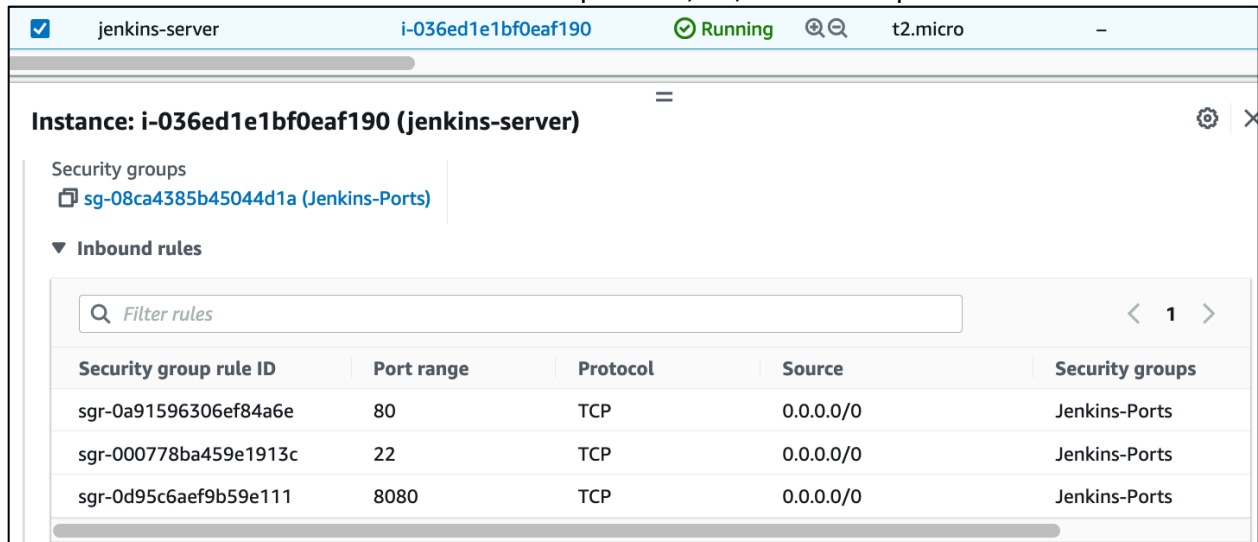


Install Jenkins on an EC2 from the Default VPC

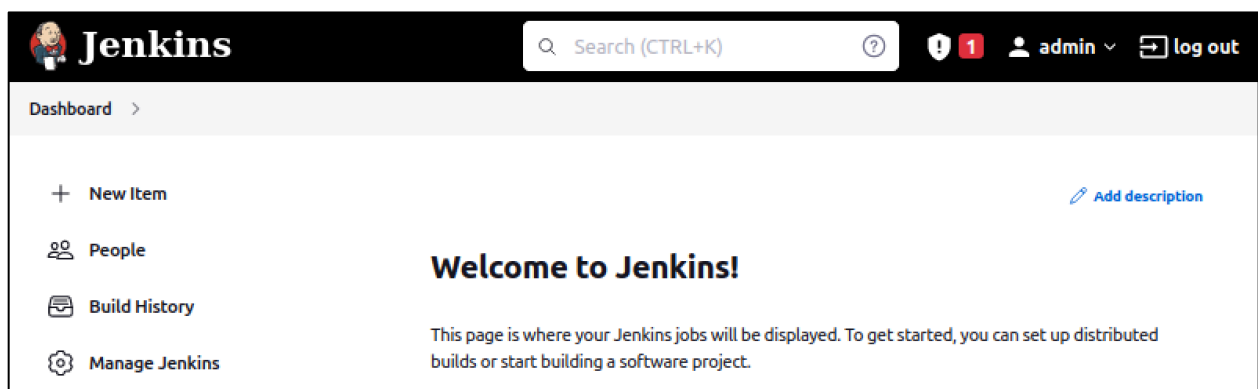
- Create an EC2 with Ubuntu AMI and ports 22, 80, and 8080 open:



- SSH into the EC2 and run `setup_jenkins.sh` to install, run, and check status of Jenkins:

```
ubuntu@ip-172-31-89-3:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2022-10-09 14:47:31 UTC; 14s ago
     Main PID: 4576 (java)
       Tasks: 44 (limit: 1143)
      Memory: 306.4M
         CPU: 49.300s
    CGroup: /system.slice/jenkins.service
            └─4576 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war
```

- Go to <http://<ec2-public-ip>:8080> to set up Jenkins admin role – retrieve password by running the command `sudo cat /var/lib/Jenkins/secrets/initialAdminPassword`
- Install suggested plugins
- Reset admin password then save



Create an EC2 in the Public Subnet of Kura VPC

- Select Ubuntu AMI and follow the configurations:
 - o Kura VPC
 - o Public Subnet
 - o Auto-Assign Public IP -> Enable

