How to Deploy Your First Python App on AWS EC2 instance

Introduction

This project shows how to a python application on aws EC2 instance. I wrote it in very simple steps so anyone can follow easily.

Note: The python application running on port 5000.

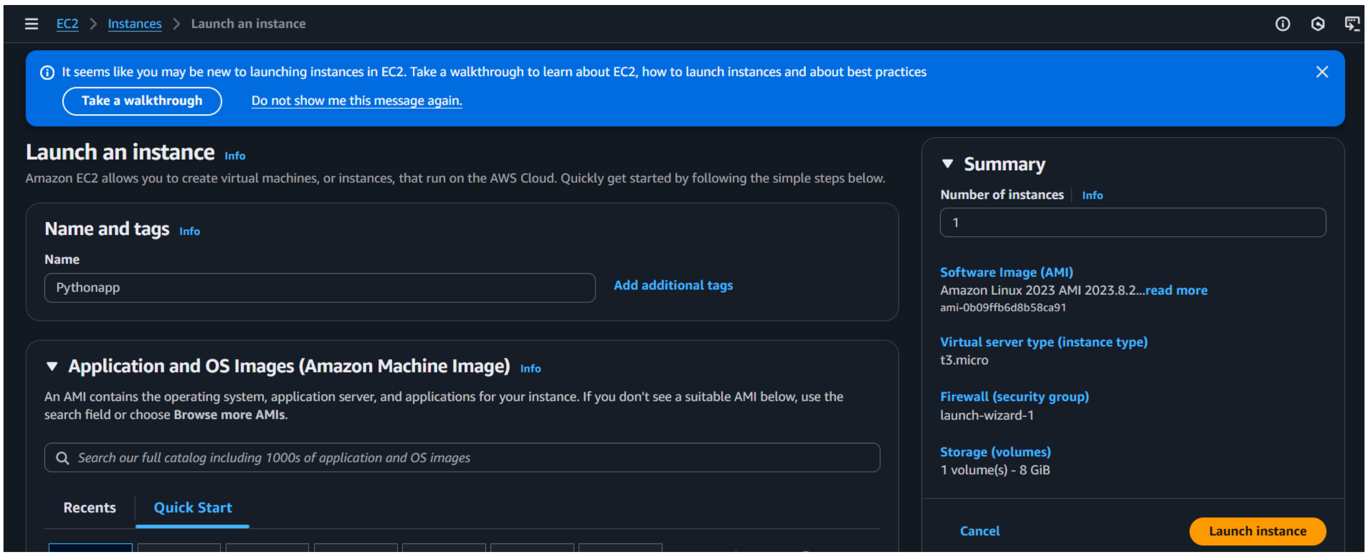
Requirements

1. AWS account with an EC2 instance
2. SSH key pair to connect to the instance
3. A Python application with a requirements.txt file.
4. Your application's code repository link.

Deployment Steps:

Step 1: Launch an EC2 Instance:

Launch a new EC2 instance and name it pythonapp.


EC2 > Instances > Launch an instance

It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: Pythonapp

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images

Summary

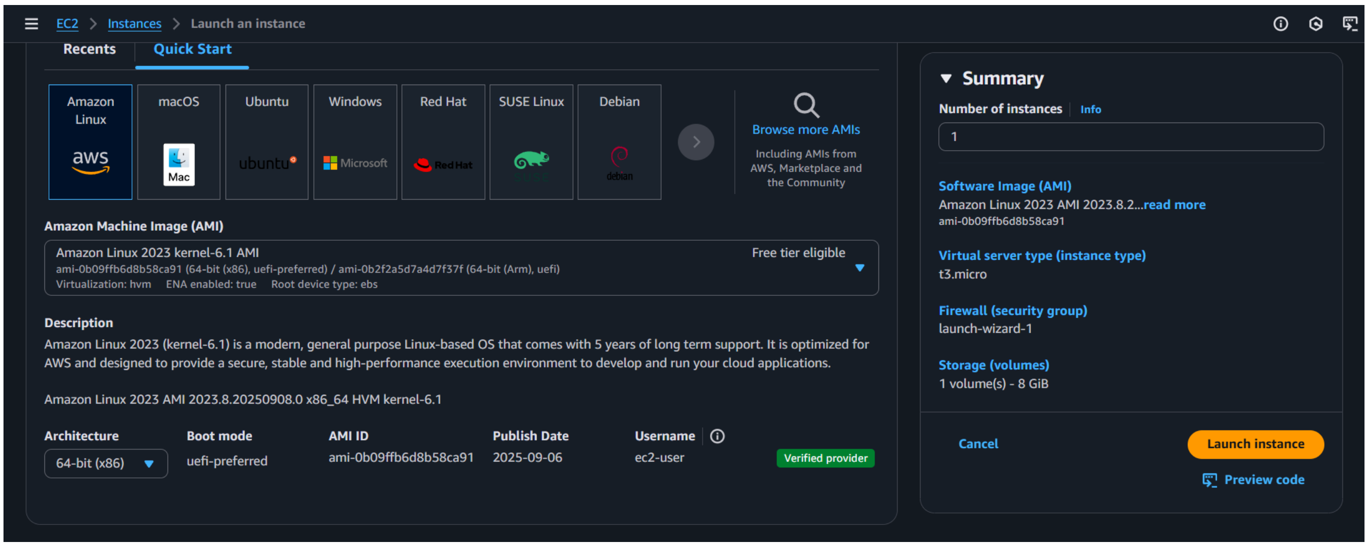
Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.8.2...read more
ami-0b09ffb6d8b58ca91

