amzn2023.0.2     amazonlinux   36 k
 perl-lib          x86_64       0.65-477.amzn2023.0.7   amazonlinux   15 k
=====
Transaction Summary
=====
Install 8 Packages

Total download size: 7.9 M
Installed size: 41 M
Is this ok [y/N]:
```

2. Clone the Application and go inside the project folder (pythonapp)

clone git application

git clone

```
[ec2-user@ip-172-31-64-114 ~]$ git clone https://github.com/iamtruptimane/pythonapp.git
Cloning into 'pythonapp'...
remote: Enumerating objects: 71, done.
remote: Counting objects: 100% (71/71), done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 71 (delta 32), reused 30 (delta 10), pack-reused 0 (from 0)
Receiving objects: 100% (71/71), 14.44 KiB | 7.22 MiB/s, done.
Resolving deltas: 100% (32/32), done.
[ec2-user@ip-172-31-64-114 ~]$ ls
pythonapp
[ec2-user@ip-172-31-64-114 ~]$ cd pythonapp
[ec2-user@ip-172-31-64-114 pythonapp]$ ls
Dockerfile README.md app.py jenkinsfile requirements.txt test
[ec2-user@ip-172-31-64-114 pythonapp]$ |
```



Go inside the project folder

cd pythonapp

```
[ec2-user@ip-172-31-64-114 ~]$ cd pythonapp
[ec2-user@ip-172-31-64-114 pythonapp]$ sudo pip install -r requirements.txt
```

Step 4: Create a virtual environment and run activate file.

create virtual environment

sudo python3 -m venv myenv

```
ec2-user@ip-172-31-64-114 ~]$ sudo python3 -m venv myenv
ec2-user@ip-172-31-64-114 ~]$ sudo source myenv/bin/activate
udo: source: command not found
ec2-user@ip-172-31-64-114 ~]$ sudo beach myenv/bin/activate
udo: beach: command not found
ec2-user@ip-172-31-64-114 ~]$ sudo bash| myenv/bin/activate
```



activate file

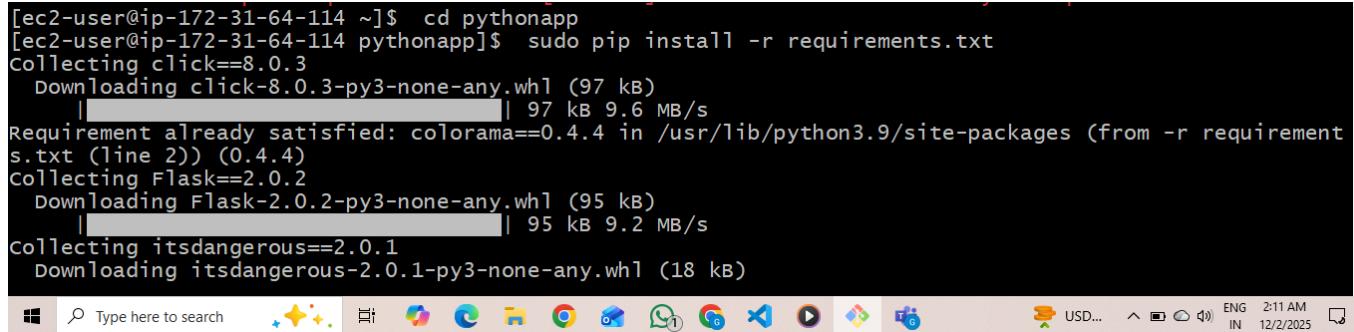
sudo source myenv/bin/activate

```
ec2-user@ip-172-31-64-114 ~]$ sudo python3 -m venv myenv
ec2-user@ip-172-31-64-114 ~]$ sudo source myenv/bin/activate
udo: source: command not found
ec2-user@ip-172-31-64-114 ~]$ sudo beach myenv/bin/activate
udo: beach: command not found
ec2-user@ip-172-31-64-114 ~]$ sudo bash| myenv/bin/activate
```



Step 5: Install Dependency.

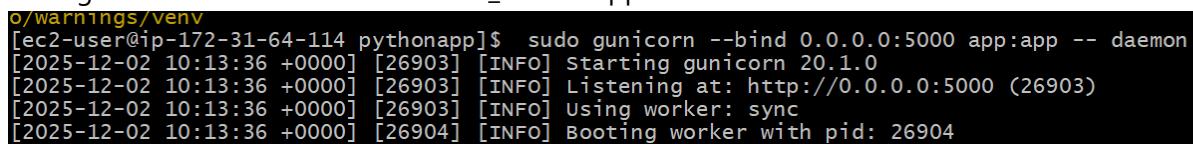
sudo pip install -r requirement.txt