▼ Network settings [Info](#)

VPC - *required* [Info](#)

vpc-088f820674463f327 (kura-review-VPC)
172.25.0.0/16

Subnet [Info](#)

subnet-0c76d5c8ca9df0aa6
VPC: vpc-088f820674463f327 Owner: 108026381256
Availability Zone: us-east-1a IP addresses available: 16377 CIDR: 172.25.0.0/18

PUBLIC-REVIEW-A

Auto-assign public IP [Info](#)

Enable

- o Ports 22 and 5000:

Inbound security groups rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) [Remove](#)

Type [Info](#) Protocol [Info](#) Port range [Info](#)

ssh TCP 22

Source type [Info](#) Source [Info](#) Description - *optional* [Info](#)

Anywhere [Add CIDR, prefix list or security](#) e.g. SSH for admin desktop

0.0.0.0/0 ✕

▼ Security group rule 2 (TCP, 5000, 0.0.0.0/0) [Remove](#)

Type [Info](#) Protocol [Info](#) Port range [Info](#)

Custom TCP TCP 5000

Source type [Info](#) Source [Info](#) Description - *optional* [Info](#)

Anywhere [Add CIDR, prefix list or security](#) e.g. SSH for admin desktop

0.0.0.0/0 ✕

- Install necessary packages: default-jre, python3-pip, python3.10-venv, nginx >
Use **setup_VPC_pub_ec2.sh** or Include under User Data as a bootstrap script:

User data [Info](#)

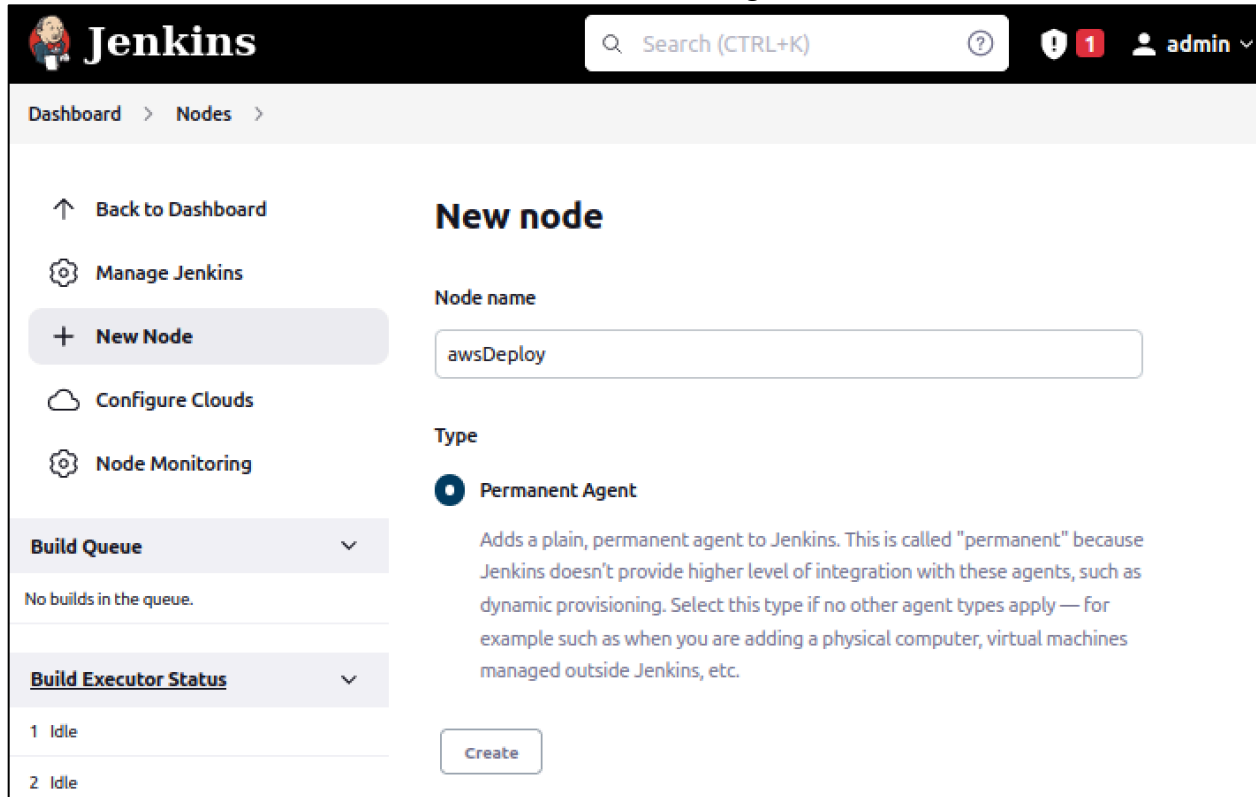
```
#!/bin/bash
```

```
sudo apt update && sudo apt upgrade -y
```