Virtual server type (instance type): t3.micro

Firewall (security group): launch-wizard-1

Storage (volumes): 1 volume(s) - 8 GiB



Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI — Free tier eligible
ami-0b09ffb6d8b58ca91 (64-bit (x86), uefi-preferred) / ami-0b2f2a5d7a4d7f37f (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.8.20250908.0 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username	
64-bit (x86)	uefi-preferred	ami-0b09ffb6d8b58ca91	2025-09-06	ec2-user	Verified provider

Summary

Number of instances Info: 1

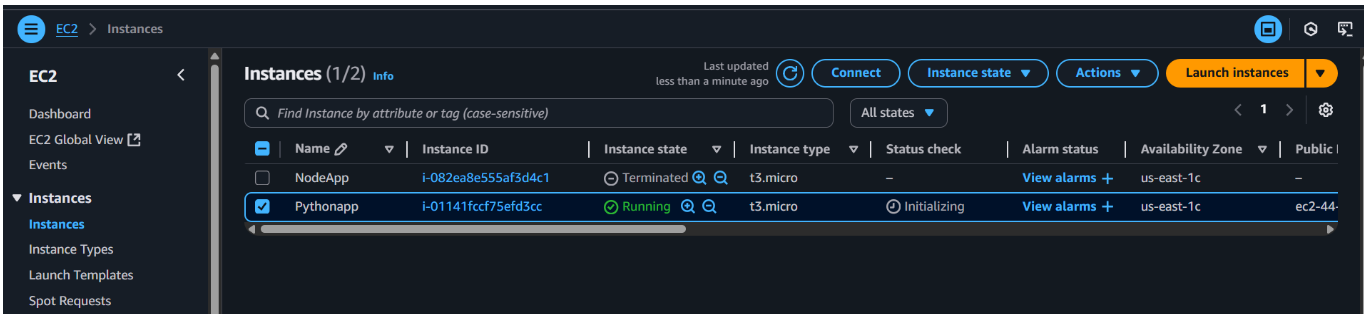
Software Image (AMI): Amazon Linux 2023 AMI 2023.8.2...read more
ami-0b09ffb6d8b58ca91

Virtual server type (instance type): t3.micro

Firewall (security group): launch-wizard-1

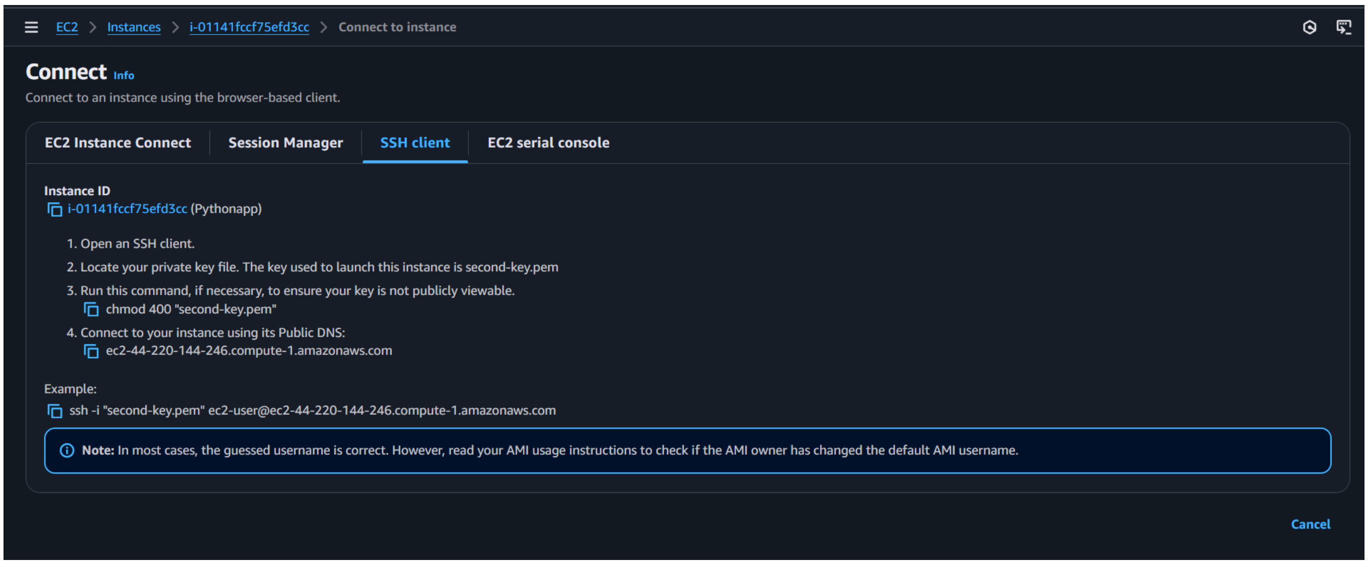
Storage (volumes): 1 volume(s) - 8 GiB

Step 2: Connecting to EC2 instance



Instances (1/2) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
☐	NodeApp	i-082ea8e555af3d4c1	Terminated	t3.micro	–	View alarms +	us-east-1c	–
☑	Pythonapp	i-01141fccf75efd3cc	Running	t3.micro	Initializing	View alarms +	us-east-1c	ec2-44-



Connect Info

Connect to an instance using the browser-based client.

