



# JENKINS HANDS-ON

DevOps Certification Training

[support@intellipaat.com](mailto:support@intellipaat.com)  
+91-7022374614  
US: 1-800-216-8930(Toll Free)

## 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.

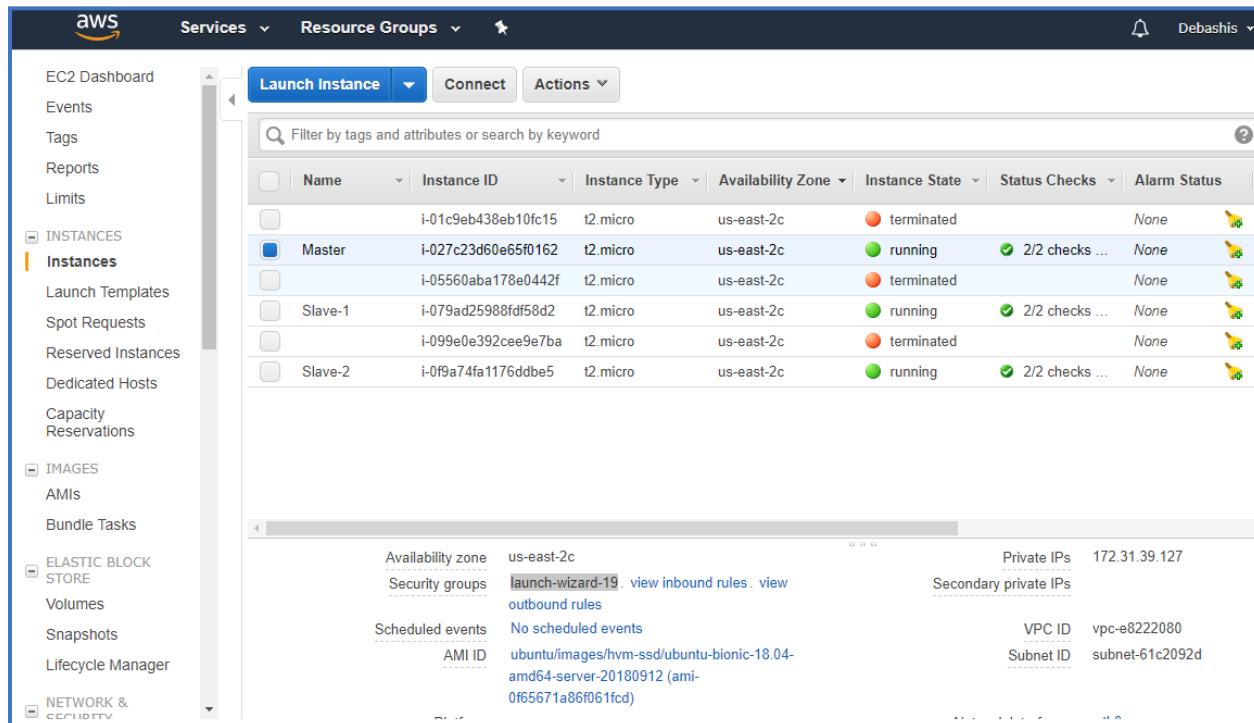
**Step 1:** Check the status of the Jenkins first.

```
$ service jenkins status
```

```
ubuntu@ip-172-31-39-127:~$ service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
  Active: active (exited) since Tue 2019-01-08 10:04:51 UTC; 1min 7s ago
    Docs: man:systemd-sysv-generator(8)
   Tasks: 0 (limit: 1152)
  CGroup: /system.slice/jenkins.service

Jan 08 10:04:49 ip-172-31-39-127 systemd[1]: Starting LSB: Start Jenkins at boot time...
Jan 08 10:04:50 ip-172-31-39-127 jenkins[11182]: Correct java version found
Jan 08 10:04:50 ip-172-31-39-127 jenkins[11182]: * Starting Jenkins Automation Server jenkins
Jan 08 10:04:50 ip-172-31-39-127 su[11228]: Successful su for jenkins by root
Jan 08 10:04:50 ip-172-31-39-127 su[11228]: + ??? root:jenkins
Jan 08 10:04:50 ip-172-31-39-127 su[11228]: pam_unix(su:session): session opened for user jenkins by (uid=0)
Jan 08 10:04:50 ip-172-31-39-127 su[11228]: pam_unix(su:session): session closed for user jenkins
Jan 08 10:04:51 ip-172-31-39-127 jenkins[11182]: ...done.
Jan 08 10:04:51 ip-172-31-39-127 systemd[1]: Started LSB: Start Jenkins at boot time.
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 Instances dashboard. On the left sidebar, under the 'Instances' section, 'Master' is selected. In the main pane, there is a table of instances. The 'Master' instance is highlighted with a blue selection bar. The table columns include Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, and Alarm Status. Below the table, detailed information for the selected 'Master' instance is shown, including its Availability zone (us-east-2c), Security groups (launch-wizard-19), Private IP (172.31.39.127), VPC ID (vpc-e8222080), and Subnet ID (subnet-61c2092d).

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



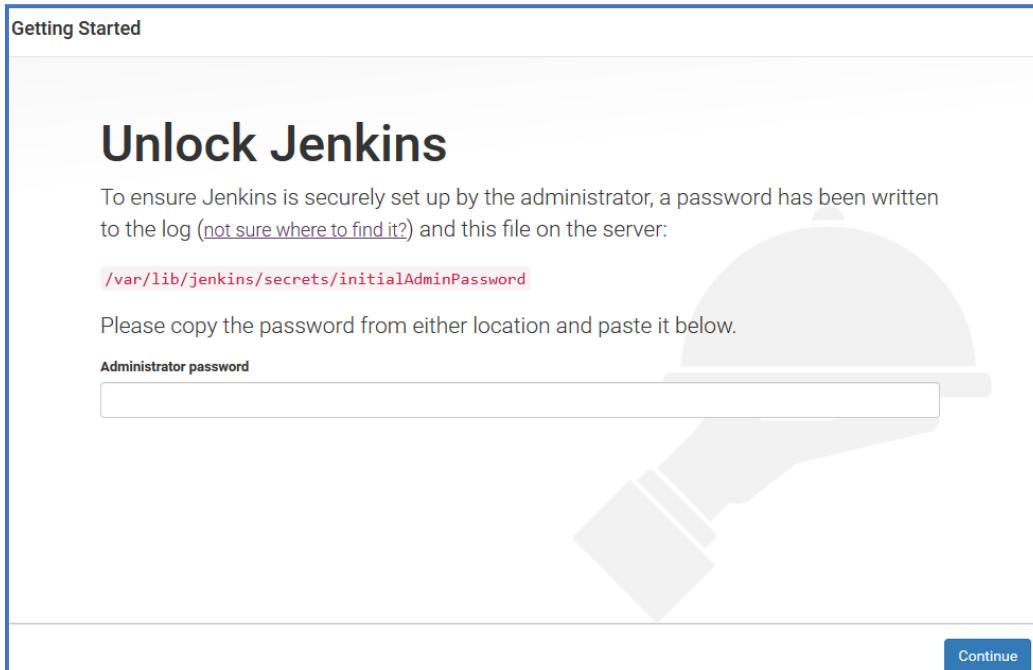
The screenshot shows the 'Edit inbound rules' dialog box. It contains two entries in the rule table:

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

Below the table, a note states: "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." At the bottom right are 'Cancel' and 'Save' buttons.

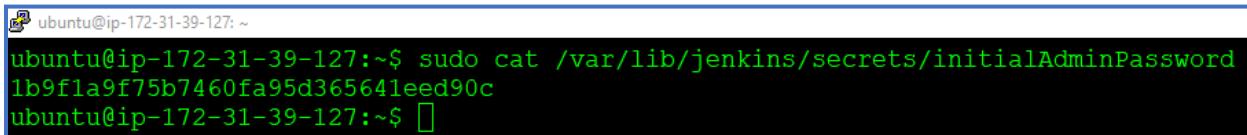
**Step 4:** Now open browser and enter **masterIP:8080**

You should land on a page like this:

A screenshot of a web browser showing the Jenkins 'Unlock Jenkins' setup page. The page has a light gray background with a large, semi-transparent circular watermark of a hand holding a wrench in the center. At the top left, there's a 'Getting Started' link. The main title is 'Unlock Jenkins'. Below it, a text block says: '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:' followed by the path '/var/lib/jenkins/secrets/initialAdminPassword'. A text input field labeled 'Administrator password' is provided for pasting the copied password. At the bottom right is a 'Continue' button.

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

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

A screenshot of a terminal window on an Ubuntu system. The command \$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword is entered, and its output is shown: 1b9f1a9f75b7460fa95d365641eed90c. The terminal prompt is 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:

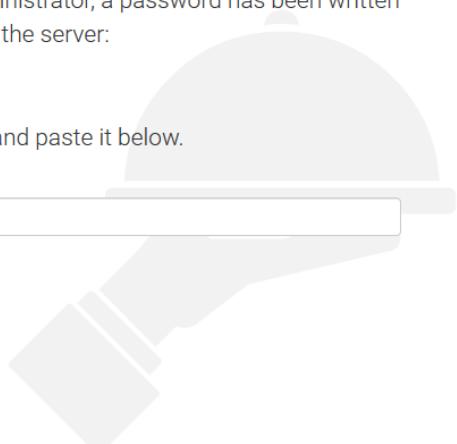
`/var/lib/jenkins/secrets/initialAdminPassword`

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:	Intellipaat
Password:	*****
Confirm password:	*****
Full name:	Intellipaat
E-mail address:	Intellipaat@gmail.com

Jenkins 2.150.1 [Continue as admin](#) [Save and Continue](#)

Then click on **Save and Continue**.

Getting Started

## Instance Configuration

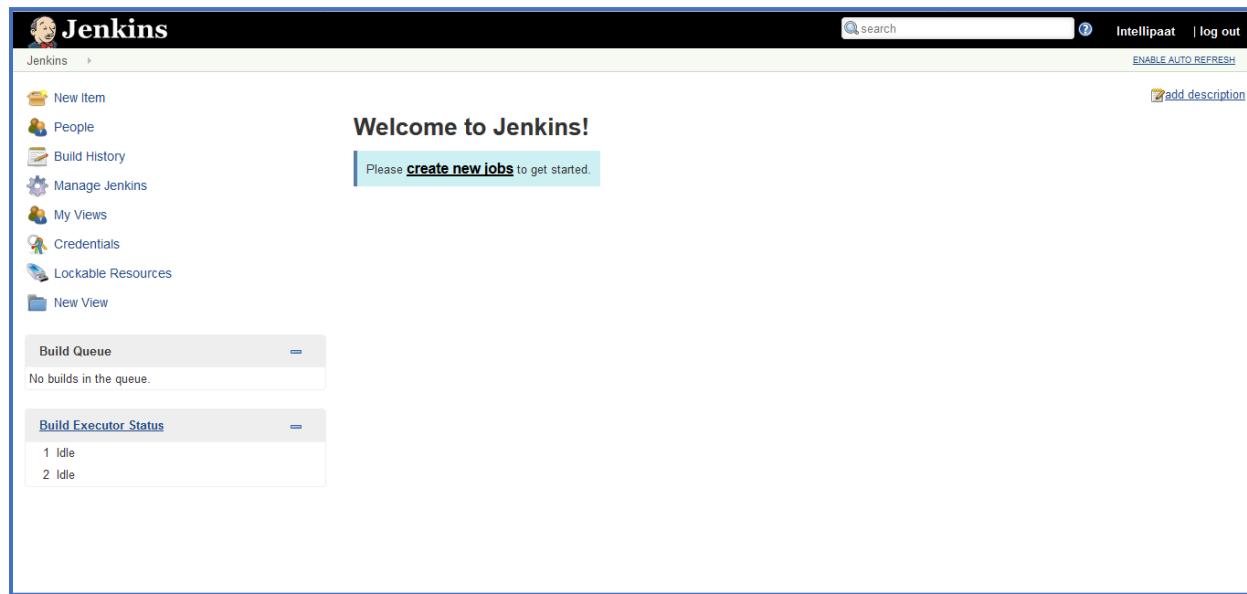
Jenkins URL:

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.

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.

Jenkins 2.150.1 Not now Save and Finish

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. At the top, there's a navigation bar with links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', 'Lockable Resources', and 'New View'. On the right side of the header, there are links for 'search', 'IntelliPaat', 'log out', and 'ENABLE AUTO REFRESH'. Below the header, the main content area has a title 'Welcome to Jenkins!' and a message 'Please [create new jobs](#) to get started.' There are two expandable sections: 'Build Queue' (which says 'No builds in the queue.') and 'Build Executor Status' (which shows '1 Idle' and '2 Idle').

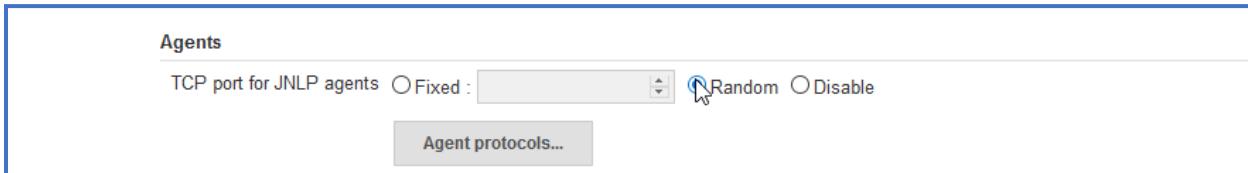
This is our **Jenkins Dashboard**.

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



The screenshot shows the Jenkins welcome screen. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is currently selected), 'My Views', 'Credentials', 'Lockable Resources', and 'New View'. Below that is a 'Build Queue' section stating 'No builds in the queue.' The main area has a 'Welcome to Jenkins!' message with a link to 'Create new jobs'. A dropdown menu is open under 'Manage Jenkins', showing options such as 'Configure System', 'Configure Global Security' (which is highlighted with a blue selection bar), 'Configure Credentials', 'Global Tool Configuration', 'Reload Configuration from Disk', 'Manage Plugins', 'System Information', and 'System Log'. At the top right, there are 'search', 'ENABLE AUTO REFRESH', and 'log out' buttons.

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



