FLASK Web Development

**Usage instructions:**

1. Launch the product via 1-click. **Please wait until** the instance passes **all** status checks and is running. You can connect using your Amazon private key and '**ubuntu**' login via your SSH client.

   To update software, use: **sudo apt-get update**

2. This AMI has been preconfigured with Gunicorn & NGINX a reverse proxy.

3. Let's test the code. Change directories and edit the python script located here:

   **cd My-Flask-Application**

   **sudo nano app.py**

   - Change the **text** to something new…. For example….

```
  GNU nano 6.2
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World! Here is my test.'
```

4. **Save & Exit**

5. Restart Flask app.

   **sudo systemctl restart my-flask-app**

   - If you need to check status run

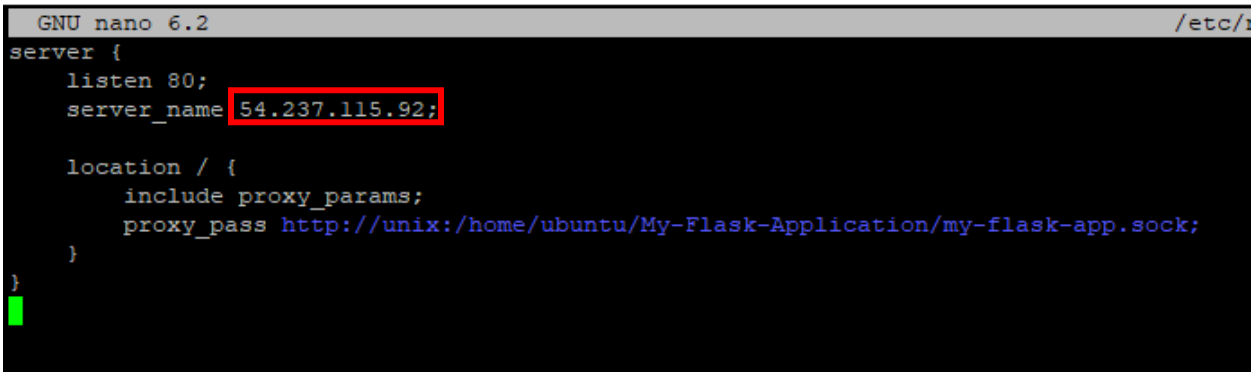
     **sudo systemctl status my-flask-app**

6. Next, configure the Nginx Poxy to your Instance Public IP Address.

**cd My-Flask-Application**

**sudo nano /etc/nginx/sites-available/my-flask-app**

- Change the server_name to your **Instance IP public address** or domain name.




- **Exit & Save**

7. Restart Nginx. Run:

**sudo systemctl restart nginx**

8. Be sure to change the permissions

**sudo chmod 755 /home/ubuntu**

9. In a browser go to your: http:// Public IPv4 address

You should see:

**"Hello, World! Here is my test" or the new edited script you added**

**To Run in a virtual Environment (optional)**

8. If you prefer to run it in a virtual environment, run:

**cd My-Flask-Application**

**export FLASK_APP=app**

**export FLASK_ENV=development**

**flask run --host=0.0.0.0**

```
ubuntu@ip-172-31-55-66:~/My-Flask-Application$ flask run --host=0.0.0.0
 * Serving Flask app 'app' (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.55.66:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 132-205-652
```

In a browser go to your: **http:// Public IPv4 address:5000**

- For ex: http://54.237.115.92**:5000**
- ***(Press CTRL+C to quit)***

Visit the flask website to start building: **https://flask.palletsprojects.com/en/1.1.x/quickstart/**

## AWS Data

- Data Encryption Configuration: This solution does not encrypt data within the running instance.
- User Credentials are stored: /root/.ssh/authorized_keys & /home/ubuntu/.ssh/authorized_keys
- Monitor the health:
    - Navigate to your Amazon EC2 console and verify that you're in the correct region.
    - Choose Instance and select your launched instance.
    - Select the server to display your metadata page and choose the Status checks tab at the bottom of the page to review if your status checks passed or failed.

## Extra Information: (Optional)

### Allocate Elastic IP

To ensure that your instance **keeps its IP during restarts** that might happen, configure an Elastic IP. From the EC2 console:

1. Select ELASTIC IPs.
2. Click on the ALLOCATE ELASTIC IP ADDRESS.
3. Select the default (Amazon pool of IPv4 addresses) and click on ALLOCATE.
4. From the ACTIONS pull down, select ASSOCIATE ELASTIC IP ADDRESS.
5. In the box that comes up, note down the Elastic IP Address, which will be needed when you configure your DNS.
6. In the search box under INSTANCE, click and find your INSTANCE ID and then click ASSOCIATE.
7. Your instance now has an elastic IP associated with it.
8. For additional help: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html

### Using Your Own Domain Name

1. You will need to configure your DNS entry for the new host server you created.
2. Change your domain's "Record Set" value to point to your new instance. Change and copy your "IPv4 Public IP" into the "A" type value.
3. For additional help: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/rrsets-working-with.html

**Deploy a Load Balancer**

1. https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/load-balancer-getting-started.html

**Deploy a SSL for a Domain Name  (Optional but Recommended)**

1. Installing Cerbot:

   **sudo apt install certbot python3-certbot-nginx**

2. Obtain an SSL certificate and automatically configure Nginx:

   **sudo certbot –nginx**

3. Follow the on-screen instructions. Certbot will modify your Nginx configuration automatically to serve your Flask app over HTTPS.