



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY

DevOps & its Applications (CS457)

Jenkins Assignment 2

Under the Guidance of - Dr. Uma S

Submitted by -

18BCS006-Anjuru Lokesh

18BCS021-Avinash c

18BCS025-Diddi Geethakrishna

18BCS028-Godina Pranav

18BCS035-Kancharla Gireesh Kumar

18BCS070-Pravalika

JENKINS MASTER - SLAVE PIPELINE

Tech Stack:

1. Git
2. Github
3. Docker
4. Jenkins
5. Three AWS EC2 Instances (t2.micro)

SETTING UP JENKINS MASTER

```
~ master*  
↳ cd Downloads  
~/Downloads master*  
↳ chmod 400 jenkins.cer  
~/Downloads master*  
↳ ssh -i "jenkins.cer" ubuntu@ec2-13-234-21-184.ap-south-1.compute.amazonaws.com  
  
The authenticity of host 'ec2-13-234-21-184.ap-south-1.compute.amazonaws.com (13.234.21.184)' can't be established.  
ED25519 key fingerprint is SHA256:4y7hCFe6kGVhK0bvY/P5tTXWvX0LA2Gm5UhI0tbsxbY.  
This host key is known by the following other names/addresses:  
    ~/.ssh/known_hosts:9: ec2-3-108-252-164.ap-south-1.compute.amazonaws.com  
    ~/.ssh/known_hosts:12: ec2-13-233-152-72.ap-south-1.compute.amazonaws.com  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-13-234-21-184.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1059-aws x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
  
System information as of Sun Nov 14 16:37:32 UTC 2021  
  
System load: 0.0 Processes: 97  
Usage of '/': 30.3% of 7.69GB Users logged in: 0  
Memory usage: 46% IP address for eth0: 172.31.1.75  
Swap usage: 0%  
  
* Ubuntu Pro delivers the most comprehensive open source security and  
  compliance features.  
  
https://ubuntu.com/aws/pro  
  
3 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Last login: Sun Nov 14 13:46:52 2021 from 157.50.10.204
```

Step1: Install Jenkins on an EC2 instance with the commands mentioned below in the screenshot.

```
ubuntu@ip-172-31-1-75:~$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  daemon
The following NEW packages will be installed:
  daemon jenkins
0 upgraded, 2 newly installed, 0 to remove and 19 not upgraded.
Need to get 72.0 MB of archives.
After this operation, 72.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 daemon amd64 0.6.4-1build1 [99.5 KB]
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.303.3 [71.9 MB]
Fetched 72.0 MB in 2min 28s (486 kB/s)
Selecting previously unselected package daemon.
(Reading database ... 73585 files and directories currently installed.)
Preparing to unpack .../daemon_0.6.4-1build1_amd64.deb ...
Unpacking daemon (0.6.4-1build1) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.303.3_all.deb ...
Unpacking jenkins (2.303.3) ...
Setting up daemon (0.6.4-1build1) ...
Setting up jenkins (2.303.3) ...
Processing triggers for systemd (237-3ubuntu10.52) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
```

Step2: After installation, we can see the status of the jenkins service. It should be “active” as mentioned in the screenshot below.

```
ubuntu@ip-172-31-1-75:~$ service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
  Active: active (exited) since Sat 2021-11-13 14:31:53 UTC; 26s ago
    Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 1140)
   CGroup: /system.slice/jenkins.service

Nov 13 14:31:51 ip-172-31-1-75 systemd[1]: Starting LSB: Start Jenkins at boot time...
Nov 13 14:31:52 ip-172-31-1-75 jenkins[19056]: Correct java version found
Nov 13 14:31:52 ip-172-31-1-75 jenkins[19056]: * Starting Jenkins Automation Server jenkins
Nov 13 14:31:52 ip-172-31-1-75 su[19106]: Successful su for jenkins by root
Nov 13 14:31:52 ip-172-31-1-75 su[19106]: + ??? root:jenkins
Nov 13 14:31:52 ip-172-31-1-75 su[19106]: pam_unix(su:session): session opened for user jenkins by (uid=0)
Nov 13 14:31:52 ip-172-31-1-75 su[19106]: pam_unix(su:session): session closed for user jenkins
Nov 13 14:31:53 ip-172-31-1-75 jenkins[19056]:     ...done.
Nov 13 14:31:53 ip-172-31-1-75 systemd[1]: Started LSB: Start Jenkins at boot time.
```

