Deployment 3

Brayan Molina

October 9th, 2022

Table of Contents

Objectives 3
Documentation 3
Jenkins Configuration and Installation: 3
Jenkins Agent Configuration: 3
Pipeline Build in Jenkins 4
Modification to pipeline: 5
CI/CD Pipeline 6
VPC Diagram 7
Technology Stack 8
Sources 9

Objectives:

This deployment will build on the information used in previous deployments of the flask application, Url-shortner. The objectives of this deployment will be the following:

- Gain familiarity with VPC's and subnets.
- Understand the role of the Jenkins agent and controller.
- Understand the role Nginx and Gunicorn play in application deployment.
- Understand reverse proxy and the configurations needed to deploy our flask application.

Documentation:

Jenkins Configuration and Installation:

- EC2 instance on AWS configured on default VPC with the following:
    - Port 22 – SSH
    - Port 80 – HTTP
    - Port 8080 – Jenkins
    - Jenkins script used for default installation

- EC2 instance configured on Kura VPC in a public subnet
    - The following command used for package installation of default-jre, python3-pip, python3.10-venv, and nginx.

```
sudo apt install -y default-jre python3-pip python3.10-venv nginx
```

Jenkins Agent Configuration:

- Log into Jenkins and select the build executor status to create a new node
    - Node: A machine which is part of the Jenkins environment and capable of executing Pipelines or jobs.
- The agent will be used for the deployment step in our jenkinsfile and will use both Nginx and Gunicorn to deploy the flask application.
- The number of executors indicates how many jobs can be run by the agent at the same time. Each executor runs one process at a time.
- The remote root directory will be the root directory for your agent.
- The label and usage sections in the configuration will be used in the jenkinsfile deploy stage to allow the agent to only run jobs with the designated keywords of "awsDeploy".
- The configurations set for the agent were as follows:
    - Name: awsDeploy
    - Description: Deployment Server
    - Number of executors: 1
    - Remote root directory: /home/ubuntu/agent
    - Labels: awsDeploy
    - Usage: Only build jobs with label expressions matching this node
    - Launch method: Launch agents via SSH
    - Host: EC2 Public IP

    - Host key verification strategy: Non verifying verification strategy
    - Availability: Keep this agent online as much as possible
- Credentials:
    - Add → Jenkins
    - SSH Username with private key
    - Username: ubuntu
    - Private Key: Paste from private key associated with the EC2, the agent will be installed on

Pipeline Build in Jenkins:

Nginx configuration:

- Configuration of nginx default file needs to be adjusted.
- Server block is where the nginx will be listening for incoming requests. Nginx is a web server that serves as a reverse proxy, handling static requests or routing HTTP requests to Gunicorn. Nginx is unable to handle python requests, but it can forward requests to your application server (Gunicorn). The port that nginx will be using to listen for requests is port 5000, which was initially configured on our public subnet in the Kura VPC.
- The location block is where requests are sent from Nginx; In this case, we are specifying to send requests to a specified proxy server (localhost at port 8000), the default TCP socket created by Gunicorn. The location block is telling Nginx where to find the files to grab as well.
- The X-Forwarded For (XFF) helps to identify the originating client IP address connecting to the web server through a proxy server.
- Host header is made to correspond to the configuration made under the server block.

Jenkinsfile:

- Four stage declarative pipeline to build, test, and deploy the application.
- The agent that was configured inside the Kura VPC will kill any instances of Gunicorn running before deploying the application. Killing the process is necessary to avoid conflicts using Port 8000.
- The deploy stage contains the installation of Gunicorn and the command to run the application using Gunicorn.
    - -w is defining the number of workers used by Gunicorn. The workers are processes that are used to load the python application.
    - -b option used to bind the IP address to the socket. In this case the -b option is specifying 0.0.0.0 which will allow outside connections.
    - –daemon used to run the process in the background to handle requests from the client.
    - Using ss -atn shows information related to sockets. Showed Gunicorn running listening on port 8000. This was also configured on the nginx configuration file where we passed proxy to 127.0.0.1:8000.
- Alternative solution to resolving Jenkins killing the process after completing the job is to use environment variables such as JENKINS_NODE_COOKIE.
    - JENKINS_NODE_COOKIE=stayalive is an environment variable in Jenkins that will allow the process to continue to run after Jenkins completes the pipeline job. In our deployment, this will allow our application to remain running after Jenkins completes the build.

Modification to pipeline:

- Previous modification was performed manually. The following lines were added to the build stage of the Jenkinsfile:

```
python3 -m pip install mypy
mypy –show-error-codes application.py >> test3/bin/linterrors.txt
```

- The output from mypy will be placed in a file called linterrors.txt; path of file could be changed if needed.

