



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY

DevOps & its Applications (CS457)

Jenkins Assignment 2

Under the Guidance of - Dr. Uma S

Submitted by -

Advay Aggarwal (18BCS002)

Meghna Hadimani (18BCS052)

Perumalla Tushar (18BCS065)

Rahul Priyadarshi (18BCS074)

Samana B S (18BCS088)

JENKINS MASTER - SLAVE PIPELINE

Tech Stack:

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

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

```
Processing triggers for dbus (1.12.16-2ubuntu2.1) ...
[ubuntu@ip-172-31-42-133:~/jenkins$ sudo apt install openjdk-8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  adwaita-icon-theme at-spi2-core ca-certificates-java fontconfig fontconfig-config fonts-dejavu-core fonts-dejavu-extra gtk+
  libatk-wrapper-java libatk-wrapper-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data 1
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0+dfsg-3ubuntu0.2) ...
[ubuntu@ip-172-31-42-133:~/jenkins$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
OK
[ubuntu@ip-172-31-42-133:~/jenkins$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
[ubuntu@ip-172-31-42-133:~/jenkins$ sudo apt update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:7 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:8 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:6 https://packages.cloud.google.com/apt/kubernetes-xenial InRelease
Get:9 https://pkg.jenkins.io/debian-stable binary/ Packages [20.9 kB]
Fetched 23.8 kB in 1s (40.0 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
[ubuntu@ip-172-31-42-133:~/jenkins$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  daemon net-tools
The following NEW packages will be installed:
  daemon jenkins net-tools
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 72.2 MB of archives.
After this operation, 73.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 daemon amd64 0.6.4-1build2 [96.3 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/main amd64 net-tools amd64 1.60+git20180626.acehd88e-1ubuntu1 [196 kB]
```

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

```
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.13) ...
[ubuntu@ip-172-31-42-133:~/jenkins$ service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
  Active: active (exited) since Sun 2021-11-14 07:00:58 UTC; 3min 5s ago
    Docs: man:systemd-sysv-generator(8)
   Tasks: 0 (limit: 1147)
  Memory: 0B
  CGroup: /system.slice/jenkins.service

Nov 14 07:00:56 ip-172-31-42-133 systemd[1]: Starting LSB: Start Jenkins at boot time...
Nov 14 07:00:57 ip-172-31-42-133 jenkins[7480]: Correct java version found
Nov 14 07:00:57 ip-172-31-42-133 jenkins[7480]: * Starting Jenkins Automation Server jenkins
Nov 14 07:00:57 ip-172-31-42-133 su[7513]: (to jenkins) root on none
Nov 14 07:00:57 ip-172-31-42-133 su[7513]: pam_unix(su-l:session): session opened for user jenkins by (uid=0)
Nov 14 07:00:57 ip-172-31-42-133 su[7513]: pam_unix(su-l:session): session closed for user jenkins
Nov 14 07:00:58 ip-172-31-42-133 jenkins[7480]: ...done.
Nov 14 07:00:58 ip-172-31-42-133 systemd[1]: Started LSB: Start Jenkins at boot time.
[ubuntu@ip-172-31-42-133:~/jenkins$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

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 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.