The screenshot shows the 'Agents' configuration page. It has a heading 'Agents' and a sub-section 'TCP port for JNLP agents'. There are three radio button options: 'Fixed' (with a dropdown menu), 'Random' (which is selected and highlighted with a blue selection bar), and 'Disable'. Below this is a 'Agent protocols...' button.

**Step 9:** Now go to **Manage Nodes**.

Welcome to Jenkins!

[Create new jobs](#) to get started.

- Configure System
- Configure Global Security
- Configure Credentials
- Global Tool Configuration
- Reload Configuration from Disk
- Manage Plugins
- System Information
- System Log
- Load Statistics
- Jenkins CLI
- Script Console
- Manage Nodes**
- About Jenkins
- Manage Old Data

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

Node name: Slave-1

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

OK

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

Name	Slave-1
Description	
# of executors	1
Remote root directory	
<b>Remote directory is mandatory</b>	
Labels	
Usage	Use this node as much as possible
Launch method	Launch agent agents via SSH Launch agent agents via SSH <b>Launch agent via Java Web Start</b> Launch agent via execution of command on the master

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

Launch method  Launch agent via Java Web Start

Disable WorkDir

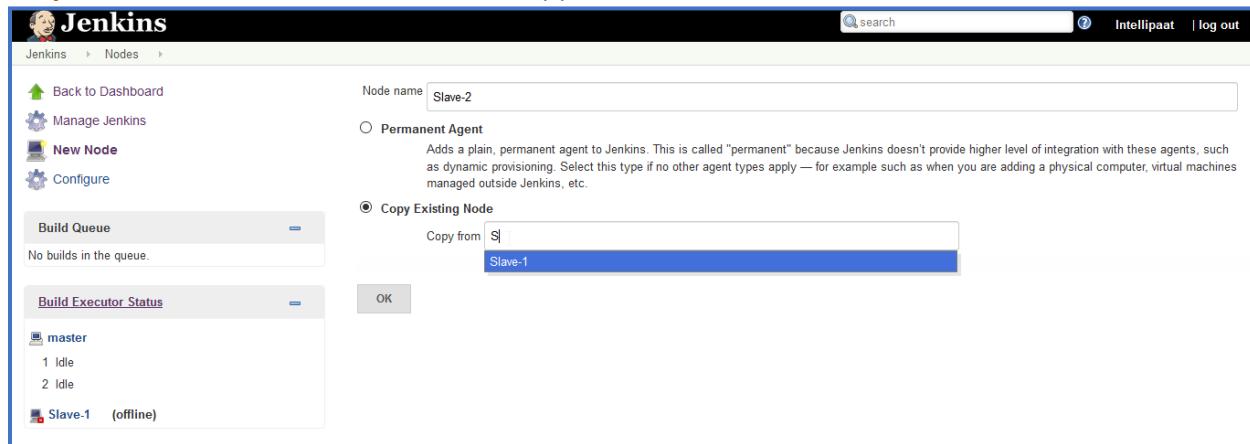
Custom WorkDir path

Internal data directory

Fail if workspace is missing

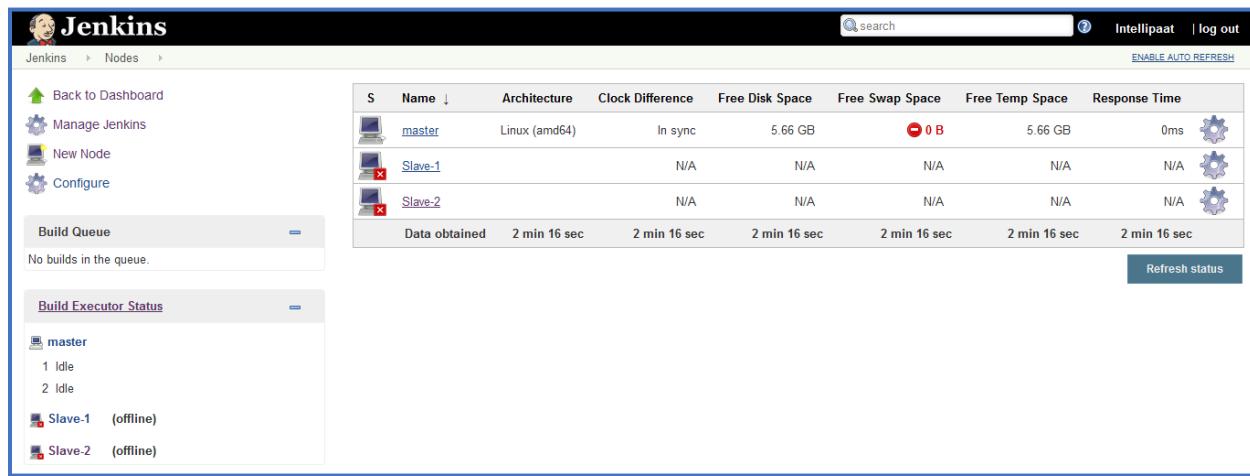
[Advanced...](#)

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



The screenshot shows the Jenkins Nodes configuration page. On the left sidebar, there are links for Back to Dashboard, Manage Jenkins, New Node (which is highlighted in yellow), and Configure. Under the Nodes section, there is a 'Build Queue' table showing 'No builds in the queue.' Below it is a 'Build Executor Status' table showing two nodes: 'master' (1 Idle, 2 Idle) and 'Slave-1' (offline). In the main content area, a 'Node name' input field contains 'Slave-2'. A radio button labeled 'Permanent Agent' is selected, with a descriptive text about plain permanent agents. Another radio button labeled 'Copy Existing Node' is selected, with a dropdown menu showing 'Slave-1' which is currently highlighted in blue. An 'OK' button is visible at the bottom of the dialog.

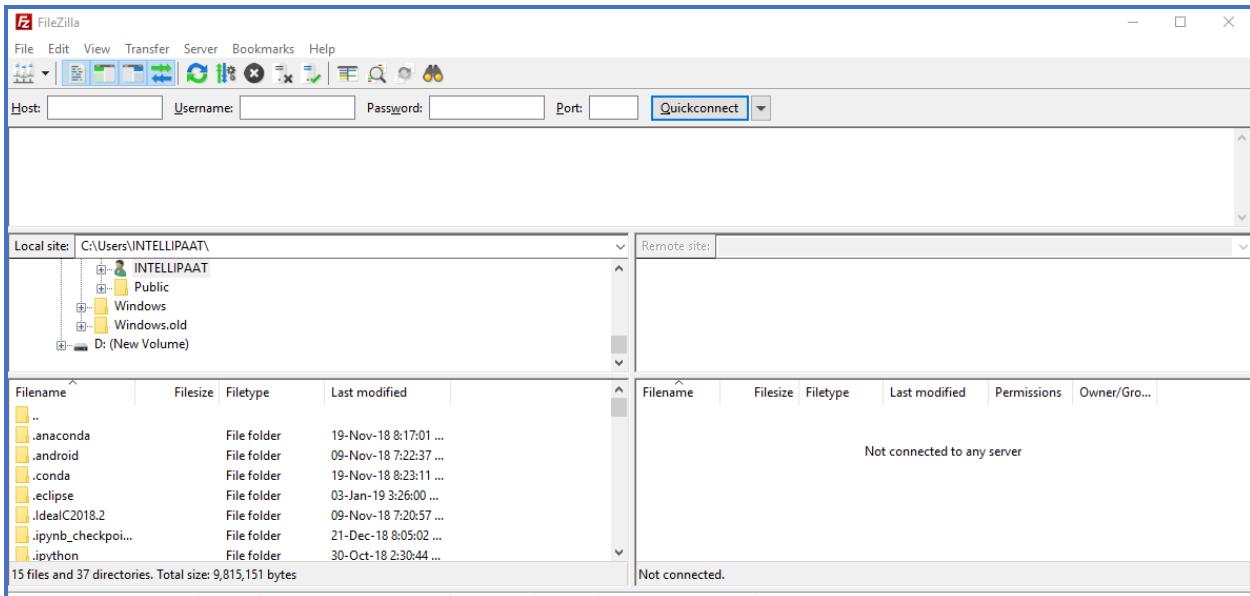
**Step 14:** Then click ok. You can see the list of nodes that we have on the Jenkins Dashboard.



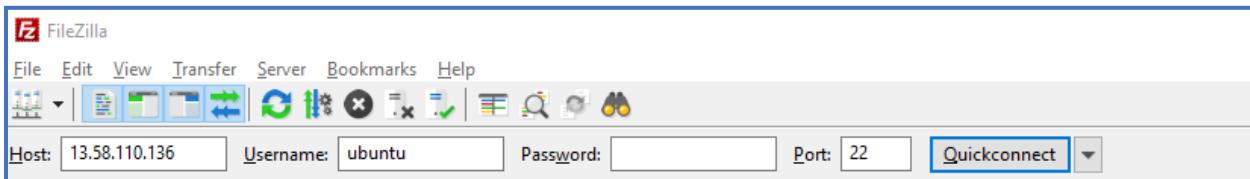
The screenshot shows the Jenkins Dashboard. On the left sidebar, there are links for Back to Dashboard, Manage Jenkins, New Node, and Configure. Under the Nodes section, there is a 'Build Queue' table showing 'No builds in the queue.' Below it is a 'Build Executor Status' table showing three nodes: 'master' (1 Idle, 2 Idle), 'Slave-1' (offline), and 'Slave-2' (offline). In the main content area, there is a table titled 'Nodes' with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The table contains three rows: '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, N/A), and 'Slave-2' (N/A, N/A, N/A, N/A, N/A, N/A). A 'Refresh status' button is located at the bottom right of the table.

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

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

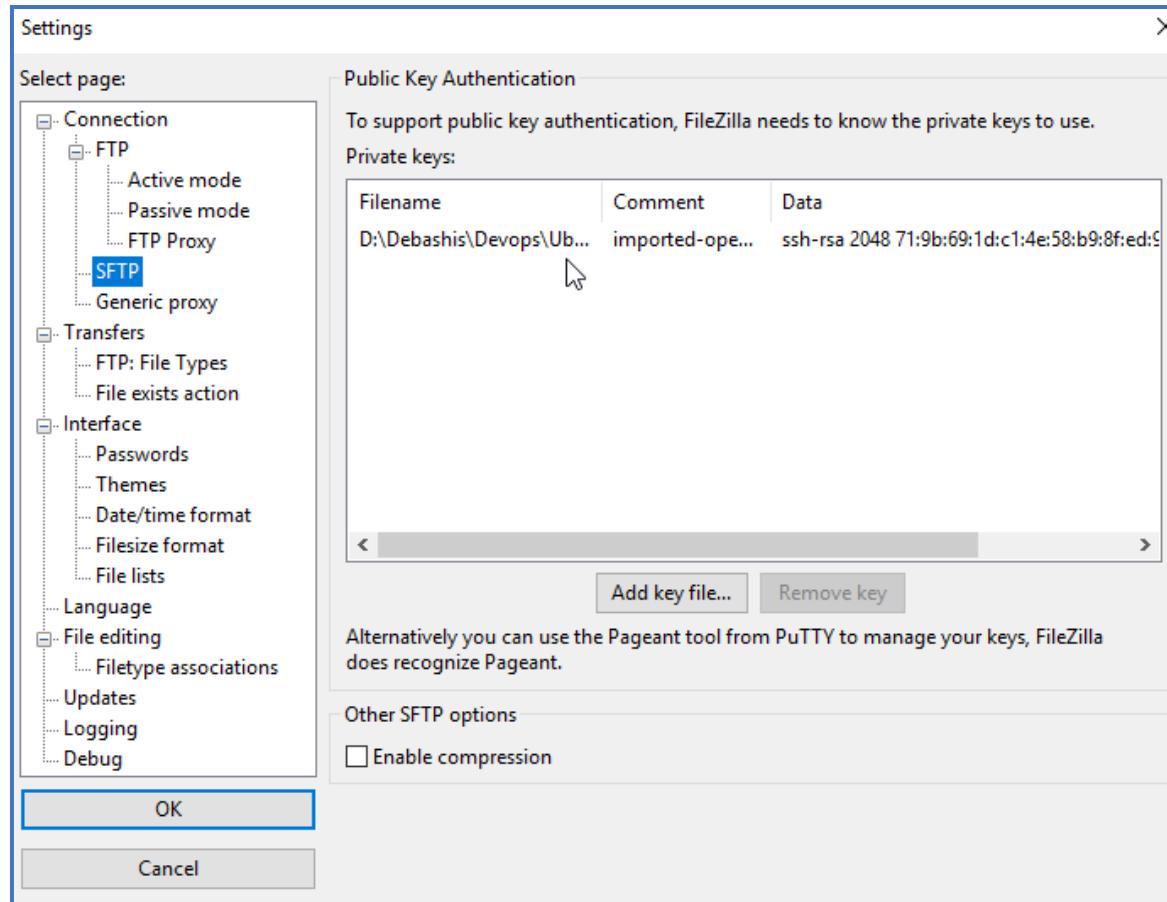


**Step 17:** Now copy the **Slave-1 IP address**. And add it as **Host**. Add **ubuntu** as **username**. Leave the password field empty. Add **Port** as **22**.

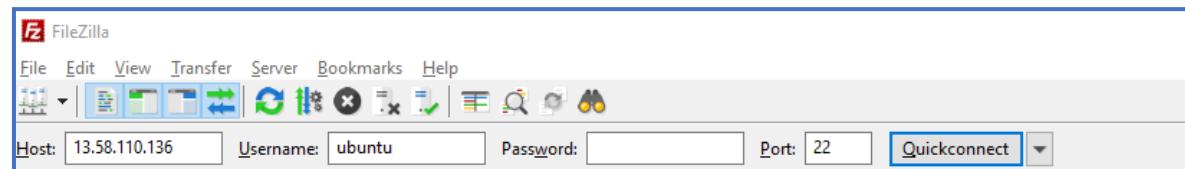


**Don't start the connection yet.**

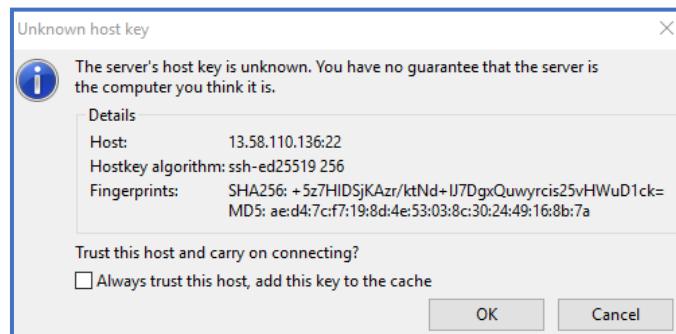
**Step 18:** Before we start the connection, we need to add the Private key (PPK file). Go to **Edit**, click on **Settings**. Click on **SFTP**. Add the PPK file there and click **ok**.



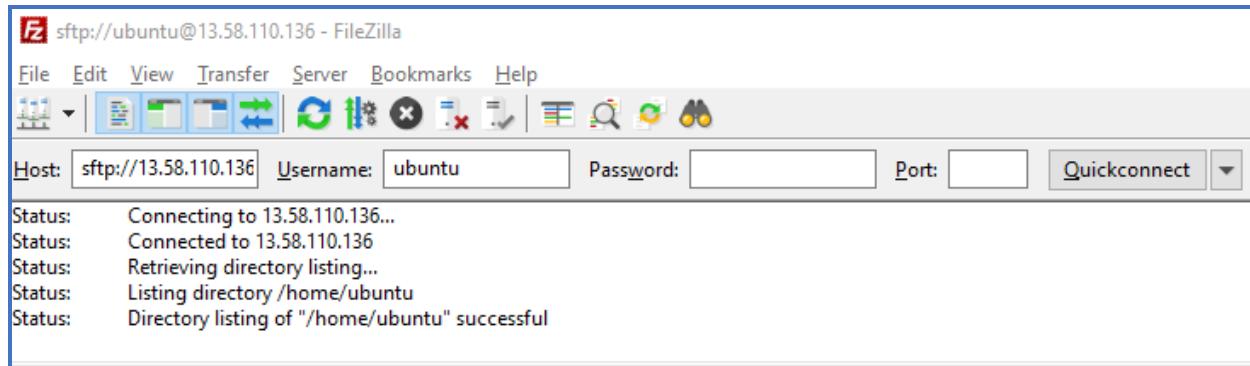
### Step 19: Click on Quickconnect.



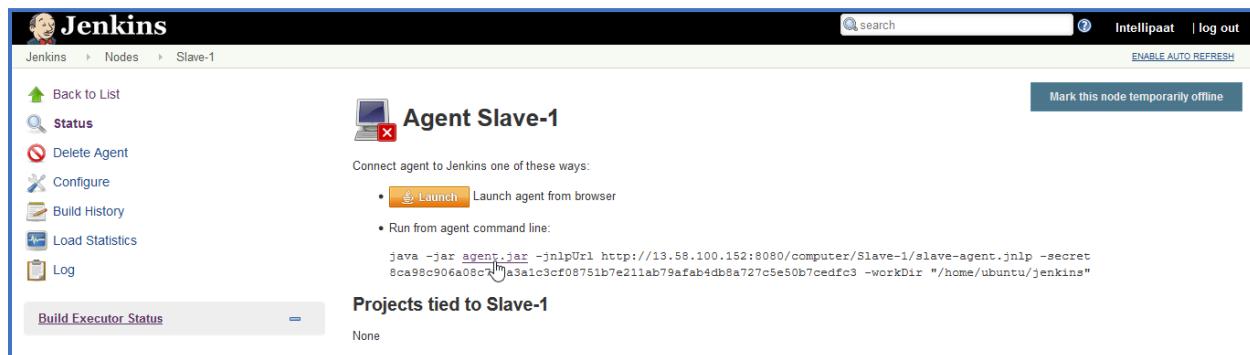
Then click on **ok**.



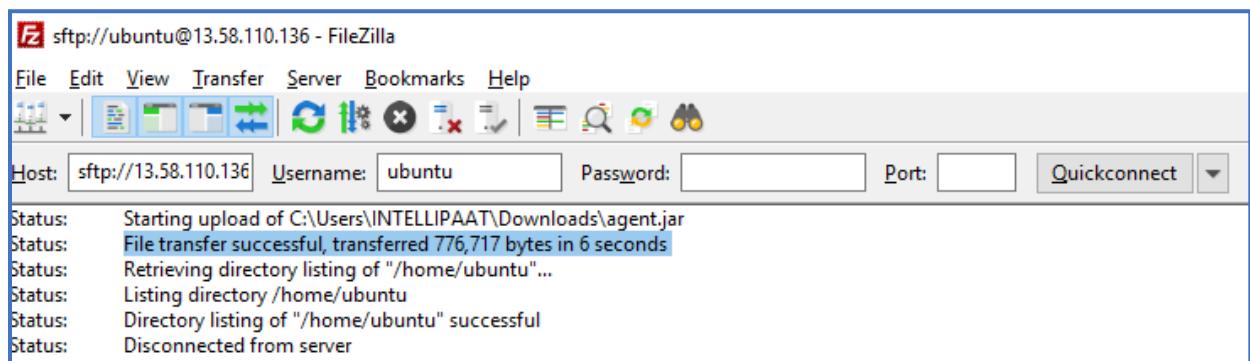
As you can see our connection is successful.



**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.

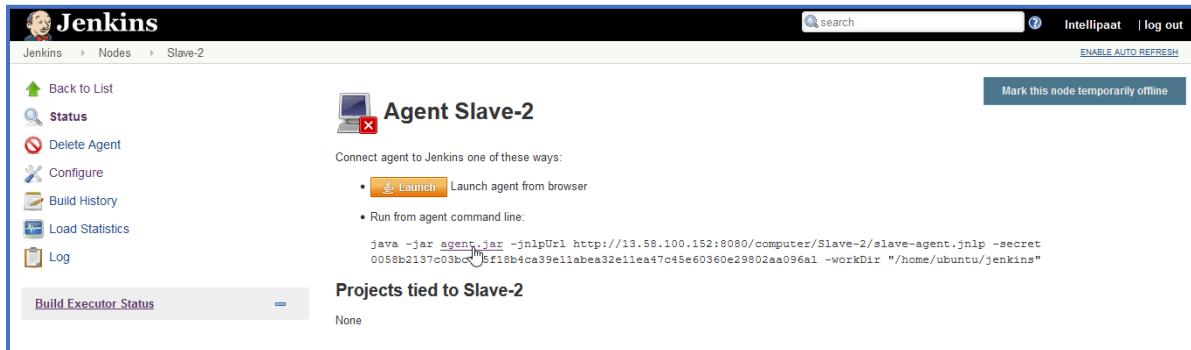


**Step 22:** Let us verify if the file has been transferred to **Slave-1** or not. Open a new session on putty. Connect to slave-1. Run **ls command**.

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

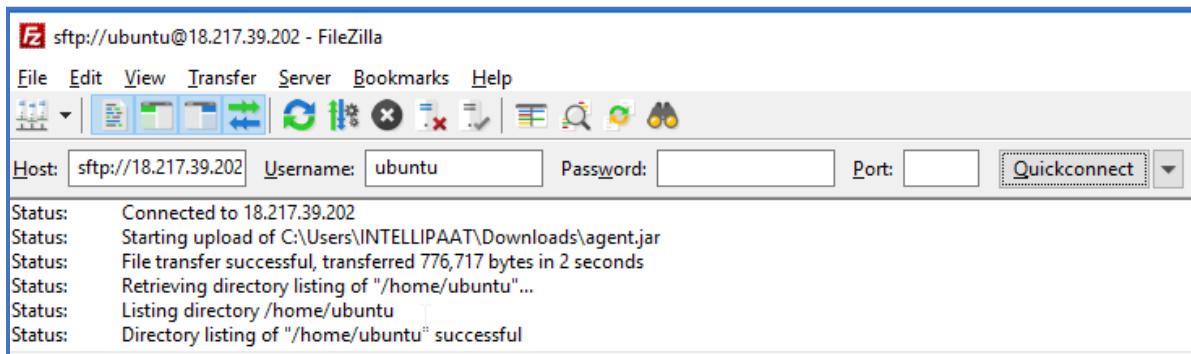
As you can see the agent.jar file appears there, which means our file has been successfully transferred to **Slave-1**.

**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 node configuration for 'Agent Slave-2'. The left sidebar includes options like Back to List, Status, Delete Agent, Configure, Build History, Load Statistics, and Log. The main area displays the node's status as 'Up' with a green icon. It provides instructions to connect via browser or command line, showing a sample Java command to run the agent. Below this, a section titled 'Projects tied to Slave-2' shows 'None'.

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



The screenshot shows the FileZilla interface with the connection details 'Host: sftp://18.217.39.202', 'Username: ubuntu', and 'Port: 22'. The status window at the bottom displays the progress of the file transfer, showing it connected to the host, starting the upload of 'agent.jar', and successfully transferring 776,717 bytes in 2 seconds. It also lists the directory listing of '/home/ubuntu'.

**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 file!

**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]
```