Step3: Now enter the public IP address of the EC2 instance followed by port number 8080 (by default, Jenkins runs on port 8080) in a browser to access the Jenkins Dashboard. Here, it will prompt you to enter the Admin Password to unlock Jenkins. The Admin Password can be found in the file whose absolute path will be mentioned in the prompt. Read the content of the file as mentioned in the screenshot below, and enter that in the Admin Password field.

```
ubuntu@ip-172-31-1-75:~$ cat /var/lib/jenkins/secrets/initialAdminPassword
cat: /var/lib/jenkins/secrets/initialAdminPassword: Permission denied
ubuntu@ip-172-31-1-75:~$ sudo cat
^Z
[1]+  Stopped                  sudo cat
ubuntu@ip-172-31-1-75:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
7357bfe6036d4281b0f4cae13ff90cc
```

Step4: After entering the correct password, Jenkins will prompt to create a new user by entering username, password, and email. Enter the same and login.

Step5: Name the other two EC2 instances as 'slave1' and 'slave2'

Go to jenkins -> manage jenkins -> Configure global security. In agents, change TCP port for inbound agents to random and save.

The screenshot shows the Jenkins 'Configure Global Security' configuration page. The 'Agents' section is the active tab, displaying the 'TCP port for inbound agents' setting. The 'Random' option is selected, while 'Fixed' and 'Disable' are unselected. Below this, there is a 'Agent protocols...' button. The 'Markup Formatter' section contains a dropdown menu set to 'Plain text'. The 'CSRF Protection' section includes a 'Crumb Issuer' dropdown set to 'Default Crumb Issuer' and an unchecked 'Enable proxy compatibility' checkbox. At the bottom of the page are 'Save' and 'Apply' buttons.

Step6: Connect slave1 and slave2 nodes to Jenkins master.

Go to Manage Jenkins -> Manage Nodes and Clouds -> New Node.

Now enter the name as slave1/slave2 and enable 'Permanent Agent'. Then click 'OK'.

Step7: Now configure the created Nodes and enter '/home/ubuntu/jenkins' in the 'Remote root directory' and select 'connect via SSH'. Add ip address of respective instance of the node and enter the credentials as mentioned in the screenshot below and click Save.

Training set | Inbox (1,13) | How to Inst | Instance de | (7) WhatsA | DevOps-Tr | pravalika2E | pravalika2E | slave1 C | Jenkins Pro | Jenkins Pro | Jenkins As | Jenkins As | +

Not Secure | 13.233.27.70:8080/computer/slave1/configure

Dashboard > Nodes > slave1

slave1

Status

Delete Agent

Configure

Build History

Load Statistics

Log

Build Executor Status

slave1

Description

Number of executors

1

Remote root directory

/home/ubuntu/jenkins

Labels

Usage

Use this node as much as possible

Launch method

Launch agents via SSH

Host

65.0.29.114

Credentials

ubuntu

Add

Host Key Verification Strategy

Save

This is how the ‘Manage Nodes’ dashboard looks like:

Training set | Inbox (1,13) | How to Inst | Instance de | (7) WhatsA | DevOps-Tr | pravalika2E | pravalika2E | Nodes | Jenkins Pro | Jenkins Pro | Jenkins As | Jenkins As | +

Not Secure | 13.233.27.70:8080/computer/

Jenkins

search

Dashboard > Nodes

Back to Dashboard

Manage Jenkins

New Node

Configure Clouds

Node Monitoring

Build Queue

No builds in the queue.

Build Executor Status

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	master	Linux (amd64)	In sync	5.29 GB	0 B	5.29 GB	0ms
2	slave1		N/A	N/A	N/A	N/A	N/A
3	slave2		N/A	N/A	N/A	N/A	N/A

last checked 5 min 0 sec 5 min 0 sec

Refresh status

REST API Jenkins 2.303.3

Step8: Go to master EC2 instance and run the following commands

- **ssh-keygen -t rsa**
Generates the key pair
- **cd ~/.ssh**
- **ls**
- **cat id_rsa.pub**
This is the public key

Go to both slave1 and slave2 Ec2 instances and run the command

- **echo "Public key">>>authorized_keys**
- **sudo chmod 777 -R jenkins**
- **Cd ~/.ssh**
- **ls**
- **cat id_rsa**

The private key is copied and added in the credentials of slave1, slave2 configuration.