Instance ID

i-01141fccf75efd3cc (Pythonapp)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is second-key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   `chmod 400 "second-key.pem"`
4. Connect to your instance using its Public DNS:
   `ec2-44-220-144-246.compute-1.amazonaws.com`

Example:

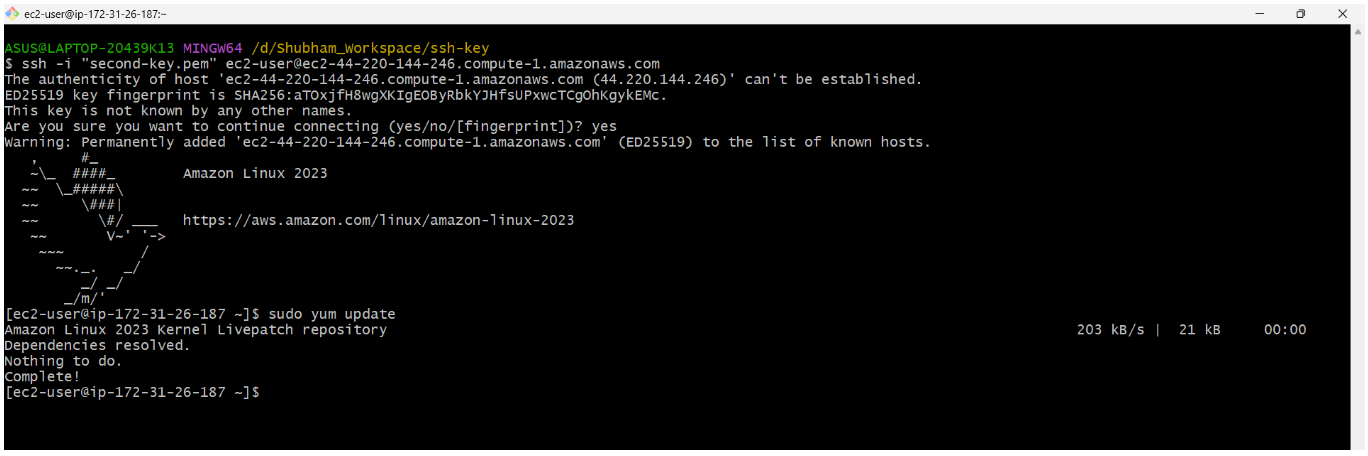
```
ssh -i "second-key.pem" ec2-user@ec2-44-220-144-246.compute-1.amazonaws.com
```

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.

Step 3: Updating Packages

Run this command on terminal:

sudo yum update


ec2-user@ip-172-31-26-187:~

```
ASUS@LAPTOP-20439K13 MINGW64 /d/Shubham_Workspace/ssh-key
$ ssh -i "second-key.pem" ec2-user@ec2-44-220-144-246.compute-1.amazonaws.com
The authenticity of host 'ec2-44-220-144-246.compute-1.amazonaws.com (44.220.144.246)' can't be established.
ED25519 key fingerprint is SHA256:aTOxjfH8wgXKIgEOByRbkYJHfsUPxwcTCgOhKgykEMc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-220-144-246.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-26-187 ~]$ sudo yum update
Amazon Linux 2023 Kernel Livepatch repository                                           203 kB/s |  21 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-26-187 ~]$
```

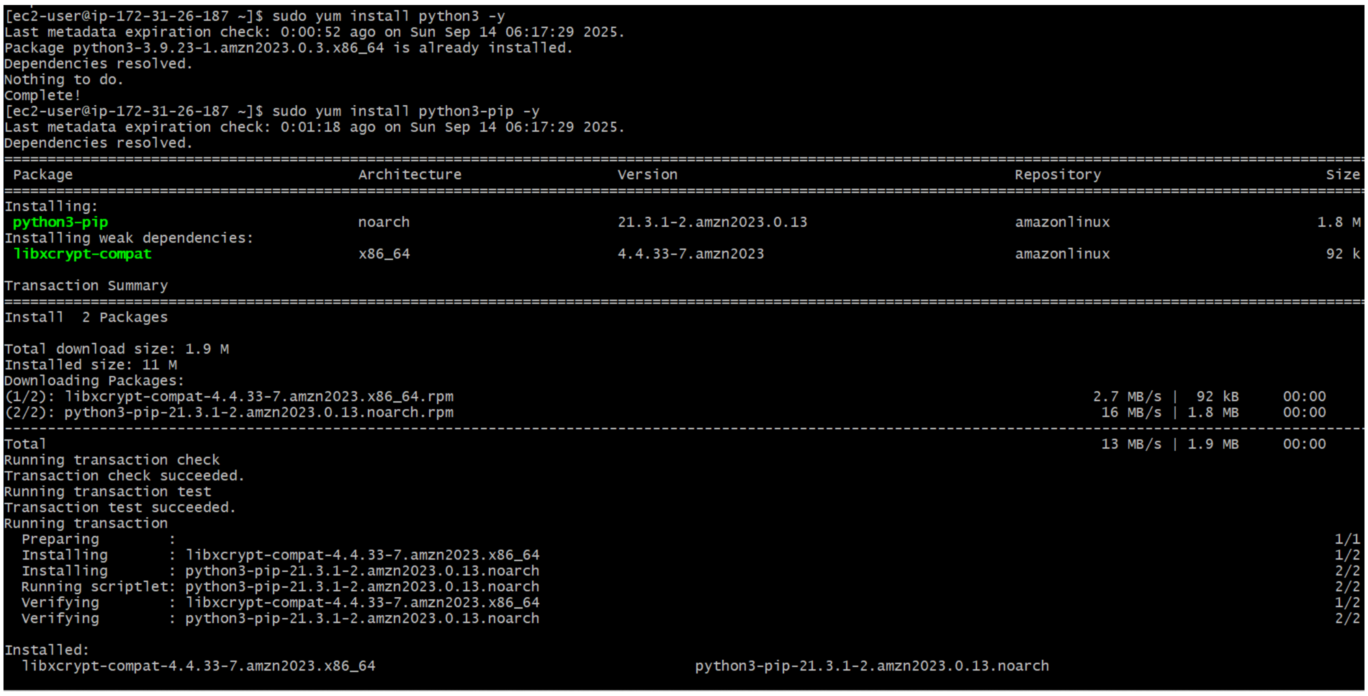
Step 4: Install Python and Git:

Install Python 3, pip, and git on the instance.

sudo yum install python3 -y

sudo yum install python3-pip -y

sudo yum install git -y

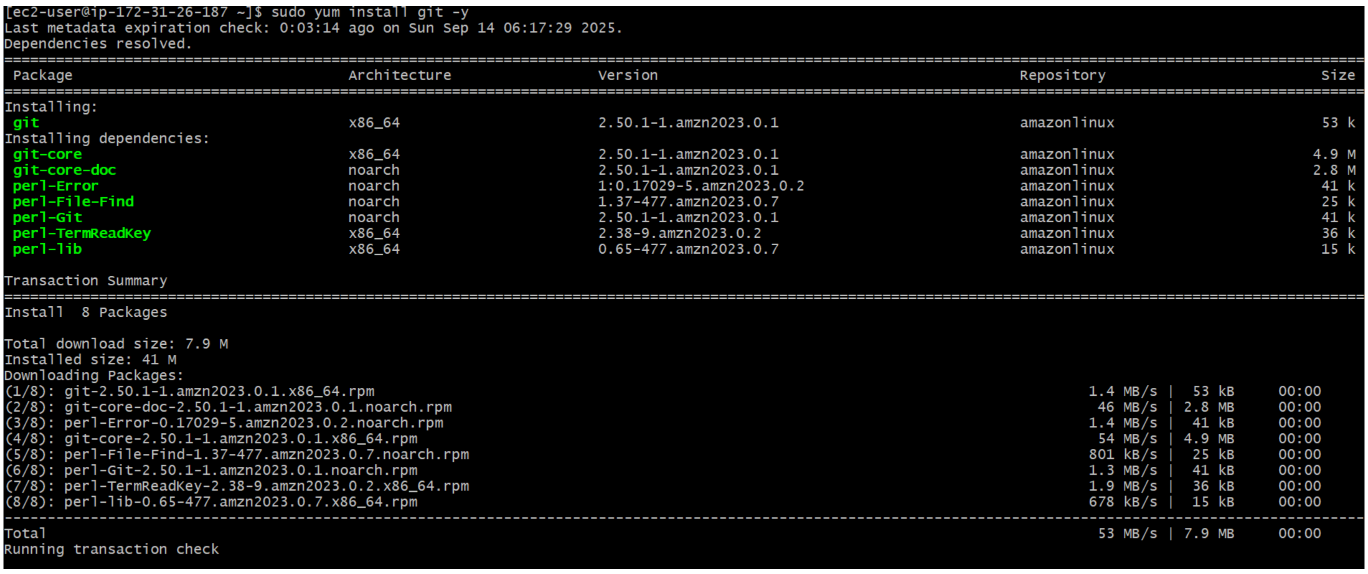


```
[ec2-user@ip-172-31-26-187 ~]$ sudo yum install python3 -y
Last metadata expiration check: 0:00:52 ago on Sun Sep 14 06:17:29 2025.
Package python3-3.9.23-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-26-187 ~]$ sudo yum install python3-pip -y
Last metadata expiration check: 0:01:18 ago on Sun Sep 14 06:17:29 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
Downloading Packages:
(1/2): libxcrypt-compat-4.4.33-7.amzn2023.x86_64.rpm                                    2.7 MB/s |  92 kB     00:00
(2/2): python3-pip-21.3.1-2.amzn2023.0.13.noarch.rpm                                     16 MB/s | 1.8 MB     00:00
--------------------------------------------------------------------------------------------------------------------------
Total                                                                                    13 MB/s | 1.9 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                1/1
  Installing       : libxcrypt-compat-4.4.33-7.amzn2023.x86_64                                                      1/2
  Installing       : python3-pip-21.3.1-2.amzn2023.0.13.noarch                                                      2/2
  Running scriptlet: python3-pip-21.3.1-2.amzn2023.0.13.noarch                                                      2/2
  Verifying        : libxcrypt-compat-4.4.33-7.amzn2023.x86_64                                                      1/2
  Verifying        : python3-pip-21.3.1-2.amzn2023.0.13.noarch                                                      2/2

Installed:
  libxcrypt-compat-4.4.33-7.amzn2023.x86_64                          python3-pip-21.3.1-2.amzn2023.0.13.noarch
```

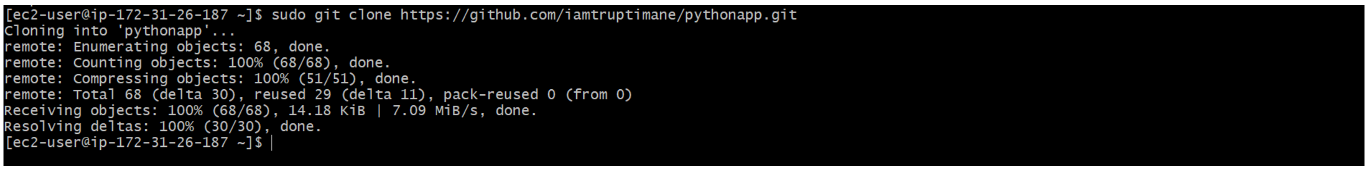
Step 5: Clone the Application Repository:

Navigate to the desired directory and clone your application's repository.