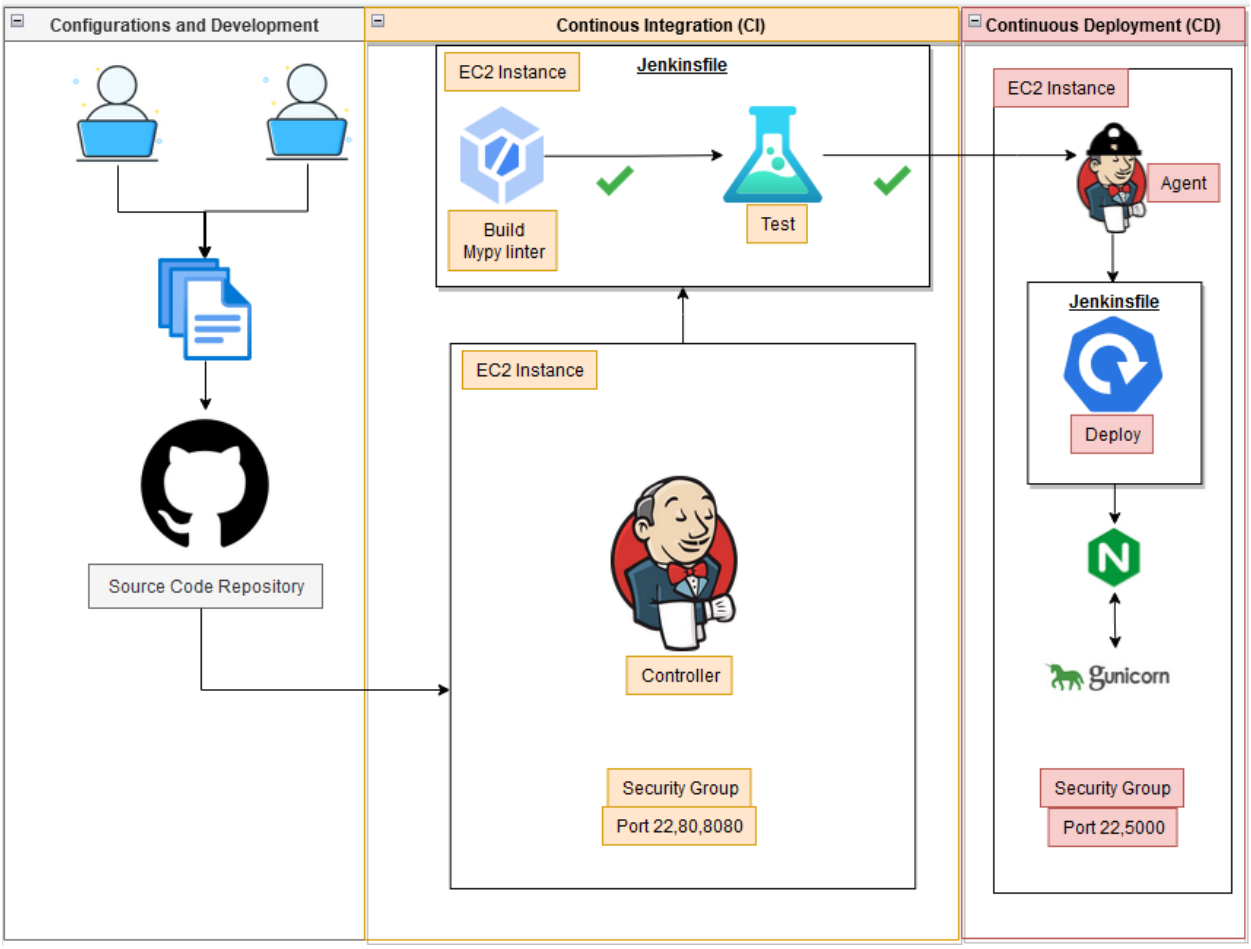
Observations and Errors:

Error	*Cause*	*Solution*
502 Error after Jenkins job completed	Jenkins kills all processes when build is completed	Include environment variable JENKINS_NODE_COOKIE to ensure process remains running after deploy stage.
test_app.py failed in testing stage	Extra space in the test_app.py file function	Remove space from function to fix syntax compatibility issue.
Unexpected '}' in line 61	Extra brace inside the Jenkinsfile is causing the stages function to be closed early.	Remove additional brace from Jenkinsfile.

Improvements:

- Create redundancy in a singular VPC instead using two different subnets for the controller and agent.
- Put Nginx and Gunicorn in private subnets and use a jump host to interact with the private subnet. This will create another layer of security.

