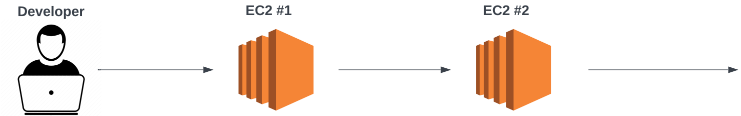
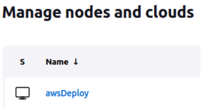


Used a previous EC2 from past deployments to install Jenkins

- Ports used: 80, 8080, 22
- 8080 port is used alongside the public ip address of my EC2 to access my Jenkins server
- This first EC2 instance created is located within the default amazon VPC

Configured an EC2 within the Public Subnet of my VPC with port numbers: 22 and 5000

- This EC2 will be used to contain the jenkins agent needed to deploy our Flask Application
    - **port 5000 is used for accessing Nginx**
- These packages we're installed within the EC2:
    - **sudo apt install *default-jre*:** this package installs the Java run time environment that is needed for Jenkins, but more specifically to configure/connect our agent
    - **sudo apt install *python3-pip***: package manager for python3
    - **sudo apt install python3.10-venv**: installing the environment we will be deploying our application in
    - **sudo apt install nginx**: Being used as our web server to route traffic and communicate with gunicorn



Manage nodes and clouds

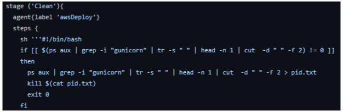
S	Name ↓
🖥	awsDeploy

Configured a Node within Jenkins named 'awsDeploy' and added our agent to it

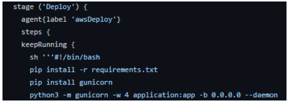
- I am creating a node to essentially monitor the slave state of my build, nodes in Jenkins serve as the "machines" that are able to host one or many pipeline processes
- Nodes also connect pipeline processes to web services, and in this deployment it will be used to connect to the Nginx web server
- I used the public IPV4 address of the EC2 within the public subnet of my VPC as the host for the Node
    - configured the node launch method to: **launch agents via ssh**

Jenkins Agent

- The Jenkins Agent we are configuring is dependent on the installation of Java
- Essentially, an agent is a machine or container which is connected to a Jenkins controller and performs tasks when directed by the Jenkins controller
- The agent specifies where the entire pipeline or a specific stage, will execute in the Jenkins environment all depending on the placement of the agent



```
stage ('Clean'){
  agent{label 'awsDeploy'}
  steps {
    sh '''#!/bin/bash
    if [[ $(ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2) != 0 ]]
    then
      ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2 > pid.txt
      kill $(cat pid.txt)
      exit 0
    fi
```

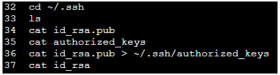


```
  stage ('Deploy') {
    agent{label 'awsDeploy'}
    steps {
    keepRunning {
      sh '''#!/bin/bash
      pip install -r requirements.txt
      pip install gunicorn
      python3 -m gunicorn -w 4 application:app -b 0.0.0.0 --daemon
```

Differences between top and stage level Agents

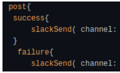
- Top Level Agents
  - for agents that are specified at the top level of the pipeline code, the steps or options declared are executed after entering the agent
- Stage Level Agents
  - for agents that are declared within a specific stage, the stage steps are executed prior to entering the agent
  - **In this deployment, the agent we configured is placed within the 'Deploy' and 'Clean' stage**
  - The 'agent { label 'awsDeploy' } syntax within the python code of the pipeline is signifying to execute the stage on an agent available in the jenkins environment with the specified label

Troubleshooting/Build Failures when configuring Jenkins Node



```
32  cd ~/.ssh
33  ls
34  cat id_rsa.pub
35  cat authorized_keys
36  cat id_rsa.pub > ~/.ssh/authorized_keys
37  cat id_rsa
```

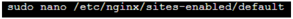
- when entering the credentials needed to connect the agent to jenkins, it prompted me to paste my private key. My initial approach was to take my EC2 instance key pair
  - however, upon storing that as my private key and saving the credentials, my node instantanously went offline and would not launch
  - After troubleshooting and looking at documentation, I noticed that I had to create a new private key within my VPC EC2 I created my agent in.
  - I had to cd into the < /.ssh > directory, run the keygen command to generate a new key and then cat the file where I copied/pasted the entire key back into the credentials.
  - Upon reconfiguring my Node, it eventually went online and began to progress through my pipeline



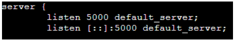
```
post{
 success{
     slackSend( channel:
 }
  failure{
     slackSend( channel:
```

Installed the Slack notif
Jenkins, then configure
reflect my slack worksp
channel I created for pr
alerts about my