sudo git clone "your-repository-link"



```
[ec2-user@ip-172-31-26-187 ~]$ sudo yum install git -y
Last metadata expiration check: 0:03:14 ago on Sun Sep 14 06:17:29 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
Downloading Packages:
(1/8): git-2.50.1-1.amzn2023.0.1.x86_64.rpm                                                       1.4 MB/s |  53 kB     00:00
(2/8): git-core-doc-2.50.1-1.amzn2023.0.1.noarch.rpm                                               46 MB/s | 2.8 MB     00:00
(3/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm                                               1.4 MB/s |  41 kB     00:00
(4/8): git-core-2.50.1-1.amzn2023.0.1.x86_64.rpm                                                   54 MB/s | 4.9 MB     00:00
(5/8): perl-File-Find-1.37-477.amzn2023.0.7.noarch.rpm                                            801 kB/s |  25 kB     00:00
(6/8): perl-Git-2.50.1-1.amzn2023.0.1.noarch.rpm                                                  1.3 MB/s |  41 kB     00:00
(7/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm                                            1.9 MB/s |  36 kB     00:00
(8/8): perl-lib-0.65-477.amzn2023.0.7.x86_64.rpm                                                  678 kB/s |  15 kB     00:00
------------------------------------------------------------------------------------------------------------------------------------------
Total                                                                                              53 MB/s | 7.9 MB     00:00
Running transaction check
```

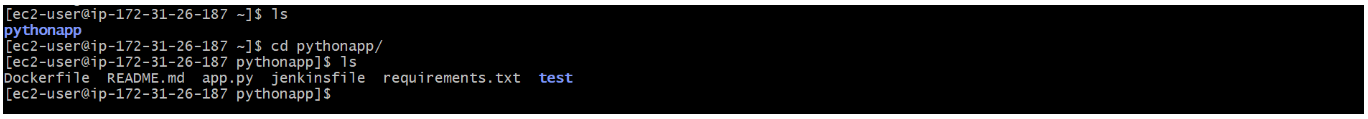


```
[ec2-user@ip-172-31-26-187 ~]$ sudo git clone https://github.com/iamtruptimane/pythonapp.git
Cloning into 'pythonapp'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 68 (delta 30), reused 29 (delta 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-172-31-26-187 ~]$ |
```

Step 6: Navigate to the Application Directory:

The cloning process will create a directory for your application. Change into that directory.

cd "your-application-directory"



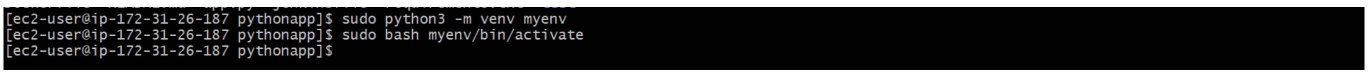
```
[ec2-user@ip-172-31-26-187 ~]$ ls
pythonapp
[ec2-user@ip-172-31-26-187 ~]$ cd pythonapp/
[ec2-user@ip-172-31-26-187 pythonapp]$ ls
Dockerfile  README.md  app.py  jenkinsfile  requirements.txt  test
[ec2-user@ip-172-31-26-187 pythonapp]$
```

Step 7: Create and Activate a Virtual Environment:

It is a best practice to install dependencies in a virtual environment.

sudo python3 -m venv myenv

source myenv/bin/activate

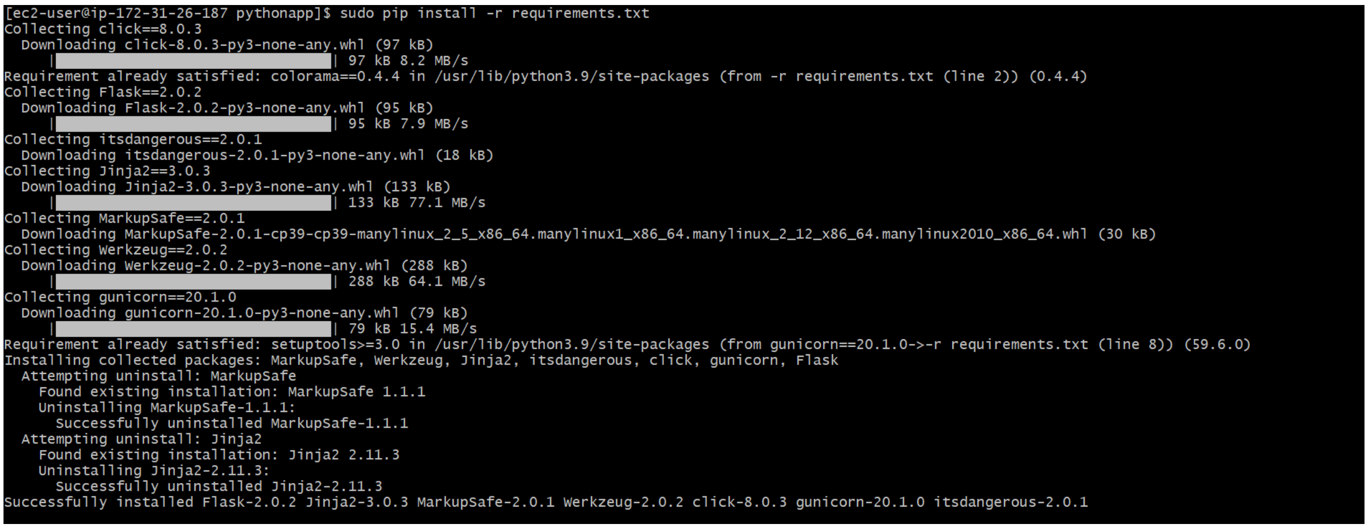


```
[ec2-user@ip-172-31-26-187 pythonapp]$ sudo python3 -m venv myenv
[ec2-user@ip-172-31-26-187 pythonapp]$ sudo bash myenv/bin/activate
[ec2-user@ip-172-31-26-187 pythonapp]$
```