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

Configure and Connect a Jenkins Agent to Jenkins

- Inside Jenkins server (EC2 from Default VPC), click on Build Executor Status > + New Node > Enter node name and select Permanent Agent:



The screenshot shows the Jenkins web interface. At the top is the Jenkins logo and a search bar. Below the logo is a navigation menu with links: 'Dashboard', 'Nodes', 'Build Queue', and 'Build Executor Status'. The 'Nodes' link is active, and the 'New node' page is displayed. The 'Node name' field is filled with 'awsDeploy'. The 'Type' is set to 'Permanent Agent'. A description for 'Permanent Agent' is provided: 'Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.' A 'Create' button is at the bottom right. The left sidebar shows 'Build Queue' with 'No builds in the queue.' and 'Build Executor Status' with two idle executors.

- Enter following configurations:
 - o **Name:** awsDeploy
 - o **Description:** Deployment Server
 - o **Number of Executors:** 1
 - o **Remote Root Directory:** /home/ubuntu/agent
 - o **Labels:** awsDeploy
 - o **Usage:** Only build jobs with label...
 - o **Launch Method:** Launch Agents via SSH
 - o **Host:** Public IP of EC2 from Kura VPC
 - o **Host Key Verification Strategy:** Non-verifying verification strategy
 - o **Availability:** Keep this agent online as much as possible

Deployment 3 Documentation

Caden Hong

Dashboard > Nodes >

↑ Back to Dashboard

⚙️ Manage Jenkins

+ New Node

☁️ Configure Clouds

⚙️ Node Monitoring

Build Queue ▾

No builds in the queue.

Build Executor Status ▾

1 Idle

2 Idle

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

Dashboard > Nodes >

Launch method ?

Launch agents via SSH ▾

Host ?

52.207.123.22

Credentials ?

ubuntu (Deployment agent server) ▾

+ Add

Host Key Verification Strategy ?

Non verifying Verification Strategy ▾ ?

Advanced...

Availability ?

Keep this agent online as much as possible ▾ ?

- Credentials:

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

SSH Username with private key

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

ID ?

JenkinsAgent

Description ?

Deployment agent server

Username

ubuntu

Private Key

☒ Enter directly







Key

Enter New Secret Below

-----END OPENSSSH PRIVATE KEY-----

*****PRIVATE KEY IS THE CONTENT OF THE PEM FILE USED TO SSH INTO EC2 INSTANCES*****

- Once Agent configuration info is saved, it will be created and can be viewed from Dashboard > Build Executor Status:

Manage nodes and clouds							Refresh status
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	awsDeploy	Linux (amd64)	In sync	4.98 GB	 0 B	4.98 GB	141ms 
	Built-In Node	Linux (amd64)	In sync	5.05 GB	 0 B	5.05 GB	0ms 
Data obtained		12 min	12 min	12 min	12 min	12 min	12 min

- You can check the log as well:

```
<===[JENKINS REMOTING CAPACITY]===>channel started
Remoting version: 3044.vb_940a_a_e4f72e
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by jenkins.slaves.StandardOutputSwapper$ChannelSwapper to
constructor java.io.FileDescriptor(int)
WARNING: Please consider reporting this to the maintainers of
jenkins.slaves.StandardOutputSwapper$ChannelSwapper
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access
operations
WARNING: All illegal access operations will be denied in a future release
Evacuated stdout
Agent successfully connected and online
```

Create a Pipeline Build in Jenkins

- Prior to building a pipeline – SSH into the EC2 in Kura VPC and change the **/etc/nginx/sites-enabled/default** file:

```
ubuntu@ip-172-25-51-241:~$ sudo nano /etc/nginx/sites-enabled/default
```

1. Change port from 80 to 5000:

```
# Default server configuration
#
server {
    listen 5000;
    listen [::]:5000 default_server;
```

2. Replace contents of location as below:

```
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;
```

- Go back to Jenkins server on EC2 in Default VPC and configure a multibranch pipeline by navigating to Dashboard > New Item > Multibranch Pipeline
- Under Branch Sources > GitHub > Credentials > + Add > Enter GitHub username and generated access token as password:

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

cadenhong

☐ Treat username as secret ?

Password ?

.....

- Once entered, validate connection to ensure Jenkins can access GitHub repo:

Branch Sources

GitHub

Credentials ?

cadenhong/*****

+ Add

Repository HTTPS URL

Repository HTTPS URL ?

https://github.com/cadenhong/kl_wk14_deployment3.git

Credentials ok. Connected to https://github.com/cadenhong/kl_wk14_deployment3.