Agent is successfully connected.

```
BASH_SOURCE<()
BASH_VERSION=5.0.17(1)-release
Bash-Completion_BASH_ADDRESS=/run/user/1000/bus
DIRSTACK[1]
EUID=1000
GROUPS+=()
HOSTNAME=ubuntu
HOSTNAME_IP=172.31.37.158
HOSTTYPE=x86_64
IFS=$' \n'
LINES=24
LOGNAME=ubuntu
MACHTYPE=x86_64-pc-linux-gnu
MOTD_SHELL=/bin/pam
OPTERR=1
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/usr/games:/usr/bin
PFILEID=1344
PPID=1344
PS4='* '
PWD=/home/ubuntu
SHLLINE=/bin/bash
SHLLINE_OPTS+=braceexpand:hashall:interactive-comments
SHLLVL=1
SSH_CLIENT=13.214.21.184 42014 22
SSH_CONNECTION=13.234.21.184 42014 172.31.37.158 22
TERM=dumb
UID=1000
USER=root
XDG_RUNTIME_DIR=/run/user/1000
XDG_SESSION_CLASS=user
XDG_SESSION_ID=8
XDG_SESSION_TYPE=tty
```
Checking Java version in the PATH
openjdk version "1.8.0_292"
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
[11/14/21 17:14:41] [SSH] Checking java version of /home/ubuntu/jenkins/jdk/bin/java
Couldn't figure out Java version of /home/ubuntu/jenkins/jdk/bin/java
bash: /home/ubuntu/jenkins/jdk/bin/java: No such file or directory
[11/14/21 17:14:41] [SSH] Copying latest remoting.jar...
[11/14/21 17:14:41] [SSH] Copied 1,507,813 bytes.
Expected the checksum value to be 0d
[11/14/21 17:14:41] [SSH] Starting agent process: cd "/home/ubuntu/jenkins" &> java -jar remoting.jar -workDir /home/ubuntu/jenkins -jar-cache /home/ubuntu/jenkins/remoting/jarCache
Nov 14, 2021 5:14:41 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remote working directory
Nov 14, 2021 5:14:41 PM org.jenkinsci.remoting.engine.WorkDirManager setupUpLogging
INFO: Logging output logs will be printed to /home/ubuntu/jenkins/remoting
<==[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 4.10.1
This is a Unix agent
Executed without errors
Agent successfully connected and online
`
```

REST API Jenkins 2.303.3

Both the slave nodes will now be in sync with Jenkins master.

| S | Name   | Architecture  | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|--------|---------------|------------------|-----------------|-----------------|-----------------|---------------|
|   | master | Linux (amd64) | In sync          | 5.29 GB         | 0 B             | 5.29 GB         | 0ms           |
|   | slave1 | Linux (amd64) | In sync          | 4.72 GB         | 0 B             | 4.72 GB         | 30ms          |
|   | slave2 | Linux (amd64) | In sync          | 4.49 GB         | 0 B             | 4.49 GB         | 28ms          |

## Step9: Node install JDK on both the slave nodes.

## Step10: Now install docker on both the slave nodes.