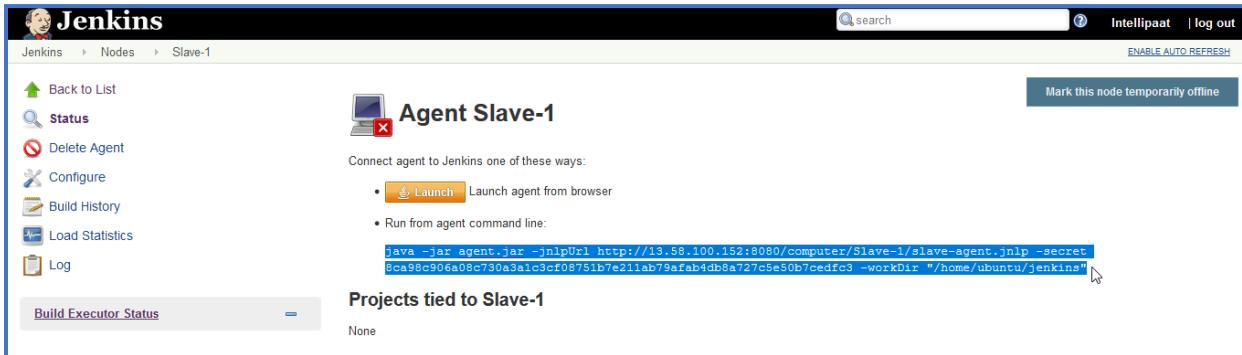
**Step 26:** Now install run the following installation command on both terminal.

```
$ 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-gsetting
  fontconfig-config fonts-dejavu-core fonts-dejavu-extra glib-network
  gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme
```

```
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.

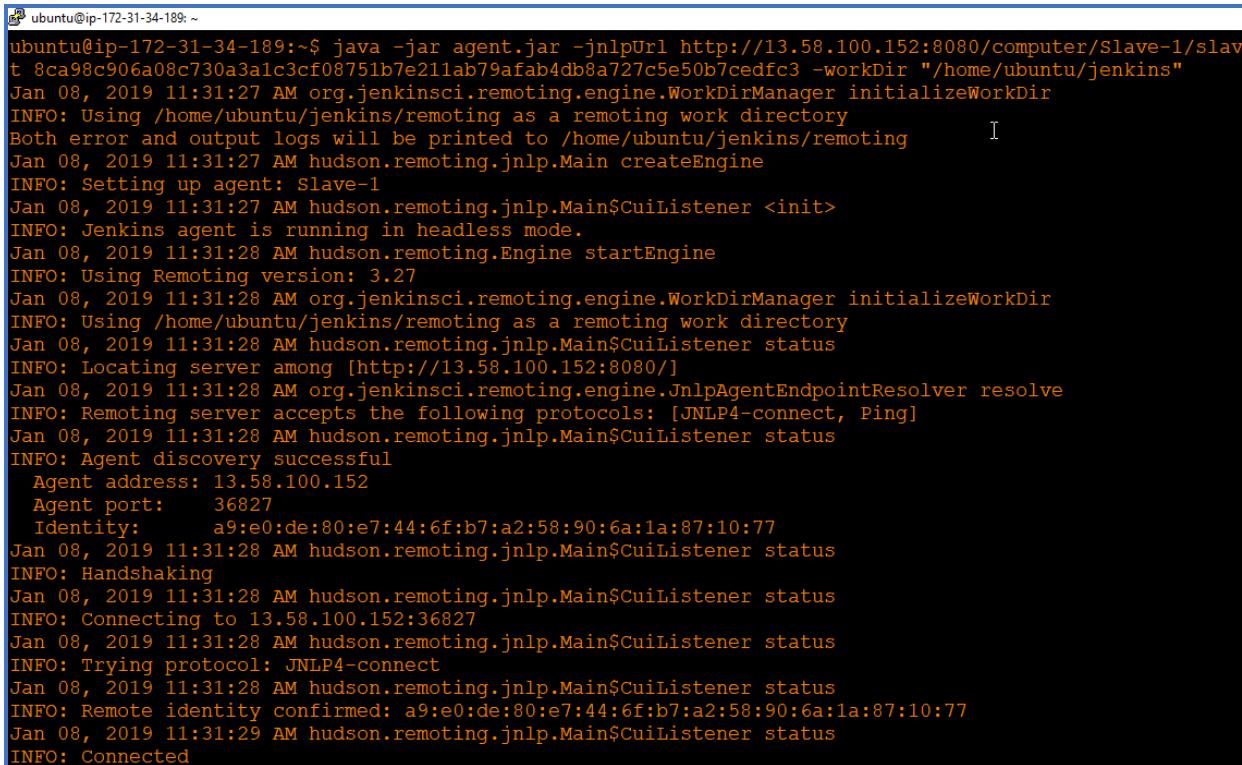


The screenshot shows the Jenkins interface for managing nodes. On the left, there's a sidebar with options like 'Back to List', 'Status', 'Delete Agent', 'Configure', 'Build History', 'Load Statistics', and 'Log'. The main area is titled 'Agent Slave-1' and displays the following information:

- Connect agent to Jenkins one of these ways:
  - Launch agent from browser (button)
  - Run from agent command line (link)
- Command line: `java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-1/slave-agent.jnlp -secret 8ca98c906a08c730a3a1c3cf08751b7e211ab79afab4db8a727c5e50b7cedfc3 -workDir "/home/ubuntu/jenkins"`

Below this, there's a section titled 'Projects tied to Slave-1' which says 'None'.

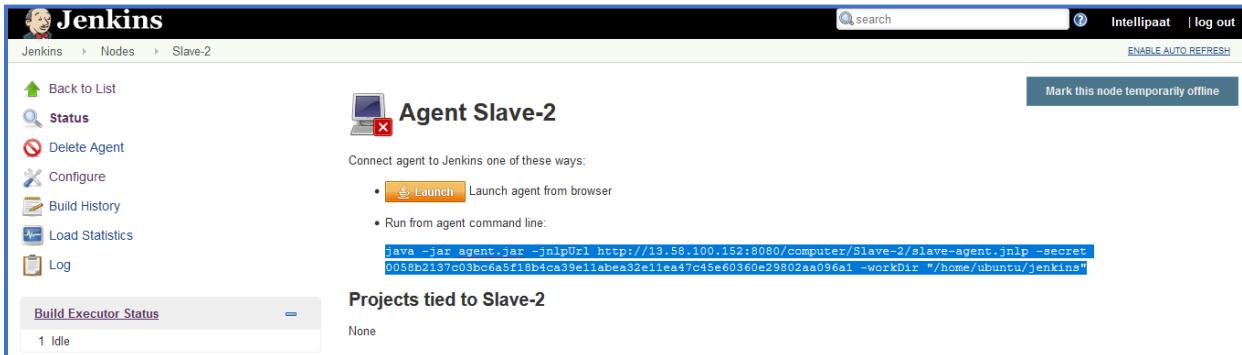
Run the command line from Slave-1 as shown below.



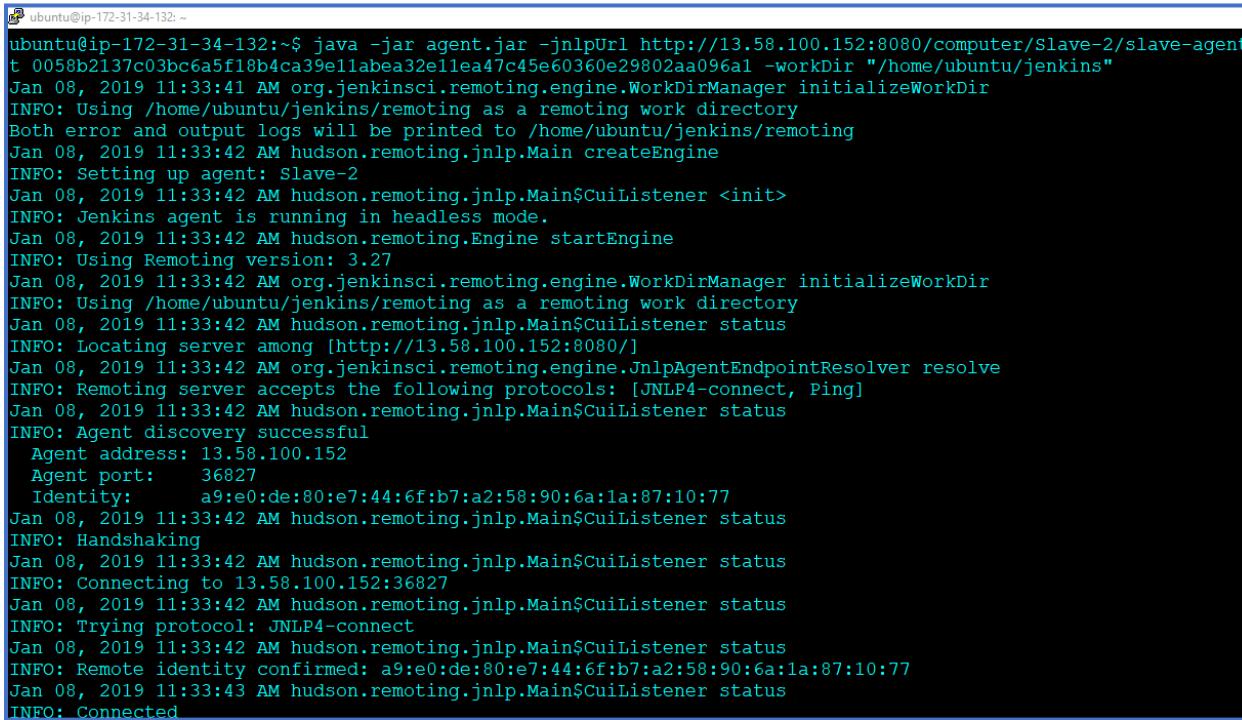
```
ubuntu@ip-172-31-34-189:~$ java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-1/slave-1/8ca98c906a08c730a3a1c3cf08751b7e211ab79afab4db8a727c5e50b7cedfc3 -workDir "/home/ubuntu/jenkins"
Jan 08, 2019 11:31:27 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Both error and output logs will be printed to /home/ubuntu/jenkins/remoting
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.

