Ishtaar Desravines
10/14/2022
Deployment 3 Documentation

Objective: Deploying a python application to Gunicorn through the CI/CD pipeline using Jenkins, Github, and AWS services. Two VPCs were used in the deployment, one hosted an EC2 with a Jenkins manager and the other hosted an EC2 with a Jenkins agent.

Creating a Jenkins Server on an EC2

- Create a Jenkins manager on an EC2 on AWS
- Launch an instance and open ports 22 (ssh), 80 (HTTP), and 8080 (TCP) in the security groups setting.
- Under “Advanced details”, add jenkins server script below to the “User data” section to automatically install Jenkins on the EC2 and create an active agent.
- Launch instance and ssh into the Jenkins EC2.
- Verify jenkins is running on the EC2 with: `$ systemctl status jenkins`
- Open a web browser and enter http://{EC2publicipaddress}:8080 to access the jenkins webpage.
- Retrieve the access key and create a Jenkins account.

```
#!/bin/bash

sudo apt update
sudo apt -y install openjdk-11-jre
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian-stable binary/ | sudo tee
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get -y install jenkins
sudo systemctl start jenkins
systemctl status jenkins >> ~/file.txt
```

Creating a Custom VPC on AWS

- Navigate to VPC services on AWS and create a VPC.
- Create private and public subnets by choosing the corresponding CIDR block. For this deployment we are installing a Jenkins agent on an EC2 on a Public Subnet.
- Select an internet gateway and route tables for the VPC.

Creating a Jenkins Agent on an EC2

- Create a Jenkins manager on an EC2 on AWS
- Launch an instance and open ports 22 (ssh) and 5000 (TCP) in the security groups settings. Port 5000 is open so that it can communicate with the nginx web server.
- Launch instance and ssh into the Jenkins Agent EC2.
- Install Java Runtime Environment and nginx

```
$ sudo apt install default-jre nginx
```

- Edit the default nginx file to tell the nginx server to communicate with the EC2 on port 5000. Use the command below and make the following corrections.

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

- Corrections:

```
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;
```

Installing Virtual Environment on both EC2s:

- Python must be installed since the application includes python files.
- Install virtual environments on both EC2s so that the application can run without version conflicts from their dependencies.
- Follow the commands below:

```
$ sudo apt install python3-pip
$ sudo apt install python3.10-venv
```

Configure and Connect Jenkins Agent to Jenkins Server:

- On Jenkins GUI on Jenkins Manager server, select “Build Executor Status” and then “+ New Node”
- Give the Node a name, ex “awsDeploy” and select permanent agent
- Enter the following configurations:
    - Name: awsDeploy
    - Description: Deployment server
    - Number of executors: 1
    - Remote root directory: /home/ubuntu/agent
    - Labels: awsDeploy
    - Usage: only build jobs with label
    - Launch methods: launch agents via ssh
    - Host: EC2 that is going to be the jenkins agent (EC2 in the public subnet of custom VPC)
    - Credentials: ubuntu (SSH-CALI)
        - Add → Jenkins → SSH username with private key
        - Scope: Global, ID: JenkinsAgent, Description: deployment agent server
        - Username: ubuntu, private key: paste EC2 private key
    - Host Key verification strategy: non verifying verification strategy
    - Availability: keep this agent online as much as possible.
- Select Apply then Save and wait for Jenkins manager to launch the jenkins agent.

Add Clean Stage and Edit Deploy Stage in Jenkins File

- Add the clean stage after the test stage:

```
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
     '''
   }
 }
```

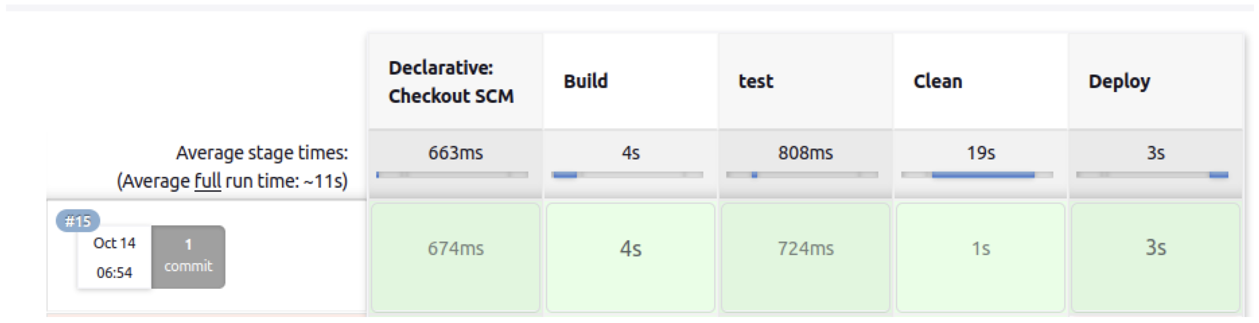
- Edit the Deploy stage:

```
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
    '''
   }
  }
}
```