Step 8: Install Application Dependencies:

Install all the required packages listed in your requirements.txt file.

sudo pip install -r requirements.txt

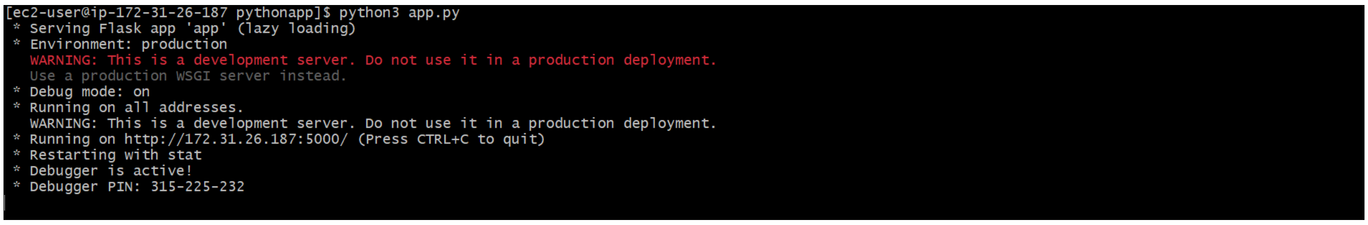


```
[ec2-user@ip-172-31-26-187 pythonapp]$ sudo pip install -r requirements.txt
Collecting click==8.0.3
  Downloading click-8.0.3-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 8.2 MB/s
Requirement already satisfied: colorama==0.4.4 in /usr/lib/python3.9/site-packages (from -r 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 7.9 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 77.1 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 64.1 MB/s
Collecting gunicorn==20.1.0
  Downloading gunicorn-20.1.0-py3-none-any.whl (79 kB)
     |████████████████████████████████| 79 kB 15.4 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)
Installing collected packages: MarkupSafe, Werkzeug, Jinja2, itsdangerous, click, gunicorn, Flask
  Attempting uninstall: MarkupSafe
    Found existing installation: MarkupSafe 1.1.1
    Uninstalling MarkupSafe-1.1.1:
      Successfully uninstalled MarkupSafe-1.1.1
  Attempting uninstall: Jinja2
    Found existing installation: Jinja2 2.11.3
    Uninstalling Jinja2-2.11.3:
      Successfully uninstalled Jinja2-2.11.3
Successfully installed Flask-2.0.2 Jinja2-3.0.3 MarkupSafe-2.0.1 Werkzeug-2.0.2 click-8.0.3 gunicorn-20.1.0 itsdangerous-2.0.1
```

Step 9: Run the Application:

Start your Python application.

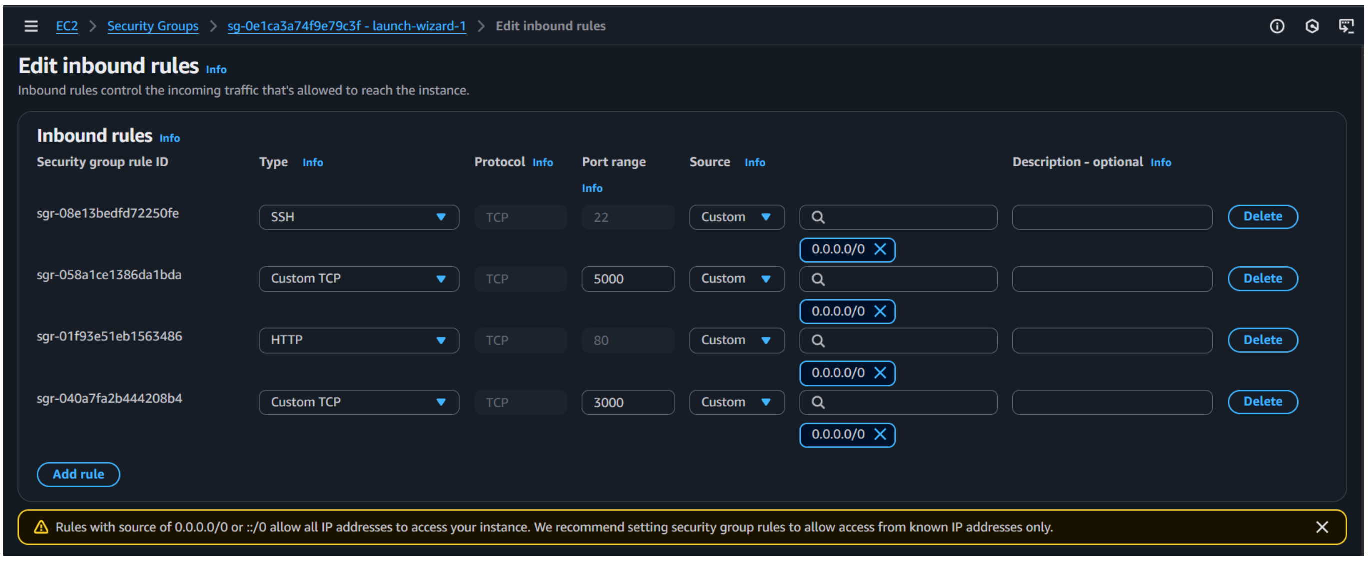
python3 app.py



```
[ec2-user@ip-172-31-26-187 pythonapp]$ python3 app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.26.187:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 315-225-232
```

Step 10: Configure Security Group:

Go to the AWS Security Group settings for your EC2 instance and add a rule to enable incoming traffic on port 5000 from Anywhere-IPv4.