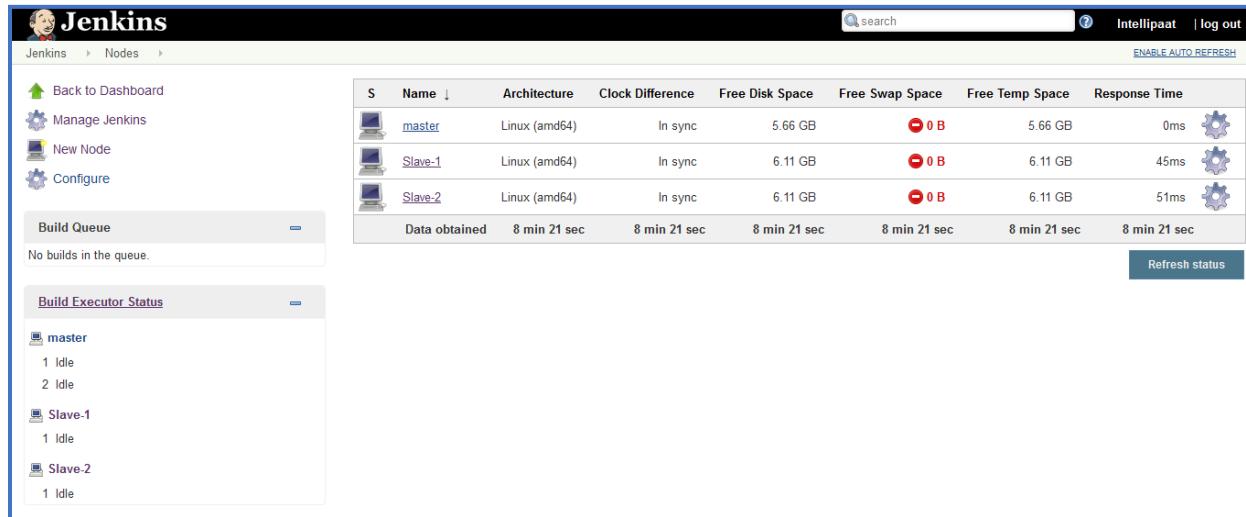
A screenshot of the Jenkins web interface showing the configuration for 'Agent Slave-2'. On the left, there's a sidebar with links like 'Back to List', 'Status', 'Delete Agent', 'Configure', 'Build History', 'Load Statistics', and 'Log'. The main area has a title 'Agent Slave-2' with a red 'X' icon. It says 'Connect agent to Jenkins one of these ways:' followed by two options: 'Launch agent from browser' (with a 'Launch' button) and 'Run from agent command line:' (with a command-line snippet). Below that is a section titled 'Projects tied to Slave-2' which shows 'None'. At the bottom left, it says 'Build Executor Status' with '1 Idle'. At the top right, there are 'Search', 'IntelliPaat | log out', and 'ENABLE AUTO REFRESH' buttons.