Create a Pipeline Build in Jenkins:

- On the Jenkins Manager server, download the “Pipeline Keep Running Step” Plugin
- Create a new pipeline by selecting: “New Item” and enter the name of the application: “url-shortener” → select “Multibranch pipeline” → select “OK”
- Enter Display Name: Build Flask
- Enter Description: CI/CD pipeline deployment 2
- Add Branch source: Github and under Github Credentials: select “Add” → Jenkins
- Under “Username”: Enter Github Username
- Under “Password”: Enter Github personal access token
- Save Github credentials

- Enter the Github repository URL → select “Validate”
- Check to make sure under Build Configuration → Mode , it says “by Jenkinsfile” and under Script it say “Jenkinsfile”
- Select Apply → Save
- Saving this configuration should trigger an automatic SCM checkout, build, then test in Jenkins. All of these stages of the pipeline should pass.



	Declarative: Checkout SCM	Build	test	Clean	Deploy
Average stage times: (Average full run time: ~11s)	663ms	4s	808ms	19s	3s
#15 Oct 14 06:54 1 commit	674ms	4s	724ms	1s	3s

Adding Slack Notifications to the CI/CD pipeline:

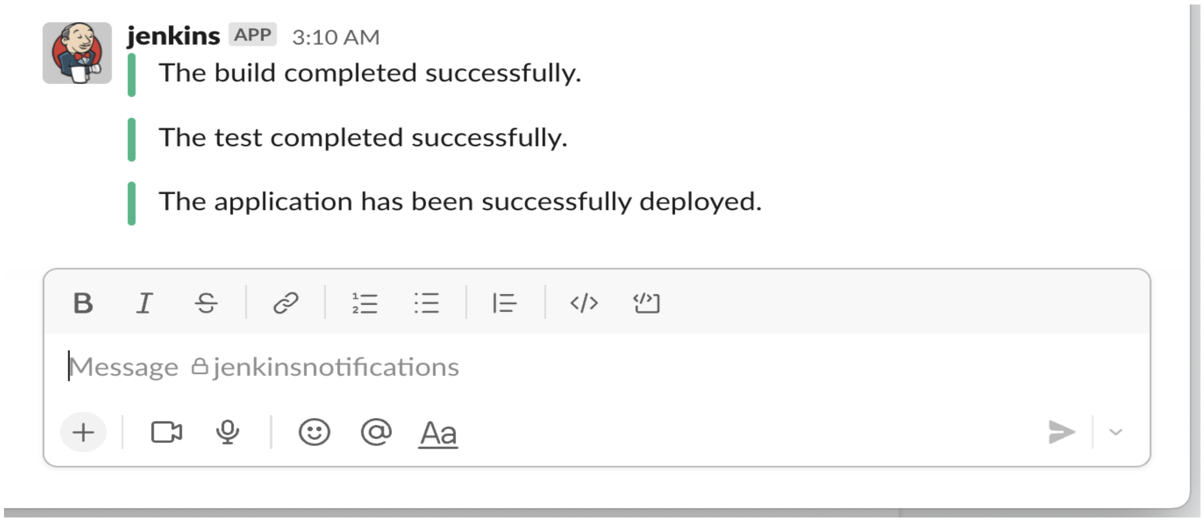
Slack notifications were added to give updates about each stage of the pipeline (whether the stage was successful or failed).

Steps:

- Clink on the link to configure jenkins integration: https://my.slack.com/services/new/jenkins-ci
- Copy the integration token and save for later step.
- Choose a channel or create a channel to receive jenkins notifications then select “Add Jenkins CI Integration”
- Head over to Jenkins GUI and go to Dashboard → Manage Plugins → Configure Settings.
- Scroll down to “Slack Notifications” and click “Add+” and select “jenkins”
- Add workspace name, channel name, and integration token, then press “Apply” then “Save”.
- Add the following code to the Jenkinsfile on Github:

```
post{
   success {
     slackSend channel: 'jenkinsnotifications',
               color: 'good',
               message: "The build completed successfully."
   }
   failure  {
     slackSend channel: 'jenkinsnotifications',
               color: 'warning',
               message: "The build FAILED"
   }
}
```

- Include this code to any stage in the Jenkins file to receive notifications for that stage.
- Save the Jenkinsfile and schedule another build on the application in Jenkins.
- Notifications on each stage will appear in the indicated slack channel.


jenkins APP 3:10 AM

The build completed successfully.

The test completed successfully.

The application has been successfully deployed.

Issues:

- The location of Gunicorn needs to be placed in the PATH so it can be found and the application can be deployed using Gunicorn.
- The test failed in the test stage in Jenkins. There was an additional space added at the end of “Hi jeff”