Validate

- Then, navigate to Dashboard > Manage Jenkins > Plugin Manager and install Pipeline Keep Running Step plugin:

Plugin Manager

Updates

Available

Installed

Advanced

Q

pipeline keep running

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<div>Pipeline Keep Running Step 1.0</div> <div>This plugin provides keepRunning step to keep the process running even if the build has finished.</div>	2 yr 5 mo ago

- Once completed, edit the Jenkinsfile in deployment repo with the following code:

kl_wk14_deployment3 / Jenkinsfile in main

<> Edit file

Preview changes

```
16     stage ('test') {
17         steps {
18             sh '''#!/bin/bash
19             source test3/bin/activate
20             py.test --verbose --junit-xml test-reports/results.xml
21             '''
22         }
23
24         post{
25             always {
26                 junit 'test-reports/results.xml'
27             }
28         }
29     }
30 }
31 stage ('Clean') {
32     agent{label 'awsDeploy'}
33     steps {
34         sh '''#!/bin/bash
35         if [[ $(ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut -d " " -f 2) != 0 ]]
36         then
37             ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut -d " " -f 2 > pid.txt
38             kill $(cat pid.txt)
39             exit 0
40         fi
41         '''
42     }
43 }
44 stage ('Deploy') {
45     agent{label 'awsDeploy'}
46     steps {
47         keepRunning {
48             sh '''#!/bin/bash
49             pip install -r requirements.txt
50             pip install gunicorn
51             python3 -m gunicorn -w 4 application:app -b 0.0.0.0 --daemon
52             '''
```

Test Stage Issues

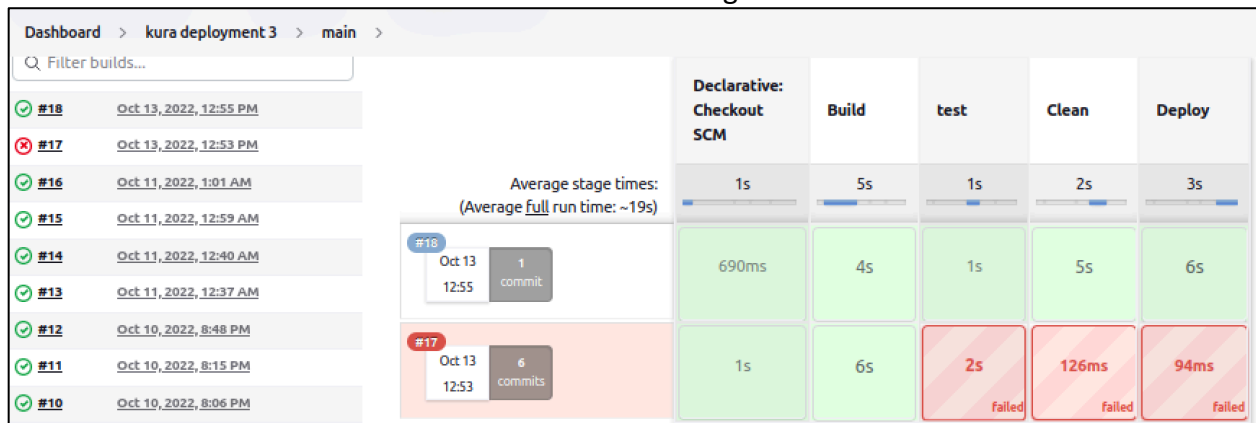
- Test failed initially due to an extra space on line 6 – removed the extra space and did another build:

```
kl_wk14_deployment3 / test_app.py in main

<> Edit file  Preview changes

1  from application import app, greet
2
3  def test_quick():
4      a = "jeff"
5      greeting = greet(a)
6      assert greeting == "Hi jeff"
7
8  # def test_home_page():
9  #     response = app.test_client().get('/')
10 #     assert response.status_code == 200
```

- Successful Build after edits made in the Test Stage:



Deployment Issues

Initial Jenkinsfile

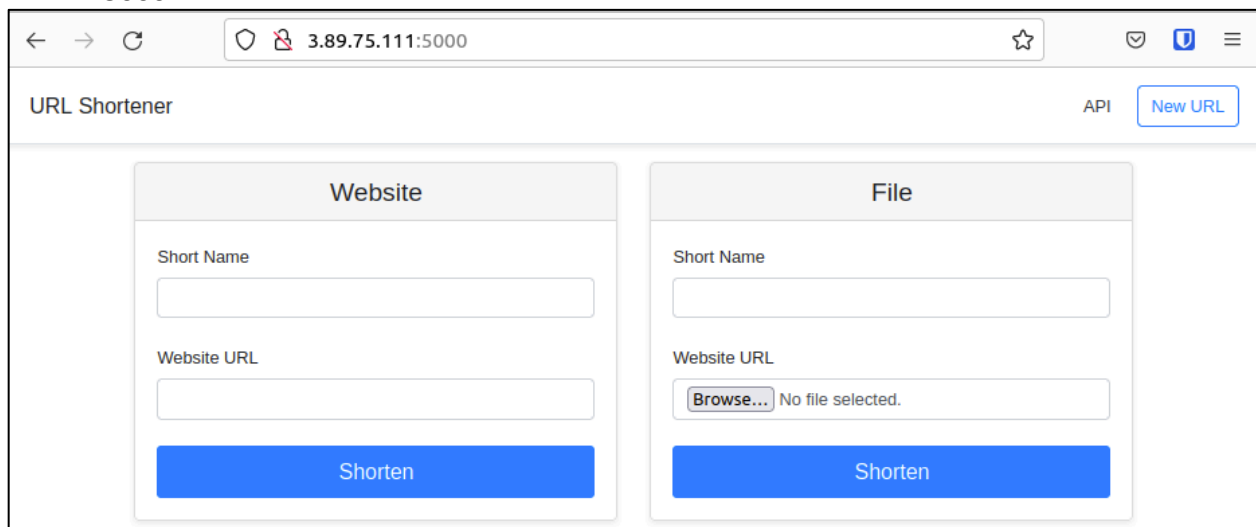
- Even with a successful Deploy stage on Jenkins, there was 502 Bad Gateway error:



- To resolve this, I manually went inside the VPC EC2, activated the venv and ran the last command found in the Jenkins Deploy stage:

```
(test3) ubuntu@ip-172-25-51-241:~/agent/workspace/kl-deployment3_main/kuralabs_deployment_2$ pwd
/home/ubuntu/agent/workspace/kl-deployment3_main/kuralabs_deployment_2
(test3) ubuntu@ip-172-25-51-241:~/agent/workspace/kl-deployment3_main/kuralabs_deployment_2$ gunicorn -w 4 application:app
-b 0.0.0.0 --daemon
```

- Then, I was able to access the url-shortener website using the VPC EC2's IP and port 5000:



Deployment 3 Documentation

Caden Hong

Updated Jenkinsfile

- That said, Tyrone informed us that there was a bug within the original Jenkinsfile, so he provided us with new instructions – upon installing the Jenkins Pipeline Keep Running Step and editing the Jenkinsfile, the Deploy stage was successful:

The screenshot shows a web browser window with the title 'URL Shortener'. The address bar displays '54.209.154.250:5000'. The page has a header with 'URL Shortener' on the left and 'API' and a 'New URL' button on the right. The main content area is divided into two columns: 'Website' and 'File'. Each column contains a 'Short Name' input field, a 'Website URL' input field, and a blue 'Shorten' button. The 'File' column's 'Website URL' field includes a 'Browse...' button and the text 'No file selected.'.

Deployment 3 Documentation

Caden Hong

Additions from Deployment 2

Webhook

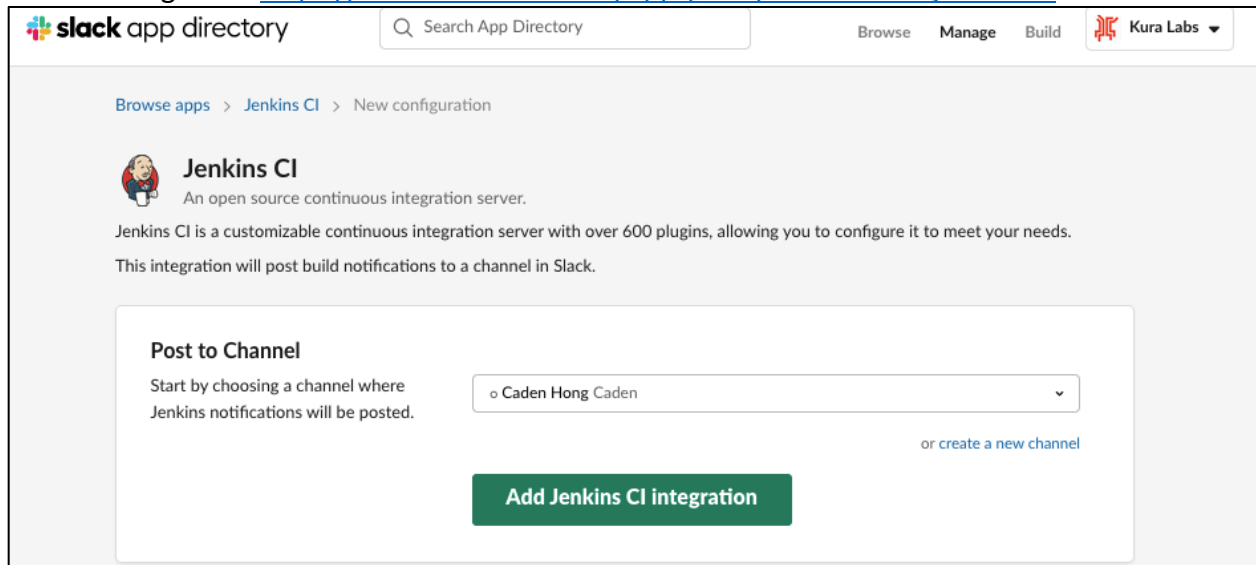
- Navigate to GitHub repo > Settings > Webhooks
 - o Payload URL: <http://<ec2-public-ip>:8080/github-webhook/>
 - o Content Type: application/json