Paste the command line in the Slave-2 Terminal.

A terminal window showing the output of a Java command to connect a Jenkins slave. The command is: `java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-2/slave-agent.jnlp -secret 0058b2137c03bc6a5f18b4ca39e1abea32e11ea47c45e60360e29802aa096a1 -workDir "/home/ubuntu/jenkins"`. The terminal output shows the agent connecting to the master, with logs indicating the use of remoting, setting up the agent, and performing discovery to find the master. The connection is successful at the end.

**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.



The screenshot shows the Jenkins interface for managing nodes. On the left, there's a sidebar with links like 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below that is a 'Build Queue' section stating 'No builds in the queue.' To the right is a main table titled 'Nodes' with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. It lists three nodes: 'master' (Linux (amd64), In sync, 5.66 GB free disk, 0 B swap, 5.66 GB temp, 0ms response), 'Slave-1' (Linux (amd64), In sync, 6.11 GB free disk, 0 B swap, 6.11 GB temp, 45ms response), and 'Slave-2' (Linux (amd64), In sync, 6.11 GB free disk, 0 B swap, 6.11 GB temp, 51ms response). A 'Build Executor Status' section below the table shows each node has 1 idle executor. A 'Refresh status' button is at the bottom right of the table area.

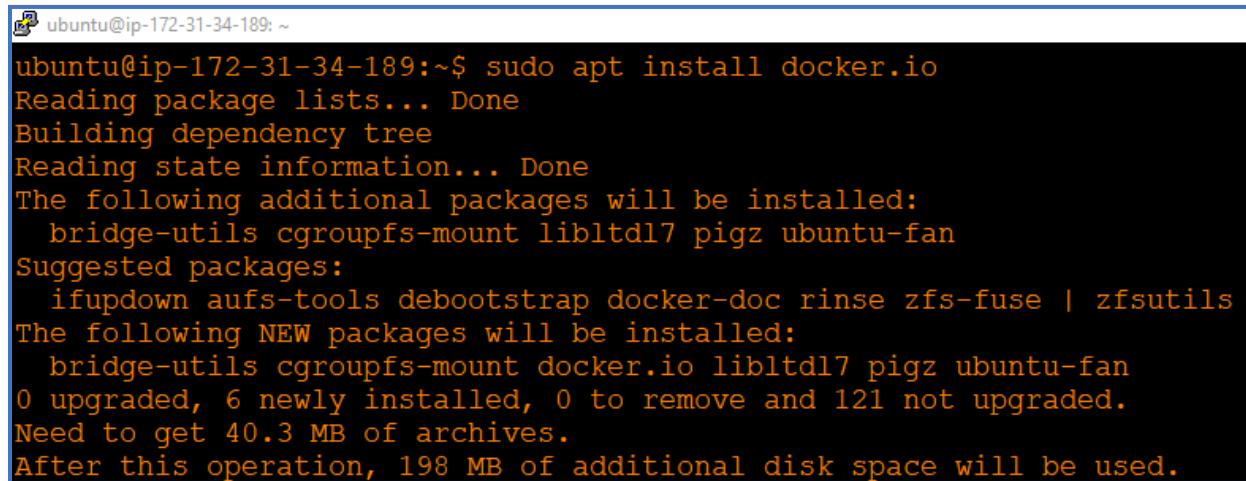
After we have successfully created the Master Slave Cluster on AWS Jenkins. We will now create a CI CD pipeline triggered by Git Webhook.

### 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>

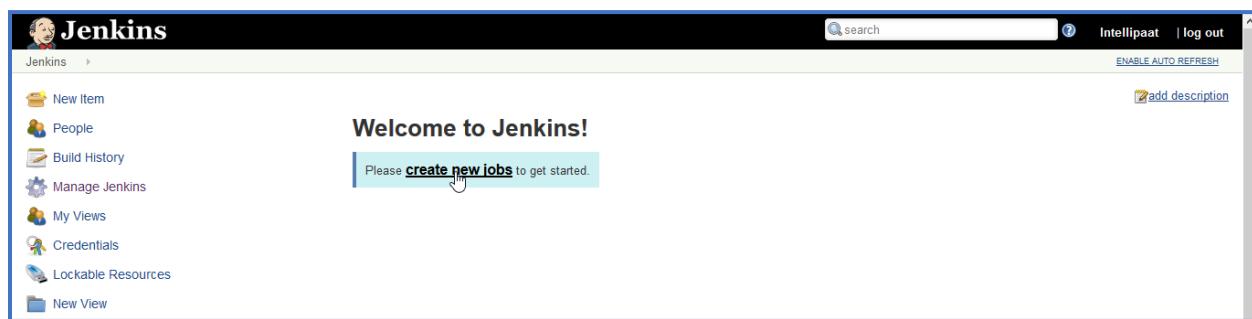
**Step 2:** Install docker on both **Slave-1** and **Slave-2**.