```
[ec2-user@ip-172-31-64-114 ~]$ cd pythonapp
[ec2-user@ip-172-31-64-114 pythonapp]$ sudo pip install -r requirements.txt
Collecting click==8.0.3
  Downloading click-8.0.3-py3-none-any.whl (97 kB)
    |████████| 97 kB 9.6 MB/s
Requirement already satisfied: colorama==0.4.4 in /usr/lib/python3.9/site-packages (from -r requirements.txt (line 2)) (0.4.4)
Collecting Flask==2.0.2
  Downloading Flask-2.0.2-py3-none-any.whl (95 kB)
    |████████| 95 kB 9.2 MB/s
Collecting itsdangerous==2.0.1
  Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
```

Step 6: Keep the App Running on background.

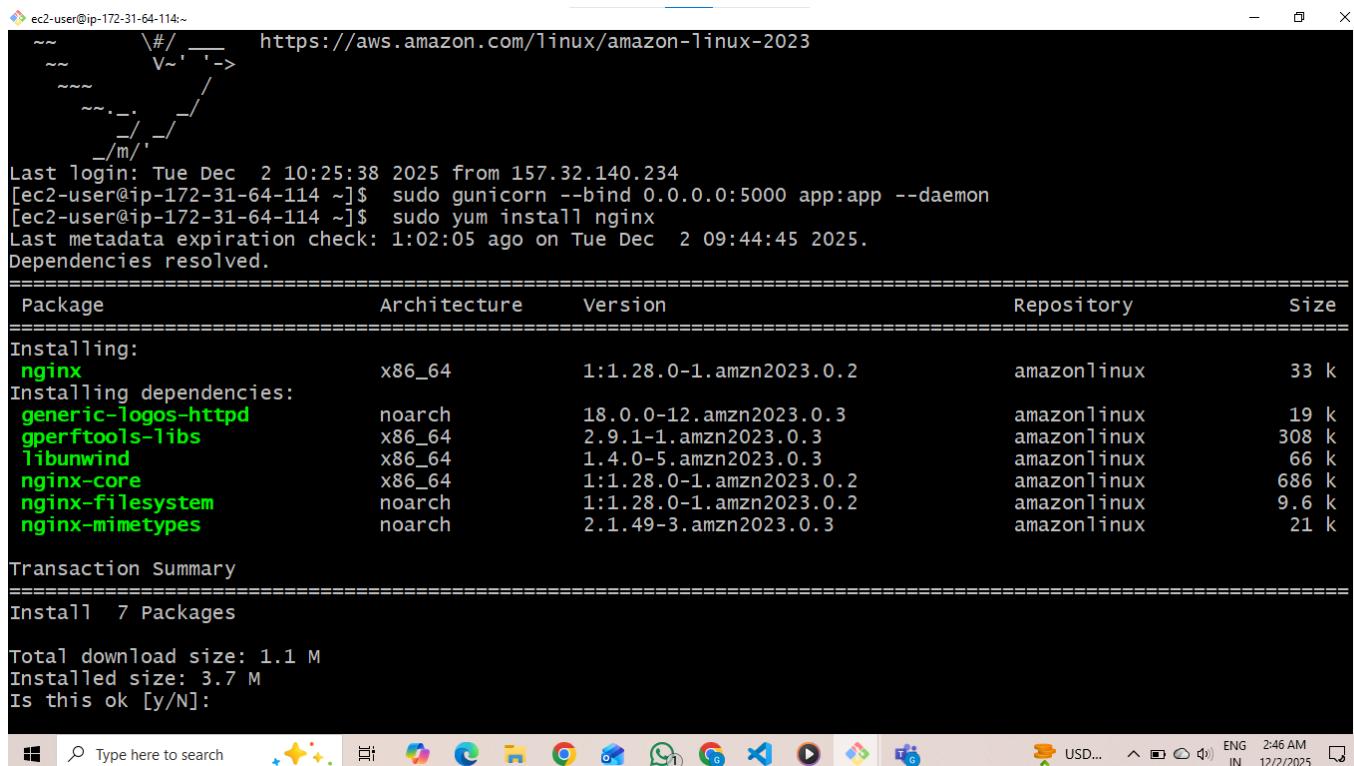
sudo gunicorn --bind 0.0.0.0:5000 <file_name>:app -- daemon



