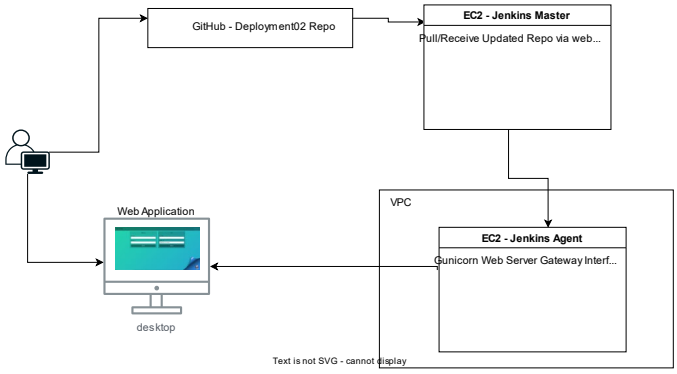
R. Da Costa

Kura Labs – Deployment 03 2022-10-15

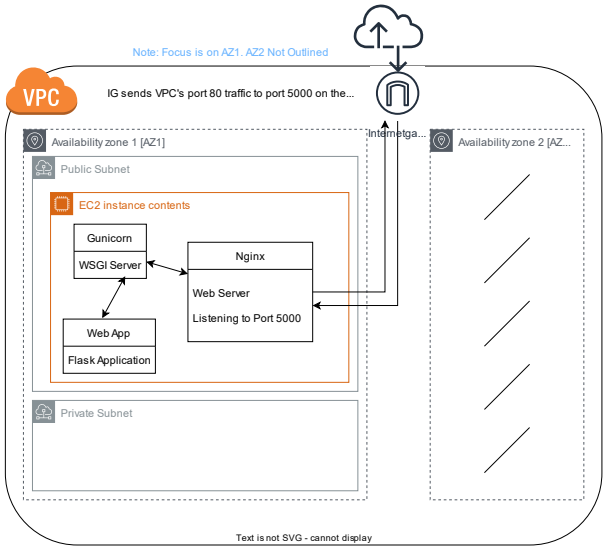
Project Outline:

Using a Jenkins CI/CD Pipeline to deploy a Flask / Python application to a Jenkins Agent that is hosted in a customized VPC.

1. Create a new Multibranch Pipeline on an existing Jenkins server / automation platform
    a. Clone the “Deployment 3” repository from Kura Labs
    b. Review Build and Test steps of existing Jenkinsfile and run a test Build



2. Create an EC2 instance in a Public subnet of an existing VPC (see Appendix A for a brief overview and diagram of the VPC infrastructure)
    a. Open port 22 for SSH
    b. Open port 5000 to host the application



3. Configure and connect a Jenkins Agent on the EC2 from #2
4. Add a “Deploy” stage to the project’s Jenkinsfile
    a. This stage will run remotely on the aforementioned Jenkins Agent

Notes:

In order to see Deployment through to completion, the following actions were necessary:

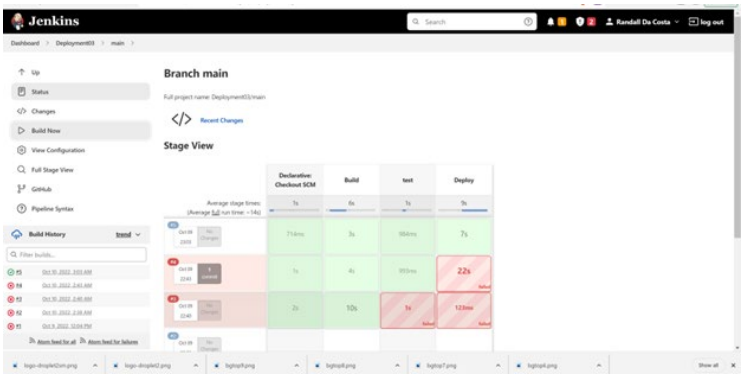
1. Ensure “gunicorn” was installed on the Jenkins Agent EC2 instance
2. Ensure “venv” was installed on the Jenkins Agent EC2 instance
3. Ensure “nginx” was installed
4. Ensure “default-jre” was installed
5. Ensure “python3-pip” was installed

Issues:

1. The Python “test_app.py” required a slight modification to the main “assertion” in order to test the application successfully
2. Improper Agent setup in Jenkins Master console lead to deployment failure
3. If the deployment had already been made once to the agent, subsequent deployments would show as having succeeded (on the Jenkins master’s web console). However, the reality is that no new changes could be deployed due to the fact that “gunicorn,” the webserver gateway interface being used, was already running on the Agent server.
4. Jenkins terminates all processes once it completes a run through each stage [Build, Test, Deploy, etc.]. This process termination killed the virtual environment that the application runs in on the agent server. This in turn led to the deployed application producing a 502 error and not be served when the http request was made

Resolutions & Improvements:

1. Test Failure Repaired by removing fault from the “assertion.” See failed “Test” stage in Build #3 and subsequent Success in Build #4 below
2. Deployment failure due to improper Agent set up in Build #4 was resolved. See successful deploy state in Build #5



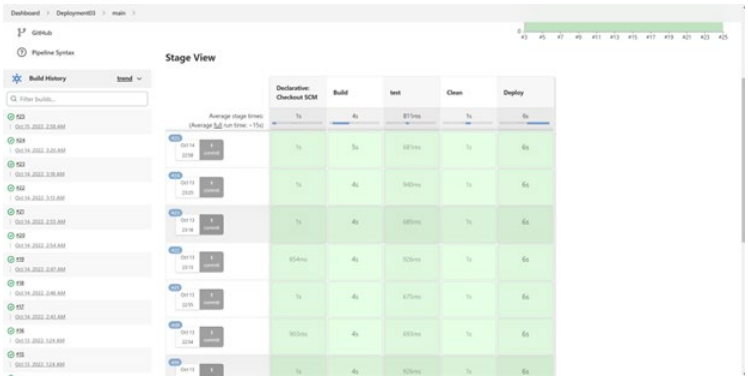
3. “Clean” stage was added to pipeline to resolve the problem outlined in “Issues 3” above.

```
29              stage ('Clean') {
30               agent{label 'awsDeploy'}
31               steps {
32                 sh '''#!/bin/bash
33                 if [[ $(ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2) != 0 ]]
34                 then
35                   ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2 > pid.txt
36                   kill $(cat pid.txt)
37                   exit 0
38                 fi
39                 '''
40               }
41             }
```

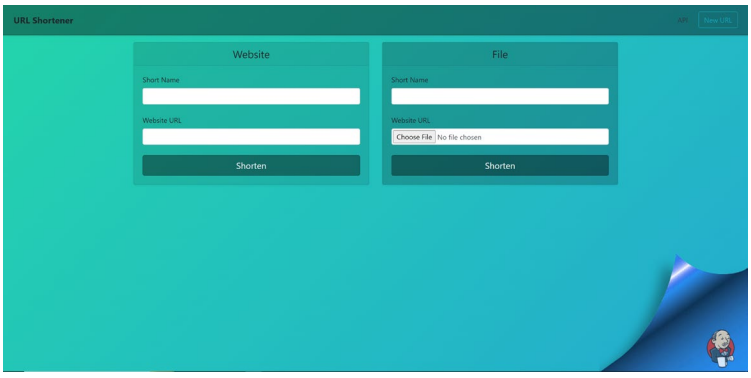
4. “StayAlive” command was added to the “Deploy” stage to solve the problem outlined in “issues 4” above.

```
42              stage ('Deploy') {
43                 agent {
44                     label 'awsDeploy'
45                 }
46                 steps {
47                     sh '''#!/bin/bash
48                     git clone https://github.com/dacostaration/kuralabs_deployment_3.git cd ./kuralabs_deployment_3
49                     python3 -m venv test3 source test3/bin/activate
50                     pip install -r requirements.txt pip install gunicorn
51                     JENKINS_NODE_COOKIE=stayAlive
52                     gunicorn -w 4 application:app -b 0.0.0.0 --daemon '''
53                 }
54             }
```

Finally, a successful pipeline…



And an active web application…



Possible Improvements:

1. Due to limitations with either service [Gunicorn and Nginx], both must be used to serve the dynamic and static file content on the Jenkins Agent. Perhaps there is a better web server set up to host a Python application without the need for a reverse proxy like Nginx?