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

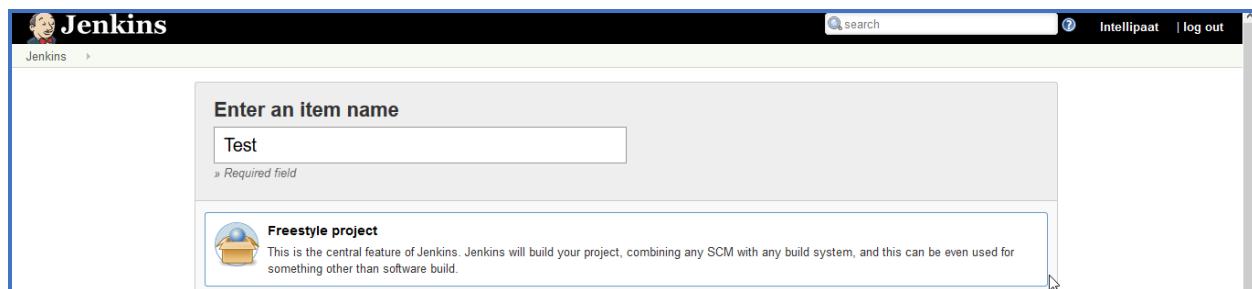
```
ubuntu@ip-172-31-34-132: ~
ubuntu@ip-172-31-34-132:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount libltdl7 pigz ubuntu-fan
Suggested packages:
  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. The left sidebar contains 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.' There is also a 'search' bar and a 'log out' link at the top right.

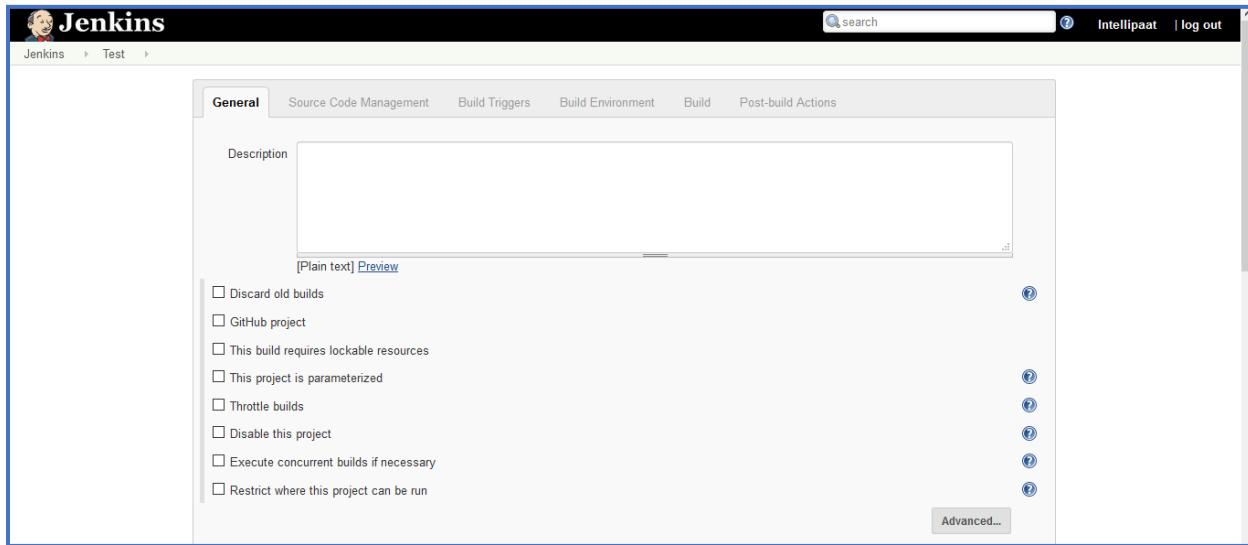
Name the Project as **Test**, Select **Freestyle Project** option.



The screenshot shows the 'Enter an item name' dialog. The input field contains 'Test'. Below it, a 'Freestyle project' section is shown with 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.' There is a 'Required field' note next to the input field.

Then click on **Ok**.

You should land on a page like this.

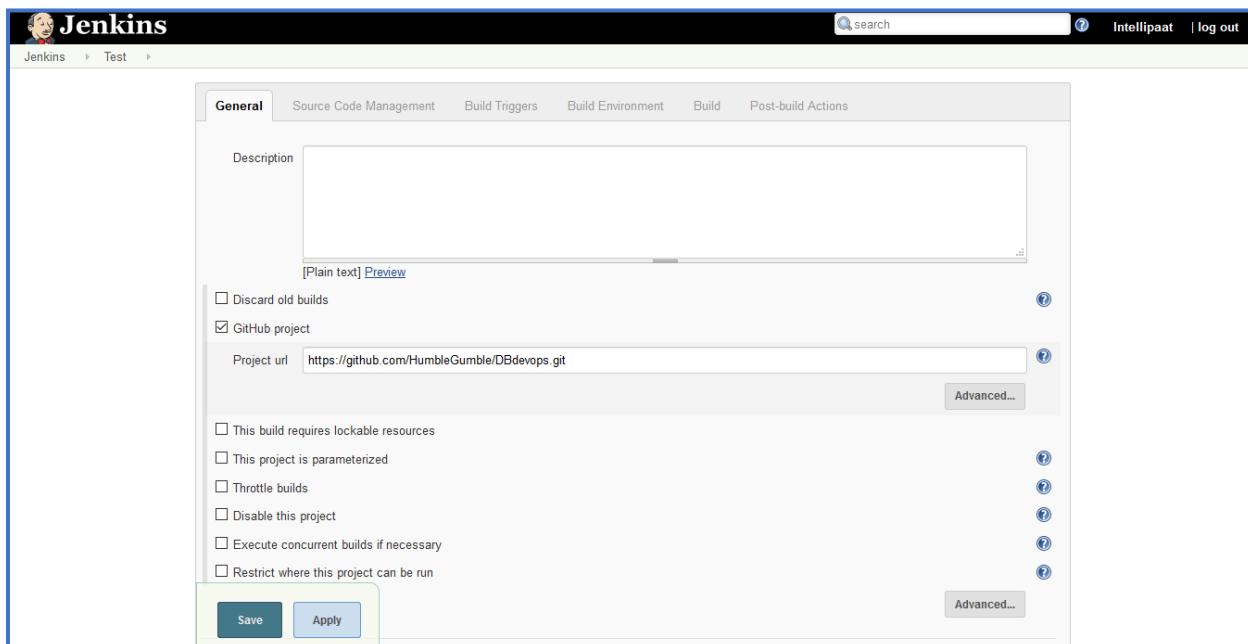


A screenshot of the Jenkins General configuration page. The 'General' tab is selected. The 'Description' field is empty. Below it is a 'Plain text' link. A list of checkboxes follows:

- 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 checked 'GitHub project' checkbox. 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. Under the 'Restrict where this project can be run' section, the 'Label Expression' field contains 'Slave-1'. A yellow warning message below the field states: '⚠ There's no agent/cloud that matches this assignment. Did you mean 'Slave-1' instead of 'Slav'?'. There are several other build-related checkboxes available but not selected.

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 'Repository URL' field is set to 'https://github.com/HumbleGumble/DBdevops.git'. A red error message above the field says: 'Please enter Git repository.'. The 'Branches to build' section shows a 'Branch Specifier' field containing '/master'. The 'Additional Behaviours' section has an 'Add' button. At the bottom, there are 'Save' and 'Apply' buttons.

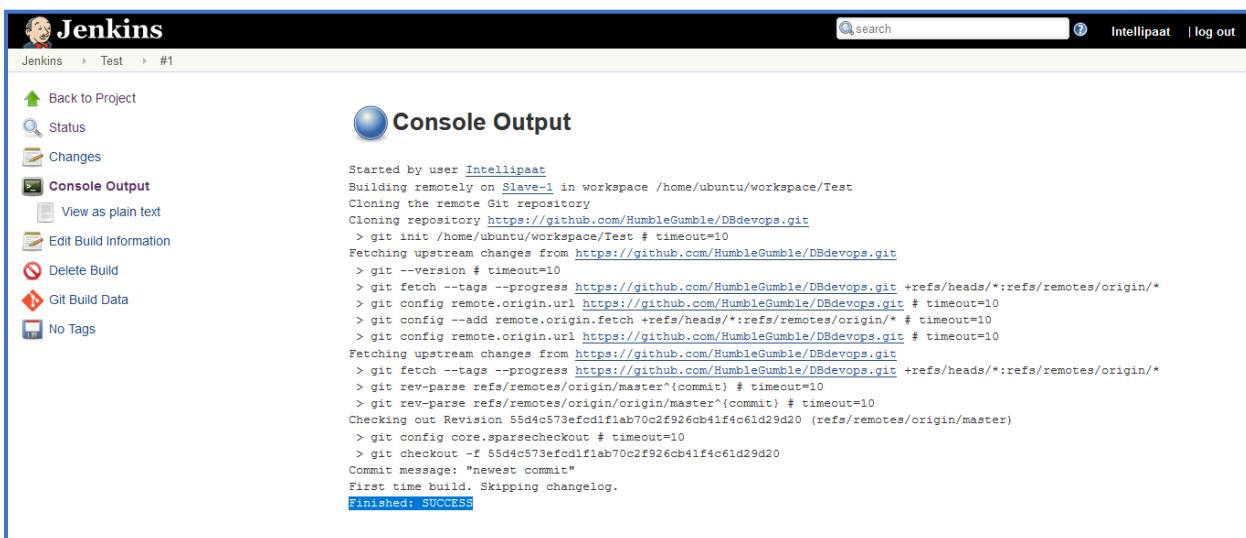
Click on **Save**.

**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 interface for a project named "Test". On the left, a sidebar lists options 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). A "Permalinks" section is also present. On the right, there's a "Build History" table with one entry: "#1 Jan 8, 2019 12:06 PM". Below the table are links for "RSS for all" and "RSS for failures".

Click on the blue circle of build #1.



The screenshot shows the Jenkins interface for build #1 of the "Test" project. The left sidebar includes "Console Output" under the "Build Information" section. The main content area is titled "Console Output" and displays the terminal logs for the build. The logs show the process of cloning a repository from GitHub, fetching upstream changes, and committing the code. 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
```

A terminal window icon showing a blue square with a white terminal symbol.  
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.

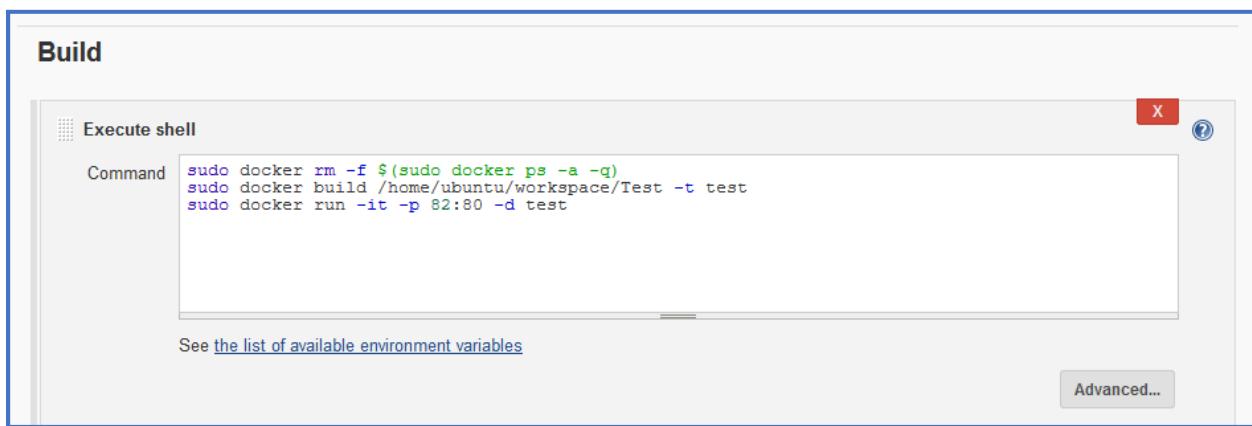
**Step 7:** To run the **Dockerfile** we have to check the copy the present working directory.