```
Last login: Sun Nov 14 17:26:06 2021 from 157.58.10.204
[ubuntu@172-31-36-168: ~]$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
 bridge-utils contained dns-root-data dnsmasq-base libidn11 pigz runc
 ubuntu-fan
Suggested packages:
 ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc
 rsync rfs-fuse | zfsutils
The following NEW packages will be installed:
 bridge-utils contained dns-root-data dnsmasq-base docker.io libidn11 pigz
 runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 24 not upgraded.
Need to get 74.5 MB of archives.
After this operation, 361 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu2 [20.5 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1.0.1-0ubuntu2+20.04.1 [4155 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 containedr amd64 1.5.5-0ubuntu3+20.04.1 [33.0 MB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 dns-root-data all 2019052802 [5308 B]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main libidn11 amd64 1.33-2.2ubuntu2 [46.2 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 dnsmasq-base amd64 2.88-1.1ubuntu1.4 [315 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/universe amd64 docker.io amd64 20.10.7-0ubuntu5+20.04.2 [36.9 MB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 ubuntu-fan all 0.12.13 [34.5 kB]
Fetched 74.5 MB in 18s (4144 kB/s)
Preconfiguring packages...
Selecting previously unselected package pigz.
(Reading database ... 79377 files and directories currently installed.)
Preparing to unpack .../pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../bridge-utils_1.6-2ubuntu1_amd64.deb ...
Unpacking bridge-utils (1.6-2ubuntu1) ...
Selecting previously unselected package runc.
Preparing to unpack .../runc_1.0.1-0ubuntu2+20.04.1_amd64.deb ...
Unpacking runc (1.0.1-0ubuntu2+20.04.1) ...
Selecting previously unselected package libidn11:amd64.
Preparing to unpack .../libidn11:amd64_1.33-2.2ubuntu2_amd64.deb ...
Unpacking libidn11:amd64 (1.33-2.2ubuntu2) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../dnsmasq-base_2.88-1.1ubuntu1.4_amd64.deb ...