```
o/warnings/venv
[ec2-user@ip-172-31-64-114 pythonapp]$ sudo gunicorn --bind 0.0.0.0:5000 app:app -- daemon
[2025-12-02 10:13:36 +0000] [26903] [INFO] Starting gunicorn 20.1.0
[2025-12-02 10:13:36 +0000] [26903] [INFO] Listening at: http://0.0.0.0:5000 (26903)
[2025-12-02 10:13:36 +0000] [26903] [INFO] Using worker: sync
[2025-12-02 10:13:36 +0000] [26904] [INFO] Booting worker with pid: 26904
```

Step 7: Create Proxy Server

1. Configuring Nginx as Proxy. sudo yum install nginx



```
ec2-user@ip-172-31-64-114:~$ curl https://aws.amazon.com/linux/amazon-linux-2023
Last login: Tue Dec  2 10:25:38 2025 from 157.32.140.234
[ec2-user@ip-172-31-64-114 ~]$ sudo gunicorn --bind 0.0.0.0:5000 app:app --daemon
[ec2-user@ip-172-31-64-114 ~]$ sudo yum install nginx
Last metadata expiration check: 1:02:05 ago on Tue Dec  2 09:44:45 2025.
Dependencies resolved.
=====
 Package           Architecture Version       Repository      Size
=====
Installing:
 nginx            x86_64      1:1.28.0-1.amzn2023.0.2   amazonlinux   33 k
Installing dependencies:
 generic-logos-httd noarch      18.0.0-12.amzn2023.0.3   amazonlinux   19 k
 gperftools-libs  x86_64      2.9.1-1.amzn2023.0.3   amazonlinux   308 k
 libunwind         x86_64      1.4.0-5.amzn2023.0.3   amazonlinux   66 k
 nginx-core        x86_64      1:1.28.0-1.amzn2023.0.2   amazonlinux   686 k
 nginx-filesystem noarch      1:1.28.0-1.amzn2023.0.2   amazonlinux   9.6 k
 nginx-mimetypes   noarch      2.1.49-3.amzn2023.0.3   amazonlinux   21 k
=====
Transaction Summary
=====
Install 7 Packages

Total download size: 1.1 M
Installed size: 3.7 M
Is this ok [y/N]:
```

2. Start, enable and check status of nginx sudo systemctl start nginx sudo systemctl enable nginx sudo systemctl status nginx

```
Complete!
[ec2-user@ip-172-31-64-114 ~]$ sudo systemctl start nginx
[ec2-user@ip-172-31-64-114 ~]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-172-31-64-114 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Active: active (running) since Tue 2025-12-02 10:47:38 UTC; 20s ago
     Main PID: 28658 (nginx)
        Tasks: 3 (limit: 1053)
       Memory: 3.2M
          CPU: 56ms
        CGroup: /system.slice/nginx.service
                  └─28658 "nginx: master process /usr/sbin/nginx"
                     ├─28659 "nginx: worker process"
                     ├─28660 "nginx: worker process"

Dec 02 10:47:38 ip-172-31-64-114.ec2.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse p>
Dec 02 10:47:38 ip-172-31-64-114.ec2.internal nginx[28656]: nginx: the configuration file /etc/nginx/nginx.conf>
Dec 02 10:47:38 ip-172-31-64-114.ec2.internal nginx[28656]: nginx: configuration file /etc/nginx/nginx.conf tes>
Dec 02 10:47:38 ip-172-31-64-114.ec2.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse pr>
Lines 1-16/16 (END)
```



3. Create a server block for your app

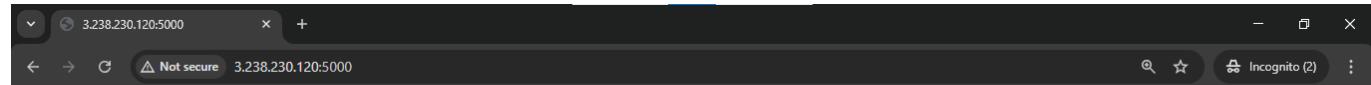
1. open nginx.conf sudo vim nginx.conf

2. Edit and add server block

```
location = /50x.html {  
}  
location /{  
    proxy-pass http://localhost:5000;  
}
```

Step 8: Testing the Deployment

Copy the public IP and paste it in any browser.



successfully deployed python application through jenkins!!!!!!!, added webhook