A terminal window icon showing a blue square with a white terminal symbol.  
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
```



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.

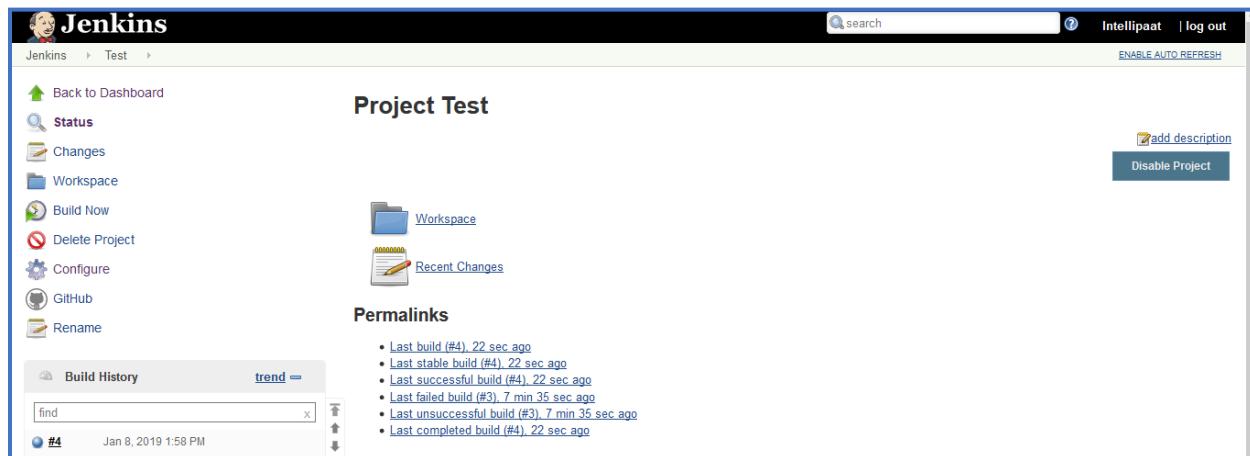
```
$ sudo docker run -it -d ubuntu
```

```
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
ebe701788bff916b06db2a9bf7ad34b4c7cf3e722e8e789c1ea6deaf5ee2beaf
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
AMES               ubuntu      "/bin/bash"   About a minute ago   Up About a minute
ebe701788bff916b06db2a9bf7ad34b4c7cf3e722e8e789c1ea6deaf5ee2beaf
ubuntu@ip-172-31-34-189:~/workspace/Test$
```

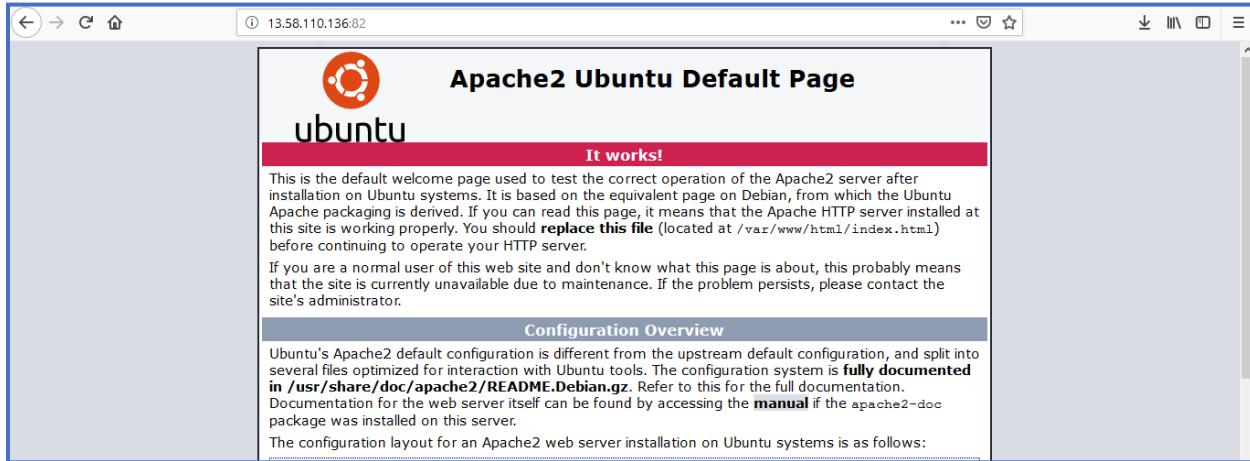
**Step 10:** Now open Jenkins Dashboard and **build the project**.



The screenshot shows the Jenkins Project Test dashboard. On the left, there's a sidebar with links: Back to Dashboard, Status (highlighted), Changes, Workspace, Build Now, Delete Project, Configure, GitHub, and Rename. Below this is a search bar and a dropdown menu for trend analysis. The main area has a title 'Project Test' with icons for Workspace and Recent Changes. Under 'Permalinks', it lists several build links. At the bottom, there's a 'Build History' section showing a single build (#4) from Jan 8, 2019, at 1:58 PM.

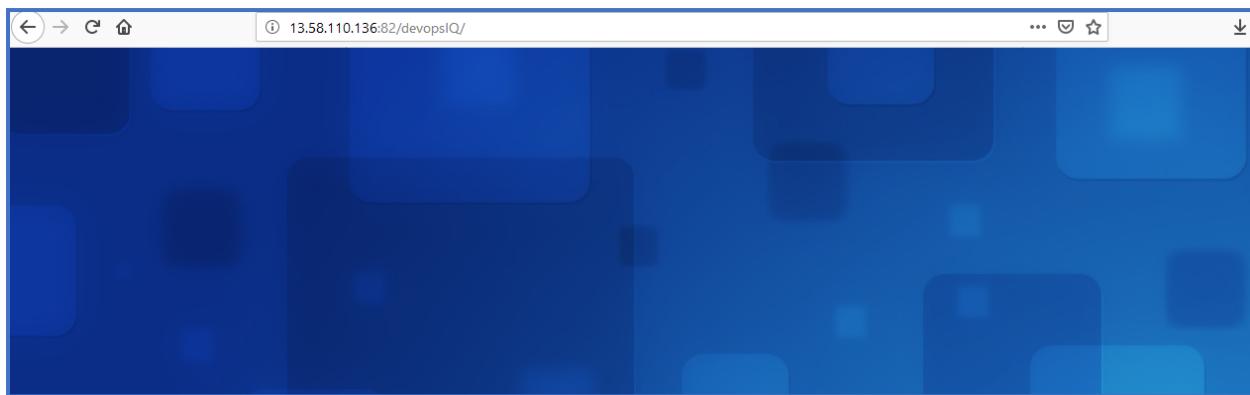
Building was successful.

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



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

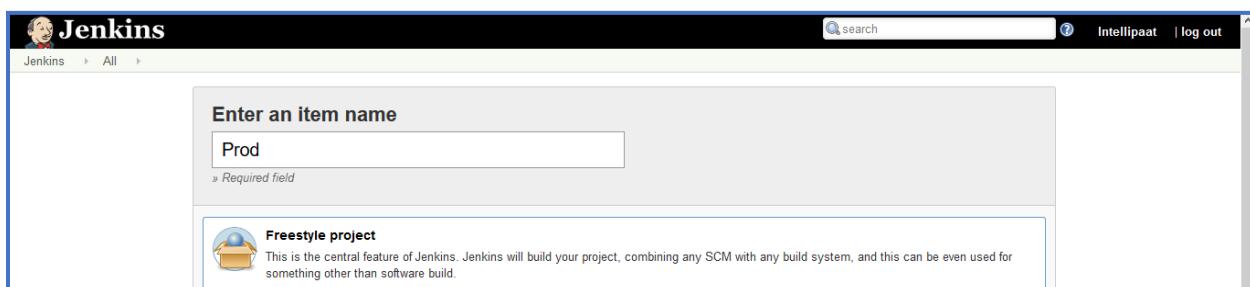
**Step 12:** Now enter **slave-1 IP:82/devopsIQ/** in the browser.



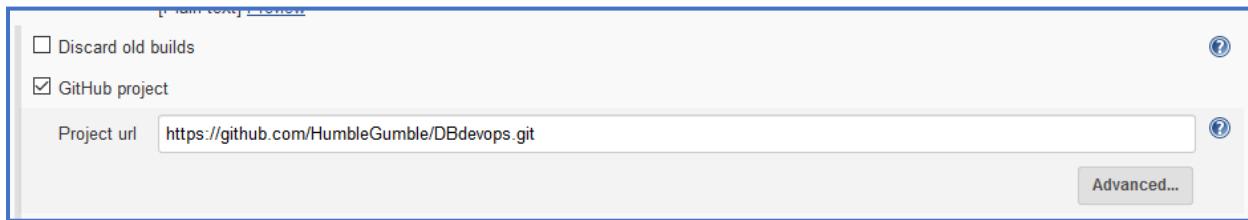
This looks fine.

Now, we will create a new project.

**Step 13:** Create a new project.

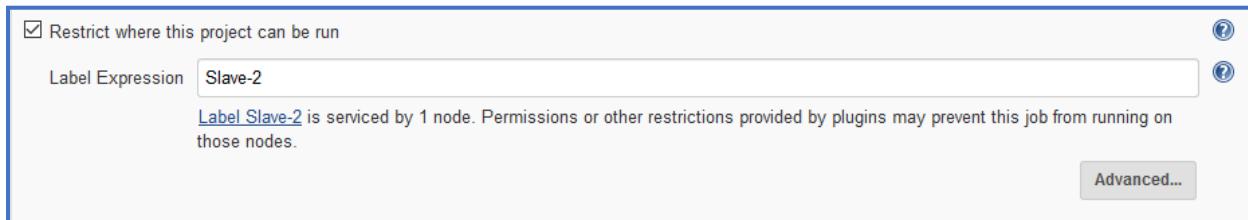


**Step 14:** Click on git project. Enter the git hub repository URL.



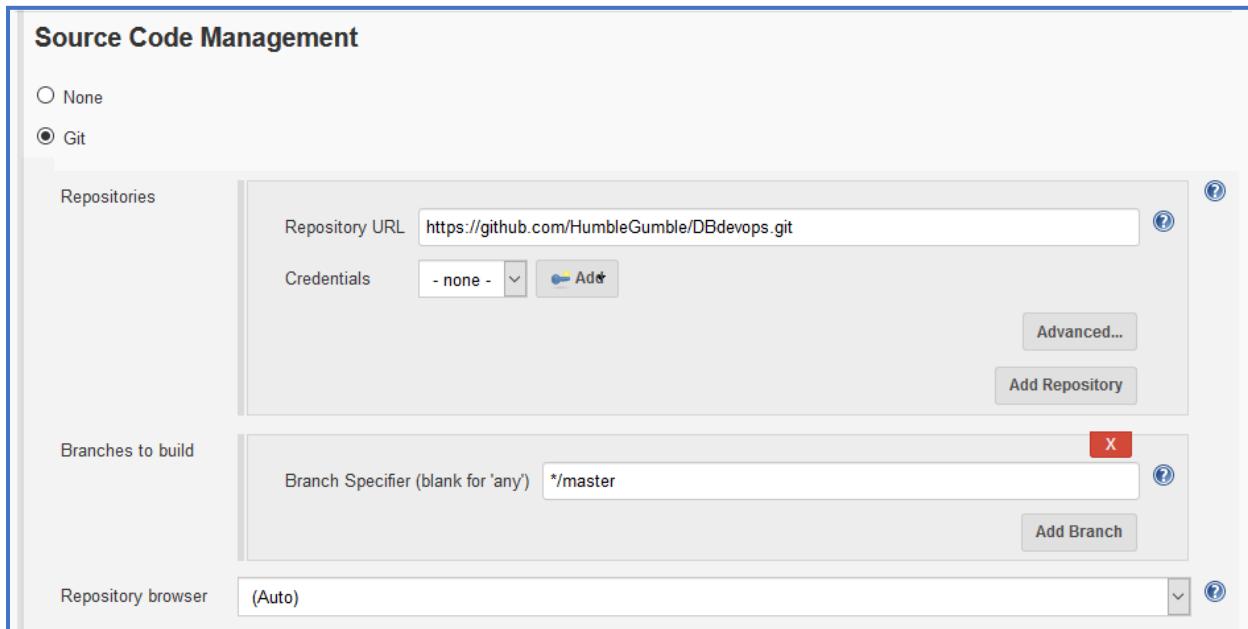
A screenshot of the Jenkins job configuration interface. Under the 'GitHub project' section, the 'Project url' field contains the value <https://github.com/HumbleGumble/DBdevops.git>. Other options like 'Discard old builds' and 'Advanced...' are visible.

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



A screenshot of the Jenkins job configuration interface. Under the 'Restrict where this project can be run' section, the 'Label Expression' field contains 'Slave-2'. A note below states: 'Label Slave-2 is serviced by 1 node. Permissions or other restrictions provided by plugins may prevent this job from running on those nodes.' An 'Advanced...' button is also present.

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



A screenshot of the Jenkins Source Code Management configuration. It shows the 'Git' option selected. Under 'Repositories', the 'Repository URL' is set to <https://github.com/HumbleGumble/DBdevops.git>. Under 'Branches to build', the 'Branch Specifier' is set to \*/master. Under 'Repository browser', it is set to '(Auto)'. Buttons for 'Advanced...', 'Add Repository', 'Add Branch', and 'Add Repository' are visible.

**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
```