Unpacking dnsmasq-base (2.88-1.1ubuntu1.4) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../docker.io_20.10.7-0ubuntu5_amd64.deb ...
Unpacking docker.io (20.10.7-0ubuntu5) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../ubuntu-fan_0.12.13_all.deb ...
Unpacking ubuntu-fan (0.12.13) ...
Setting up runc (1.0.1-0ubuntu2+20.04.1) ...
Setting up dns-root-data (2019052802) ...
Setting up libidn11:amd64 (1.33-2.2ubuntu2) ...
Setting up bridge-utils (1.6-2ubuntu1) ...
Setting up pigz (2.4-1) ...
Setting up containerd (1.5.5-0ubuntu3+20.04.1) ...
```

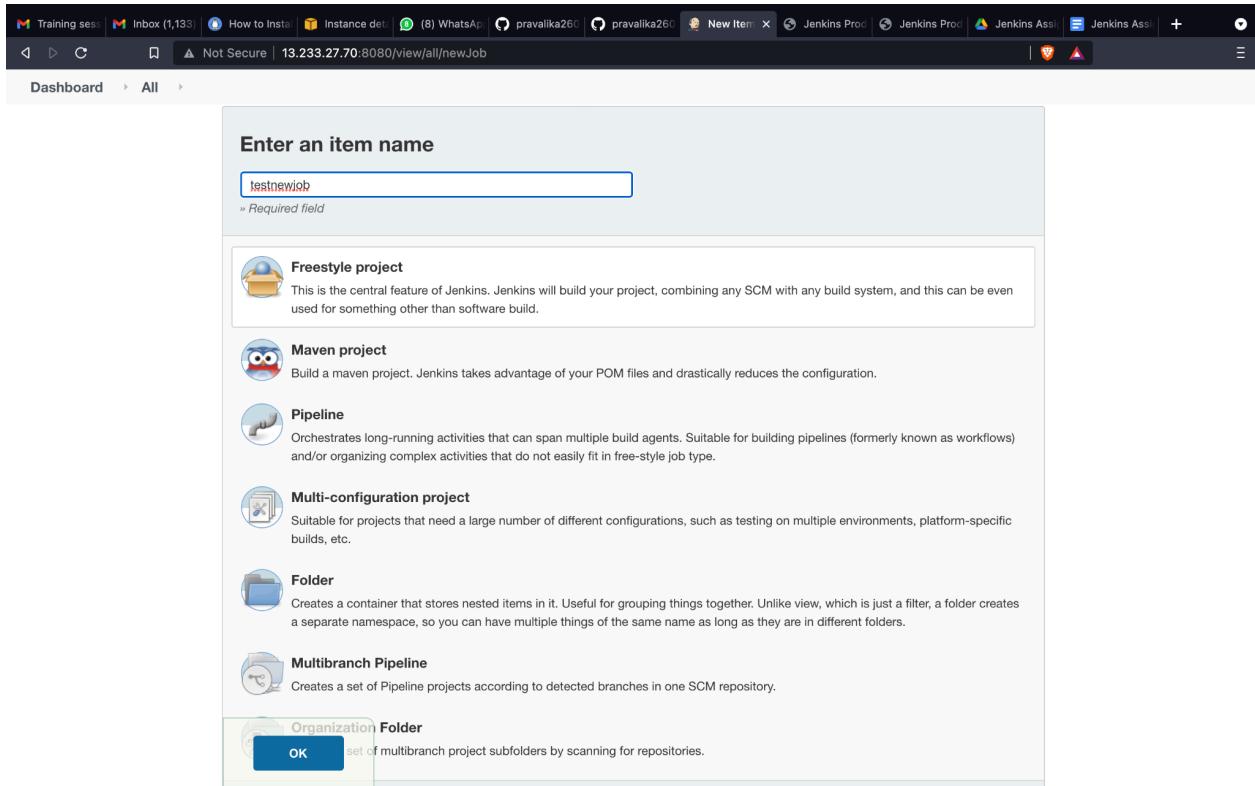
## CI/CD JENKINS PIPELINE

We are treating slave1 as a testing server and slave2 as production server.

Configuring CICD pipeline in the way

- Configure webhook that triggers the slave1 job.
- When the slave1 job is successful, only then slave2 job is called.

**Step11:** In Jenkins dashboard, create a new Job by going to Jenkins Dashboard ⇒ Create New Job ⇒ enter Job Name ⇒ select Freestyle Project ⇒ click OK.



**Step12:** Now configure the new job by entering the details as follows:

- GitHub Project URL - git@github.com:pravalika2604/devopsIQ.git
- Select 'Restrict where this project can be run' and enter the appropriate node name.
- In 'Source Code Management' select Git and enter the repository URL mentioned above.
- Select 'GitHub hook trigger for GITScm polling' under 'Build Triggers'. (This step is only for the testing server).
- Under 'Build' select 'Execute shell' and enter the commands as shown in the screenshot below.
- Under 'Post-build Actions' select 'Build other projects' and enter the prod node name. Then select 'Trigger only if build is stable'. (This step should be done only on the testing server).
- Click Save.

This step should be done for both testing and production servers.

The screenshot shows the Jenkins job configuration for the 'Prod' job. The 'General' tab is selected, displaying a GitHub webhook URL: `https://github.com/pravalika26/Prod/pull/1`. Below the URL, there is a note: 'This build requires Jenkins resources' and 'This project is permanent'. The 'Build Triggers' tab is also visible, showing a checked checkbox for 'GitHub hook trigger for GITScm polling'.

