Deploy and Serve Your Python App with Nginx

Introduction

This project offers a complete walkthrough for deploying a Python application in a production-ready environment using a proxy server. It covers everything from preparing both the development and server environments, installing essential packages, configuring the application, and setting up a proxy server like Nginx to handle and route incoming traffic effectively. Following this guide will help enhance the application's performance, scalability, and overall security.

Prerequisites

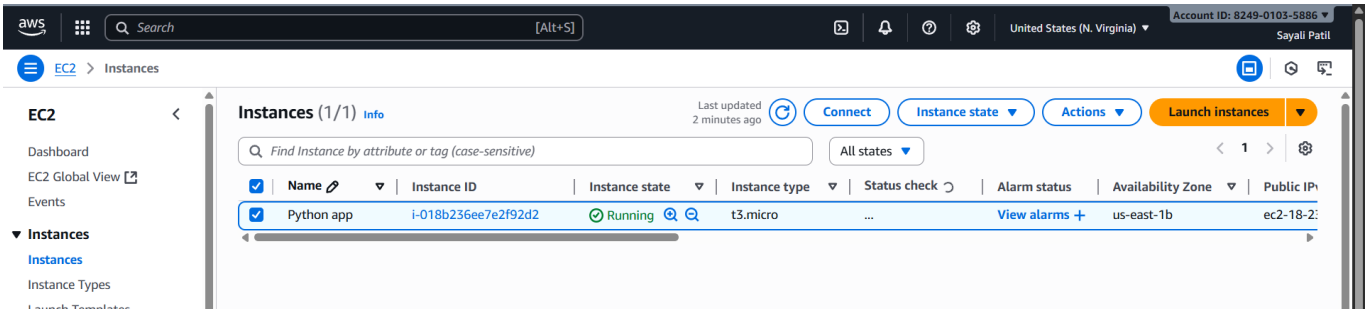
Before you deploy this Python application, make sure the following tools and services are properly installed and set up:

1. Python 3.x – The app is built using Python version 3 or later.
2. Pip – Required to install and manage Python packages.
3. Git (optional) – Useful for cloning the project repository from a version control system.
4. Virtual Environment (venv) – Recommended to keep project dependencies isolated and manageable.
5. Proxy Server – Either Nginx or Apache is needed to route and manage incoming traffic to the application.