EC2 > Security Groups > sg-0e1ca3a74f9e79c3f - launch-wizard-1 > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

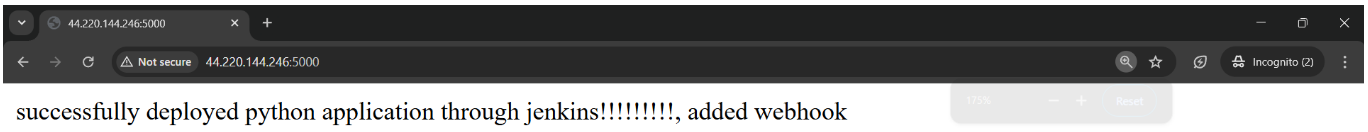
Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-08e13bedfd72250fe	SSH	TCP	22	Custom	0.0.0.0/0	
sgr-058a1ce1386da1bda	Custom TCP	TCP	5000	Custom	0.0.0.0/0	
sgr-01f93e51eb1563486	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-040a7fa2b444208b4	Custom TCP	TCP	3000	Custom	0.0.0.0/0	

Add rule

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Step 11: Test the Application:

Open a web browser and hit your EC2 instance's public IP address with the port number appended (e.g., http://"your-public-ip":5000). You should see the application's output.



44.220.144.246:5000

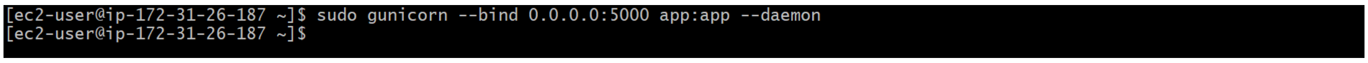
Not secure 44.220.144.246:5000

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

Step 12: If you want to run your application in the background, then

Run this command on terminal:

sudo gunicorn --bind 0.0.0.0:5000 app:app --daemon



```
[ec2-user@ip-172-31-26-187 ~]$ sudo gunicorn --bind 0.0.0.0:5000 app:app --daemon
[ec2-user@ip-172-31-26-187 ~]$
```

Step 13: Now if you want to add proxy server to your application which provide more security to your application, then

Run this command on terminal:

Install Nginx:

sudo yum install nginx -y

Configure Nginx:

cd /etc/nginx/

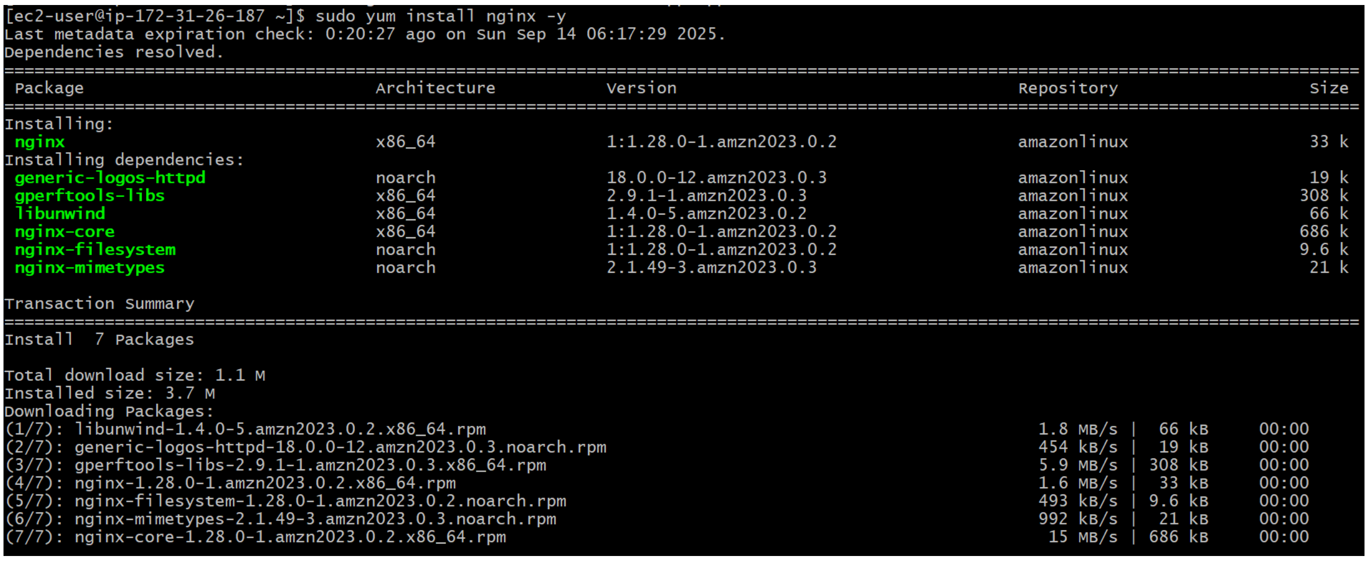
sudo vim nginx.conf

Add a proxy_pass to the http Block:

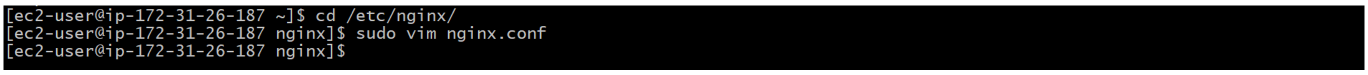
location / {

```
proxy_pass http://localhost:5000;
```