The screenshot shows the Jenkins job configuration for the 'Test' job. The 'Build Triggers' tab is selected, showing a checked checkbox for 'GitHub hook trigger for GITScm polling'. The 'Build Environment' section contains several build-related checkboxes. The 'Build' section includes an 'Execute shell' step with the command: `sudo docker rm -f $(sudo docker ps -a -q); sudo docker run -it -p 82:80 -d test`. The 'Post-build Actions' section contains a 'Build other projects' step for the 'Prod' project, with the 'Trigger only if build is stable' option selected. At the bottom, there are 'Save' and 'Apply' buttons.

**Step13:** Now set up the Pipeline view. First install the ‘build pipeline’ plugin in Jenkins. Then go to Home and select the ‘+’ icon beside ‘All’. Then select ‘Build Pipeline View’ and give a name to the view (CICD here). Then click on OK.

**Step14:** Now under ‘Pipeline Flow’, select the initial job as ‘Test’. Then click on OK.

**Step15:** Now open the GitHub repository and go to ‘Settings’. Then select ‘Webhooks’. Click on ‘Add webhook’. Then enter the ‘Payload URL’ as `'http://{ip-address-of-the-Jenkins-master-node}:8080/github-webhook/'` and select ‘Just the push event’ and then click on ‘Add webhook’.

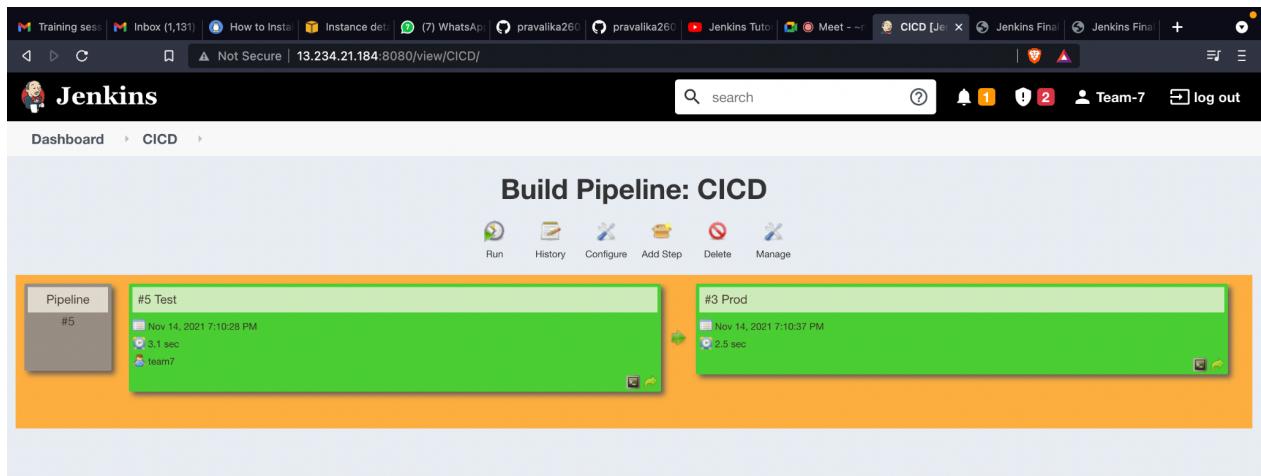
The screenshot shows the GitHub settings interface for managing webhooks. On the left, a sidebar menu lists various options like Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Webhooks' option is selected. The main content area is titled 'Webhooks / Manage webhook'. It contains a 'Settings' tab and a 'Recent Deliveries' tab. A descriptive text explains that GitHub will send a POST request to the specified URL with event details. Below this, the 'Payload URL' is set to 'http://13.234.21.184:8080/github-webhook/'. The 'Content type' is set to 'application/x-www-form-urlencoded'. There is a 'Secret' field which is currently empty. Under 'Which events would you like to trigger this webhook?', the radio button for 'Just the push event.' is selected. Below this, there is a checked checkbox for 'Active' with the note 'We will deliver event details when this hook is triggered.' At the bottom are two buttons: 'Update webhook' (green) and 'Delete webhook' (red).