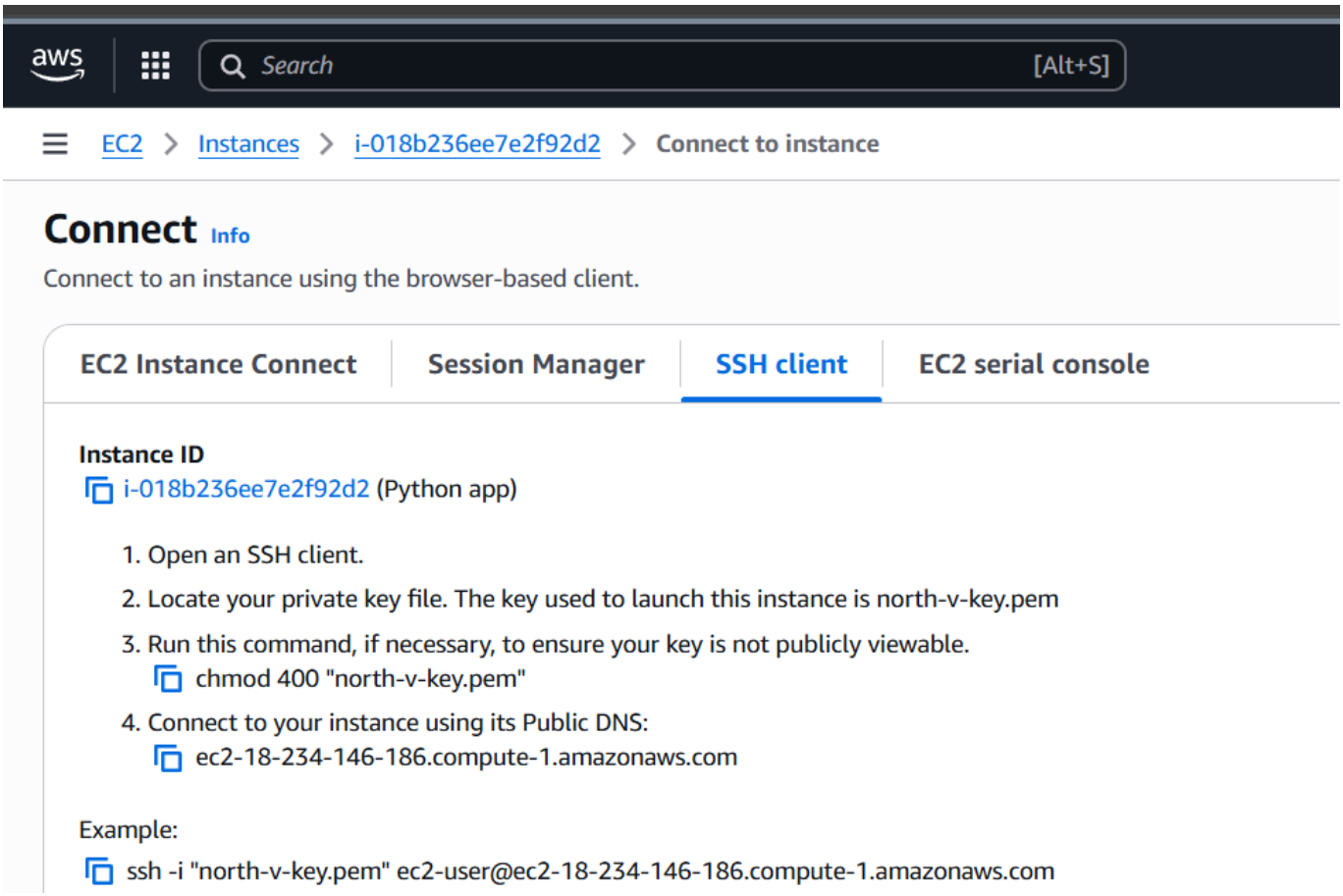
Deployment Steps

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

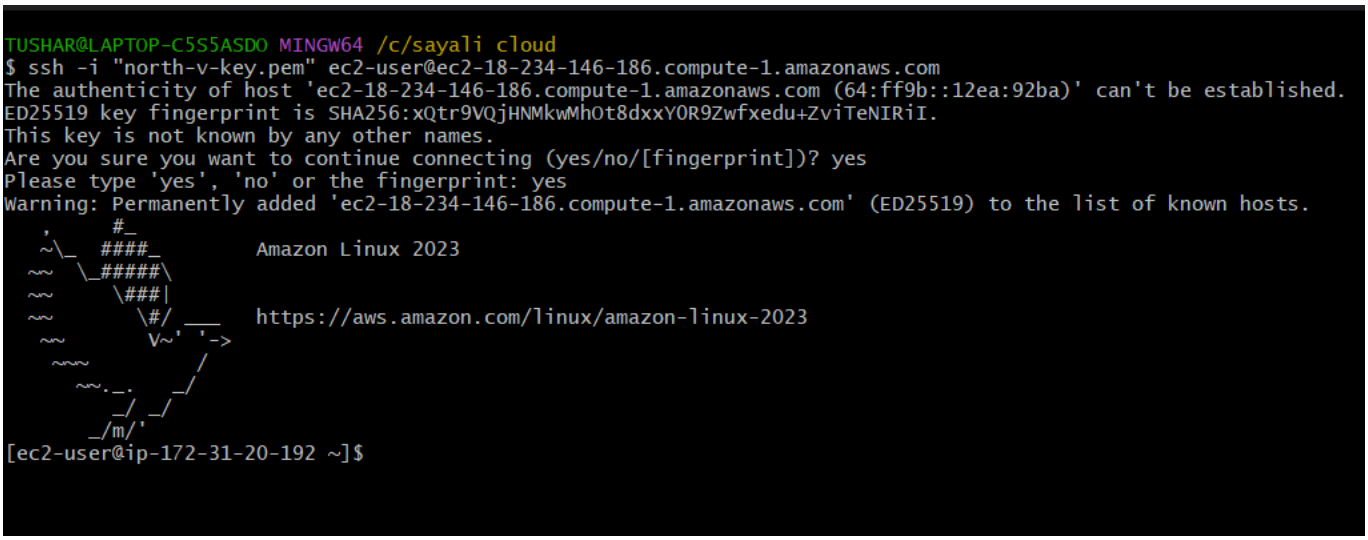
1. Launch instance



2. Copy ssh command



3. Paste ssh command in Git bash terminal



Step 2: Upgrade System Packages and Set Up Python

```
1. update
#sudo yum update

2. install python
#sudo yum install python3 -y

3. install pip
#sudo yum install python3-pip
```

```
[ec2-user@ip-172-31-20-192 ~]$ sudo yum update
Amazon Linux 2023 Kernel Livepatch repository                                    201 kB/s |  23 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-20-192 ~]$ sudo yum install python3 -y
Last metadata expiration check: 0:00:17 ago on Sat Sep 20 17:05:57 2025.
Package python3-3.9.23-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-20-192 ~]$ sudo yum installpython3-pip
No such command: installpython3-pip. Please use /usr/bin/yum --help
It could be a YUM plugin command, try: "yum install 'dnf-command(installpython3-pip)'"
[ec2-user@ip-172-31-20-192 ~]$ sudo yum install python3-pip
Last metadata expiration check: 0:01:05 ago on Sat Sep 20 17:05:57 2025.
Dependencies resolved.
================================================================================================================================
 Package                       Architecture          Version                                 Repository                   Size
================================================================================================================================
Installing:
 python3-pip                   noarch                21.3.1-2.amzn2023.0.13                  amazonlinux                 1.8 M
Installing weak dependencies:
 libxcrypt-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
```

Step 3: Upload/Clone Your Application

```
Install a git

#sudo yum install git -y
```

```
[ec2-user@ip-172-31-20-192 ~]$ sudo yum install git -y
Last metadata expiration check: 0:02:11 ago on Sat Sep 20 17:05:57 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
```

2. Clone the Application and change into the pythonapp folder.