The screenshot shows the GitHub repository settings page for 'cadenhong / kl_wk14_deployment3'. The 'Settings' tab is selected, and the 'Webhooks' section is active. The 'Add webhook' form is displayed with the following details:

- Webhooks / Add webhook**
- Payload URL:** `http://54.204.71.108:8080/github-webhook/`
- Content type:** `application/json`
- Secret:** (Empty text field)
- Which events would you like to trigger this webhook?**
 - ☒ Just the push event.
 - ☐ Send me everything.
 - ☐ Let me select individual events.
- ☒ **Active**
We will deliver event details when this hook is triggered.
- Add webhook** (Green button)

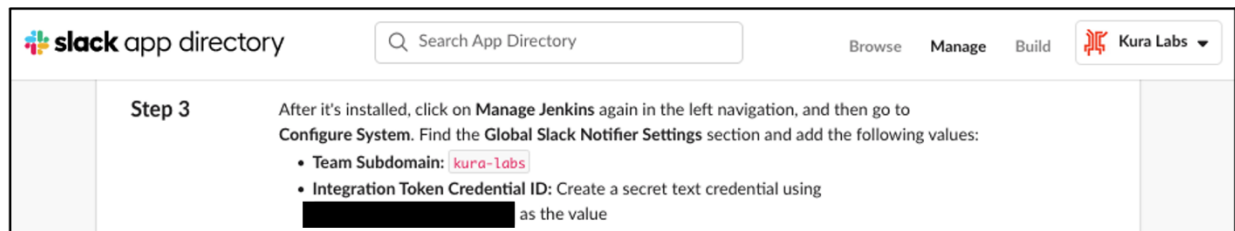
Slack Notifications

- Navigate to <https://kura-labs.slack.com/apps/new/A0F7VRFKN-jenkins-ci:>



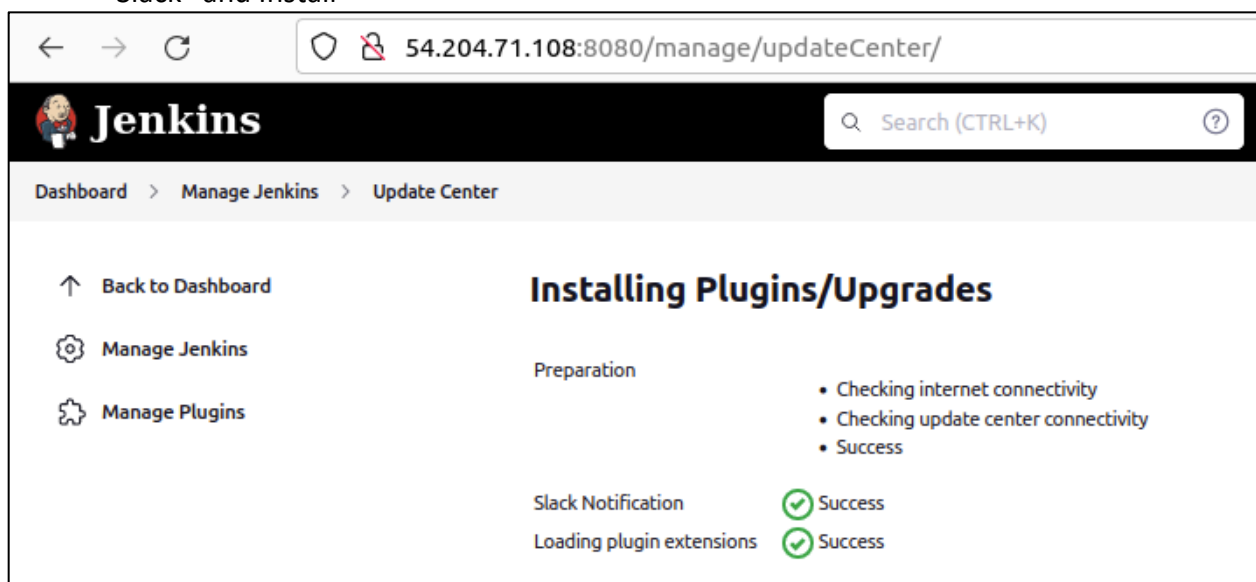
The screenshot shows the Slack App Directory interface. At the top, there's a search bar and navigation links: Browse, Manage, Build, and a Kura Labs dropdown. Below the search bar, there's a breadcrumb trail: Browse apps > Jenkins CI > New configuration. The main section features the Jenkins CI logo and a description: "An open source continuous integration server. Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs. This integration will post build notifications to a channel in Slack." Below this, there's a "Post to Channel" section with a dropdown menu showing "Caden Hong Caden" and a link to "create a new channel". At the bottom, there's a green button labeled "Add Jenkins CI integration".