After successful ping, the webhook screen will look as follows:

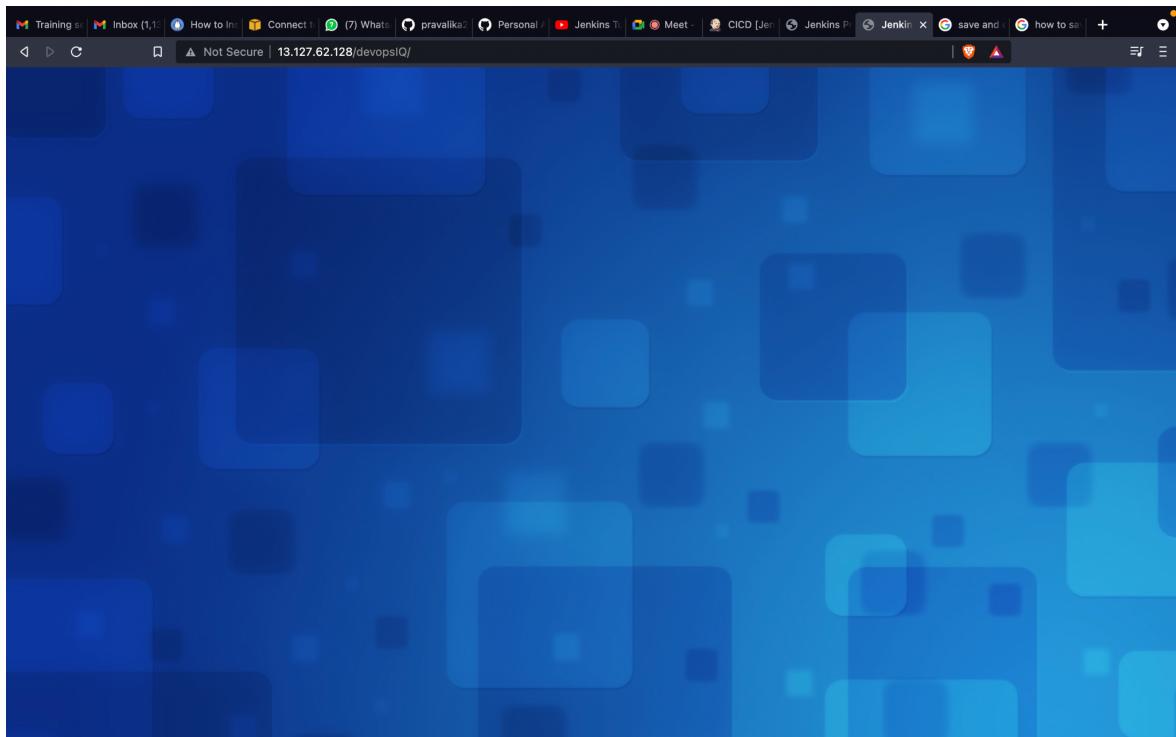
The screenshot shows the GitHub settings interface for managing webhooks. The sidebar menu is identical to the previous screenshot. The main content area is titled 'Webhooks' and includes a 'Add webhook' button. A descriptive text explains that webhooks allow external services to be notified of certain events. Below this, a list shows one active webhook entry: 'http://13.234.21.184:8080/github-webhook...' (push). To the right of this entry are 'Edit' and 'Delete' buttons. The URL is preceded by a green checkmark.

**Step16:** Now commit changes and push to the repository to trigger the webhook. This will trigger the Jenkins CICD pipeline and build the docker image on the test server first, and on successful build, it will build the same on the production server as well.

CICD pipeline looks as follows:

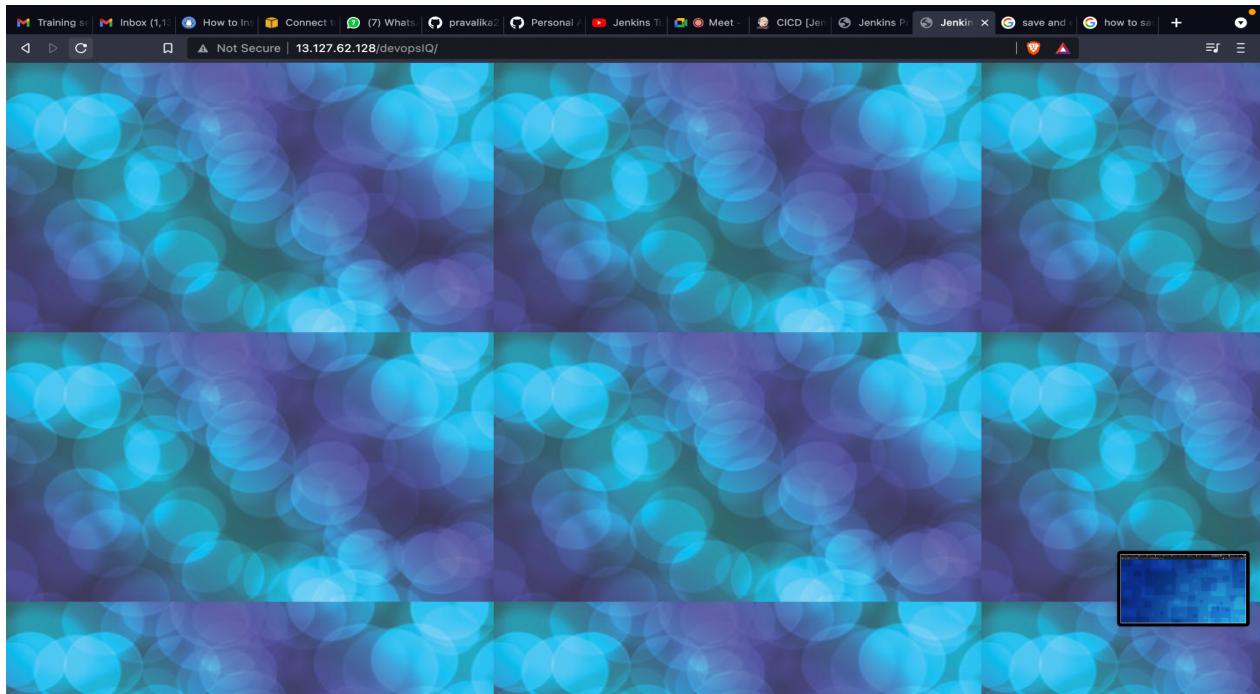


On successful build, the deployed Node application looks as follows on both test and prod servers:



**Step17:** Here we notice that the Website name is changed. And also the image is changed. We made the changes in the code and pushed it to the GitHub repository. This triggered the pipeline and changes reflect in the deployed website as shown below:

The screenshot shows the Jenkins Build Pipeline interface for a project named 'CICD'. At the top, there are navigation links for 'Dashboard' and 'CICD'. Below the header, the title 'Build Pipeline: CICD' is displayed, along with icons for 'Run', 'History', 'Configure', 'Add Step', 'Delete', and 'Manage'. Two pipeline items are listed: '#6 Test' (status: green, duration: 3.1 sec) and '#4 Prod' (status: green, duration: 2.2 sec). The Jenkins version 'Jenkins 2.303.3' is visible at the bottom right.



**Changes reflected. Hence, Jenkins master slave pipeline has been successful.**