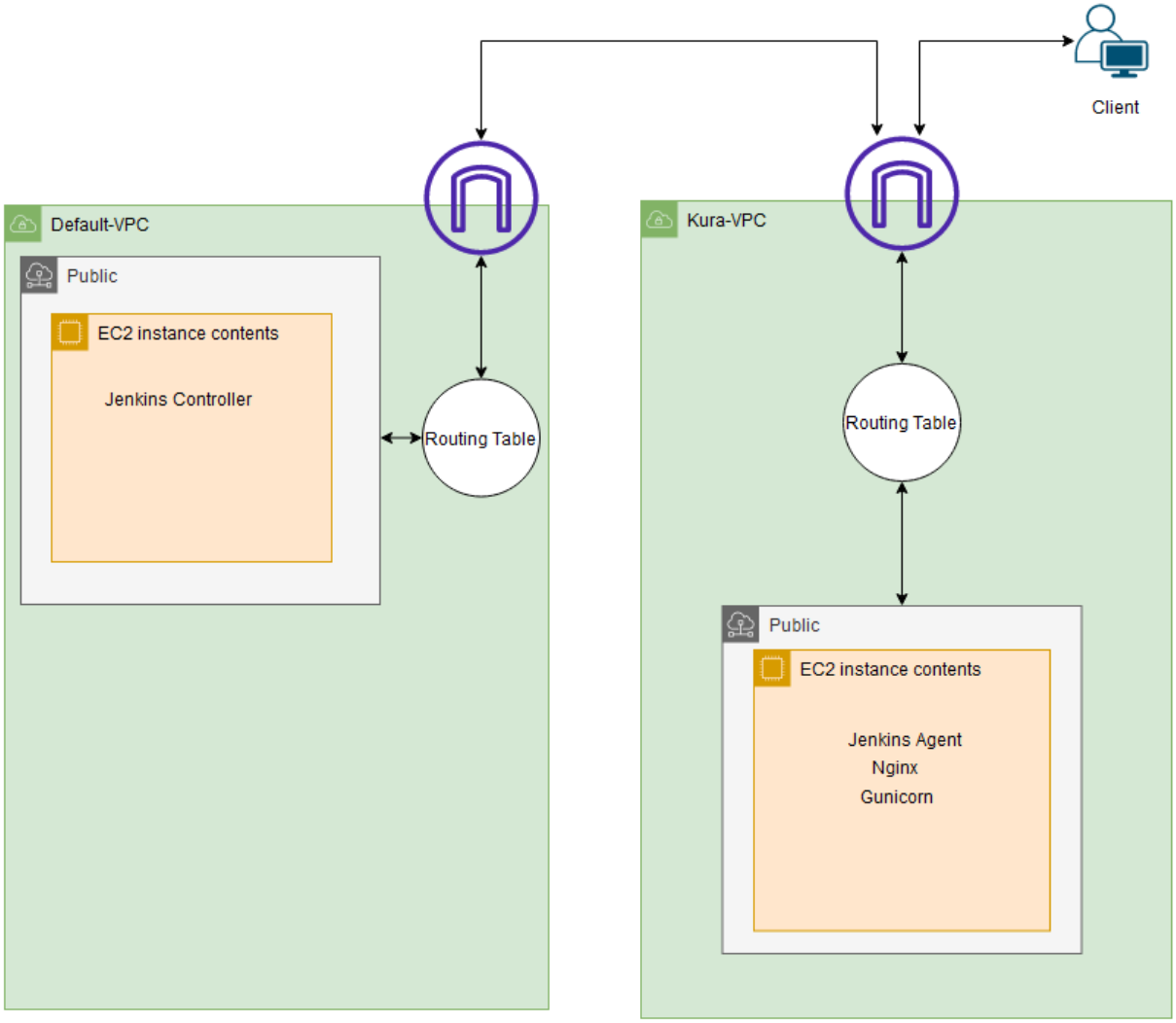
CI/CD Pipeline:



Key takeaways:

- Jenkins agent is responsible for deploying the application using Gunicorn on the EC2 instance.
- Gunicorn will forward response to requests to Nginx open on port 5000.
- The linter was added as two separate commands in the build stage of the Jenkinsfile. If the linter fails in the build stage, Jenkins will return the build stage as having failed.

VPC Diagram:



Key Takeaways:

- Single availability zones for the subnets in the VPC's.
- The client will be connecting to the EC2 instance that contains the services for Nginx and Gunicorn.
- The client can connect to Nginx on port 5000 but opening port 8000 will also allow the client to connect directly to the application since Gunicorn is operating on port 8000.
- Client is accessing web server (Nginx) on port 5000. Nginx will route requests to the application server (Gunicorn) if needed. Nginx can serve static files.

Technology Stack

Frontend Stack:

Nginx

JavaScript

HTML

CSS

Amazon EC2

Backend Stack:

Flask

Linux

Gunicorn

Python

Amazon EC2

DevOps Tools:

Jenkins

Mypy

Sources:

1. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For
2. https://stackoverflow.com/questions/68989275/keep-jenkins-child-process-running-after-build-completes
3. https://www.techtarget.com/searchapparchitecture/definition/software-stack
4. https://dev.to/brandonwallace/deploy-flask-the-easy-way-with-gunicorn-and-nginx-jgc
5. https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-18-04
6. https://www.digitalocean.com/community/tutorials/how-to-deploy-python-wsgi-apps-using-gunicorn-http-server-behind-nginx