```
clone git application
#git clone <git url>

Go inside the projrct folder
#cd pythonapp
```

```
[ec2-user@ip-172-31-20-192 ~]$ git clone https://github.com/iamtruptimane/pythonapp.git
Cloning into 'pythonapp'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 68 (delta 30), reused 29 (delta 11), pack-reused 0 (from 0)
Receiving objects: 100% (68/68), 14.18 KiB | 7.09 MiB/s, done.
Resolving deltas: 100% (30/30), done.
[ec2-user@ip-172-31-20-192 ~]$ ls
pythonapp
[ec2-user@ip-172-31-20-192 ~]$ cd pythonapp
[ec2-user@ip-172-31-20-192 pythonapp]$ ls
Dockerfile  README.md  app.py  jenkinsfile  requirements.txt  test
[ec2-user@ip-172-31-20-192 pythonapp]$
```

Step 4: Set up a virtual environment and activate it.

```
create virtual environment
#sudo python3 -m venv myenv

activate file
#sudo source myenv/bin/activate
```

```
[ec2-user@ip-172-31-20-192 pythonapp]$ sudo python3 -m venv myenv
[ec2-user@ip-172-31-20-192 pythonapp]$ sudo bash myenv/bin/activate
[ec2-user@ip-172-31-20-192 pythonapp]$
```

Step 5: Install Dependencies.

```
sudo pip install -r requirement.txt
```

```
[ec2-user@ip-172-31-20-192 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 8.3 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 7.1 MB/s
Collecting itsdangerous==2.0.1
  Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
Collecting Jinja2==3.0.3
  Downloading Jinja2-3.0.3-py3-none-any.whl (133 kB)
     |████████████████████████████████| 133 kB 74.7 MB/s
Collecting MarkupSafe==2.0.1
  Downloading MarkupSafe-2.0.1-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl
 (30 kB)
Collecting Werkzeug==2.0.2
  Downloading Werkzeug-2.0.2-py3-none-any.whl (288 kB)
     |████████████████████████████████| 288 kB 71.1 MB/s
Collecting gunicorn==20.1.0
  Downloading gunicorn-20.1.0-py3-none-any.whl (79 kB)
     |████████████████████████████████| 79 kB 14.3 MB/s
Requirement already satisfied: setuptools>=3.0 in /usr/lib/python3.9/site-packages (from gunicorn==20.1.0->-r requirements.txt
 (line 8)) (59.6.0)
```

Step 6: Run the application in the background.

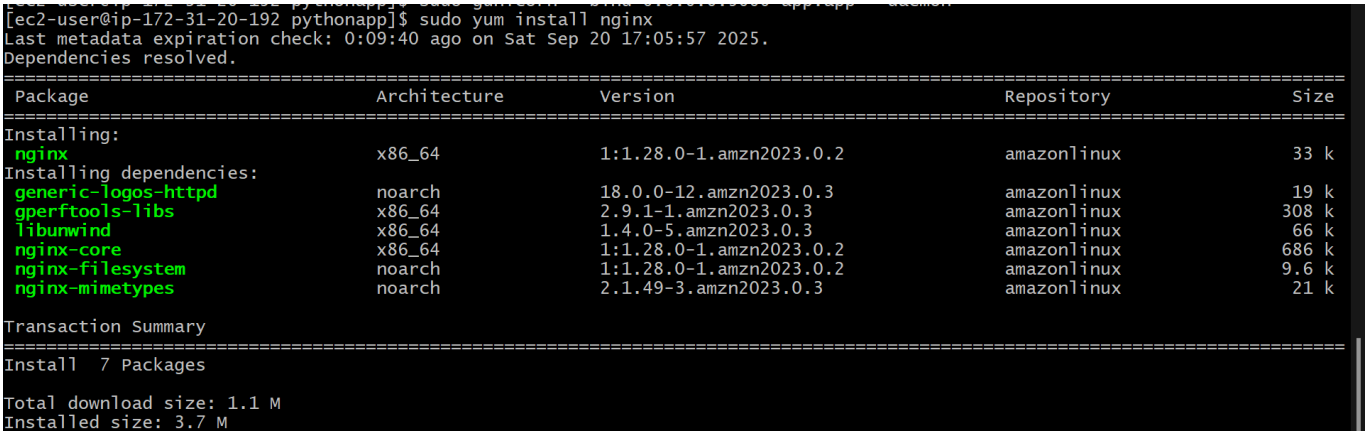
```
sudo gunicorn --bind 0.0.0.0:5000 <file_name>:app -- daemon
```

```
[ec2-user@ip-172-31-20-192 pythonapp]$ sudo gunicorn --bind 0.0.0.0:5000 app:app --daemon
[ec2-user@ip-172-31-20-192 pythonapp]$
```

Step 7: Build the Proxy Server

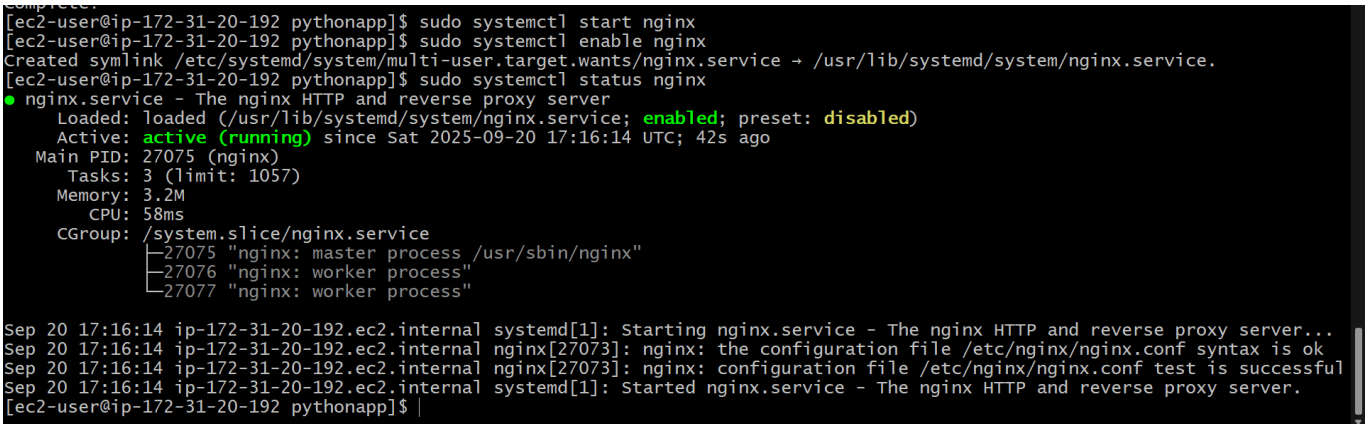
1. Setting up Nginx as a Proxy Server

```
sudo yum install nginx
```