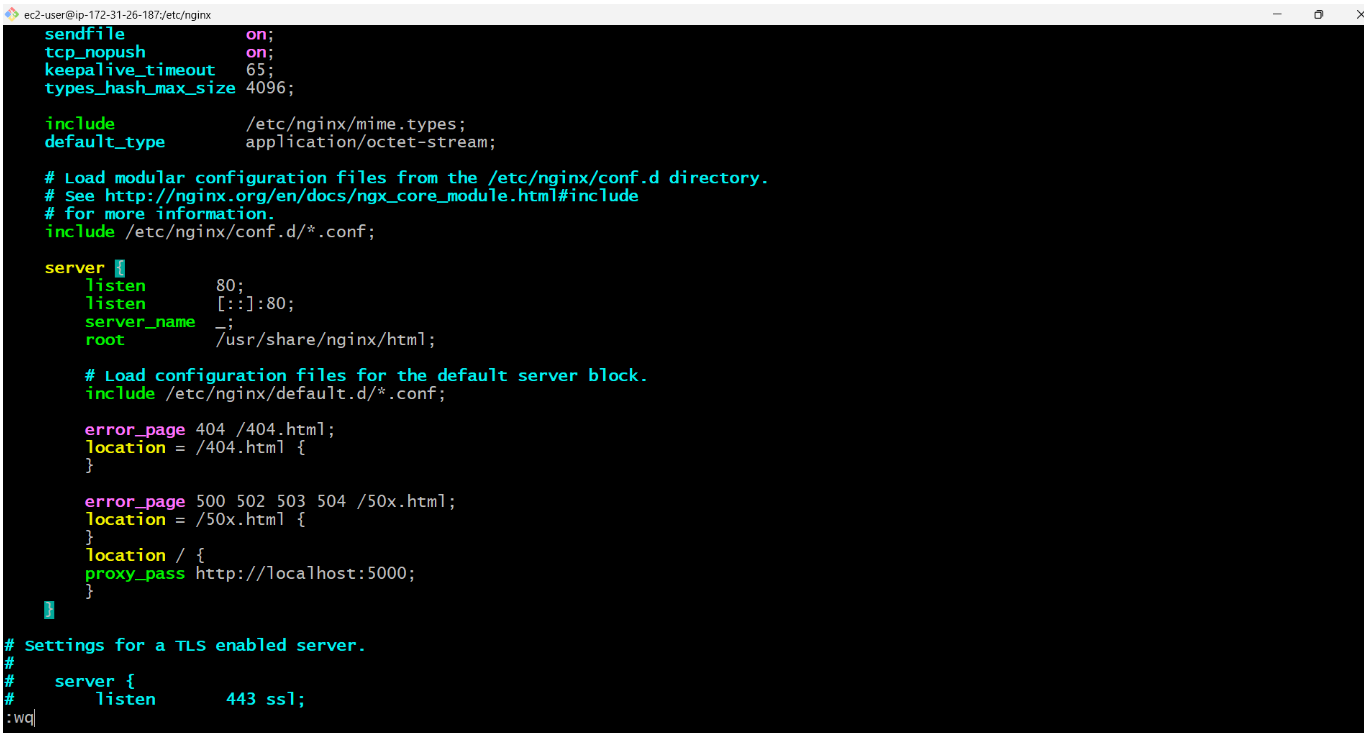
}



```
[ec2-user@ip-172-31-26-187 ~]$ sudo yum install nginx -y
Last metadata expiration check: 0:20:27 ago on Sun Sep 14 06:17:29 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-httpd              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.amzn2023.0.2                       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
Downloading Packages:
(1/7): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rpm                                               1.8 MB/s |  66 kB     00:00
(2/7): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm                                   454 kB/s |  19 kB     00:00
(3/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm                                         5.9 MB/s | 308 kB     00:00
(4/7): nginx-1.28.0-1.amzn2023.0.2.x86_64.rpm                                                  1.6 MB/s |  33 kB     00:00
(5/7): nginx-filesystem-1.28.0-1.amzn2023.0.2.noarch.rpm                                       493 kB/s | 9.6 kB     00:00
(6/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm                                        992 kB/s |  21 kB     00:00
(7/7): nginx-core-1.28.0-1.amzn2023.0.2.x86_64.rpm                                              15 MB/s | 686 kB     00:00
```



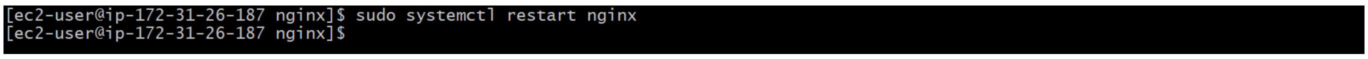
```
[ec2-user@ip-172-31-26-187 ~]$ cd /etc/nginx/
[ec2-user@ip-172-31-26-187 nginx]$ sudo vim nginx.conf
[ec2-user@ip-172-31-26-187 nginx]$
```



```
    sendfile            on;
    tcp_nopush          on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        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.
#
#    server {
#        listen       443 ssl;
:wq
```

Step 14: Restart your system

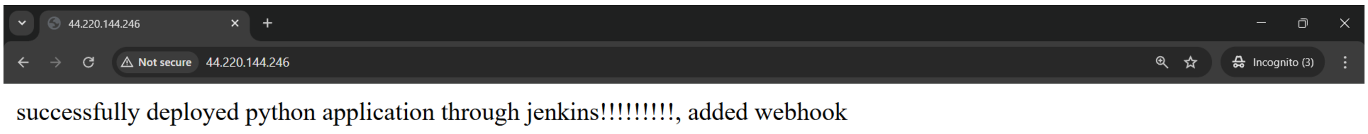
sudo systemctl restart nginx



```
[ec2-user@ip-172-31-26-187 nginx]$ sudo systemctl restart nginx
[ec2-user@ip-172-31-26-187 nginx]$
```

Step 15: Output

The final output will display here.



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