```
Nov 14 07:00:58 ip-172-31-42-133 jenkins[7480]: ...done.
Nov 14 07:00:58 ip-172-31-42-133 systemd[1]: Started LSB: Start Jenkins at boot time.
[ubuntu@ip-172-31-42-133:~/jenkins$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
e18a915459194909a19639fd0fa3179f
[ubuntu@ip-172-31-42-133:~/jenkins$ client loop: send disconnect: Broken pipe
```

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.

Jenkins is now ready to use.

Step5: Name the other two EC2 instances as ‘slave1’ and ‘slave2’ to be set-up as test and production servers. Then, from the Jenkins Dashboard, go to Manage Jenkins ⇒ Configure global security. In ‘Agents’, change the ‘TCP port for inbound agents’ to ‘Random’, and click ‘Save’.

The screenshot shows the Jenkins 'Configure Global Security' configuration page. The 'Agents' section is highlighted, specifically the 'TCP port for inbound agents' setting. The 'Random' radio button is selected, and the 'Save' button is visible at the bottom of the form.

Logged-in users can do anything

- Allow anonymous read access
- Matrix-based security
- Project-based Matrix Authorization Strategy

Markup Formatter

Markup Formatter

Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

Agents

TCP port for inbound agents

Fixed: Random Disable

Agent protocols...

CSRF Protection

Crumb Issuer

Default Crumb Issuer

Enable proxy compatibility

Hidden security warnings

Security warnings...

Buttons

Save Apply

Step6: Add 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’.

The screenshot shows the Jenkins 'Nodes' configuration page. On the left, there's a sidebar with links like 'Back to Dashboard', 'Manage Jenkins', 'New Node' (which is selected and highlighted in blue), 'Configure Clouds', and 'Node Monitoring'. The main content area has a 'Node name' input field containing 'slave-1'. Below it is a radio button group where 'Permanent Agent' is selected (indicated by a blue outline). A descriptive text explains that this adds a plain, permanent agent to Jenkins. There's also a 'Copy Existing Node' option with a 'Copy from' input field and an 'OK' button at the bottom. On the right side of the page, under 'Build Executor Status', there are sections for 'master' (with 1 idle executor) and 'slave1' (with 1 idle executor), and 'slave2' (with 1 idle executor).

Step7: Now configure the created Nodes and enter '/home/ubuntu/jenkins' in the 'Remote root directory' as mentioned in the screenshot below and click Save.

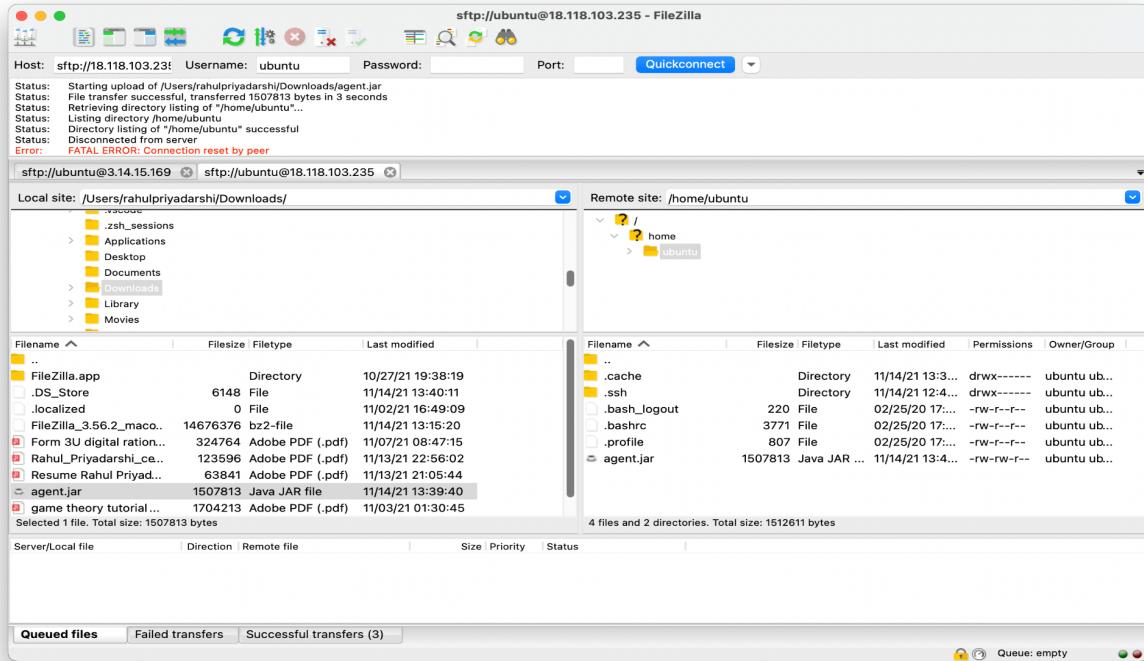
The screenshot shows the Jenkins 'Configure Node' page for a node named 'slave1'. The 'Remote root directory' field is highlighted with a blue border, containing the value '/home/ubuntu/jenkins'. Other fields include 'Name' (slave1), 'Description' (empty), 'Number of executors' (1), and 'Labels' (empty). The 'Usage' dropdown is set to 'Use this node as much as possible'. The 'Launch method' dropdown is set to 'Launch agent by connecting it to the master', with 'Custom WorkDir path' checked. A 'Save' button is at the bottom.

This is how the 'Manage Nodes' dashboard looks like:

The screenshot shows the Jenkins 'Manage Nodes' dashboard. It lists three nodes: 'master' (Linux (amd64)), 'slave1' (Linux (amd64)), and 'slave2' (Linux (amd64)). The 'Build Queue' section indicates no builds in the queue. The 'Build Executor Status' section shows the status of executors on each node. A 'Refresh status' button is located at the bottom right of the main table.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	4.54 GB	0 B	4.54 GB	0ms
	slave1	Linux (amd64)	In sync	2.95 GB	0 B	2.95 GB	7ms
	slave2	Linux (amd64)	In sync	2.75 GB	0 B	2.75 GB	6ms

Step8: Go to slave1 and download the agent.jar file. Now open FileZilla application and connect it to slave1 EC2 node. Then transfer the agent.jar file to the node using SFTP.



Repeat the step for both the nodes.

Step9: Node install JDK on both the slave nodes.

```
Found linux image: /boot/vmlinuz-5.11.0-1020-aws
Found initrd image: /boot/microcode.cpio /boot/initrd.img-5.11.0-1020-aws
Found Ubuntu 20.04.3 LTS (20.04) on /dev/xvda1
done
[ubuntu@ip-172-31-43-2:~$ sudo apt install openjdk-8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
adwaita-icon-theme at-spi2-core ca-certificates-java fontconfig fontconfig-
libasyncns0 libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-jni
libcups2 libdatriel libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-rad
libgdk-pixbuf2.0-common libgif7 libgl1 libgl1-mesa-dri libgl1-mesa-glx libg
libice6 libjbig0 libjpeg-turbo8 libjpeg8 liblcms2-2 liblvm12 libpango-1.0-
librsvg2-common libsensors-config libsensors5 libsm-dev libsm6 libsndfile1
libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxc
libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 lib
openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless ubuntu-mono x11
Suggested packages:
default-jre cups-common gvfs libice-doc liblcms2-utils pcscd pulseaudio lib
libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei fo
The following NEW packages will be installed:
adwaita-icon-theme at-spi2-core ca-certificates-java fontconfig fontconfig-
libasyncns0 libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-jni
libcups2 libdatriel libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-rad
```

Step10: Now copy the command from the slave nodes on Jenkins, and execute them on the EC2 instances as shown below, This should be done for both the nodes.

```
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0-0+dfsg-Subuntu0.2) ...
[ubuntu@ip-172-31-43-2: ~]$ java -jar agent.jar -jnlpUrl http://18.219.223.11:8080/computer/slave2/jenkins-agent.jnlp -secret d4176c315e2adc68332ee61a139fff968d43a06418690c47b6a8a0edfaab6c10 -workDir "/home/
ubuntu/jenkins"
Nov 14, 2021 8:21:31 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Nov 14, 2021 8:21:31 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/jenkins/remoting
Nov 14, 2021 8:21:31 AM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: slave2
Nov 14, 2021 8:21:31 AM hudson.remoting.jnlp.Main$CuiListener <init>
INFO: Jenkins agent is running in headless mode.
Nov 14, 2021 8:21:31 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 4.10.1
Nov 14, 2021 8:21:31 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Nov 14, 2021 8:21:31 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://18.219.223.11:8080]
Nov 14, 2021 8:21:31 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Nov 14, 2021 8:21:31 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Agent discovery successful
  Agent address: 18.219.223.11
  Agent port: 34293
  Identity: 0e:3e:1a:f6:5acd:67:5e:2a:72:62:2e:56:83:0d
Nov 14, 2021 8:21:31 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Nov 14, 2021 8:21:31 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 18.219.223.11:34293
Nov 14, 2021 8:21:31 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP4-connect
Nov 14, 2021 8:21:32 AM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Nov 14, 2021 8:21:32 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Remote identity confirmed: 0e:3e:1a:f6:5acd:67:5e:2a:72:62:2e:56:83:0d
Nov 14, 2021 8:21:33 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

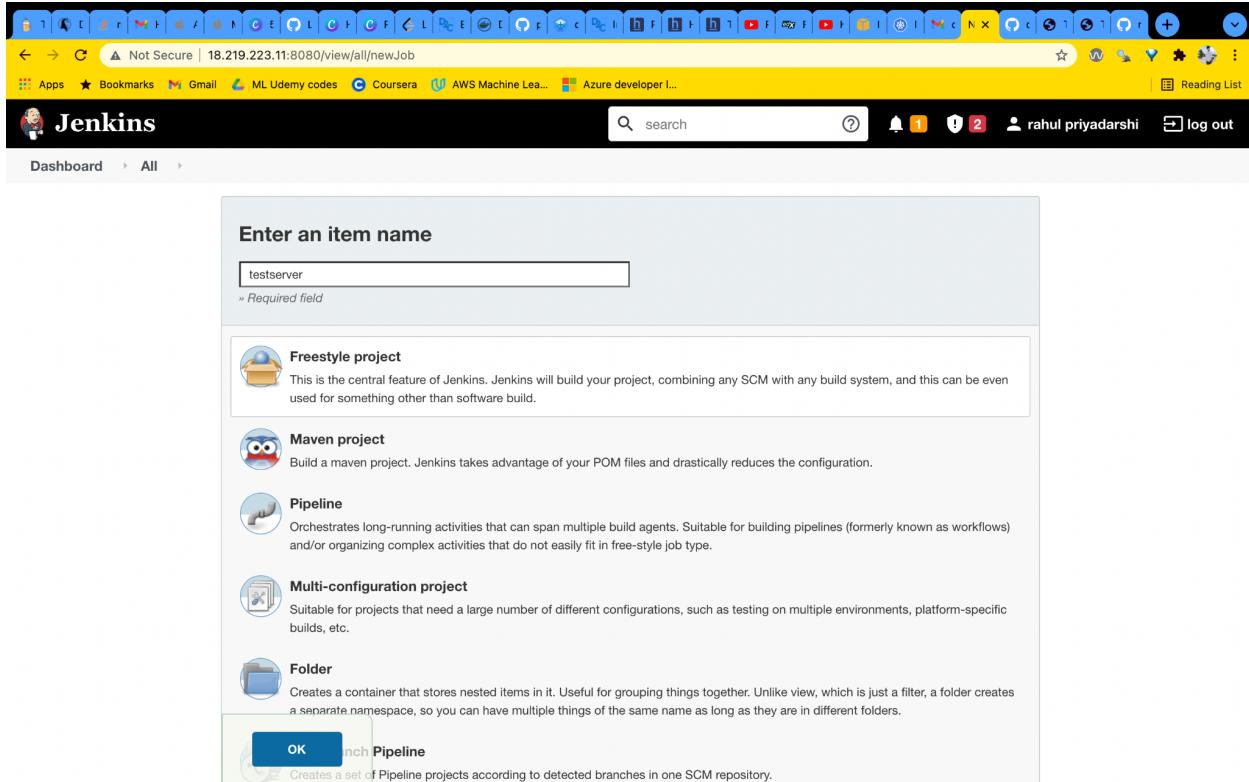
Both the slave nodes will now be in sync with Jenkins master.

Step11: Now install docker on both the slave nodes.

```
-/Downloads - ubuntu@ip-172-31-42-133: ~ -- zsh
[detached from 29081:jenkins_agent]
[ubuntu@ip-172-31-43-2: ~] sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc ubuntu-fan
Suggested packages:
ifupdown-tools cgroups-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 0 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://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu1 [30.5 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1.0.1-0ubuntu2~20.04.1 [4155 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 containerd amd64 1.5.5~Ubuntu3~20.04.1 [33.0 MB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/main amd64 dns-root-data all 2019052802 [5300 B]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/main amd64 libidn11 amd64 1.33-2.2ubuntu2 [46.2 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 dnsmasq-base amd64 2.80-1.lubuntu1.4 [315 kB]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates/universe amd64 docker.io amd64 20.10.7~Ubuntu5~20.04.2 [36.9 MB]
Get:9 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/main amd64 ubuntu-fan all 0.12.13 [34.5 kB]
Fetched 74.5 MB in 1s (55.4 kB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 110915 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.6-2ubuntu1_amd64.deb ...

```

Step12: In Jenkins dashboard, create a new Job by going to Jenkins Dashboard ⇒ Create New Job ⇒ enter Job Name ⇒ select Freestyle Project ⇒ click OK.



Step13: Now configure the new job by entering the details as follows:

- GitHub Project URL - <https://github.com/rahulp99/node-app.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).

- e. Under 'Build' select 'Execute shell' and enter the commands as shown in the screenshot below.
- f. Under 'Post-build Actions' select 'Build other projects' and enter the production node name. Then select 'Trigger only if build is stable'. (This step should be done only on the testing server).
- g. Click Save.

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

The screenshot shows the Jenkins configuration interface for a job named 'Test'. The 'General' tab is selected. In the 'GitHub project' section, the 'Project url' is set to 'https://github.com/rahulp99/node-app.git'. Under the 'Advanced...' button, the 'Restrict where this project can be run' checkbox is checked, and the label 'slave1' is entered in the 'Label Expression' field. At the bottom, there are 'Save' and 'Apply' buttons. A note at the bottom right states: 'Permissions or other restrictions provided by plugins may further reduce that list.'

General

- Restrict where this project can be run
- Label Expression: slave1
- Label slave1 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Source Code Management

- None
- Git
- Repositories
- Repository URL: https://github.com/rahulp99/node-app.git
- Credentials: - none -

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Build Environment

- Delete workspace before build starts
- Use secret text(s) or file(s)

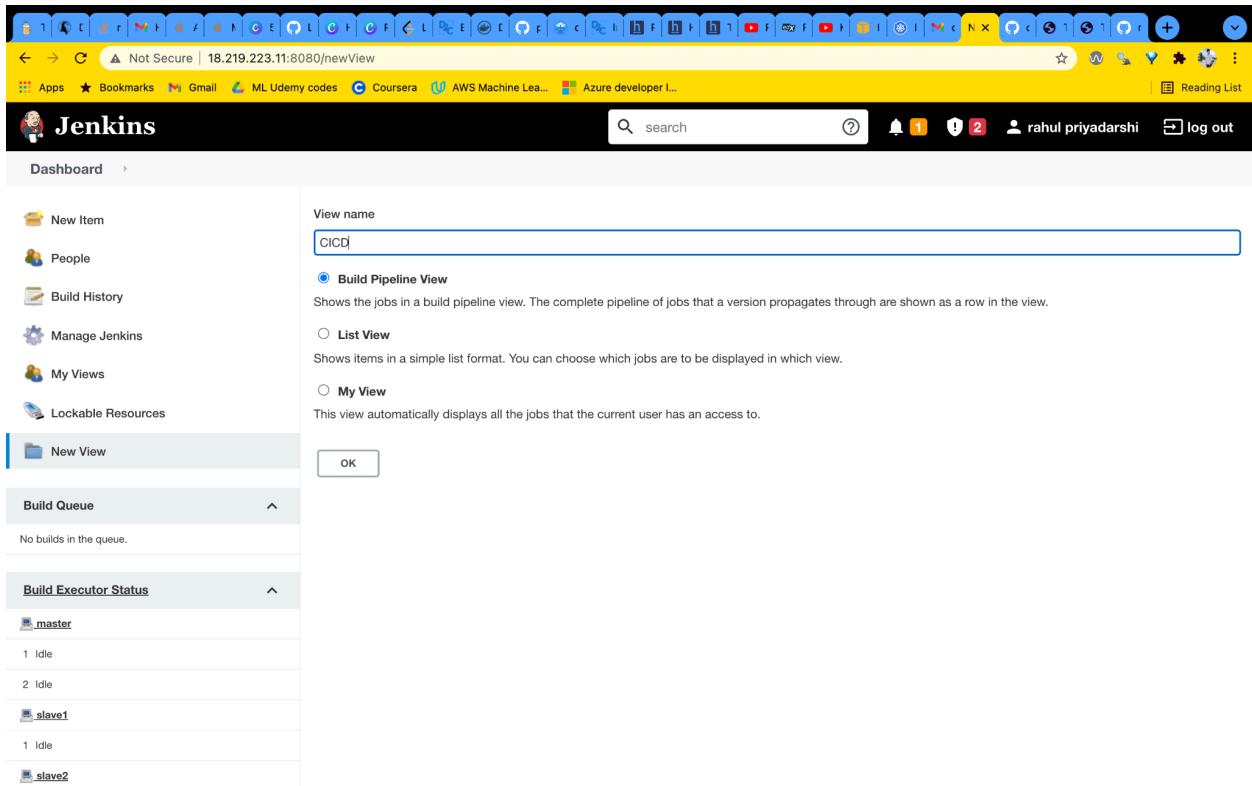
Buttons: Save, Apply, Output

The screenshot shows the Jenkins job configuration interface for a job named "Test". The top navigation bar includes links for General, Source Code Management, Build Triggers, Build Environment, Build (selected), and Post-build Actions. The Build section contains an Execute shell step with the following command:

```
sudo docker rm -f $(sudo docker ps -a -q)
sudo docker build /home/ubuntu/jenkins/workspace/Test -t test
sudo docker run -it -p 5000:5000 -d test
```

Below the command, there is a link to "See the list of available environment variables" and an Advanced... button. The Post-build Actions section contains a Build other projects step, which is configured to trigger only if the build is stable. The configuration page ends with Save and Apply buttons.

Step14: Now setup 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.



Step15: Now under 'Pipeline Flow', select initial job as 'Test'. Then click on OK.

The screenshot shows a web browser window with the URL `18.219.223.11:8080/view/CICD2/configure`. The page is titled "Build Pipeline View Title" and contains a "Pipeline Flow" section. Under "Layout", it says "Based on upstream/downstream relationship". In the "Upstream / downstream config" section, the "Select Initial Job" dropdown is set to "Test". The "Trigger Options" section includes "Build Cards" (set to "Standard build card") and "Restrict triggers to most recent successful builds" (set to "No"). The "Display Options" section has "OK" and "Apply" buttons. On the left, there are sections for "Build Queue" (empty) and "Build Executor Status" showing two slaves: "master" (2 idle) and "slave1" (1 idle), and "slave2" (1 idle).

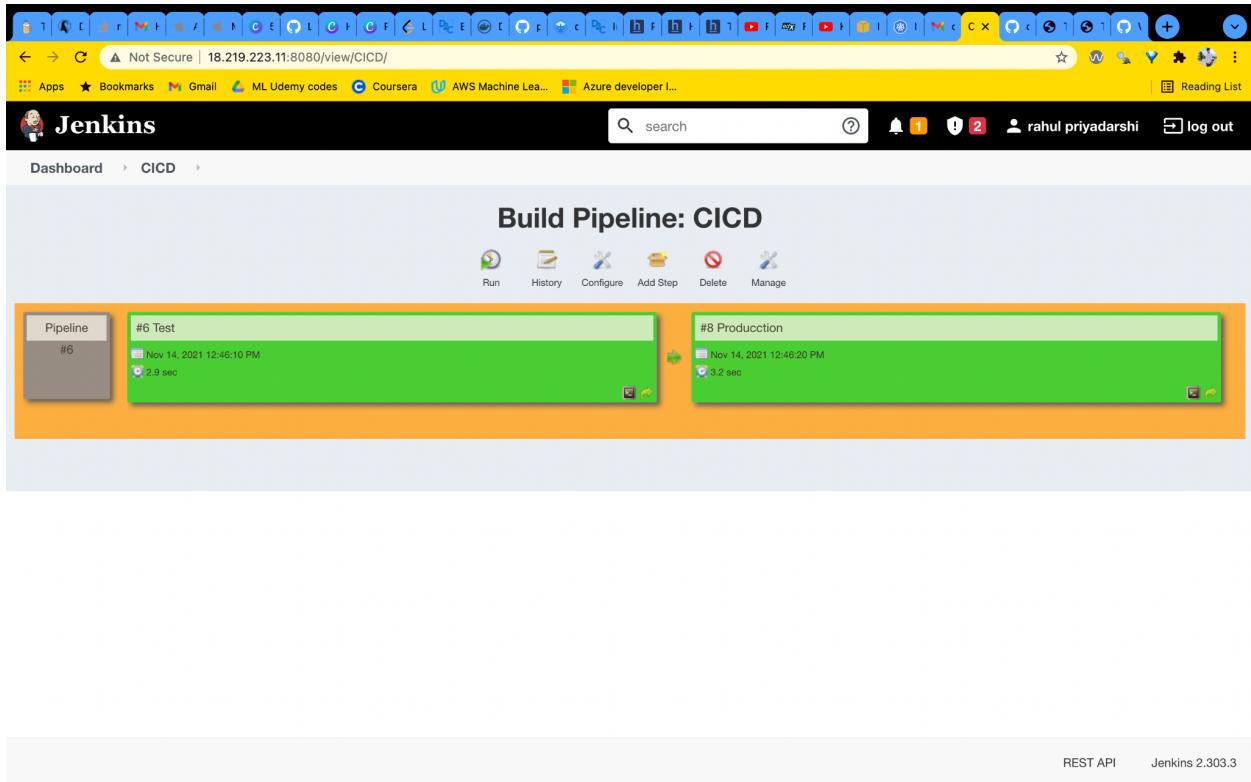
Step16: Now open the GitHub repository and go to 'Settings'. Then select 'Webhooks'. Click on 'Add webhook'. Then enter the 'Payload URL' as 'http://{ip-address-of-the-Jenkins-master-node}:8080/github-webhook/' and select 'Just the push event' and then click on 'Add webhook'.

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

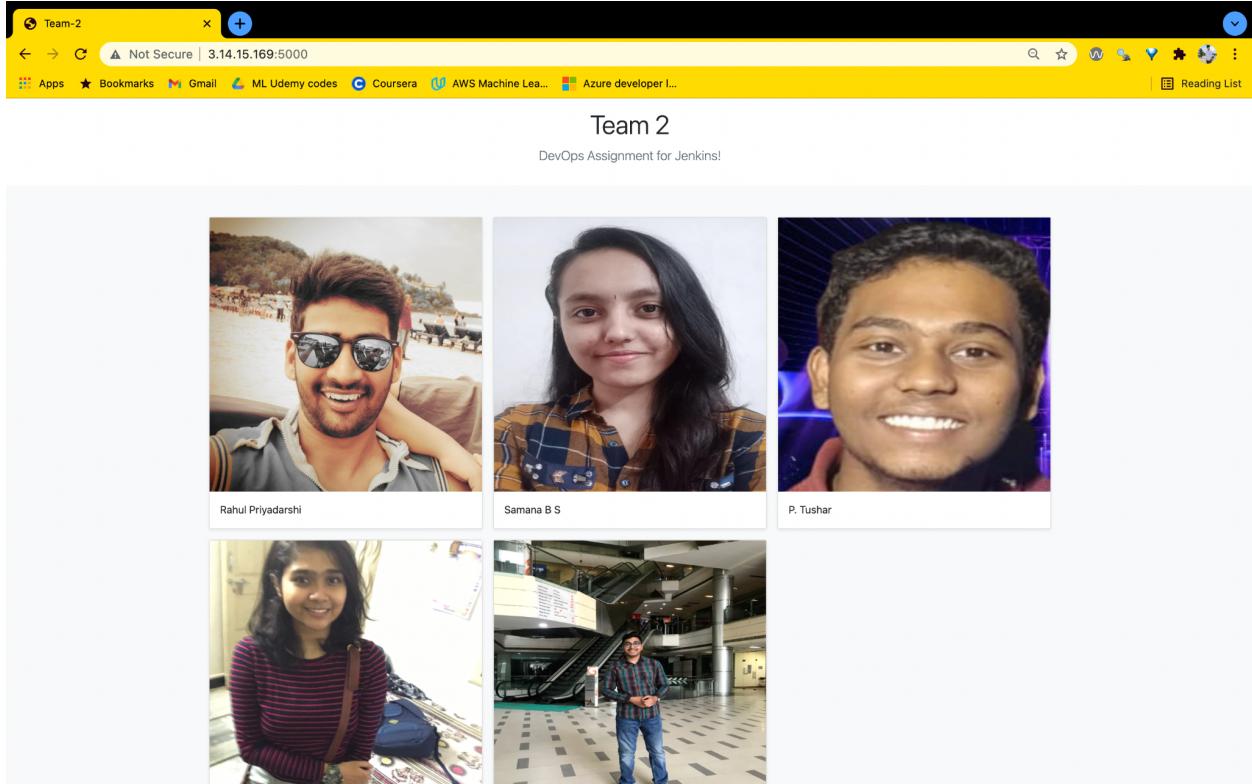
Webhook	Action
✓ http://18.219.223.11:8080/github... (push)	Edit Delete

Step17: 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.

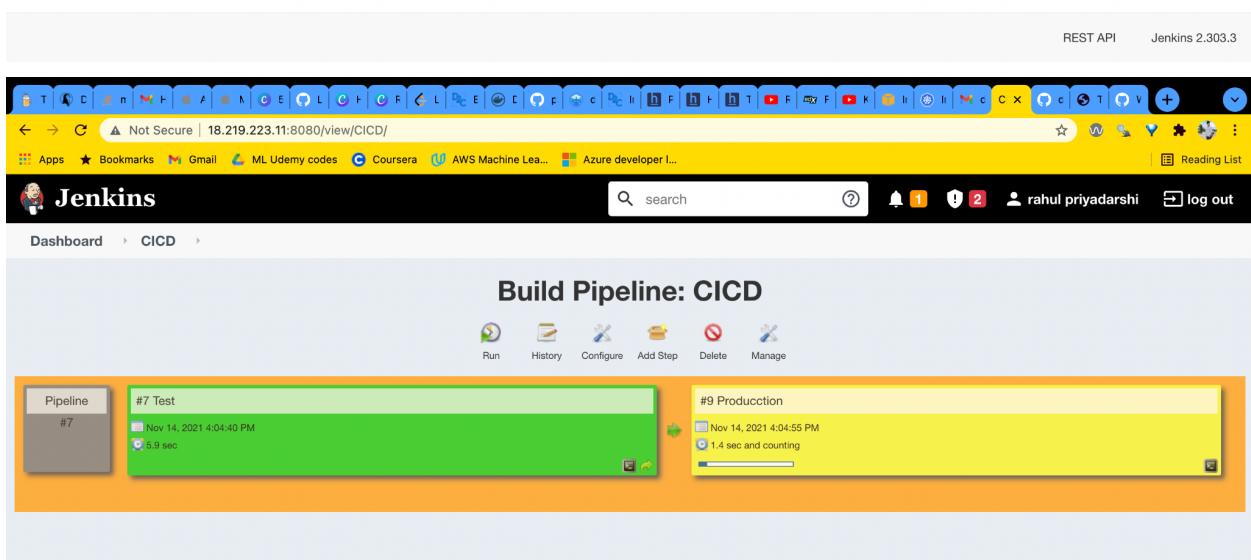
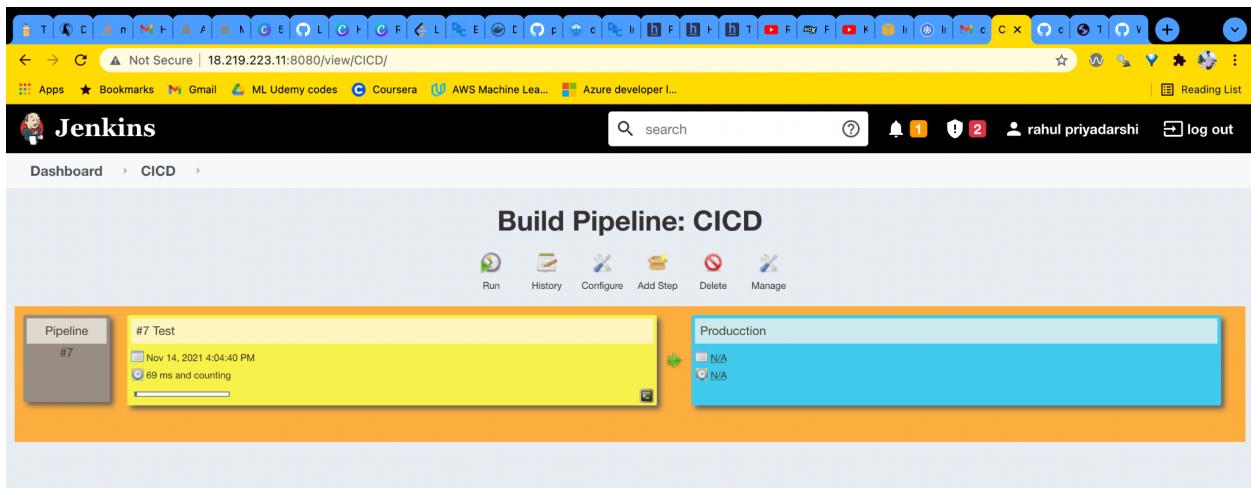
CICD pipeline looks as follows:

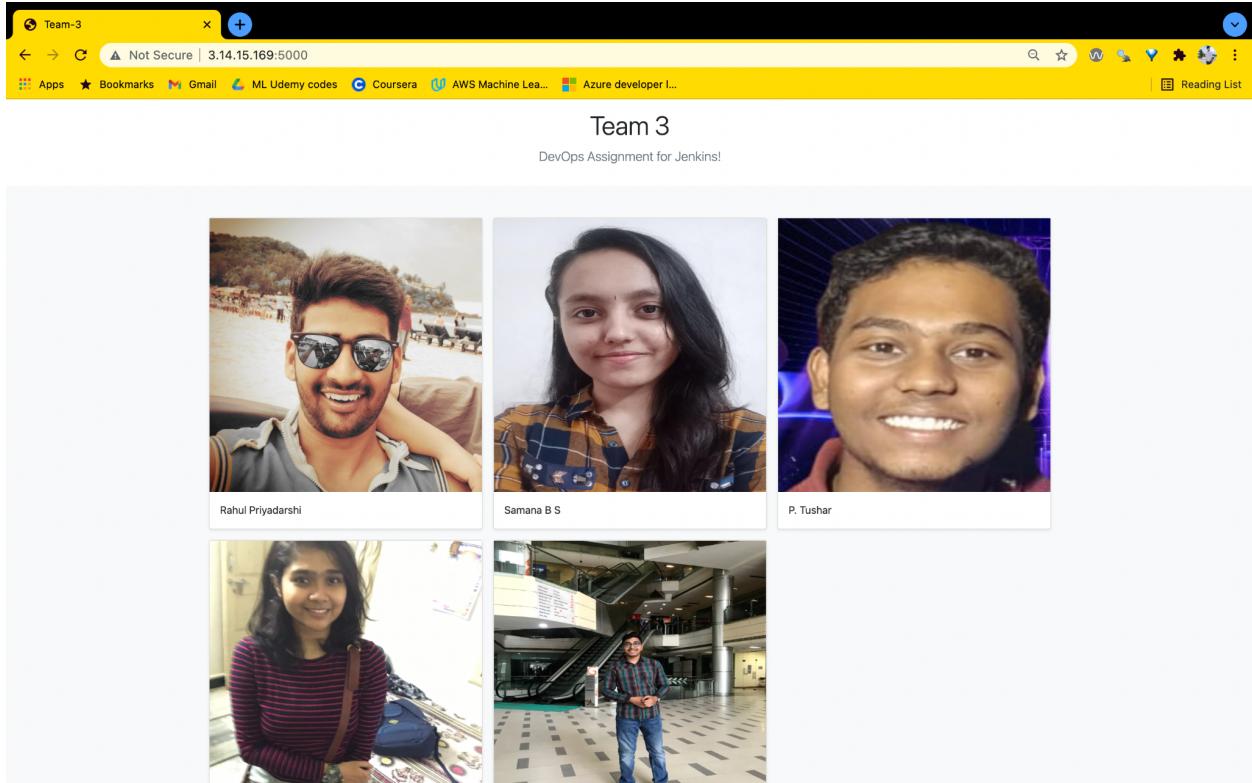


On successful build, the deployed Node application looks as follows on both testing and production servers:



Step18: Here we notice that the Team number entered in the website is Incorrect. It should be 'Team 3' and not 'Team 2'. So we make the correction in the code and push it to the GitHub repository. This triggered the pipeline and changes reflect in the deployed website as shown below:





Changes reflected. Hence, we created a CI/CD pipeline successfully with Jenkins.