

## Python flask

First we have to create security groups for python flask.

ssh-----22-----my ip----->this is for admin purpose.

tcp----- 5000(default)-----anywhere(0.0.0.0/0)---->This is for enduser purpose.

First we have to login to the console—>click on ec2 dashboard—> click on create instance—>after that we can pass name of an instance, ami and storage.

The screenshot shows the 'Launch an instance' page in the AWS Management Console. The breadcrumb trail at the top is 'EC2 > Instances > Launch an instance'. The main heading is 'Launch an instance' with an 'Info' link. Below this, a sub-heading 'Name and tags' has an 'Info' link. A text input field for 'Name' contains 'e.g. My Web Server', and there is a link 'Add additional tags'. The 'Application and OS Images (Amazon Machine Image)' section has an 'Info' link and a search bar with the placeholder 'Search our full catalog including 1000s of application and OS images'. Below the search bar, there are tabs for 'Recents' and 'Quick Start'. The 'Quick Start' tab is active, showing a grid of AMI categories: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and S. Each category has a representative logo. To the right of the grid is a 'Browse more AMIs' link. On the right side of the console, the 'Summary' panel is visible. It includes a 'Number of instances' input field set to '1'. Below this, it lists the 'Software Image (AMI)' as 'Amazon Linux 2 Kernel 5.10 AMI...read more' with the ID 'ami-09d3b3274b6c5d4aa'. The 'Virtual server type (instance type)' is 't2.micro'. The 'Firewall (security group)' is 'New security group'. The 'Storage (volumes)' section shows '1 volume(s) - 8 GiB'. A 'Free tier' notification box is also present, stating: 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.' At the bottom of the summary panel are 'Cancel' and 'Launch instance' buttons.

—>After that click on create instance—> the instance is created.

Instances (1/3) info

Find instance by attribute or tag (case-sensitive)

running X Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring
stage-apache	i-09f49ca0c451d886e	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-3-93-43-127.com...	3.93.43.127	-	-	disabled
stage-jenkins	i-0c718c132a4d4d16	Running	c5.2xlarge	2/2 checks passed	No alarms	us-east-1a	ec2-54-198-192-179....	54.198.192.179	-	-	disabled
flask	i-0234cc016258ccb99	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-44-211-200-244....	44.211.200.244	-	-	disabled

Instance: i-0234cc016258ccb99 (flask)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary info

Instance ID  
i-0234cc016258ccb99 (flask)

IPv6 address  
-

Hostname type  
IP name: ip-172-31-85-25.ec2.internal

Answer private resource DNS name  
IPv4 (A)  
Auto-assigned IP address

Public IPv4 address  
44.211.200.244 | open address

Instance state  
Running

Private IP DNS name (IPv4 only)  
ip-172-31-85-25.ec2.internal

Instance type  
t2.micro

VPC ID

Private IPv4 addresses  
172.31.85.25

Public IPv4 DNS  
ec2-44-211-200-244.compute-1.amazonaws.com | open address

Elastic IP addresses  
-

AWS Compute Optimizer finding  
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Then we connect to the server, by using below command  
`ssh -i xxx.pem ec2-user@public ip`

```
jayachandra@LAPTOP-TJGI2TUH MINGW64 ~/downloads (master)
$ ssh -i chandu.pem ec2-user@44.211.200.244
Last login: Thu Oct 27 12:01:28 2022 from 103.110.170.8

 _ | _ | _ )
 _ | ( /   Amazon Linux 2 AMI
 _ | \ _ | _ |

https://aws.amazon.com/amazon-linux-2/
13 package(s) needed for security, out of 16 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-85-25 ~]$
```

Then we connect to the root user using the below command.  
`sudo su -`

Then we can install python, by using below command  
`yum install python3 -y`

```
[root@ip-172-31-85-25 ~]# yum install python3 -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Package python3-3.7.10-1.amzn2.0.1.x86_64 already installed and latest version
Nothing to do
[root@ip-172-31-85-25 ~]#
```

After that install python flask, using below command  
`pip3 install flask`

```

root@ip-172-31-85-25 ~]# pip3 install flask
WARNING: Running pip install with root privileges is generally not a good idea. Try 'pip3 install --user' instead.
Collecting Flask
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
    | 101 kB 13.5 MB/s
Collecting importlib-metadata<=3.6.0; python_version < "3.10"
  Downloading importlib_metadata-5.0.0-py3-none-any.whl (21 kB)
Collecting click>=8.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    | 96 kB 10.3 MB/s
Collecting itsdangerous<=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2<=3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
    | 133 kB 41.9 MB/s
Collecting Werkzeug<=2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
    | 232 kB 41.1 MB/s
Collecting typing-extensions>=3.6.4; python_version < "3.8"
  Downloading typing_extensions-4.4.0-py3-none-any.whl (26 kB)
Collecting zipp>=0.5
  Downloading zipp-3.10.0-py3-none-any.whl (6.2 kB)
Collecting MarkupSafe<=2.0
  Downloading MarkupSafe-2.1.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Installing collected packages: typing-extensions, zipp, importlib-metadata, click, itsdangerous, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 flask-2.2.2 importlib-metadata-5.0.0 itsdangerous-2.1.2 typing-extensions-4.4.0 zipp-3.10.0
root@ip-172-31-85-25 ~]#

```

Then we can open the app.py file,

[vi app.py](#)

Here we can pass the code—>then save it.

```

from flask import Flask

app = Flask(__name__)
@app.route("/")
def homepage():
    return "<h1>hello this is python flask<h1>"
if __name__ == "__main__":
    app.run(host='0.0.0.0',port=8080)

```

```

from flask import Flask

app = Flask(__name__)
@app.route("/")
def homepage():
    return "<h1>hello this is python flask<h1>"
if __name__ == "__main__":
    app.run(host='0.0.0.0',port=5000)

```

—>to see the content in command line by using below command

[cat app.py](#)

```

C[root@ip-172-31-91-27 ~]# cat app.py
from flask import Flask

app = Flask(__name__)
app.route("/")
def homepage():
    return "<h1>hello this is python flask<h1>"
if __name__ == "__main__":
    app.run(host='0.0.0.0',port=5000)
root@ip-172-31-91-27 ~]#

```

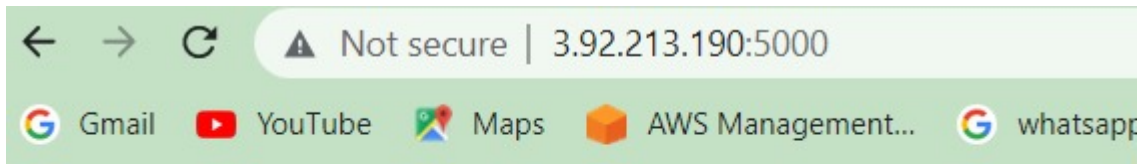
—After that to run the flask,we are using below command

[python3 app.py](#)

```
app.run(host='0.0.0.0', port=5000)
[root@ip-172-31-85-25 ~]# python3 app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.31.85.25:5000
Press CTRL+C to quit
```

—>if we are entering ctrl+c , it means we are came out of the server—>we are unable to access the page.so we are not enter ctrl+c, to access the page.

-----> to access the page by using [public ip:5000](http://3.92.213.190:5000), shown below.



# hello this is python flask