For this task I Used Python as the language and Flask as the frame work. Below are the steps I took:

STEP 1: Created a Flask App from my local PC to be pushed to Github, this App consisted of:

*hello.py (Where the application files live in)*

*hello.ini (WSGI Configuration file)*

*wsgi.py (WSGI Entry Point i.e Application Entry Point)*

*hello.test.py (Test file)*

*requirements.txt (Python packages are required to run the project)*

*README.md*

STEP 2: Created a repository on Github where the code will reside and then I Pushed my repo from my PC to the Github repository

STEP 3: Created an Ubuntu Virtual Machine in AWS opening port 8080 and port 5000. Port allows Jenkins installation and Port 5000 is my application port

STEP 4: Connected to my instance using SSH and then updated and upgraded the server repository also I installed python3-pip and some other set up tools for python using: *sudo apt install python3-pip python3-dev build-essential libssl-dev libffi-dev python3-setuptools*

STEP 5: I also installed python3-venv and I created a Python virtual environment using *python3.6 -m venv flaskprojenv* and I activated the virtual environment using *source flaskprojenv/bin/activate*

STEP 6: At this point I clone my project from the Github repo and ran *pip install -r requirements.txt uswgi.* I ran my application using *python hello.py* and opened up my browser to visit my Server IP Address with the port number I specified which was 5000

STEP 7: Using the WSGI entry point I created, I tested using the uWSGI to make sure it can serve my application. I ran *uwsgi --socket 0.0.0.0:5000 --protocol=http -w wsgi:app* and I also visited my server IP with the port number and it served the application. Then I deactivate the virtual environment using *deactivate*

STEP 8: Using my configuration file which is *hello.ini* and then I created a systemd Unit File at *etc/systemd/system/hello.service*  and this file contain some configurations. Now this file will allow Ubuntu’s init system to automatically start uWSGI and serve the flask application whenever the server boots. The file contained this config:

*[Unit]*

*Description=uWSGI instance to serve hello*

*After=network.target*

*[Service]*

*User=ubuntu*

*Group=www-data*

*WorkingDirectory=/home/ubuntu/flask\_app/vgg-flask-app*

*Environment="PATH=/home/ubuntu/flask\_app/flaskprojenv/bin"*

*ExecStart=/home/ubuntu/flask\_app/flaskprojenv/bin/uwsgi --ini hello.ini*

*[Install]*

*WantedBy=multi-user.target*

I then ran restarted the service configuration file and enabled it as a root user using: *sudo systemctl start hello.service && sudo systemctl enable hello.service* furthermore I checked the status of my service using *sudo systemctl status hello.service* and it outputted *(active) running* with a green button to show that the service is running.

STEP 8: It was time for me to configure Nginx to allow proxy request but before this I had to install Nginx using: *sudo apt install nginx*

*and I* also adjusted the firewall to allow access to my Nginx service , first I listed all the service in Nginx using *sudo ufw app list*

and then I enable Nginx using *sudo ufw allow ‘Nginx HTTP’* then to check that Nginx is activated I used: *sudo systemctl status nginx* visiting my server IP address I got the welcome page from Nginx.

For the Nginx proxy configuration I created a a new server block configuration file in *etc/nginx/sites-available/hello* that contained this configuration:

*server {*

*listen 5000;*

*listen [::]:5000;*

*real\_ip\_header X-Forwarded\_For;*

*set\_real\_ip\_from 127.0.0.1;*

*server\_name localhost;*

*location / {*

*include uwsgi\_params;*

*uwsgi\_pass unix:/home/ubuntu/flask\_app/vgg-flask-app/hello.sock;*

*}*

*}*

Then I went ahead to link this configuration file I created to the *sites-enabled* directory using *sudo ln -s /etc/nginx/sites-available/hello /etc/nginx/sites-enabled* then to test for syntax error in the configuration file I did *sudo nginx -t* I then restarted Nginx using: *sudo systemctl restart nginx*

With this basics step I had the application up and running on my server ip address

STEP 9: I installed Jenkins in this step, but since Jenkins is a Java application it need Java JDK to be installed and to install OpenJDK I used: *sudo apt install openjdk-8-jdk* after this was successfully I used *javac -version*  to check if it was installed. To install Jenkins I used this command:

*wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -*

*sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'*

*sudo apt-get update*

*sudo apt-get install jenkins*

This package installation will:

• Setup Jenkins as a daemon launched on start. See /etc/init.d/jenkins for more details.

• Create a ‘jenkins’ user to run this service.

• Direct console log output to the file /var/log/jenkins/jenkins.log. Check this file if you are

troubleshooting Jenkins.

• Populate /etc/default/jenkins with configuration parameters for the launch, e.g JENKINS\_HOME

• Set Jenkins to listen on port 8080. Access this port with your browser to start configuration.

On my browser I inputted the server ip address with the port number (*8080*) and the Jenkins page displayed with an instruction on how to get my password which was saved in *var/lib/jenkins/secrets/initialAdminPassword* all I did was to use *cat* on this path and copied and pasted the password and clicked on continue and on the next page I installed suggested plugins, created first admin user

STEP 9: I created a simple free style project that creates a file on */home/ubuntu/flask\_app/vgg-flask-app* called **helloJenkins** and at the Source code management I inputted my repository URL, I got an error at this point and after enough google searching I realized I had to allow Jenkins to overide root access, so I edited the *sudoers* file in *etc/sudoers* and on the line where I saw Jenkins I did this settings *jenkins ALL=(ALL) NOPASSWD: ALL* then I exited and restart Jenkins using *systemctl restart jenkins*  and after the next run of my job it was successful. I went ahead to install *Github integration plugin* by going to *Manage Jenkins > Manage Plugins*

STEP 10: To allow Github communicate with Jenkins I had to set up webhook and add webhook in my github repository. I set my payload URL to: <http://ip_address:8080/github-webhook/> and the content type to: *application/json* and for the event I checked *send me everything* then I went to the freestyle project I created and then checked *Github hook trigger for GITScm polling* under *Build Triggers*, then went into my server to stage my change and commit the change and then pushed this file to my master branch and when I check my Jenkins job console output I got a success and a webhook trigger.

***PS: All file edition was done using*** *nano* ***(A text editor that come pre-installed in Ubuntu)***

***All commands was ran as a root user i.e using*** *sudo*