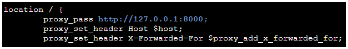
- I inserted a couple lin code within my jenkin specifically notify me my build has ran succ has failed



```
sudo nano /etc/nginx/sites-enabled/default
```



```
server {
        listen 5000 default_server;
        listen [::]:5000 default_server;
```



```
location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

Prior to creating a multi-branch pipeline for the deployment of my application, I needed to nano into the above file and configure some settings

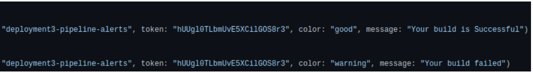
- I configured the default port 80 to port 5000 which will allow the jenkins agent created within the public subnet of my VPC to communicate with our webserver nginx
  - if I do not configure the file so that Jenkins listens to the specific hostname that contains nginx, which is through port 5000, it will then automatically listen to all interfaces on it's default port 8000

NGINX

Nginx handles a high volume of requests and is commonly used as a reverse proxy and load manager to manage incoming traffic

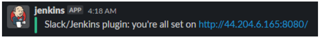
- uses few resources and is useful for delivering static content (files that are stored within a server and remains the same when delivered to users)
  - any information/files that can be delivered to a user without having to be modified
- Nginx web server essentially reduces a web application's page load time
- **In this deployment, the configurations done to the nginx file as previously stated will indicate that I am setting up Nginx as a reverse proxy to guarantee that when the port 8000 is visited, the application will thus load**
- **Nginx accelerates the deployment of our Flask Application, as it automates tasks that are manually done by the developers**

Slack notification integration with Jenkins Pipeline



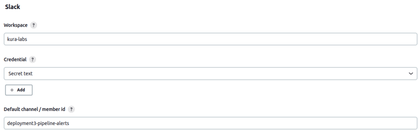
```
"deployment3-pipeline-alerts", token: "hUUgl0TLbmUvE5XCilGOS8r3", color: "good", message: "Your build is Successful")

"deployment3-pipeline-alerts", token: "hUUgl0TLbmUvE5XCilGOS8r3", color: "warning", message: "Your build failed")
```

fier plugin within
d the settings to
pace and private
imarily receiving
pipeline
es of python
s file to
on slack when
cessfully or if it



jenkins APP 4:18 AM

Slack/Jenkins plugin: you're all set on http://44.204.6.165:8080/



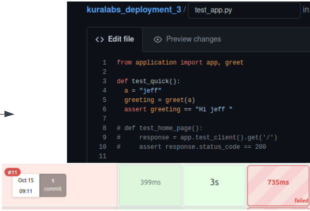
Slack

Workspace: kura-labs

Credential: Secret text

Default channel / member id: deployment3-pipeline-alerts

Trouble Shooting/Deployment Issu

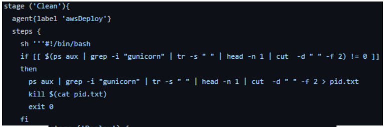


kuralabs_deployment_3 / test_app.py

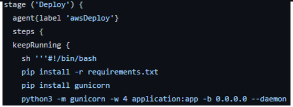
```
from application import app, greet

def test_quick():
  a = "jeff"
  greeting = greet(a)
  assert greeting == "Hi jeff "

# def test_home_page():
#     response = app.test_client().get('/')
#     assert response.status_code == 200

```

During the deployment of my application, I co
testing stage failures

- My testing stage kept failing due to imprope
spacing within my test_app.py file, once I re
and properly place the quotes within the arg
was able to fully complete and deploy



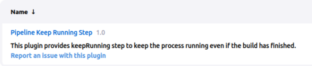
```
stage ('Clean'){
  agent{label 'awsDeploy'}
  steps {
    sh '''#!/bin/bash
    if [[ $(ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2) != 0 ]]
    then
      ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut  -d " " -f 2 > pid.txt
      kill $(cat pid.txt)
      exit 0
    fi
```



```
stage ('Deploy') {
  agent{label 'awsDeploy'}
  steps {
  keepRunning {
    sh '''#!/bin/bash
    pip install -r requirements.txt
    pip install gunicorn
    python3 -m gunicorn -w 4 application:app -b 0.0.0.0 --daemon
```

After forking the repository, I edited the Jenkins file with a 'Clean' and 'Deploy' stage interacting with the Jenkins agent I created

- ' keepRunning { ' python code syntax within this deployment is uesd to keep the pipeline, initiated by Jenkins running after the build has finished
  - Further along, after the creation of the Multi-Branch pipeline, I would need to install the Jenkins plugin "Pipeline Running Step" so the python written 'keepRunning' input runs smoothly



Name ↓

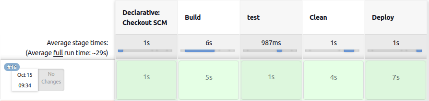
Pipeline Keep Running Step 1.0

This plugin provides keepRunning step to keep the process running even if the build has finished.

Report an issue with this plugin

Within the deployment, Gunicorn is being installed within the 'Deploy' stage as it will allow for Python applications to be ran simultaneously through running multiple Python processes

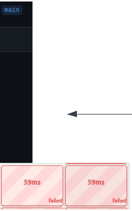
- Gunicorn is serving as the application server
- Gunicorn can serve static files, however and this is where our application server communicates with our web server (Nginx), it is efficient within the deployment of our flask app to let Nginx handle the task of serving the static files and managing requests, as it takes a large portion of the load from the application servers (Gunicorn) resulting in a better performance



	Declarative: Checkout SCM	Build	test	Clean	Deploy
Average stage times: (Average full run time: ~29s)	1s	6s	987ms	1s	1s
#16 Oct 15 09:34 No Changes	1s	5s	1s	4s	7s

After updating the Jenkins file within the GitHub repository, and adding two more stages to the pipeline, I created a Multi-Branch Pipeline within Jenkins to deploy my Flask Application

es



ntinuosly ran into

r quotation
alized the error
ument, the build