2. Start, enable and check status of nginx

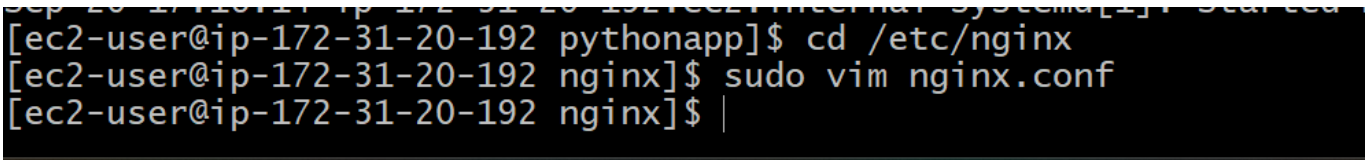
```
sudo systemctl start nginx
sudo systemctl enable nginx
sudo systemctl status nginx
```



3. Set up a server block for your application

    1. open nginx.conf

```
sudo vim nginx.conf
```



2. Edit and add server block as show below

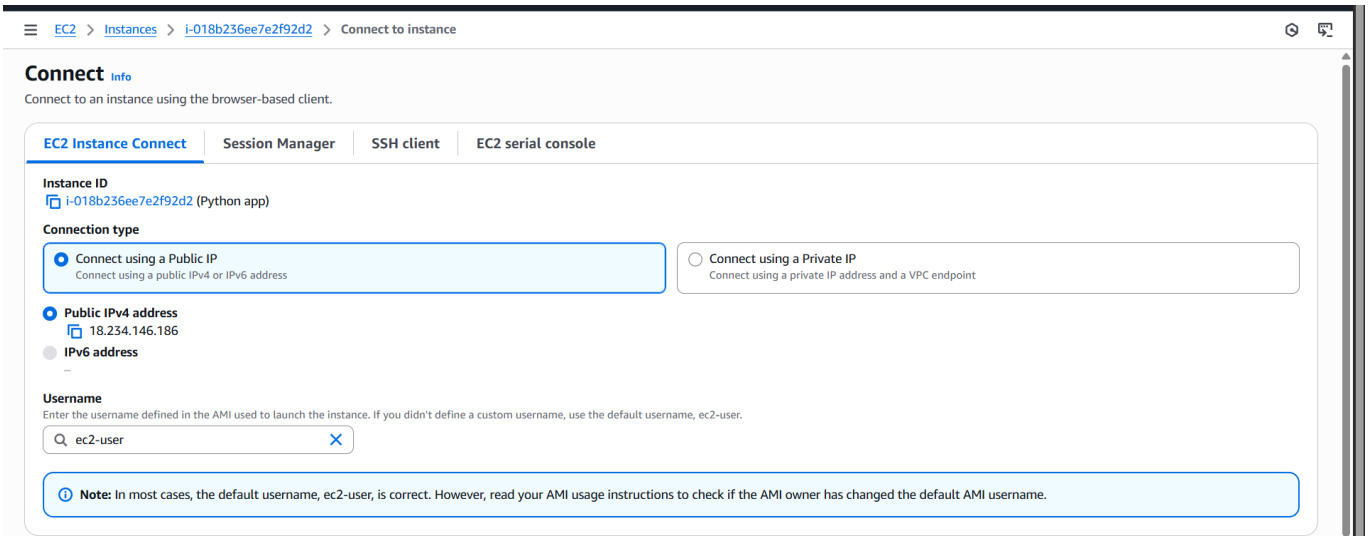
```
        error_page 404 /404.html;
        location = /404.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
        location /{
                proxy-pass http://localhost:5000;
        }

    }
# Settings for a TLS enabled server.
```

Step 8: Deployment Testing

1. Copy Public IP



2. Final Output

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

Summary

This project is a step-by-step guide to help you deploy a Python app using a proxy server. It shows you how to set up your environment, install what you need, configure the app, and use a proxy server like Nginx to handle requests smoothly.