- Once added, follow the steps provided to set up – take note of the Integration Token Credential ID to enter in Jenkins:



The screenshot shows the Slack App Directory interface with a "Step 3" section. The text reads: "After it's installed, click on Manage Jenkins again in the left navigation, and then go to Configure System. Find the Global Slack Notifier Settings section and add the following values:" followed by a list of values: "Team Subdomain: kura-labs" and "Integration Token Credential ID: Create a secret text credential using [redacted] as the value".

- In Jenkins, go to Dashboard > Manage Jenkins > Plugin Manager > Available > Search for "Slack" and Install



The screenshot shows the Jenkins Dashboard. The breadcrumb trail is: Dashboard > Manage Jenkins > Update Center. The main section is titled "Installing Plugins/Upgrades". It shows a list of plugins and their installation status. The "Slack Notification" plugin is listed with a green checkmark and the word "Success". The "Loading plugin extensions" is also listed with a green checkmark and the word "Success".

- Once completed, navigate to Dashboard > Manage Jenkins > Configure System > Slack > Add workspace name > Add credentials as "Secret Text" > Enter the Integration Token Credential ID provided earlier:

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted) ▼

Kind

Secret text ▼

Scope ?

Global (Jenkins, nodes, items, all child items, etc) ▼

Secret

.....

- Make sure to enter workspace name (i.e. kura-labs) and select the credential entered:

slack

Workspace ?

kura-labs

Credential ?

Secret text ▼

+ Add

- Test connection to ensure it is successful:

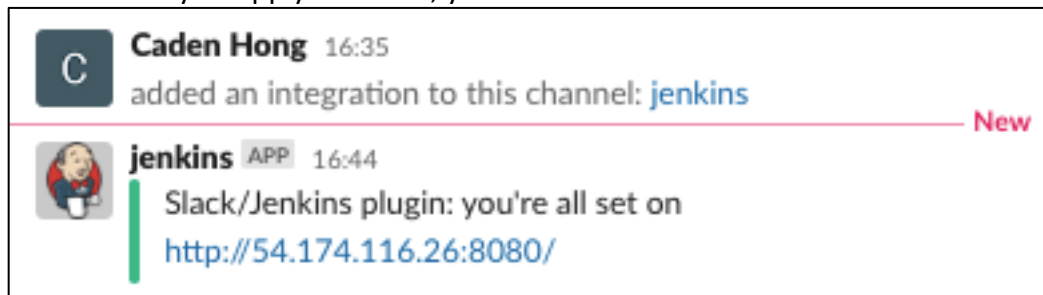
Success

Test Connection

Deployment 3 Documentation

Caden Hong

- Once you Apply and Save, you will receive a confirmation Slack notification:



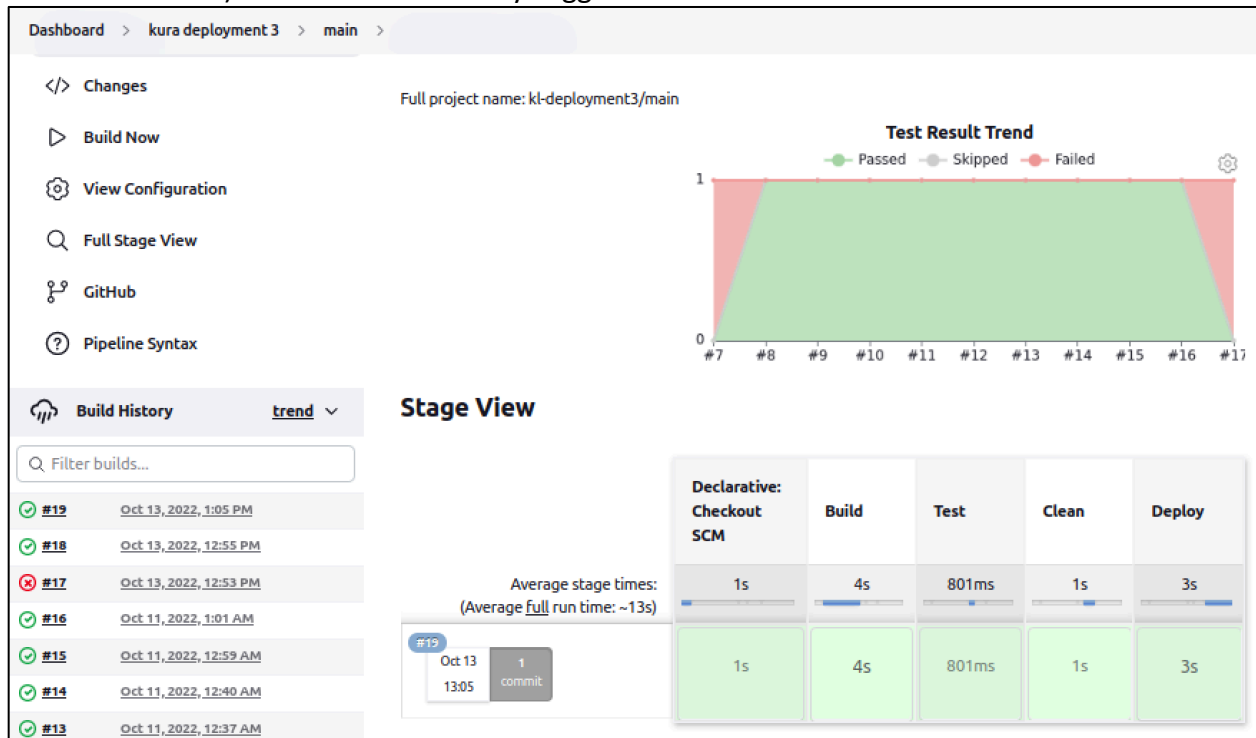
- On GitHub, go to the project repository and edit the Jenkinsfile to include Slack notifications based on build status on each run:

```
kl_wk14_deployment3 / Jenkinsfile in main Cancel changes
<> Edit file Preview changes Spaces 2 No wrap
44 agent{label 'awsDeploy'}
45 steps {
46   sh '''#!/bin/bash
47   if [[ $(ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut -d " " -f 2) != 0 ]]
48   then
49     ps aux | grep -i "gunicorn" | tr -s " " | head -n 1 | cut -d " " -f 2 > pid.txt
50     kill $(cat pid.txt)
51     exit 0
52   fi
53   '''
54 }
55 post {
56   success {
57     slackSend (message: "INFO: Build Number ${env.BUILD_NUMBER} - ${STAGE_NAME} Stage completed successfully!")
58   }
59   failure {
60     slackSend (message: "WARNING: Build Number ${env.BUILD_NUMBER} - ${STAGE_NAME} Stage has failed!")
61   }
62 }
63 }
64 stage ('Deploy') {
65   agent{label 'awsDeploy'}
66   steps {
67     keepRunning {
68       sh '''#!/bin/bash
69       pip install -r requirements.txt
70       pip install gunicorn
71       python3 -m gunicorn -w 4 application:app -b 0.0.0.0 --daemon
72       '''
73     }
74   }
75   post {
76     success {
77       slackSend (message: "INFO: Build Number ${env.BUILD_NUMBER} - ${STAGE_NAME} Stage completed successfully!")
78     }
79     failure {
```

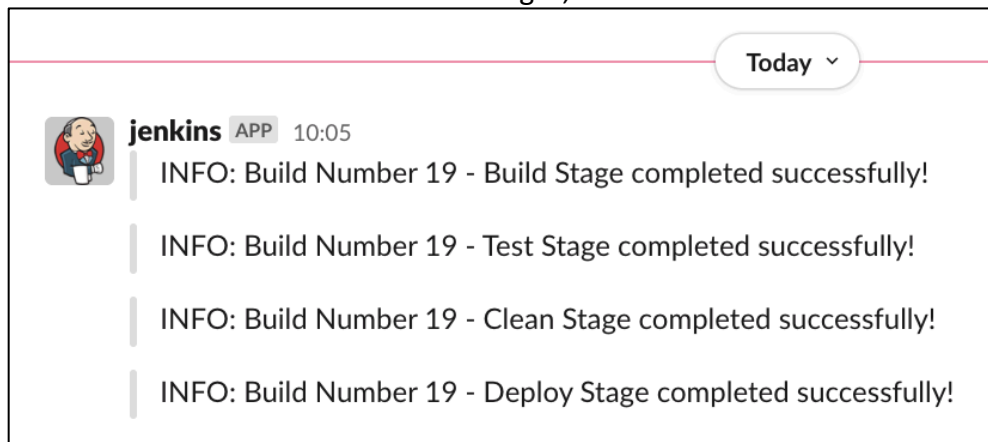
Deployment 3 Documentation

Caden Hong

- As soon as the change is committed, Jenkins will detect the changes (thanks to the webhook) and will automatically trigger a new build:



- As Jenkins builds each of the stages, a Slack notification will be sent as below:



Software Stack

My guess of how the **nginx-gunicorn-Flask** stack works:

