

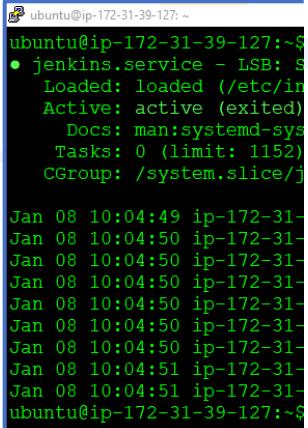
JENKINS HANDS-ON

1. Create 3 instances (Master, Slave-1, Slave-2) on EC2 server.
2. Install Jenkins on Master. (Refer to the Jenkins installation documentation)
3. Set up a Jenkins Master-Slave Cluster on AWS
4. Create a CI CD pipeline triggered by Git Webhook.

First, we have created 3 instances Master (Green terminal), Slave-1(Orange Terminal) and Slave-2(Blue Terminal) on EC2 Server. And then we have installed the Jenkins on Master Machine. Now Let us set up the Jenkins Master-Slave Cluster.

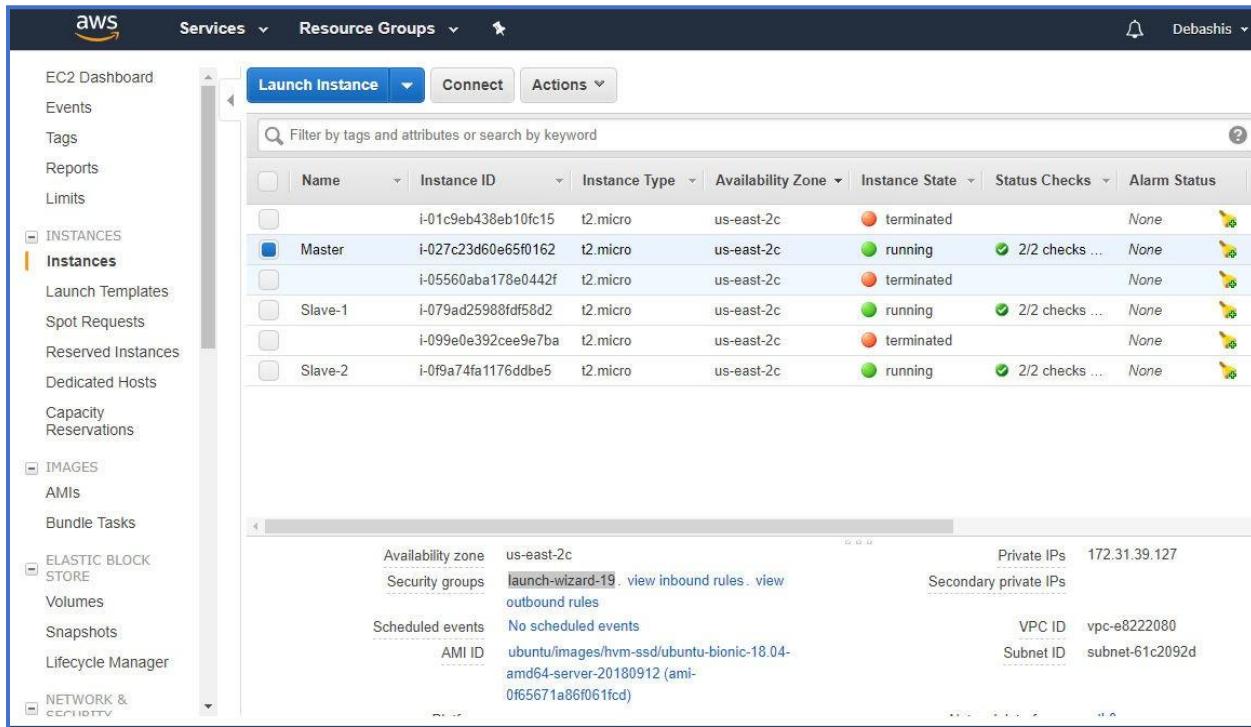
Step 1: Check the status of the Jenkins first.

```
$ service jenkins status
```



```
ubuntu@ip-172-31-39-127:~$● jenkins.service - LSB: S
  Loaded: loaded (/etc/init.d/jenkins)
  Active: active (exited)
    Docs: man:systemd-sys
          Tasks: 0 (limit: 1152)
         CGroup: /system.slice/jenkins.service
Jan 08 10:04:49 ip-172-31-39-127.jenkins[1152]: Jan 08 10:04:50 ip-172-31-39-127.jenkins[1152]: Jan 08 10:04:51 ip-172-31-39-127.jenkins[1152]: Jan 08 10:04:51 ip-172-31-39-127.jenkins[1152]: ubuntu@ip-172-31-39-127:~$
```

Step 2: Got to EC2 server. Select Master click on *launch-wizard-xx*.

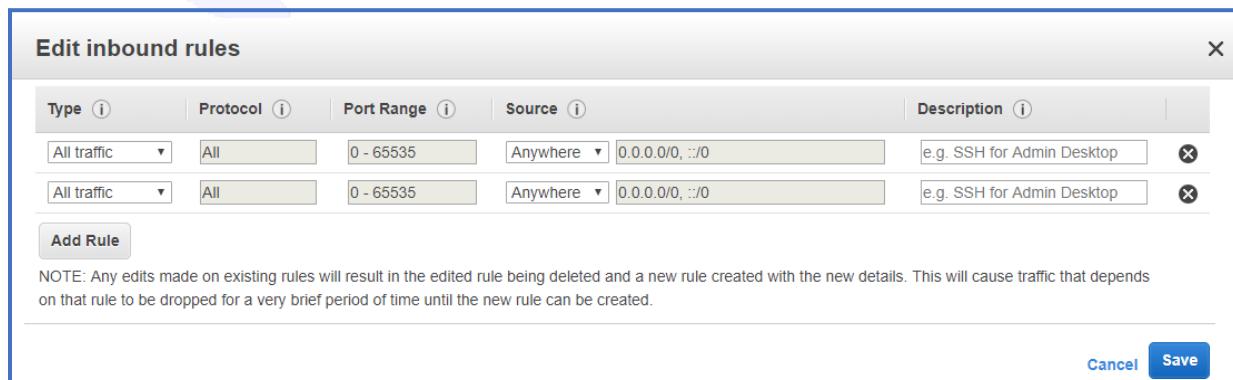


The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Launch Templates, Spot Requests, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, Bundle Tasks, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, and Network & Security. The Instances section is currently selected. The main pane displays a table of instances. The table has columns for Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, and Alarm Status. The instances listed are:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
i-01c9eb438eb10fc15	t2.micro	us-east-2c	terminated	None		
Master	i-027c23d60e65f0162	t2.micro	us-east-2c	running	2/2 checks ...	None
i-05560aba178e0442f	t2.micro	us-east-2c	terminated	None		
Slave-1	i-079ad25988fdf58d2	t2.micro	us-east-2c	running	2/2 checks ...	None
i-099e0e392cee9e7ba	t2.micro	us-east-2c	terminated	None		
Slave-2	i-0f9a74fa1176ddbe5	t2.micro	us-east-2c	running	2/2 checks ...	None

Below the table, there's a detailed view for the Master instance, showing its configuration details.

Step 3: Go to inbound connections. Click on edit. Edit the inbound rules as shown below. Then save the changes.



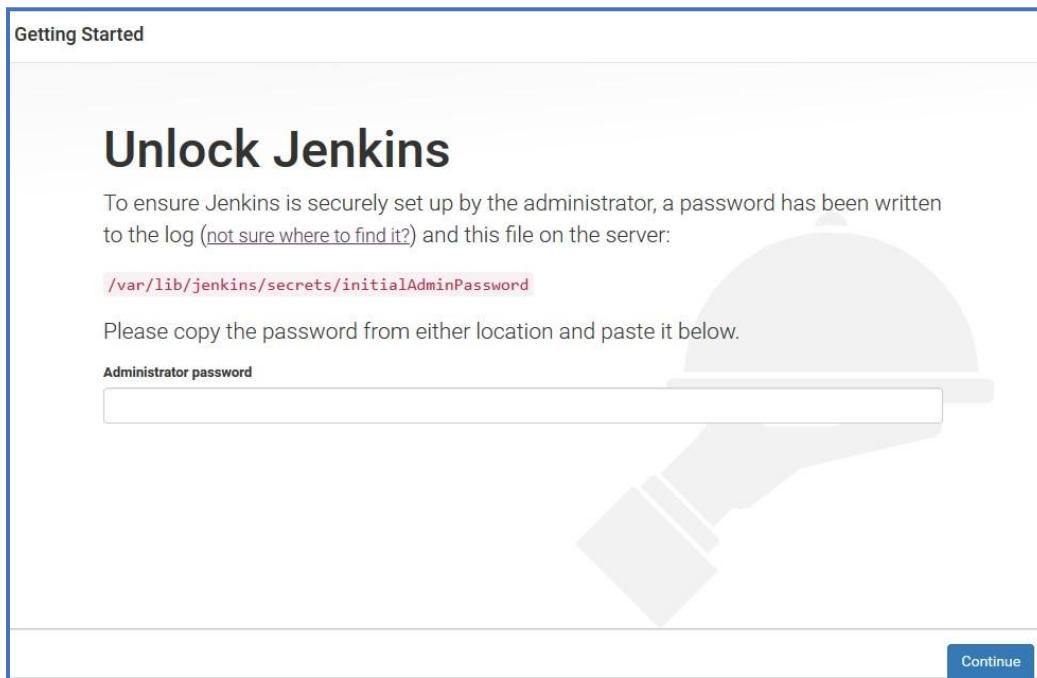
The screenshot shows the 'Edit inbound rules' dialog box. It has a table with columns: Type, Protocol, Port Range, Source, and Description. There are two existing rules:

Type	Protocol	Port Range	Source	Description
All traffic	All	0 - 65535	Anywhere	e.g. SSH for Admin Desktop
All traffic	All	0 - 65535	Anywhere	e.g. SSH for Admin Desktop

At the bottom, there's a note: "NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created." Below the note are 'Cancel' and 'Save' buttons.

Step 4: Now open browser and enter **masterIP:8080**

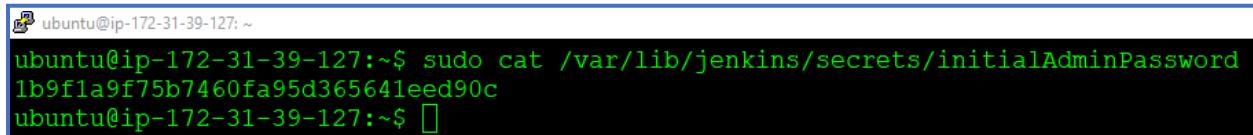
You should land on a page like this:



The screenshot shows a 'Getting Started' section titled 'Unlock Jenkins'. It contains instructions about finding the initial admin password in a log or file, followed by a text input field for the password and a 'Continue' button.

Step 5: Copy the path mentioned in the page and perform cat operation in master terminal.

```
$ sudo cat <path>
```



```
ubuntu@ip-172-31-39-127: ~
ubuntu@ip-172-31-39-127:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
1b9f1a9f75b7460fa95d365641eed90c
ubuntu@ip-172-31-39-127:~$ [ ]
```

This will give us the password which we will use to unlock our Jenkins.

Copy the password from there and paste it on the Jenkins Server page.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

Now click on continue. Then click on Install Suggested Plugins.

Step 6: Once done, enter the Admin User details.

Getting Started

Create First Admin User

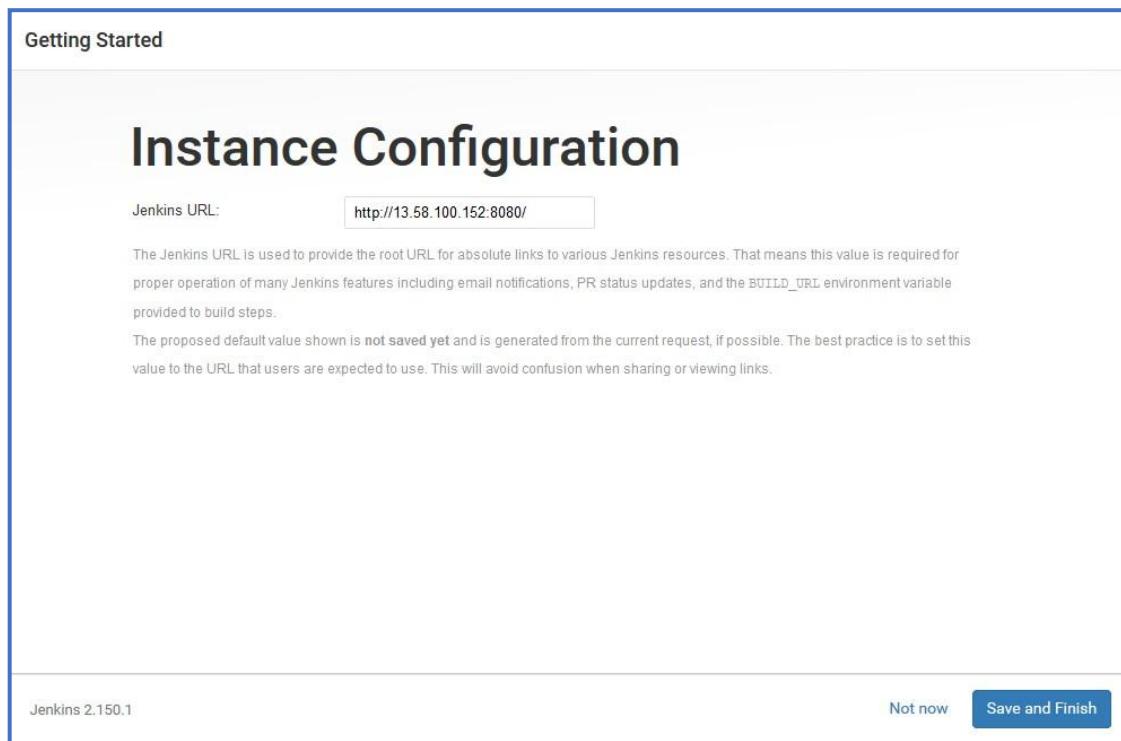
Username:	<input type="text" value="Intellipaat"/>
Password:	<input type="password" value="*****"/>
Confirm password:	<input type="password" value="*****"/>
Full name:	<input type="text" value="Intellipaat"/>
E-mail address:	<input type="text" value="Intellipaat@gmail.com"/>

Jenkins 2.150.1

Continue as admin

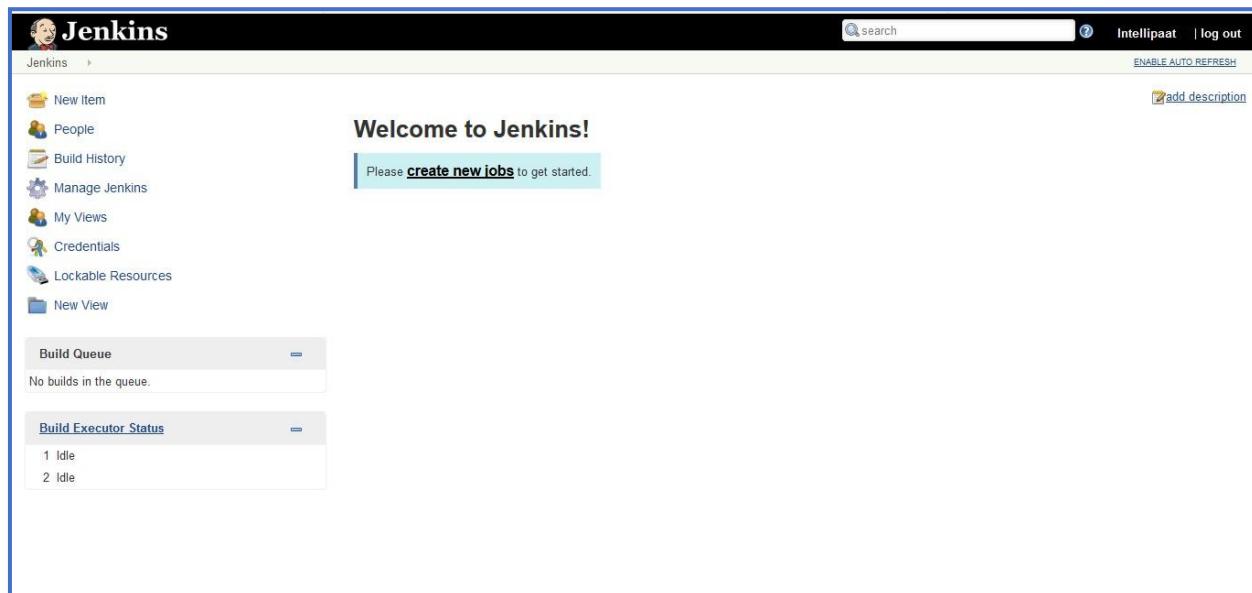
Save and Continue

Then click on **Save and Continue**.



The screenshot shows the Jenkins Instance Configuration page. At the top left is the "Getting Started" section. Below it is the main "Instance Configuration" section with the title "Instance Configuration". A "Jenkins URL:" input field contains the value "http://13.58.100.152:8080/". To the right of the input field is a descriptive text block: "The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps." Below this is another text block: "The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links." At the bottom of the page are two buttons: "Not now" and "Save and Finish". The footer indicates "Jenkins 2.150.1".

Again, click **Save and Finish**. Click on **Install Suggested Plugins**. Once it's done we will land on a page as shown below.



The screenshot shows the Jenkins dashboard. On the left is a sidebar with icons for "New Item", "People", "Build History", "Manage Jenkins", "My Views", "Credentials", "Lockable Resources", and "New View". In the center, the main area displays the message "Welcome to Jenkins!" and "Please [create new jobs](#) to get started.". Below this are sections for "Build Queue" (which says "No builds in the queue.") and "Build Executor Status" (which lists "1 Idle" and "2 Idle"). At the top right of the dashboard are search, refresh, and log out buttons, along with a link to "ENABLE AUTO REFRESH".

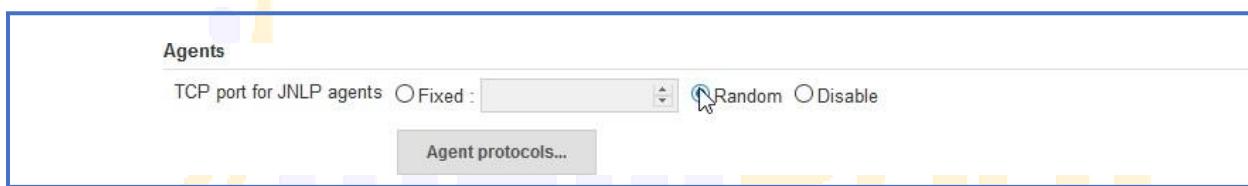
This is our **Jenkins Dashboard**.

Step 7: Go to Manage Jenkins. Click on Configure Global Security.



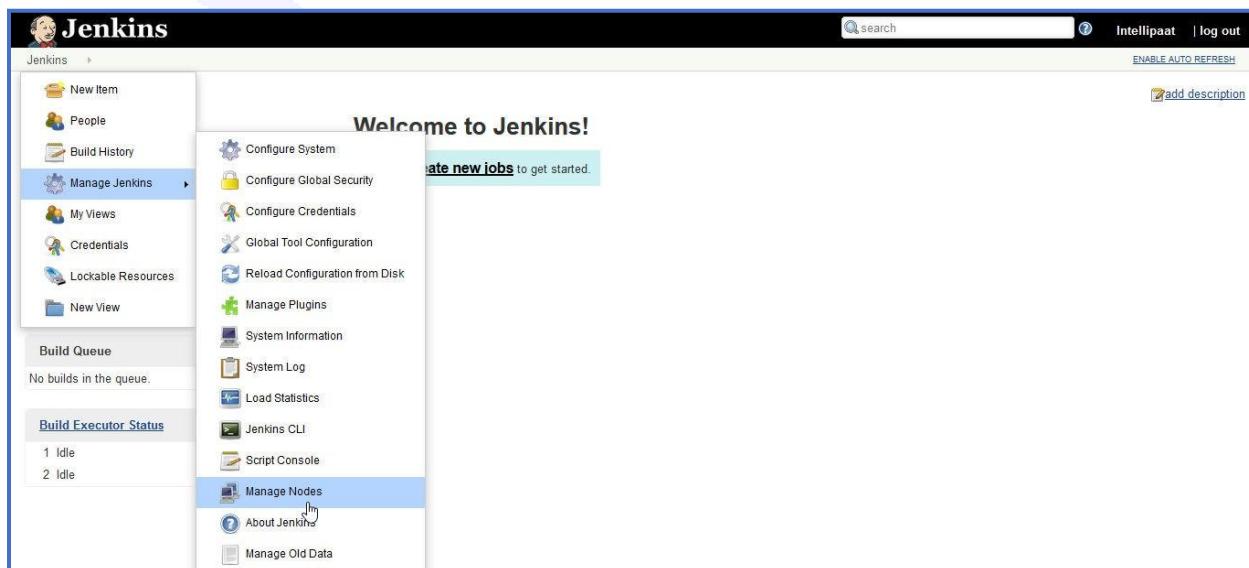
A screenshot of the Jenkins Manage Jenkins page. The left sidebar shows various management options like New Item, People, Build History, and Manage Jenkins. Under Manage Jenkins, there are sub-options: My Views, Credentials, Lockable Resources, and New View. Below these is a Build Queue section stating 'No builds in the queue.' The main content area has a 'Welcome to Jenkins!' message with a 'Create new jobs' button. A dropdown menu is open over the 'Configure Global Security' link, which is highlighted with a yellow arrow. Other items in the dropdown include Configure System, Configure Credentials, Global Tool Configuration, Reload Configuration from Disk, Manage Plugins, System Information, System Log, and Load Statistics.

Step 8: Change the Agents to Random. Then click on Save.



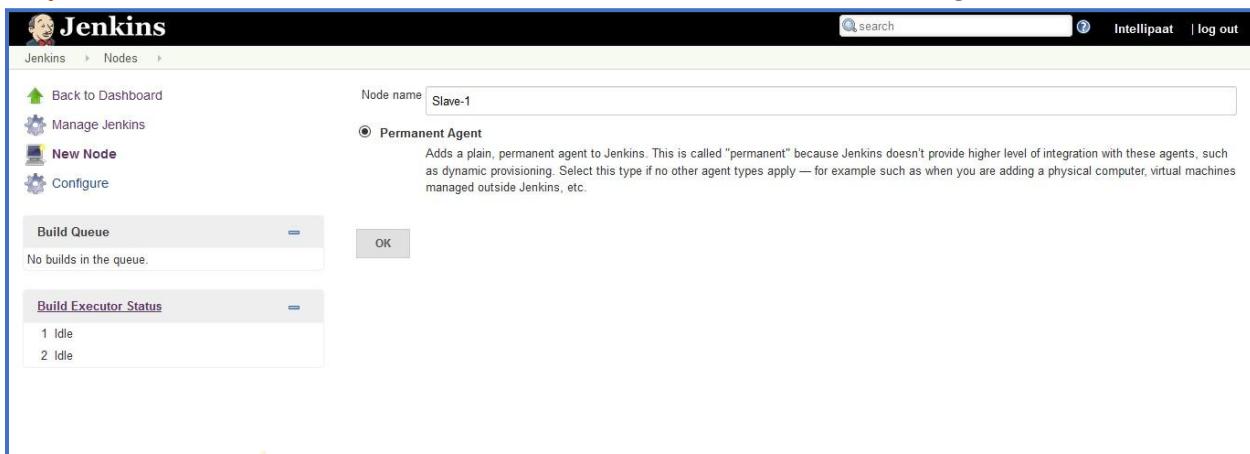
A screenshot of the Jenkins Agents configuration page. It shows a dropdown menu for 'TCP port for JNLP agents' with three options: 'Fixed' (radio button), 'Random' (radio button, highlighted with a yellow arrow), and 'Disable'. Below the dropdown is a 'Agent protocols...' button.

Step 9: Now go to Manage Nodes.



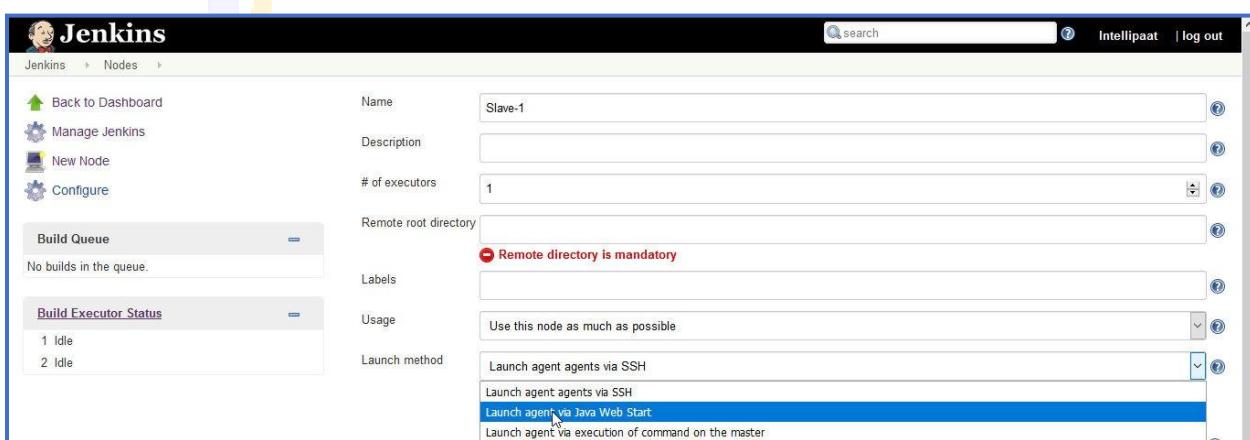
A screenshot of the Jenkins Manage Jenkins page. The left sidebar includes 'Build Executor Status' showing '1 Idle' and '2 Idle'. The 'Manage Jenkins' section contains 'My Views', 'Credentials', 'Lockable Resources', 'New View', 'Build Queue' (empty), and 'Build Executor Status'. Below these are 'Script Console', 'Manage Nodes' (highlighted with a yellow arrow), 'About Jenkins', and 'Manage Old Data'. The main content area is identical to the previous screenshot, showing the 'Welcome to Jenkins!' message and the 'Configure Global Security' dropdown.

Step 10: Click on **New Node**. Add **Slave-1** as new node and make **Permanent Agent**. Click on **ok**.



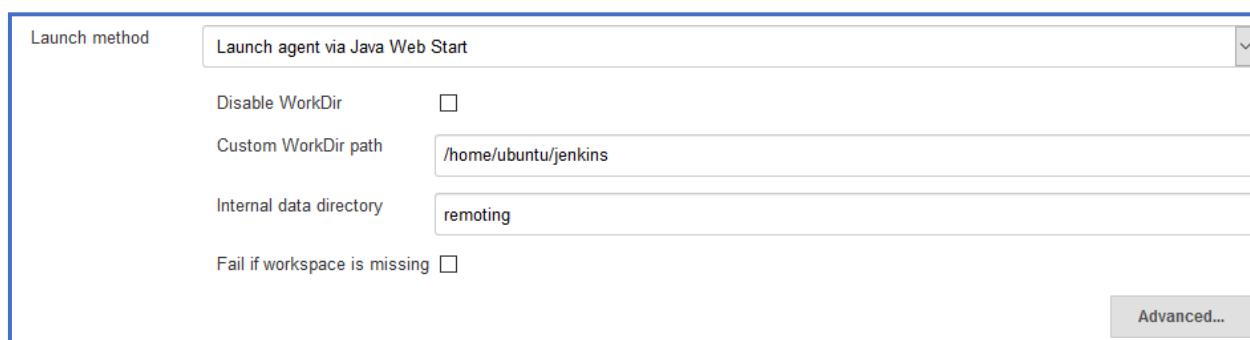
The screenshot shows the Jenkins 'Nodes' configuration page. A new node named 'Slave-1' is being created. The 'Permanent Agent' option is selected, with a note explaining it adds a plain, permanent agent to Jenkins. The 'OK' button is visible at the bottom right.

Step 11: Go to **Launch method** change it to **Launch agent via Java Web Start**.



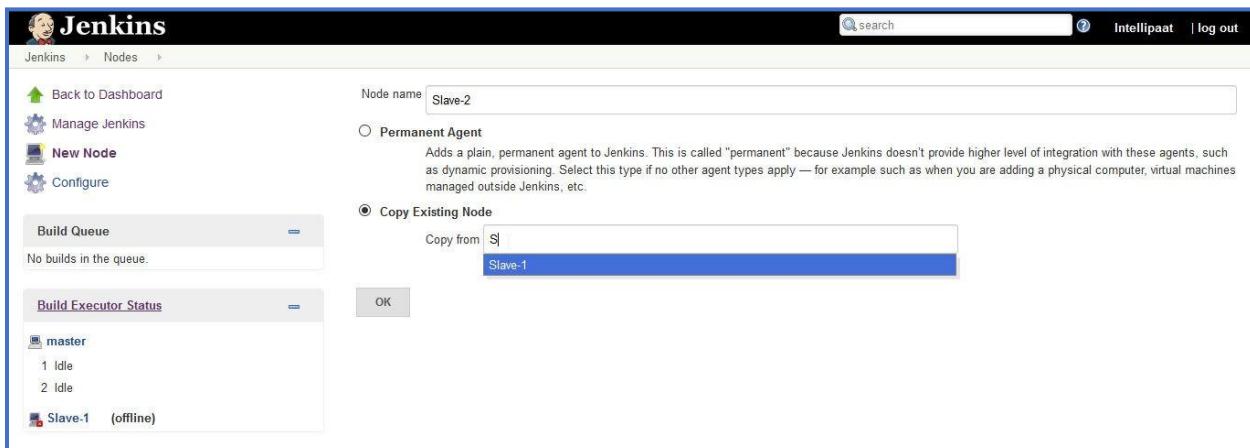
The screenshot shows the configuration of the 'Slave-1' node. The 'Launch method' dropdown is set to 'Launch agent via Java Web Start'. A red error message 'Remote directory is mandatory' is displayed next to the 'Remote root directory' field, which is currently empty.

Step 12: Then add the current working directory path to **/home/ubuntu/jenkins**. Then click on **Save**.



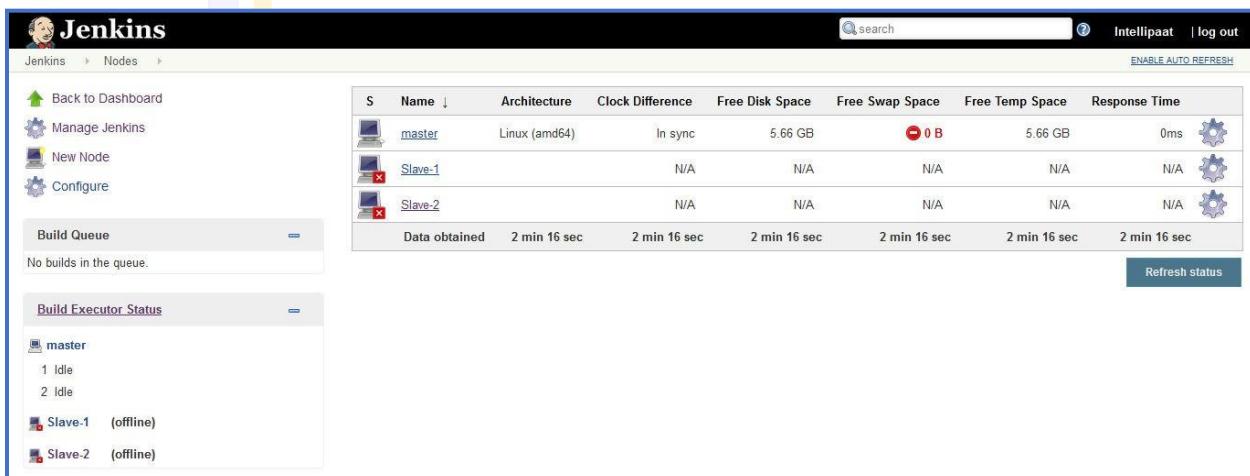
The screenshot shows the 'Launch method' configuration dialog for the 'Slave-1' node. The 'Custom WorkDir path' field is filled with '/home/ubuntu/jenkins'. Other options like 'Disable WorkDir' and 'Internal data directory' are also visible.

Step 13: Make another node **Slave-2** and copy from **Slave-1** as shown below:



The screenshot shows the Jenkins 'Nodes' configuration page. A dialog box is open for creating a new node named 'Slave-2'. The 'Copy Existing Node' option is selected, and the 'Copy from' field contains 'Slave-1'. The 'OK' button is visible at the bottom of the dialog.

Step 14: Then click ok. You can see the list of nodes that we have on the Jenkins Dashboard.



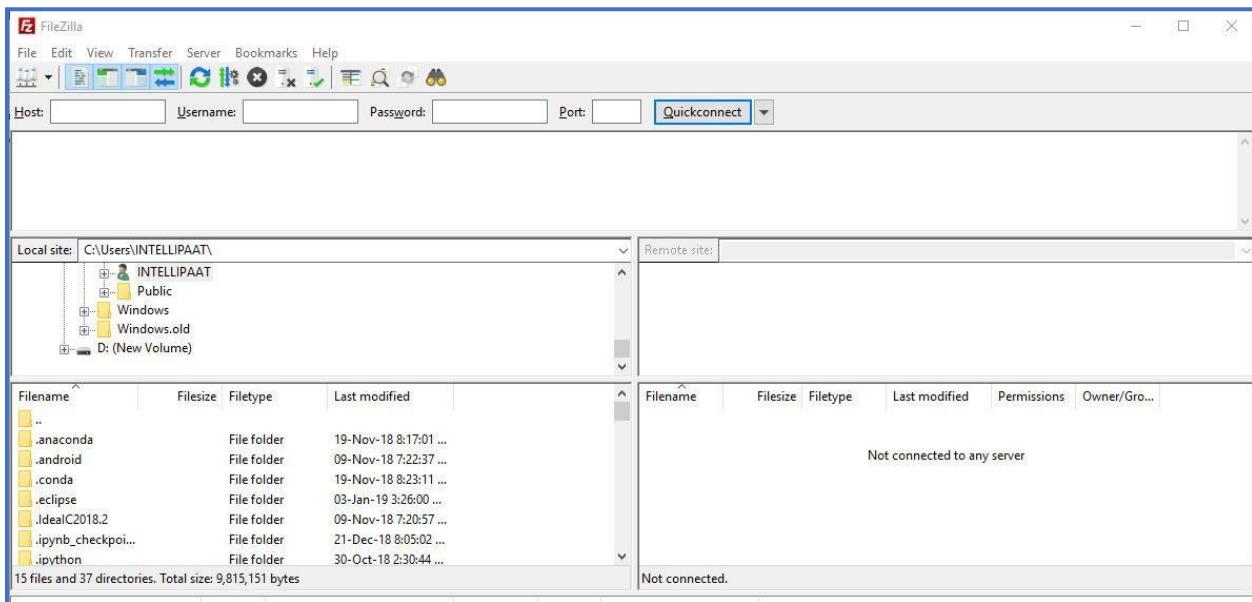
The screenshot shows the Jenkins dashboard. On the right, there is a table of nodes:

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.66 GB	0 B	5.66 GB	0ms
	Slave-1		N/A	N/A	N/A	N/A	N/A
	Slave-2		N/A	N/A	N/A	N/A	N/A

On the left sidebar, under 'Nodes', it shows 'Build Queue' (No builds in the queue) and 'Build Executor Status' (master: 1 Idle, 2 Idle; Slave-1: offline; Slave-2: offline).

Step 15: Before moving ahead, download **FileZilla**.

Step 16: Once you install **FileZilla** the home page looks like this:

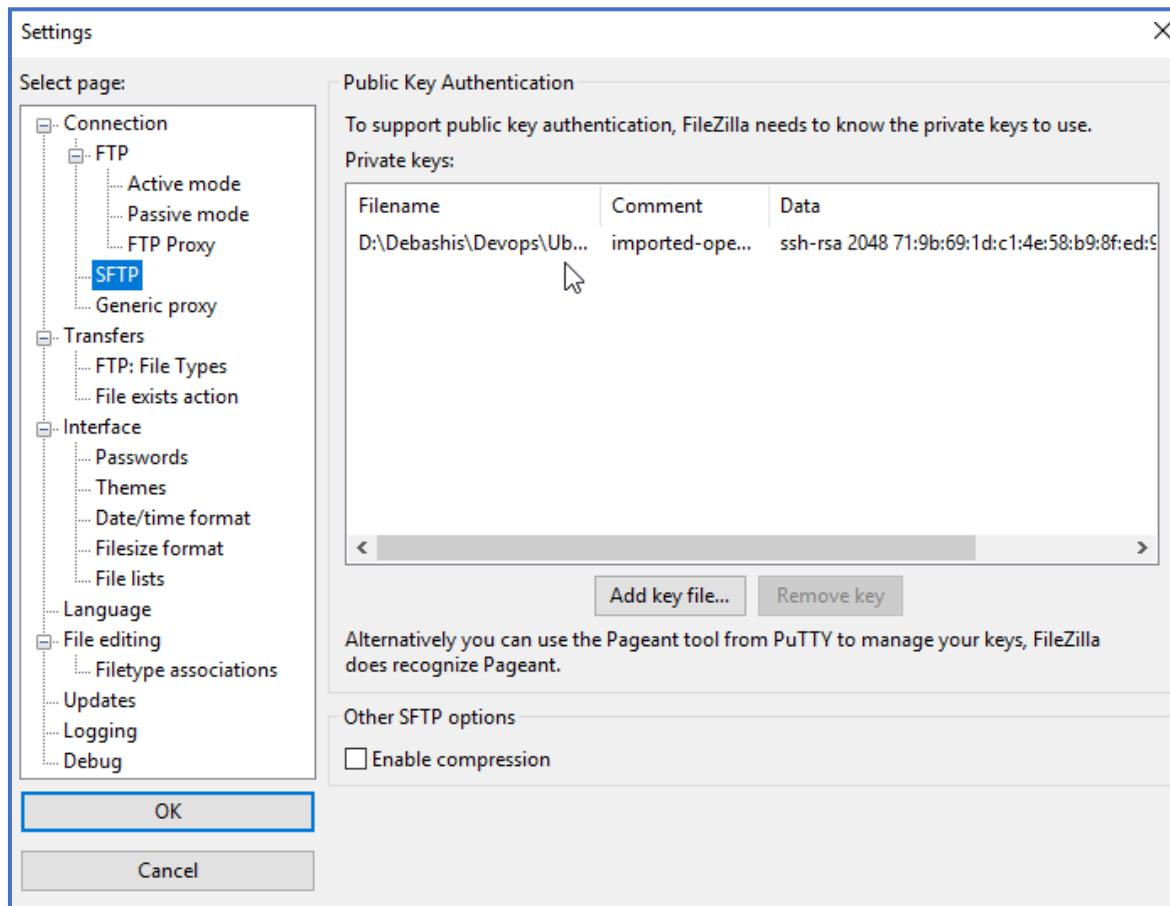


Step 17: Now copy the **Slave-1 IP address**. And add it as **Host**. Add **ubuntu** as **username**. Leave the password field empty. Add **Port as 22**.

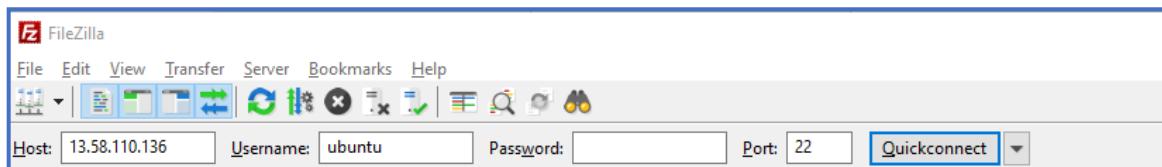


Don't start the connection yet.

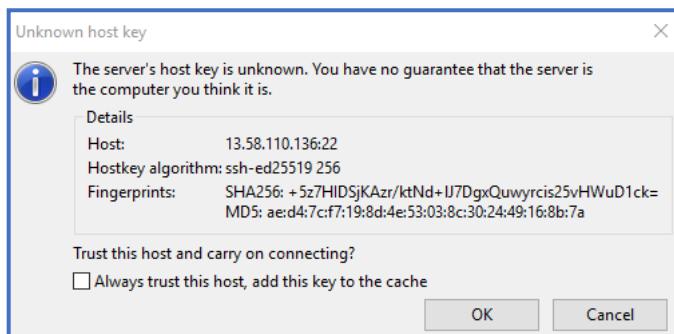
Step 18: Before we start the connection, we need to add the Private key (PPK file). Go to **Edit**, click on **Settings**. Click on **SFTP**. Add the PPK file there and click **ok**.

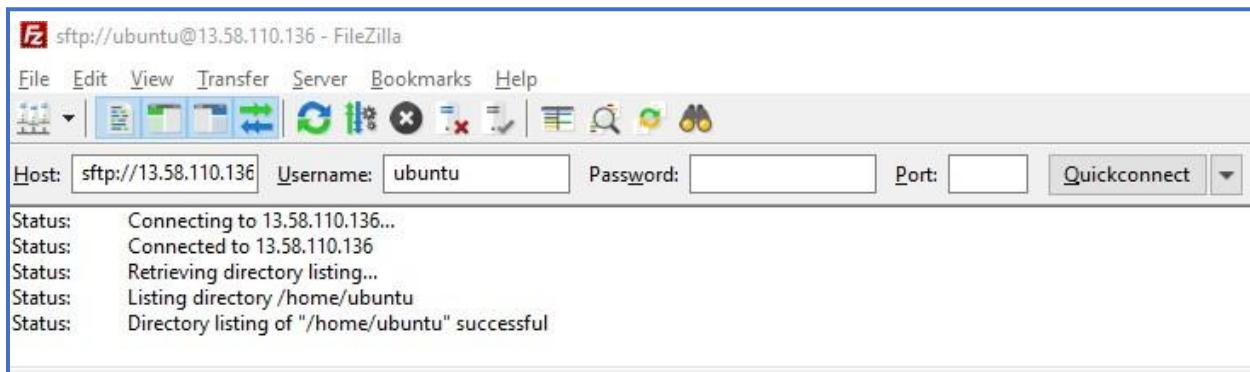


Step 19: Click on Quickconnect.



Then click on **ok**.



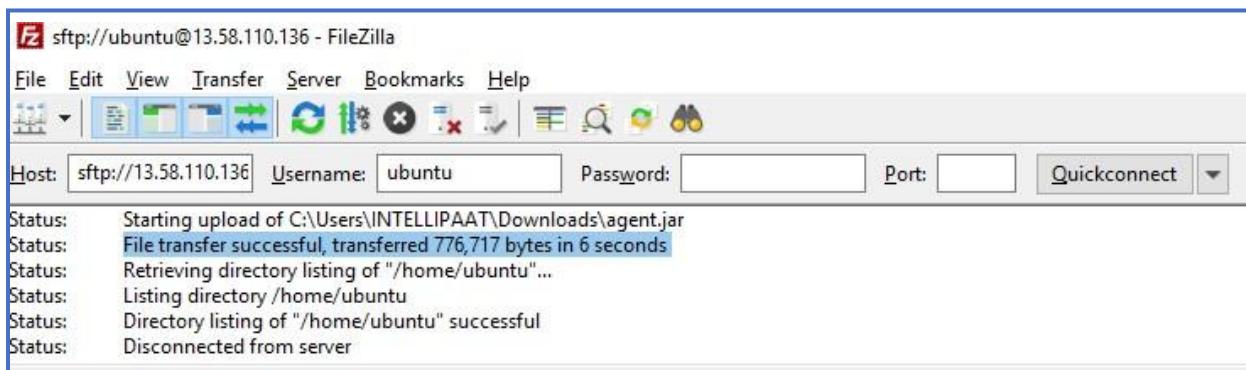


As you can see our connection is successful.

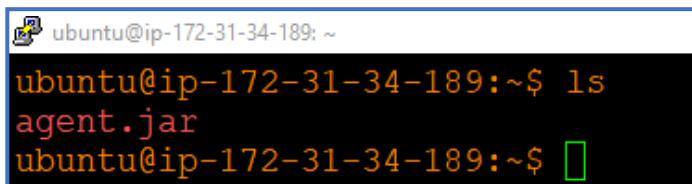
Step 20: Go to the Jenkins Dashboard, Click on **Slave-1**. Download the **Agent.jar** file by clicking on it.



Step 21: Now drag and drop the **agent.jar** file on the ubuntu folder in FileZilla.



Step 22: Let us verify if the file has been transferred to **Slave-1** or not. Open a new session on putty. Connect to slave-1. Run **ls** command.



```
ubuntu@ip-172-31-34-189: ~
ubuntu@ip-172-31-34-189:~$ ls
agent.jar
ubuntu@ip-172-31-34-189:~$
```

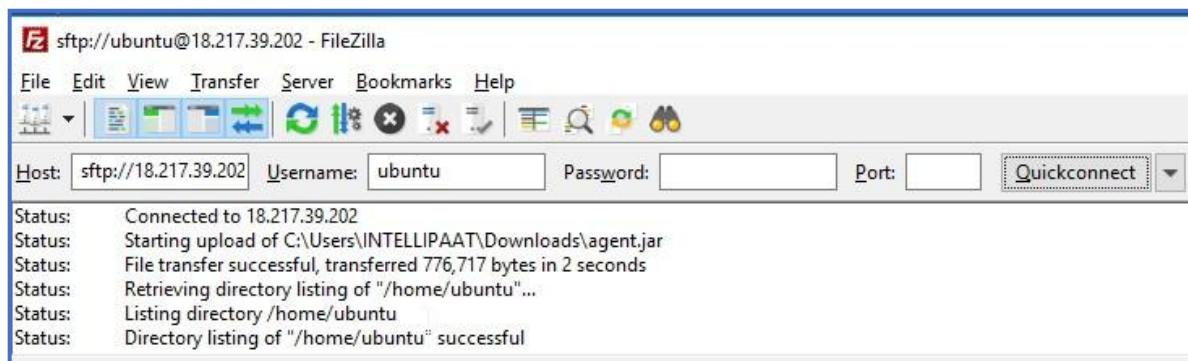
As you can see the agent.jar file appears there, which means our file has been successfully transferred to **Slave-1**.

Step 23: Perform the steps 19 to 22 for Slave-2 as well. (Tip: Rename the agent.jar file of **Slave-2**. Before performing transfer operation in FileZilla)



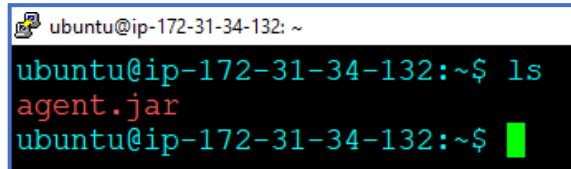
The screenshot shows the Jenkins interface for managing agents. It displays the configuration for 'Agent Slave-2'. The left sidebar includes links for Back to List, Status, Delete Agent, Configure, Build History, Load Statistics, and Log. The main panel shows the 'Agent Slave-2' node with its status as 'Up'. It provides instructions to connect via browser or command line, showing the Java command to run: `java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-2/slave-agent.jnlp -secret 0058b2137c03bcbf18b4ca39e11ea47c45e60360e29802aa096a1 -workDir "/home/ubuntu/jenkins"`. Below this, it lists 'Projects tied to Slave-2' with 'None'.

File transferring is successful for Slave-2 agent.jar file as well.



The screenshot shows the FileZilla interface. The top bar includes File, Edit, View, Transfer, Server, Bookmarks, and Help. The main area shows the connection details: Host: sftp://18.217.39.202, Username: ubuntu, Password: (redacted), Port: (redacted), and Quickconnect. The status window at the bottom displays the transfer progress: Connected to 18.217.39.202, Starting upload of C:\Users\INTELLIPAAAT\Downloads\agent.jar, File transfer successful, transferred 776,717 bytes in 2 seconds, Retrieving directory listing of "/home/ubuntu"... Listing directory /home/ubuntu, Directory listing of "/home/ubuntu" successful.

Step 24: Again, verify by opening a new putty session for Slave-2.



```
ubuntu@ip-172-31-34-132: ~
ubuntu@ip-172-31-34-132:~$ ls
agent.jar
ubuntu@ip-172-31-34-132:~$
```

Looks fine!

Step 25: Now before moving ahead install open jdk on both Slave-1 and Slave-2.

```
$ sudo apt-get update
```

```
ubuntu@ip-172-31-34-189:~$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/universe Sources [9051 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/restricted Sources [5324 B]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main Sources [829 kB]
```

```
ubuntu@ip-172-31-34-132:~$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main Sources [829 kB]
```

Step 26: Now install run the following installation command on both terminal.

```
$ sudo apt install open-9-jdk
```

```
ubuntu@ip-172-31-34-189: ~$ 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 dconf-gsettings-
fontconfig-config fonts-dejavu-core fonts-dejavu-extra glib-network-
gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme l
```

```
ubuntu@ip-172-31-34-132: ~  
ubuntu@ip-172-31-34-132:~$ sudo apt install openjdk-8-jdk  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:
```

Step 27: Now we will connect Slave-1 and Slave-2 to the AWS Jenkins Server. Go to the Jenkins Dashboard, Click on Slave-1, **Copy the command line** as shown.



Jenkins

Nodes > Slave-1

search Intelliplate | log out

[ENABLE AUTO REFRESH](#)

[Back to List](#)

[Status](#)

[Delete Agent](#)

[Configure](#)

[Build History](#)

[Load Statistics](#)

[Log](#)

Agent Slave-1

Connect agent to Jenkins one of these ways:

- [Launch](#) Launch agent from browser
- Run from agent command line:
`java -jar agent.jar --jnlpUrl http://13.58.100.152:8080/computer/Slave-1/slave-agent.jnlp -secret bca98c506a08c730a3a1c3cf08751b7e211ab79afab4db8a727c5e50b7cedfc3 -workDir "/home/ubuntu/jenkins"`

Mark this node temporarily offline

Projects tied to Slave-1

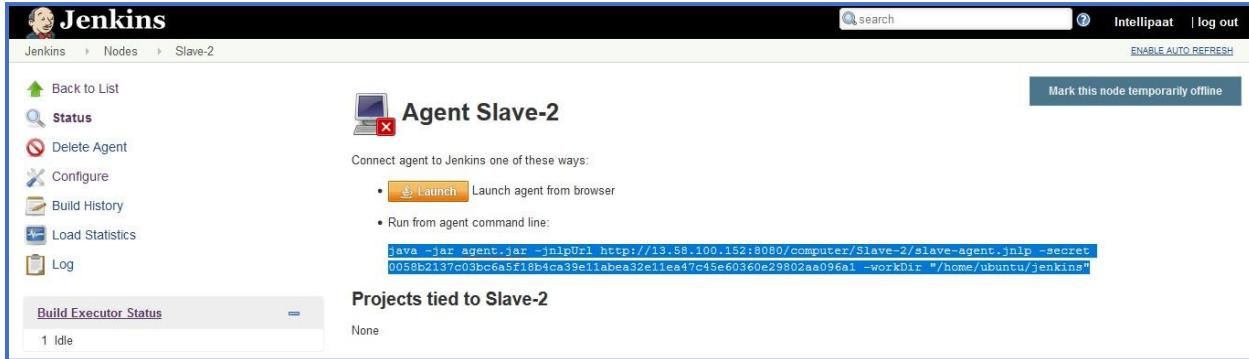
None

Run the command line from Slave-1 as shown below.

```
ubuntu@ip-172-31-34-189:~$ java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-1/slave
t 8ca98c906a08c730a3a1c3cf08751b7e211ab79afab4db8a727c5e50b7cedfc3 -workDir "/home/ubuntu/jenkins"
Jan 08, 2019 11:31:27 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Both error and output logs will be printed to /home/ubuntu/jenkins/remoting I
Jan 08, 2019 11:31:27 AM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: Slave-1
Jan 08, 2019 11:31:27 AM hudson.remoting.jnlp.Main$CuiListener <init>
INFO: Jenkins agent is running in headless mode.
Jan 08, 2019 11:31:28 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3.27
Jan 08, 2019 11:31:28 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Jan 08, 2019 11:31:28 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://13.58.100.152:8080/]
Jan 08, 2019 11:31:28 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Jan 08, 2019 11:31:28 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Agent discovery successful
  Agent address: 13.58.100.152
  Agent port: 36827
  Identity: a9:e0:de:80:e7:44:6f:b7:a2:58:90:6a:1a:87:10:77
Jan 08, 2019 11:31:28 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Jan 08, 2019 11:31:28 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 13.58.100.152:36827
Jan 08, 2019 11:31:28 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP4-connect
Jan 08, 2019 11:31:28 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Remote identity confirmed: a9:e0:de:80:e7:44:6f:b7:a2:58:90:6a:1a:87:10:77
Jan 08, 2019 11:31:29 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

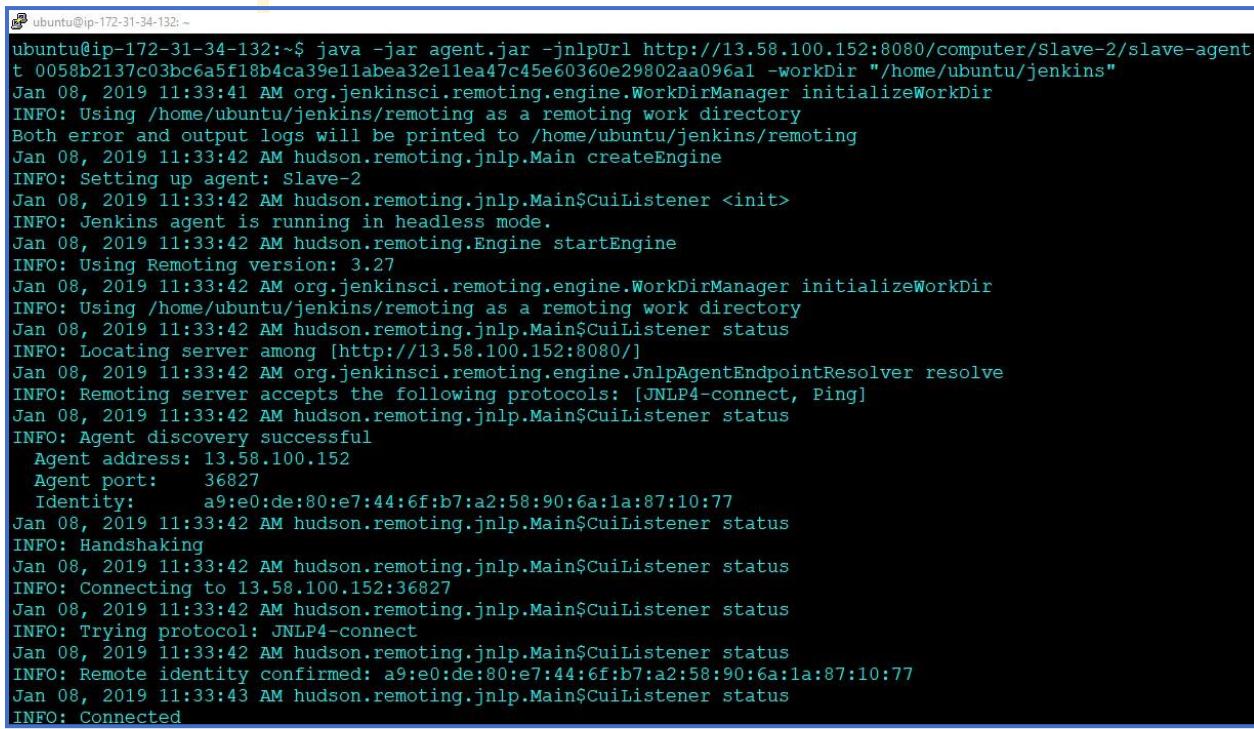
It shows “Connected”.

Step 28: Perform the **Step-27** for **Slave-2** as well.



The screenshot shows the Jenkins interface for 'Agent Slave-2'. On the left, there's a sidebar with links: Back to List, Status, Delete Agent, Configure, Build History, Load Statistics, and Log. The main area has a title 'Agent Slave-2' with a red 'X' icon. It says 'Connect agent to Jenkins one of these ways:' with two options: 'Launch' (which is highlighted in orange) and 'Run from agent command line:' followed by a command-line snippet. Below this is a section titled 'Projects tied to Slave-2' which shows 'None'. At the top right, there are buttons for 'Mark this node temporarily offline', 'ENABLE AUTO REFRESH', and 'log out'.

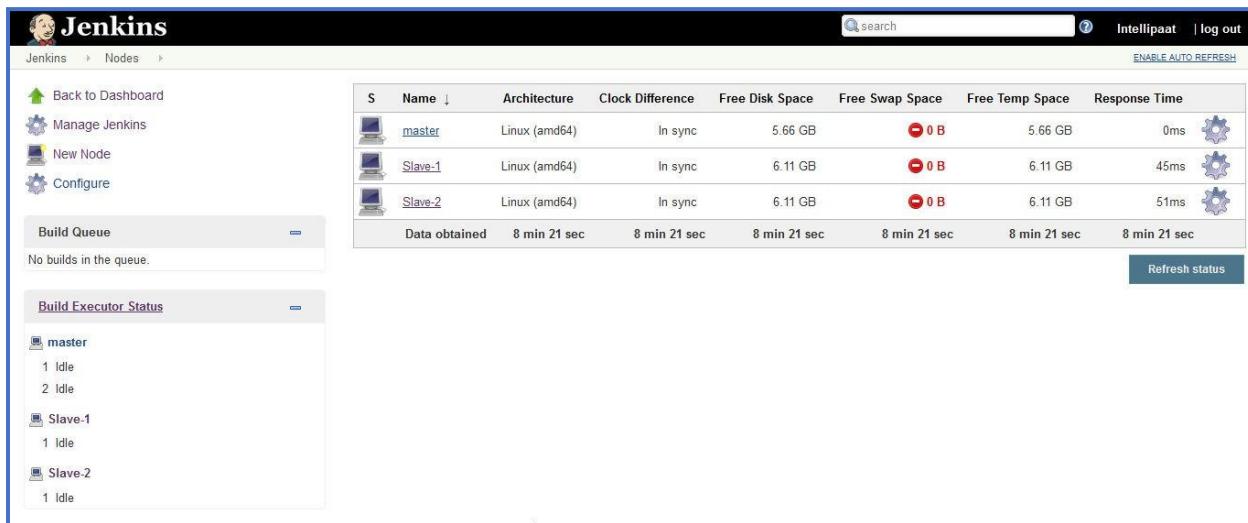
Paste the command line in the Slave-2 Terminal.



```
ubuntu@ip-172-31-34-132:~$ java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-2/slave-agent.t 0058b2137c03bc6a5f18b4ca39e11abea32e11ea47c45e60360e29802aa096a1 -workDir "/home/ubuntu/jenkins"
Jan 08, 2019 11:33:41 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Both error and output logs will be printed to /home/ubuntu/jenkins/remoting
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: Slave-2
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main$CuiListener <init>
INFO: Jenkins agent is running in headless mode.
Jan 08, 2019 11:33:42 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3.27
Jan 08, 2019 11:33:42 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://13.58.100.152:8080/]
Jan 08, 2019 11:33:42 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Agent discovery successful
  Agent address: 13.58.100.152
  Agent port: 36827
  Identity: a9:e0:de:80:e7:44:6f:b7:a2:58:90:6a:1a:87:10:77
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 13.58.100.152:36827
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP4-connect
Jan 08, 2019 11:33:42 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Remote identity confirmed: a9:e0:de:80:e7:44:6f:b7:a2:58:90:6a:1a:87:10:77
Jan 08, 2019 11:33:43 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

Important Note: Don't end the Sessions that we just connected. To perform further operations on Slave-1 and Slave-2 duplicate the sessions.

So now that our Slave-1 and Slave-2 has been connected to Jenkins Server, it looks like this.



S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.66 GB	0 B	5.66 GB	0ms
	Slave-1	Linux (amd64)	In sync	6.11 GB	0 B	6.11 GB	45ms
	Slave-2	Linux (amd64)	In sync	6.11 GB	0 B	6.11 GB	51ms

After we have successfully created the Master Slave Cluster on AWS Jenkins. We will now create a CI CD pipeline triggered by Git Webhook.

Hands-on: Create a CI CD pipeline triggered by Git Webhook.

Step 1: Before that open your GitHub account and import the below given repository.

<https://github.com/hshar/devopsIQ.git>

Step 2: Install docker on both **Slave-1** and **Slave-2**.

```
ubuntu@ip-172-31-34-189: ~
ubuntu@ip-172-31-34-189:~$ 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 cgroupfs-mount libltdl7 pigz ubuntu-fan
Suggested packages:
  ifupdown aufs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount docker.io libltdl7 pigz ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 121 not upgraded.
Need to get 40.3 MB of archives.
After this operation, 198 MB of additional disk space will be used.
```

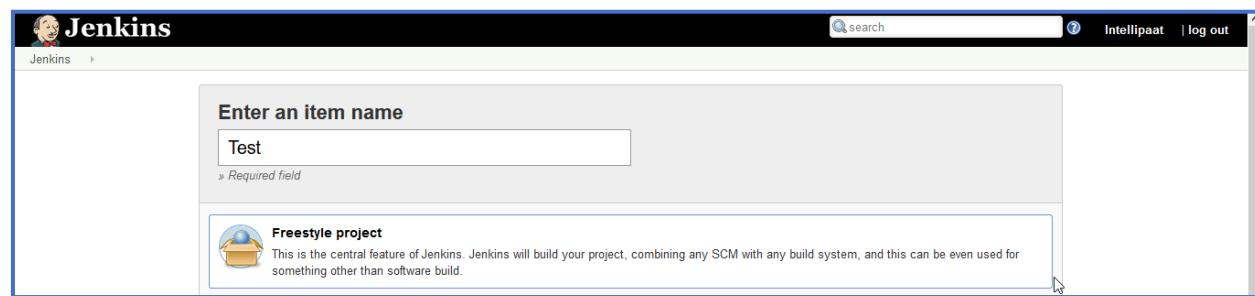
```
ubuntu@ip-172-31-34-132: ~
ubuntu@ip-172-31-34-132:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount libltdl7 pigz ubuntu-fan
Suggested packages:
  ifupdown aufs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount docker.io libltdl7 pigz ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 121 not upgraded.
Need to get 40.3 MB of archives.
After this operation, 198 MB of additional disk space will be used.
```

Step 3: Open Jenkins Dashboard. Create a new job (Freestyle Project) for Slave-1.



The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, My Views, Credentials, Lockable Resources, and New View. The main area has a title 'Welcome to Jenkins!' and a message 'Please [create new jobs](#) to get started.' A cursor is hovering over the 'create new jobs' link.

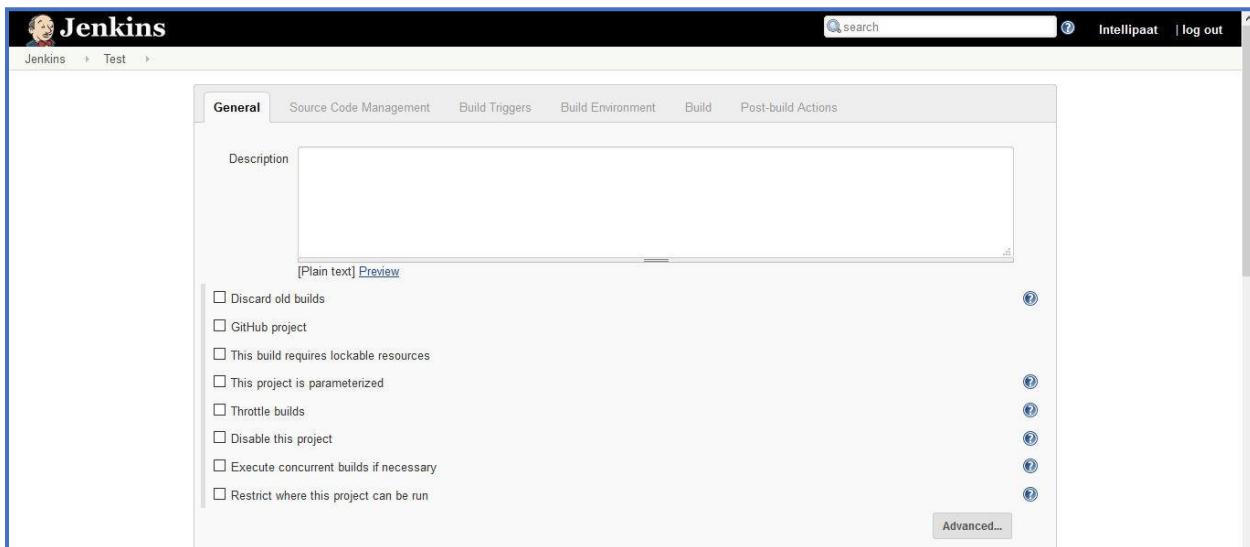
Name the Project as **Test**, Select **Freestyle Project** option.



The screenshot shows the 'Create New Item' form. In the 'Enter an item name' field, 'Test' is typed. Below it, under the 'Freestyle project' section, there is a description: 'This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.' A cursor is hovering over the 'Freestyle project' link.

Then click on **Ok**.

You should land on a page like this.

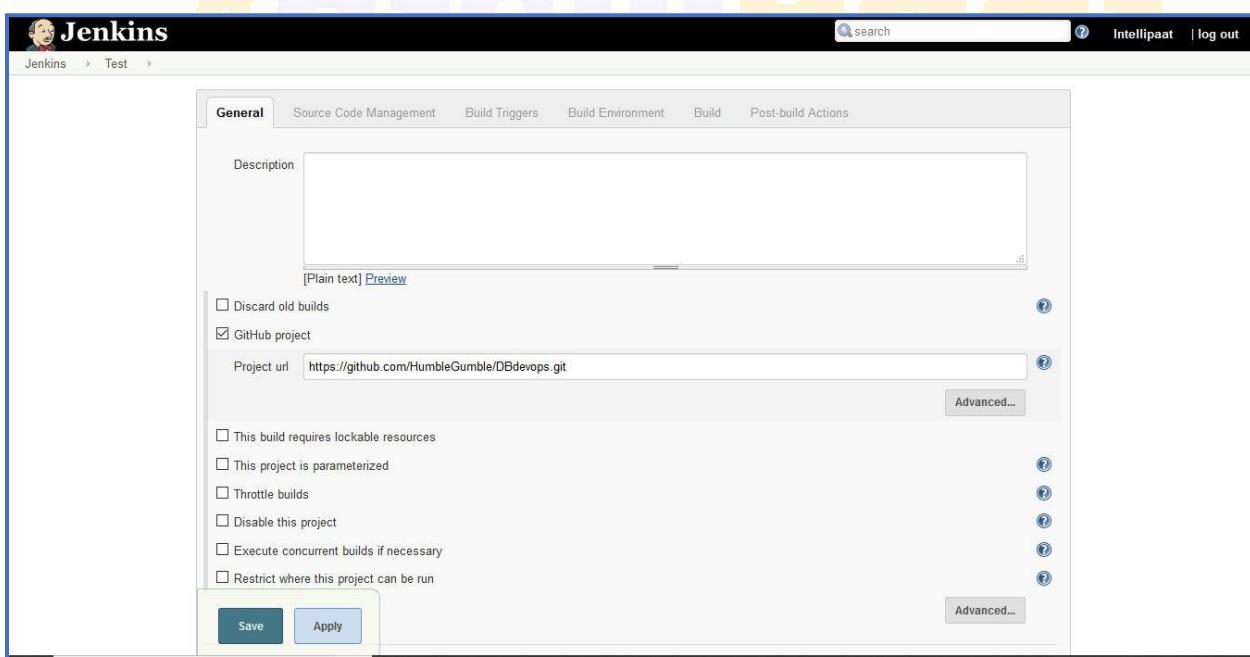


A screenshot of the Jenkins General configuration page. The 'General' tab is selected. The 'Description' field is empty. Below it is a 'Plain text' link and a 'Preview' link. A list of checkboxes follows:

- Discard old builds
- GitHub project
- This build requires lockable resources
- This project is parameterized
- Throttle builds
- Disable this project
- Execute concurrent builds if necessary
- Restrict where this project can be run

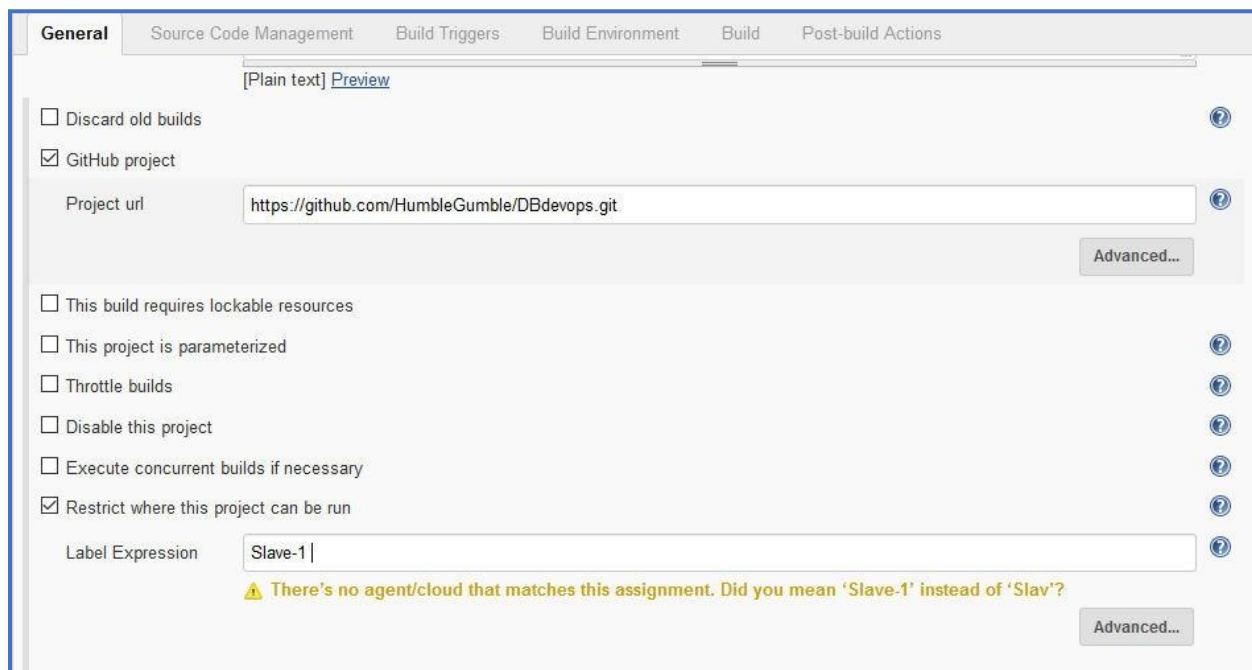
An 'Advanced...' button is located at the bottom right of the configuration area.

Step 4: Place your git repository link as shown below.



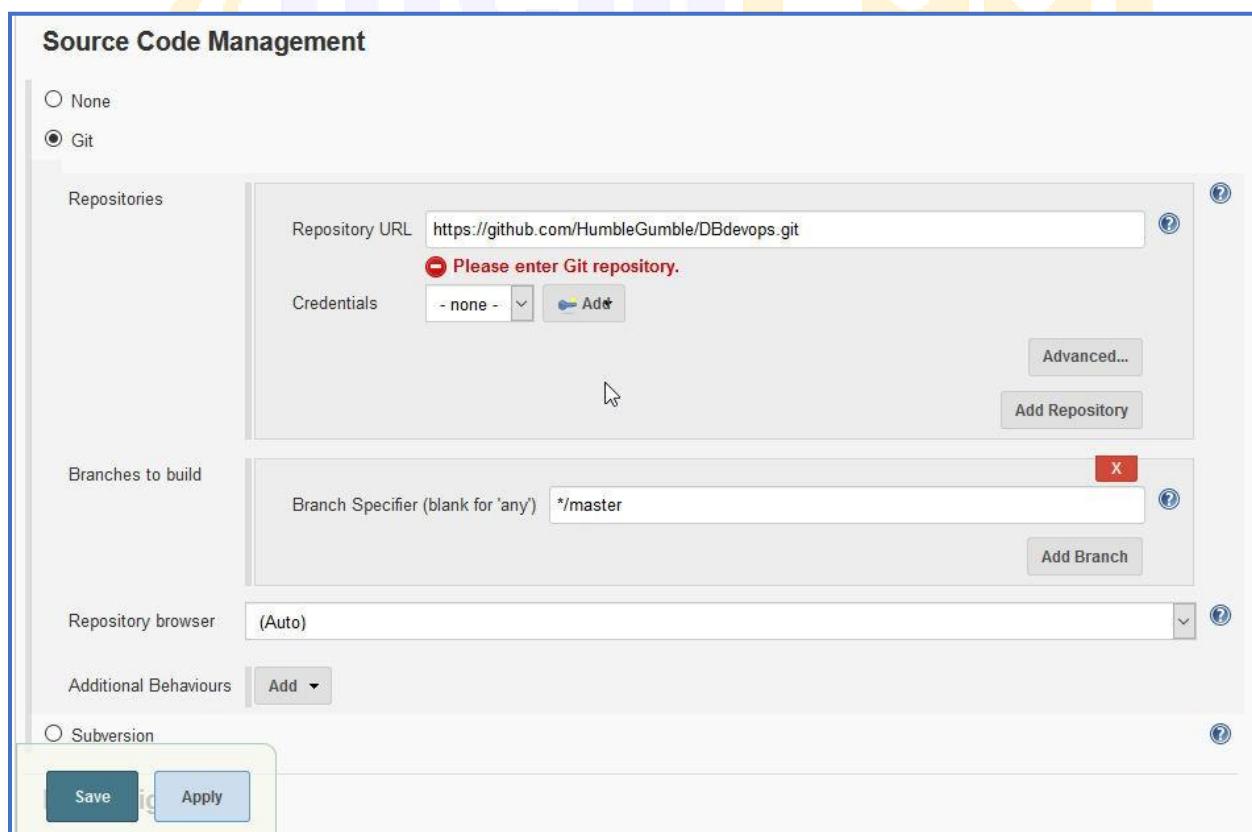
A screenshot of the Jenkins General configuration page, identical to the previous one except for the 'GitHub project' checkbox which is now checked. The 'Project url' field contains the value <https://github.com/HumbleGumble/DBdevops.git>. The 'Save' and 'Apply' buttons are visible at the bottom.

Click on **Restrict where this project can be run**. Add **Slave-1** there.



The screenshot shows the Jenkins General configuration page. The 'General' tab is selected. Under 'GitHub project', the 'Project url' is set to <https://github.com/HumbleGumble/DBdevops.git>. There are several checkboxes for build options, with 'Restrict where this project can be run' checked. A warning message below says: '⚠ There's no agent/cloud that matches this assignment. Did you mean 'Slave-1' instead of 'Slav'?'. Buttons for 'Advanced...' and 'Save' are visible at the bottom.

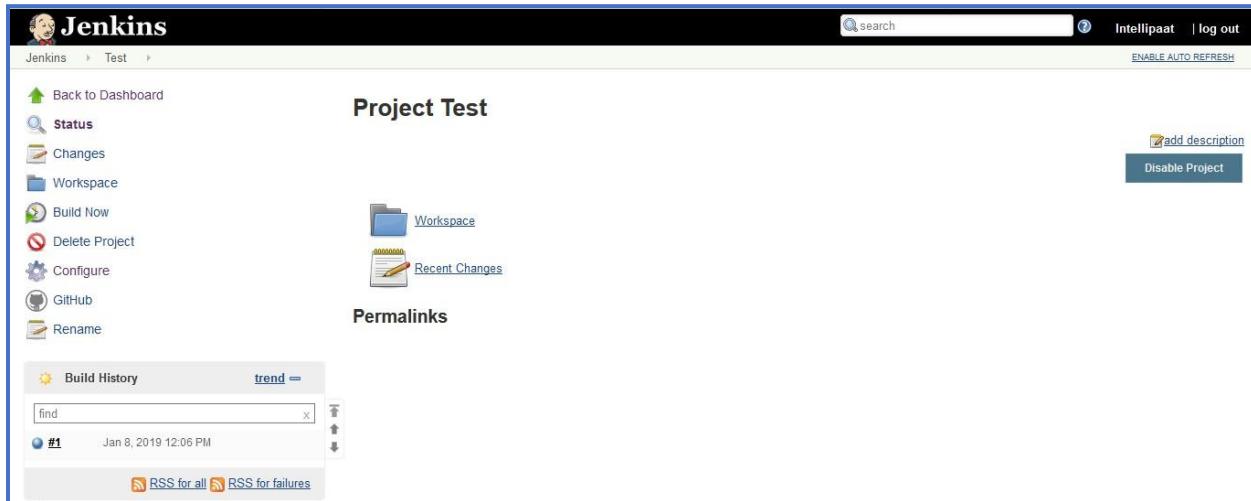
Go to **Source Code Management**, click on **git**, add the **git repository link** there as well.



The screenshot shows the Jenkins Source Code Management configuration page for Git. The 'Git' option is selected under 'Repositories'. A red error message 'Please enter Git repository.' is displayed next to the 'Repository URL' field, which contains the correct URL <https://github.com/HumbleGumble/DBdevops.git>. Other fields include 'Branches to build' (set to */master) and 'Repository browser' (set to '(Auto)'). Buttons for 'Save', 'Apply', 'Advanced...', and 'Add Repository' are at the bottom.

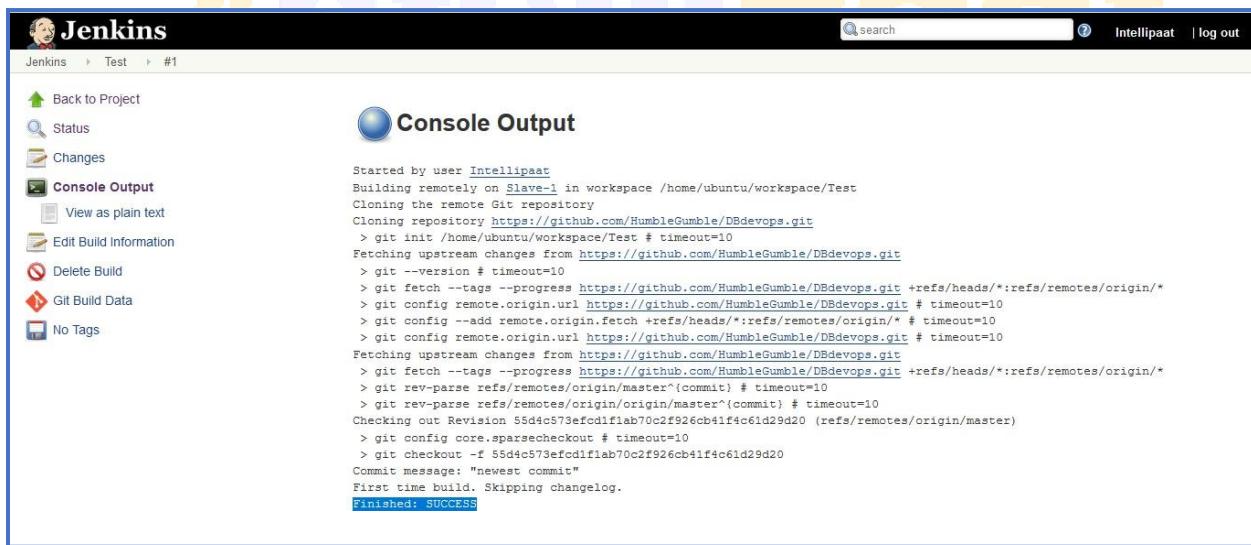
Click on **Save**.

Step 5: Click on **Build Now**, if the building is done without any error there will be **blue circle** in the building history.



The screenshot shows the Jenkins Project Test dashboard. On the left, there's a sidebar with links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, GitHub, and Rename. The main area is titled "Project Test" and contains sections for "Workspace" (with a folder icon) and "Recent Changes" (with a document icon). Below these are "Permalinks" and a "Build History" section. The "Build History" section has a table with one row: "#1 Jan 8, 2019 12:06 PM". At the bottom, there are RSS feed links for all and failures. A blue circle is visible next to the build number in the history table.

Click on the blue circle of build #1.



The screenshot shows the Jenkins Console Output for build #1. The sidebar on the left includes links for Back to Project, Status, Changes, and Console Output (which is selected). The main content area is titled "Console Output" and displays the build logs. The logs show the build process starting, cloning the repository from GitHub, and performing a successful build. The final line of the log is "Finished: SUCCESS".

You can see it has been built successfully. Let us verify that.

Step 6: Go to slave-1.

```
$ ls
$ cd workspace
$ ls
$ cd Test
$ ls
```

```
ubuntu@ip-172-31-34-189: ~/workspace/Test
ubuntu@ip-172-31-34-189:~$ ls
agent.jar jenkins workspace
ubuntu@ip-172-31-34-189:~$ cd workspace
ubuntu@ip-172-31-34-189:~/workspace$ ls
Test
ubuntu@ip-172-31-34-189:~/workspace$ cd Test
ubuntu@ip-172-31-34-189:~/workspace/Test$ ls
Dockerfile devopsIQ docker-compose
ubuntu@ip-172-31-34-189:~/workspace/Test$ 
```

You can see the repository files there. This means the git repository has been successfully cloned into the Test job.

Now we will deploy the website that we have stored in our repository.

Step 7: To run the **Dockerfile** we have to check the copy the present working directory.

```
ubuntu@ip-172-31-34-189: ~/workspace/Test
ubuntu@ip-172-31-34-189:~/workspace/Test$ pwd
/home/ubuntu/workspace/Test 
```

Now go back to configuring the job.

Step 8: Click on **Build**, then go to **Execute shell**

```
sudo docker rm -f $(sudo docker ps -a -q)
sudo docker build /home/ubuntu/workspace/Test -t test
sudo docker run -it -p 82:80 -d test 
```

Build

Execute shell

Command

```
sudo docker rm -f $(sudo docker ps -a -q)
sudo docker build /home/ubuntu/workspace/Test -t test
sudo docker run -it -p 82:80 -d test
```

See [the list of available environment variables](#)

Advanced...

Click on save.

Before building our job again we must add one arbitrary container in slave-1.

Step 9: Add container by performing the following command.

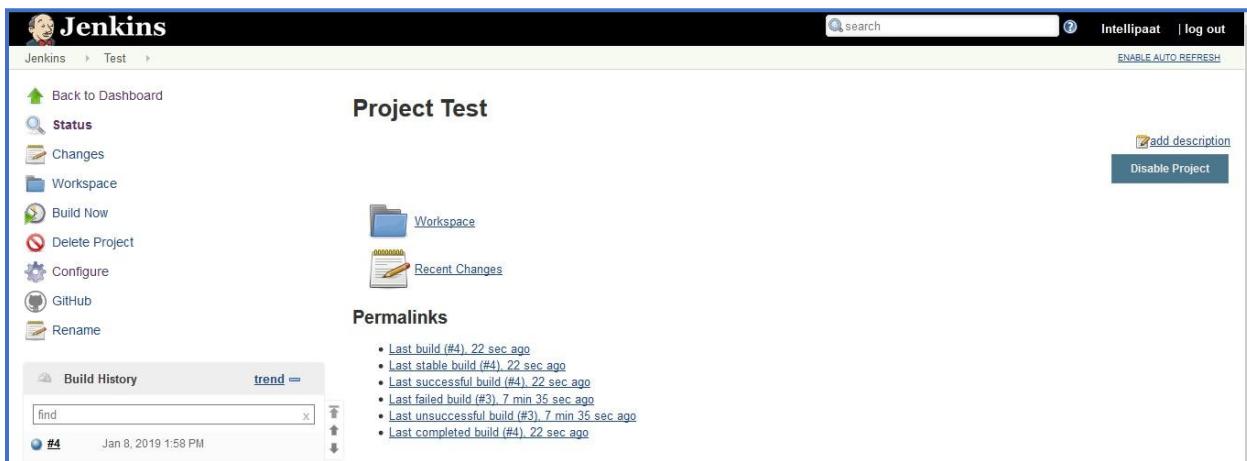
```
$ sudo docker run -it -d ubuntu
```

```
ubuntu@ip-172-31-34-189:~/workspace/Test
ubuntu@ip-172-31-34-189:~/workspace/Test$ sudo docker run -it -d ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
84ed7d2f608f: Pull complete
be2bf1c4a48d: Pull complete
a5bdc6303093: Pull complete
e9055237d68d: Pull complete
Digest: sha256:868fd30a0e47b8d8ac485df174795b5e2fe8a6c8f056cc707b232d65b8a1ab68
Status: Downloaded newer image for ubuntu:latest
ebe701788bff916b06db2a9bf7ad34b4c7cf3e722e8e789clea6deaf5ee2beaf
ubuntu@ip-172-31-34-189:~/workspace/Test$
```

Now we have added in a container.

```
ubuntu@ip-172-31-34-189:~/workspace/Test
ubuntu@ip-172-31-34-189:~/workspace/Test$ sudo docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS
ebe701788bff        ubuntu      "/bin/bash"   About a minute ago   Up About a minute
elaxed_varahamihira
ubuntu@ip-172-31-34-189:~/workspace/Test$
```

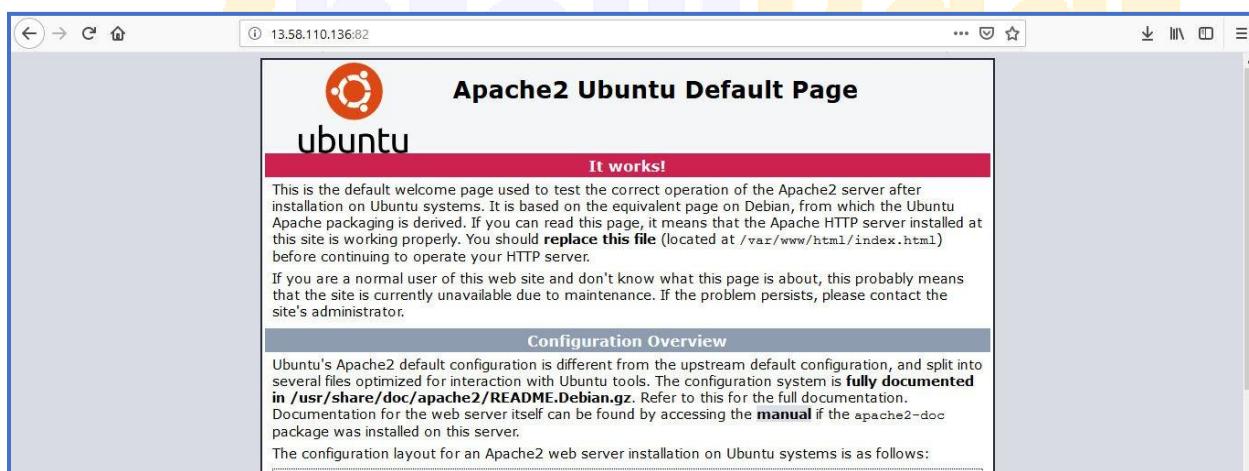
Step 10: Now open Jenkins Dashboard and **build the project.**



The screenshot shows the Jenkins Project Test dashboard. On the left, there's a sidebar with links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, GitHub, and Rename. The main area is titled "Project Test" and contains sections for "Workspace" (with a link to "Recent Changes") and "Permalinks" (listing recent builds). A "Build History" section shows the last six builds, all of which were successful. The date "Jan 8, 2019 1:58 PM" is also visible.

Building was successful.

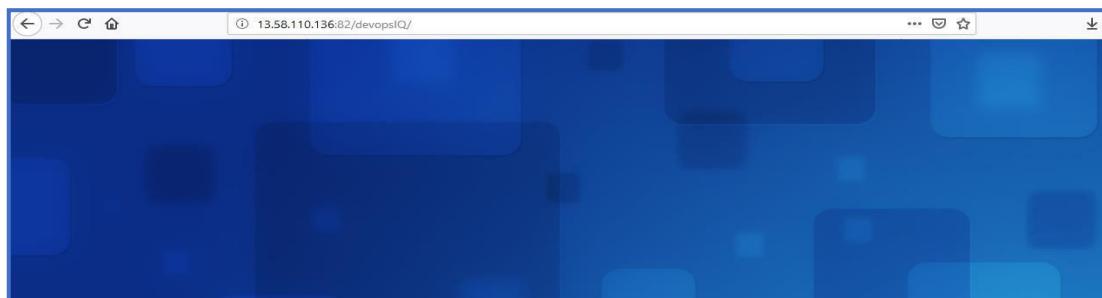
Step 11: Now open browser and enter **Slave-1 IP:82**



The screenshot shows a web browser window with the URL "13.58.110.136:82". The page title is "Apache2 Ubuntu Default Page" and features the Ubuntu logo. A red banner at the top says "It works!". Below it, text explains that this is the default welcome page for testing the Apache2 server. It includes instructions for replacing the index.html file and contact information for administrators. A "Configuration Overview" section provides details about the configuration files used.

This is the apache page that means our container is working perfectly.

Step 12: Now enter **slave-1 IP:82/devopsIQ/** in the browser.

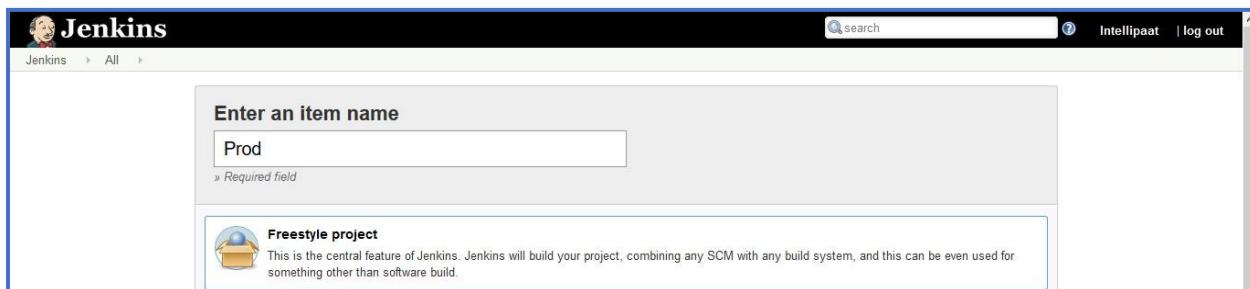


The screenshot shows a web browser window with the URL "13.58.110.136:82/devopsIQ/". The page is mostly blue and appears to be a placeholder or a loading screen for the devopsIQ application.

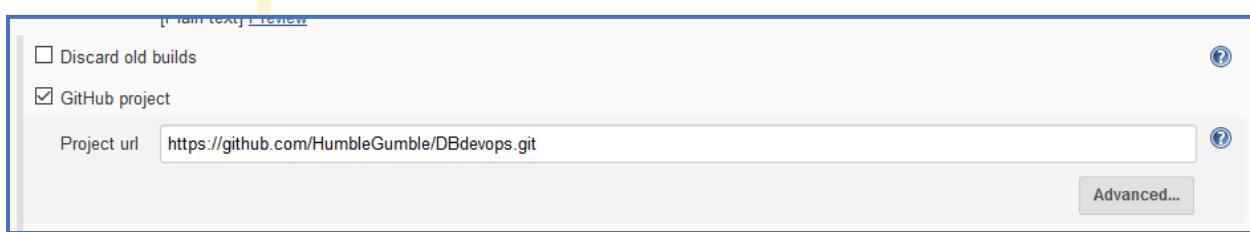
This looks fine.

Now, we will create a new project.

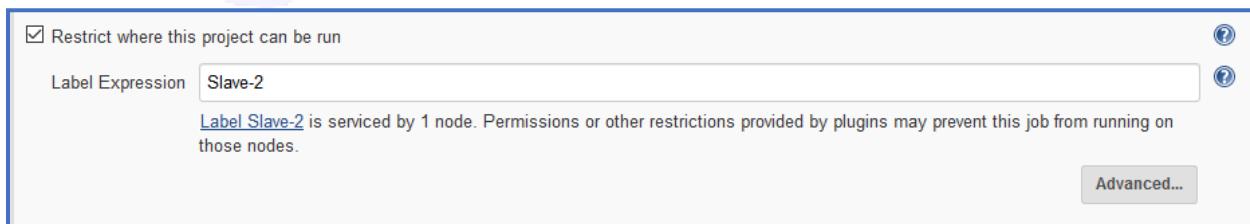
Step 13: Create a new project.



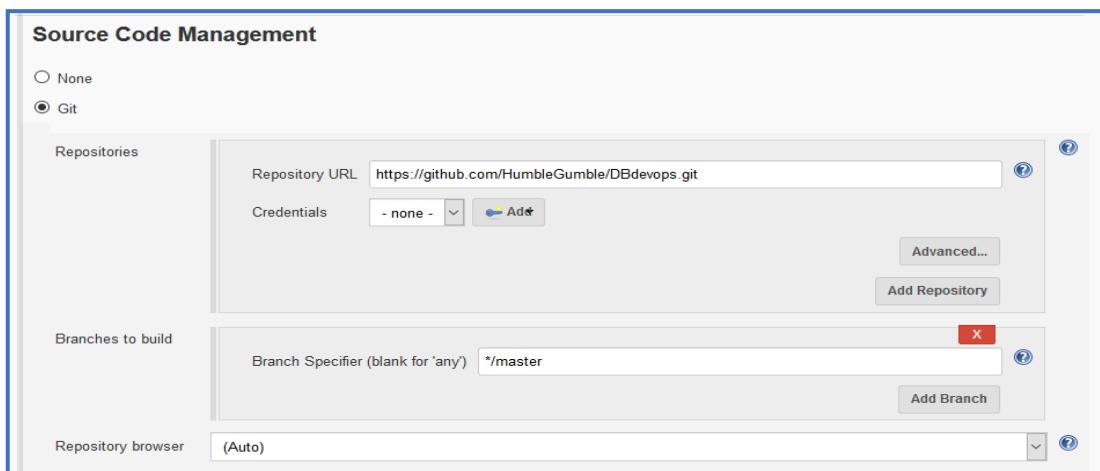
Step 14: Click on git project. Enter the git hub repository URL.



Step 15: Click on **Restrict where this project can be run** enter **Slave-2**.



Step 16: Go to Source code management enter the git repository URL there as well.

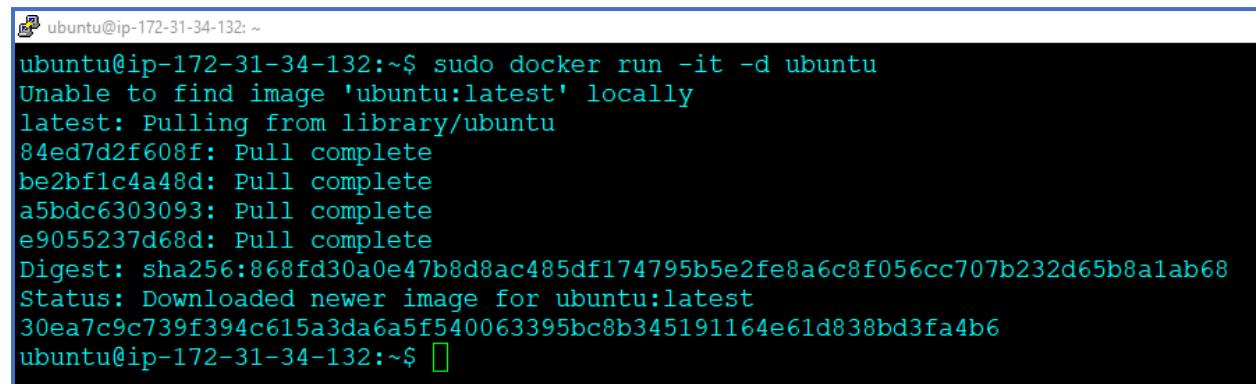


Step 17: Now enter the following command in the **Execution shell**

```
sudo docker rm -f $(sudo docker ps -a -q)  
sudo docker build /home/ubuntu/workspace/Prod -t production  
sudo docker run -it -p 82:80 -d production
```

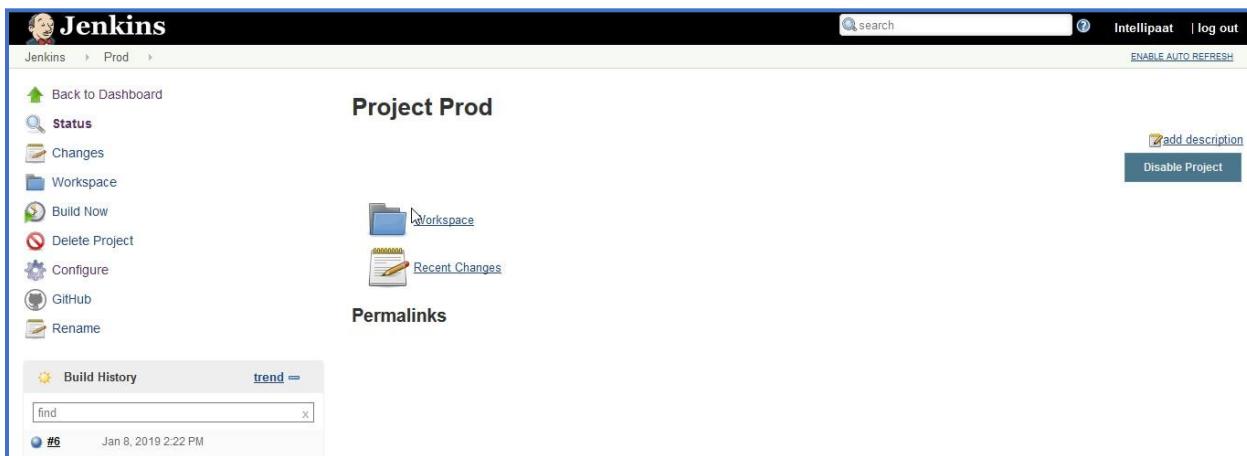
**Step 18:** Again, add one container to the Slave-2 as shown below.

```
$ sudo docker run -it -d ubuntu
```

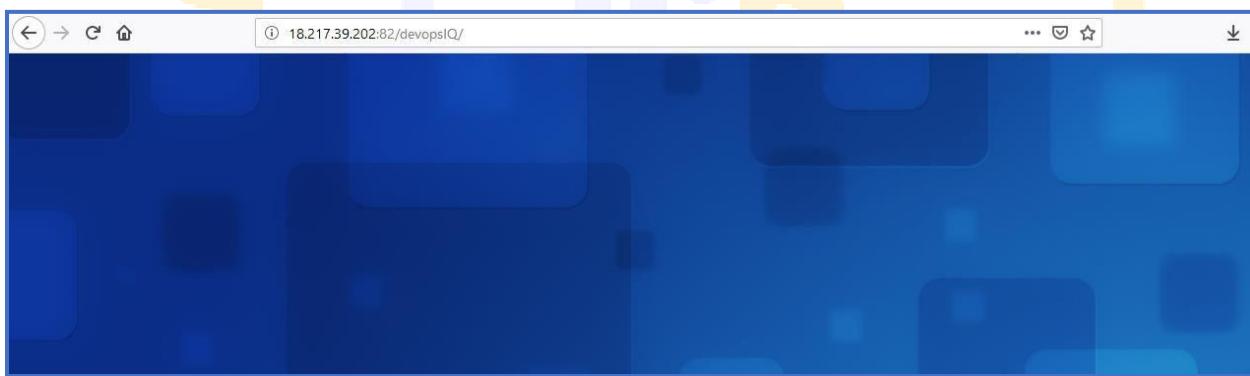


```
ubuntu@ip-172-31-34-132:~$ sudo docker run -it -d ubuntu  
Unable to find image 'ubuntu:latest' locally  
latest: Pulling from library/ubuntu  
84ed7d2f608f: Pull complete  
be2bf1c4a48d: Pull complete  
a5bdc6303093: Pull complete  
e9055237d68d: Pull complete  
Digest: sha256:868fd30a0e47b8d8ac485df174795b5e2fe8a6c8f056cc707b232d65b8a1ab68  
Status: Downloaded newer image for ubuntu:latest  
30ea7c9c739f394c615a3da6a5f540063395bc8b345191164e61d838bd3fa4b6  
ubuntu@ip-172-31-34-132:~$
```

Now that we have added an arbitrary container, go to Jenkins Dashboard and build the project.

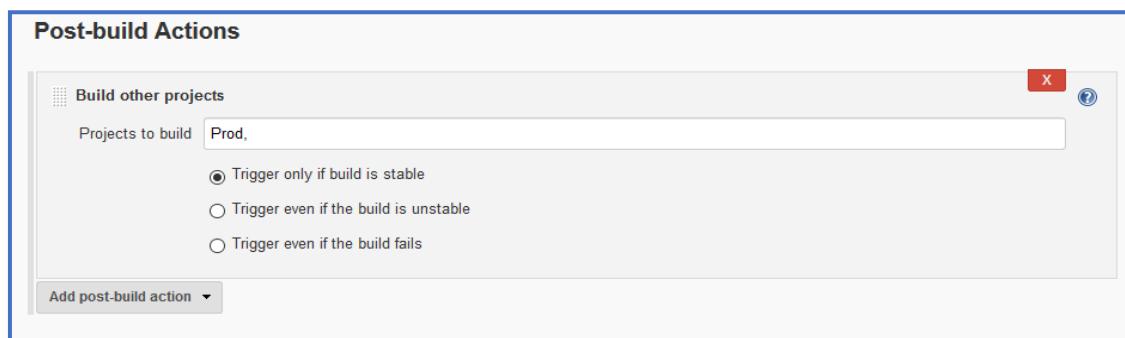
Step 19: Build the project **Prod**.A screenshot of the Jenkins interface showing the 'Project Prod' page. The left sidebar contains links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'GitHub', and 'Rename'. The main area shows 'Project Prod' with a 'Workspace' icon and a 'Recent Changes' link. Below is a 'Build History' section with one entry: '#6 Jan 8, 2019 2:22 PM'. Top right buttons include 'Add description' and 'Disable Project'.

Our Project building was successful.

Step 20: Now go to the browser and enter **Slave-2 IP:82/devopsIQ/**

It's working!

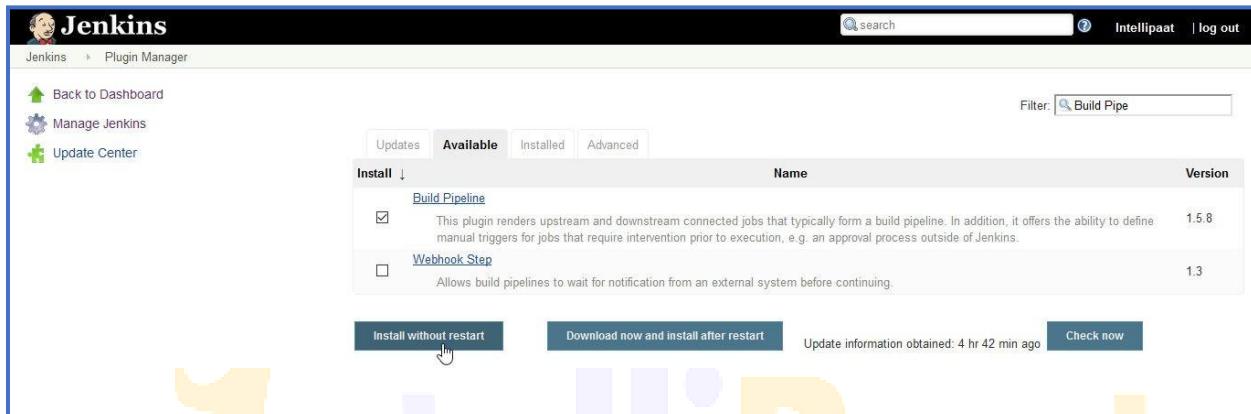
Now we will be triggered **Prod** job only when **Test** job will be completed.

Step 21: Go to the **Test** job, click on **Configure**. Add **Post-Build Actions**. Then go to **Build Other Projects**.A screenshot of the Jenkins 'Post-build Actions' configuration screen. It shows a 'Build other projects' section with 'Prod' selected in the 'Projects to build' dropdown. Under 'Trigger only if build is stable' (selected), there are three options: 'Trigger only if build is stable', 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. A 'Add post-build action' button is at the bottom.

Click on **Save**.

Now we will run jobs using pipeline.

Step 22: Go to the manage jenkins, click on available, search for Build Pipeline. Click on install without restart.



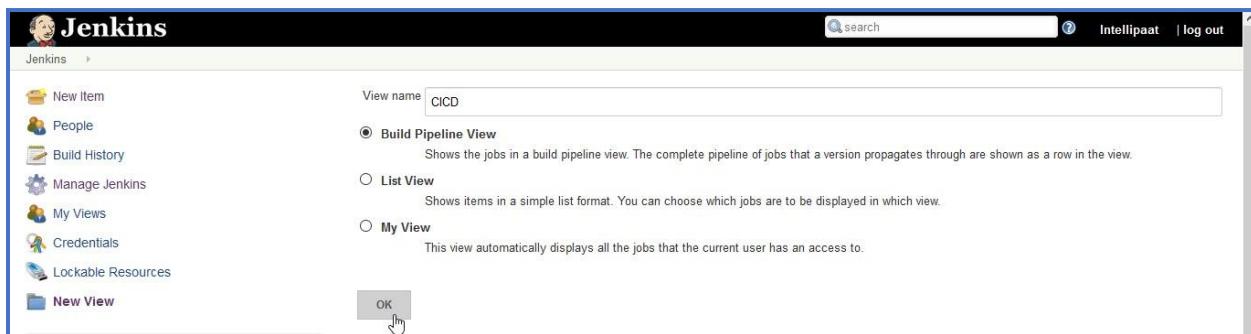
The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right contains the text 'Build Pipe'. A table lists two plugins: 'Build Pipeline' (version 1.5.8) and 'Webhook Step' (version 1.3). The 'Build Pipeline' row has a checkbox checked under 'Install'. Below the table are three buttons: 'Install without restart' (highlighted with a cursor), 'Download now and install after restart', and 'Check now'. A status message 'Update information obtained: 4 hr 42 min ago' is displayed.

Step 23: Go to the jenkins dashboard. Click on the +.



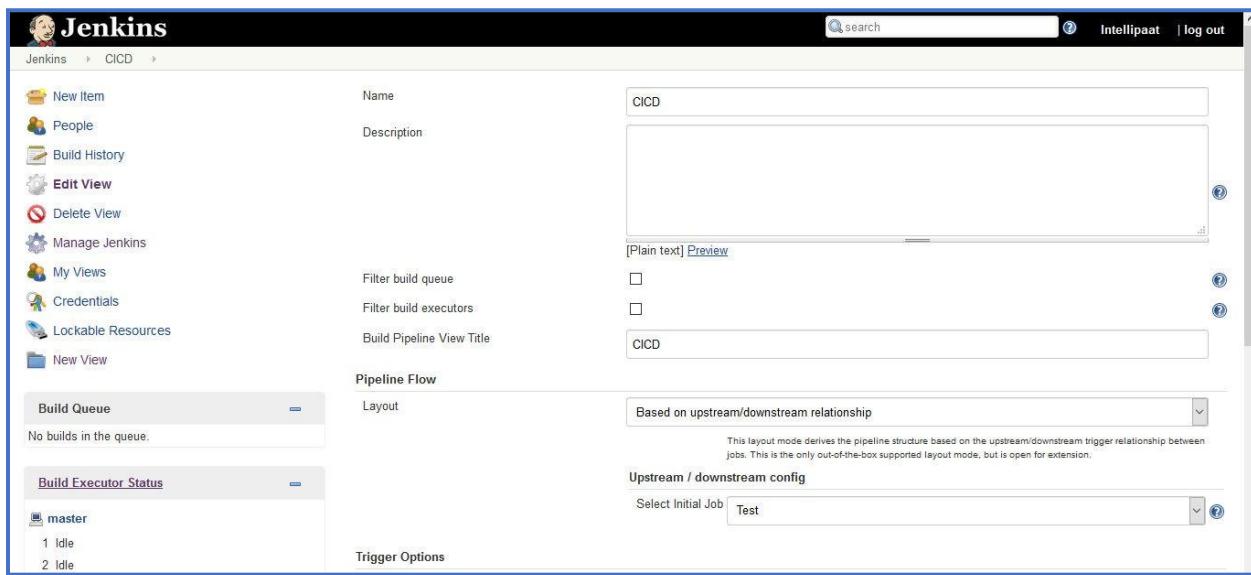
The screenshot shows the Jenkins dashboard. On the left, a sidebar lists options like 'New Item', 'People', 'Build History', etc. The main area displays a table of existing views: 'Prod' (Last Success: 23 min - #6, Last Failure: N/A, Last Duration: 10 sec) and 'Test' (Last Success: 47 min - #4, Last Failure: 55 min - #3, Last Duration: 10 sec). Below the table are icons for 'S' (Small), 'M' (Medium), and 'L' (Large). At the bottom, there's a legend for RSS feeds and a link to 'ENABLE AUTO REFRESH'. A cursor is hovering over the '+' icon in the 'New View' section of the sidebar.

Step 24: Enter view name and click ok.



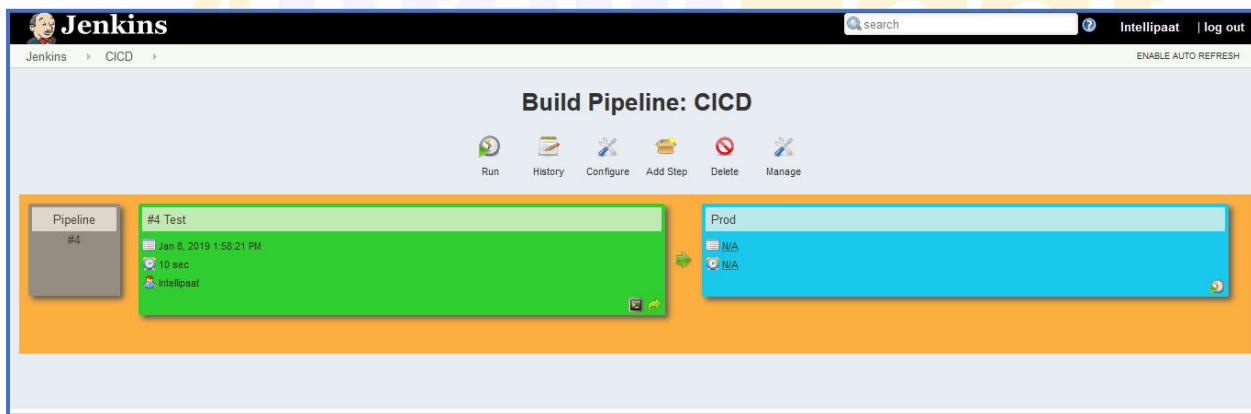
The screenshot shows the 'New View' dialog box. The 'View name' field is filled with 'CICD'. Below it, there are three radio button options: 'Build Pipeline View' (selected), 'List View', and 'My View'. Descriptions for each option are provided. At the bottom is an 'OK' button, which has a cursor hovering over it.

Step 25: There add the Build Pipeline View Title, then Select initial job as Test.



The screenshot shows the Jenkins CICD configuration page. On the left sidebar, under 'Edit View', there is a 'Build Pipeline View' section. In the 'Name' field, 'CICD' is entered. In the 'Build Pipeline View Title' field, 'CICD' is also entered. Under 'Pipeline Flow', the 'Layout' is set to 'Based on upstream/downstream relationship'. In the 'Upstream / downstream config' section, 'Select Initial Job' is set to 'Test'. The 'Trigger Options' section shows 'master' with 1 Idle and 2 Idle.

Click on ok. You should see the Pipeline Page like this.



The screenshot shows the Jenkins Build Pipeline: CICD page. It displays a pipeline structure with two stages: 'Test' (green) and 'Prod' (blue). Stage 'Test' is labeled '#4 Test' and has a status of 'N/A'. Stage 'Prod' is labeled '#1 Prod' and has a status of 'N/A'. The pipeline is triggered by 'IntelliPaat'. At the top, there are buttons for Run, History, Configure, Add Step, Delete, and Manage.

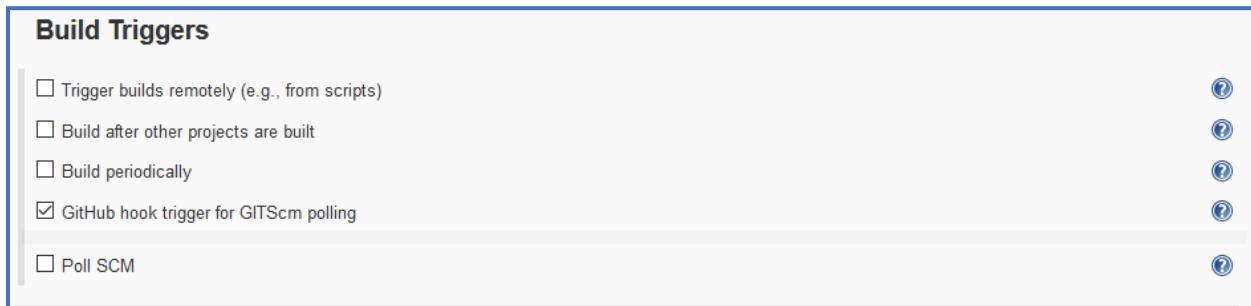
Step 26: Click on Run. Then Refresh the Page once.



The screenshot shows the Jenkins Build Pipeline: CICD page after running the pipeline. Stage 'Test' is now labeled '#5 Test' and has a status of 'N/A'. Stage 'Prod' is now labeled '#2 Prod' and has a status of 'N/A'. The pipeline is triggered by 'IntelliPaat'. The 'Run' button is highlighted in red, indicating it was just clicked.

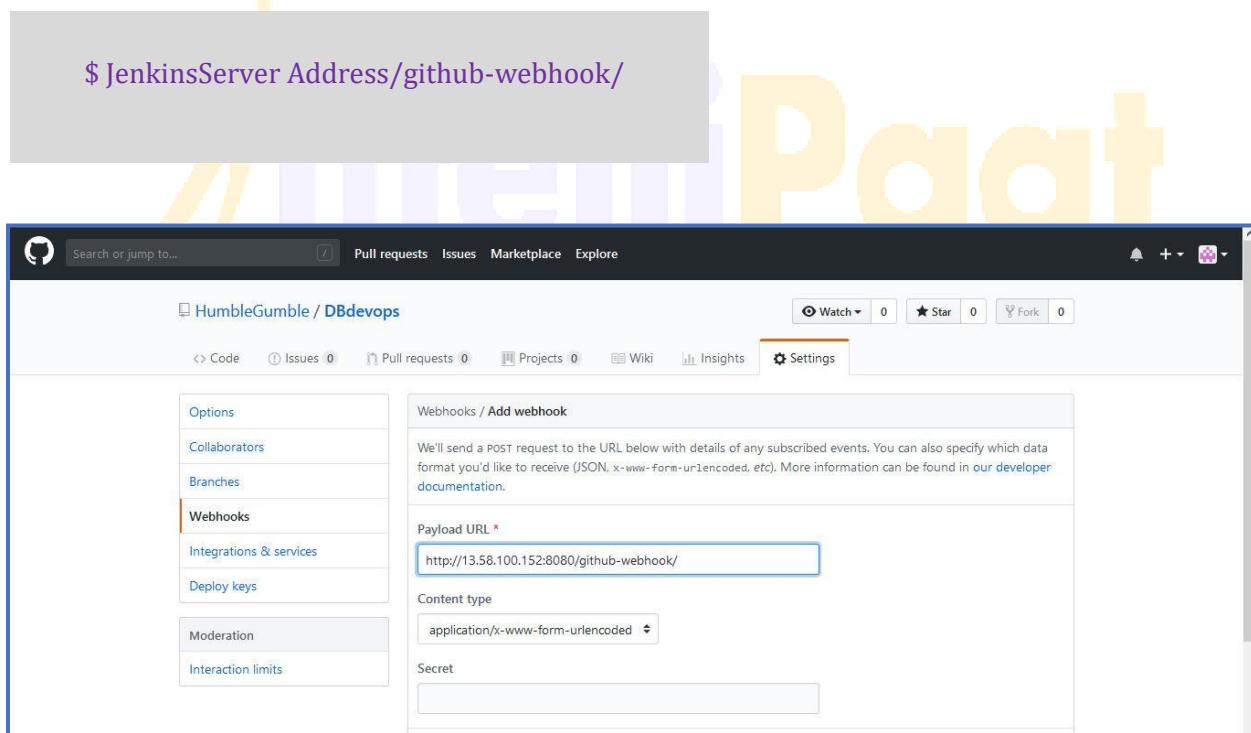
Now we will commit on GitHub, which should trigger our Jenkins Job.

Step 27: Go to the Jenkins Dashboard. Click on Test and then Configure. Check the **GitHub hook trigger for GITScm polling** option.



The screenshot shows the Jenkins Build Triggers configuration page. The 'GitHub hook trigger for GITScm polling' option is checked, while other options like 'Trigger builds remotely' and 'Poll SCM' are unchecked.

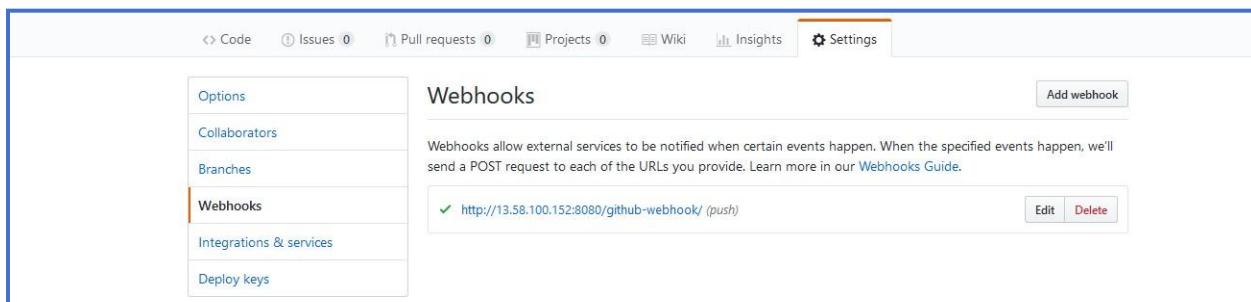
Step 28: Now configure GitHub Webhook. Go to settings, then click on Webhooks, then add webhooks. There insert the Jenkins Server Address as shown.



The screenshot shows the GitHub Settings - Webhooks page. A new webhook is being configured with the following details:

- Payload URL:** http://13.58.100.152:8080/github-webhook/
- Content type:** application/x-www-form-urlencoded
- Secret:** (empty field)

Click on Add webhook. You should see this.



The screenshot shows the GitHub Settings - Webhooks page after adding the webhook. The list now includes:

- ✓ http://13.58.100.152:8080/github-webhook/ (push)

Step 29: Go to the master terminal to trigger a built.

```
$ git clone <git repository URL>
```

```
ubuntu@ip-172-31-39-127:~$ git clone https://github.com/HumbleGumble/DBdevops.git
Cloning into 'DBdevops'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 83 (delta 15), reused 83 (delta 15), pack-reused 0
Unpacking objects: 100% (83/83), done.
ubuntu@ip-172-31-39-127:~$
```

Step 30: Now we will try to modify the website from the master terminal. Go to the master terminal and then go to the devopsIQ directory where you can find index.html file. Open it for modification

```
$ nano index.html
```

```
ubuntu@ip-172-31-39-127:~/DBdevops$ ls
Dockerfile  devopsIQ
ubuntu@ip-172-31-39-127:~/DBdevops$ cd devopsIQ
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$ ls
images  index.html
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$
```

Step 31: Make the modification in the **title** and **body** of that html file as shown below.

```
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$ nano index.html
GNU nano 2.9.3
<html>
<title>Jenkins New Website</title>
<body background="images/2.jpeg">
</body>
</html>
```

Step 32: Finally, perform git add and git commit.

```
$ git add
```

```
$ git commit -m "new commit"
```

```
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$ git add .
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$ git commit -m "new commit"
[master 74afca0] new commit
Committer: Ubuntu <ubuntu@ip-172-31-39-127.us-east-2.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

git config --global --edit

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

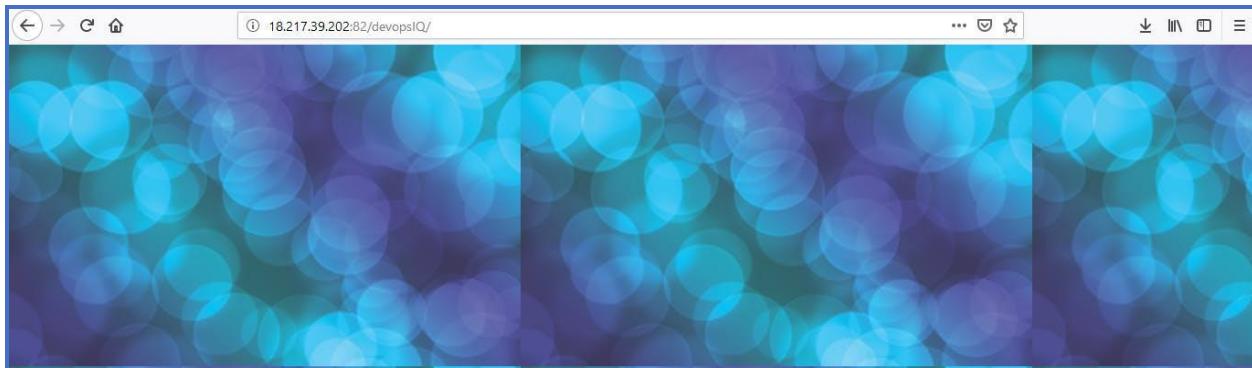
1 file changed, 2 insertions(+), 2 deletions(-)
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$
```

Step 33: Perform git push.

```
$ git push origin master
```

```
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$ git push origin master
Username for 'https://github.com': HumbleGumble
Password for 'https://HumbleGumble@github.com':
Counting objects: 4, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 496 bytes | 496.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/HumbleGumble/DBdevops.git
  55d4c57..74afca0 master -> master
ubuntu@ip-172-31-39-127:~/DBdevops/devopsIQ$
```

Step 34: Go to the browser. Refresh it. And you can see the background image got changed.



Congratulations! You have successfully completed the hands on.