**Build**

**Execute shell**

Command

```
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
```

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

Advanced...

Add build step ▾



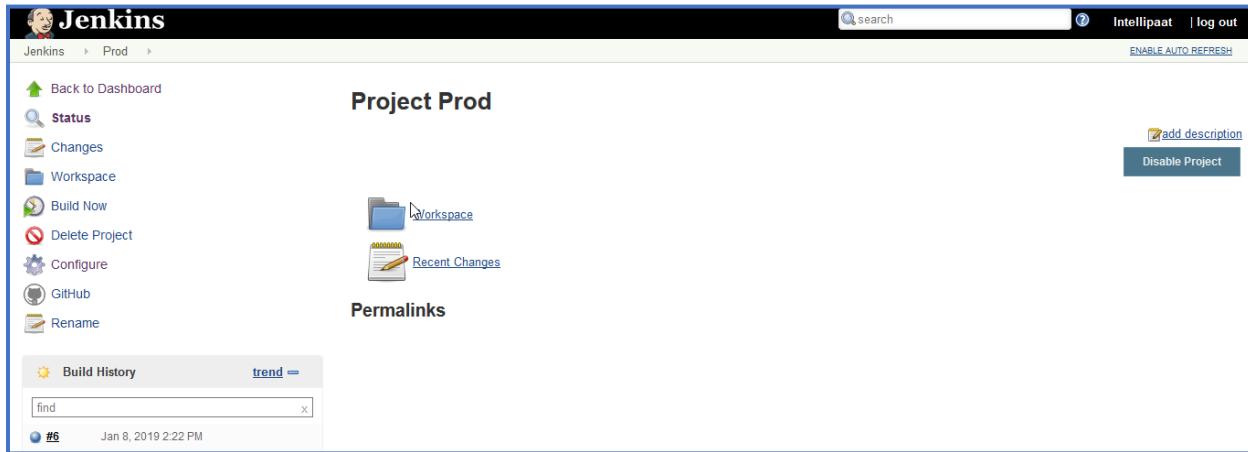
**Step 18:** Again, add one container to the Slave-2 as shown below.

```
$ 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.

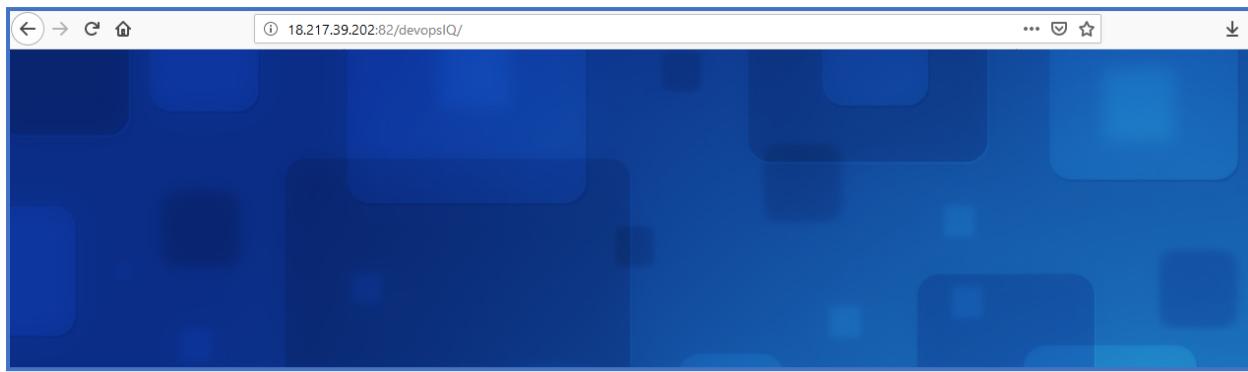
**Step 19:** Build the project **Prod**.



The screenshot shows the Jenkins Project Prod dashboard. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, GitHub, and Rename. The main area is titled "Project Prod" and shows a "Recent Changes" section with a blue folder icon labeled "Workspace". Below it is a "Permalinks" section. At the bottom left is a "Build History" table with one row: "#6 Jan 8, 2019 2:22 PM". A search bar is at the top right, and a "Disable Project" button is in the top right corner.

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**.

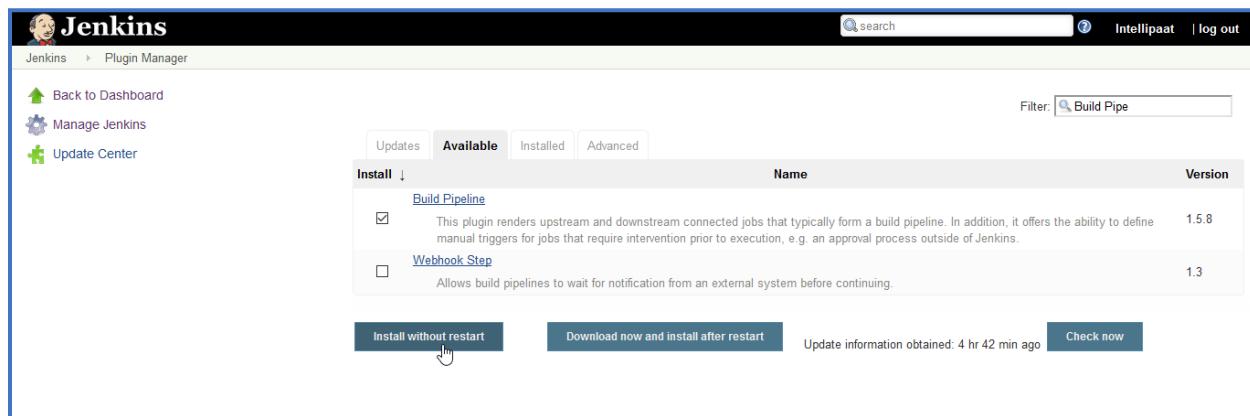


The screenshot shows the Jenkins configuration interface for a job named 'Test'. Under the 'Post-build Actions' section, there is a single action selected: 'Build other projects'. The 'Projects to build' field contains 'Prod,'. Below this, there are three trigger options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom left of the configuration panel, there is a button labeled 'Add post-build action ▾'.

Click on Save.

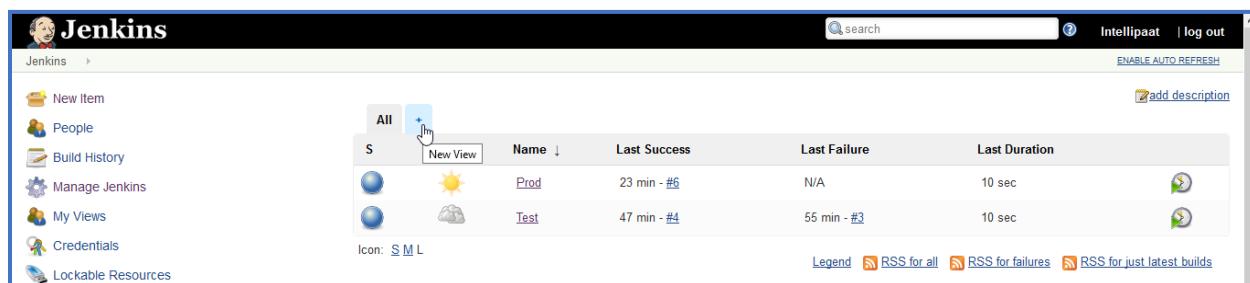
Now we will run jobs using pipeline.

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



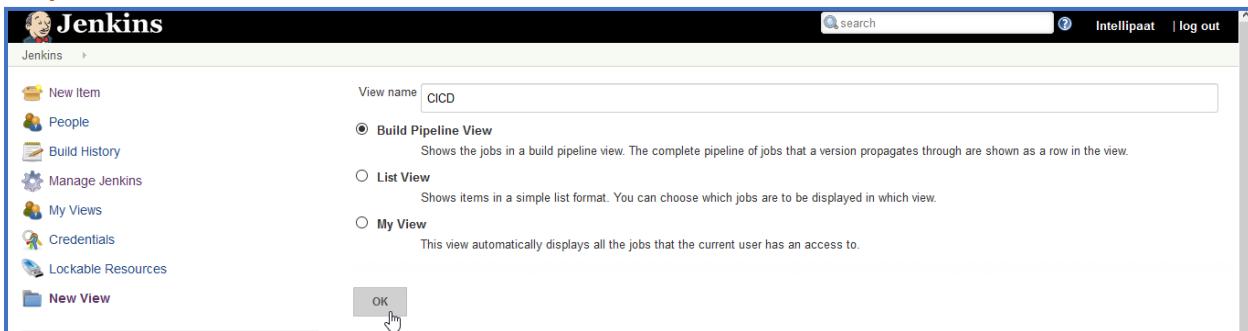
The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right contains the text 'Build Pipe'. A table lists two plugins: 'Build Pipeline' (version 1.5.8) and 'Webhook Step' (version 1.3). The 'Build Pipeline' row has a checkbox labeled 'Install' checked. A tooltip for this checkbox states: 'This plugin renders upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins.' Below the table are three buttons: 'Install without restart' (highlighted with a cursor), 'Download now and install after restart', and 'Check now'.

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



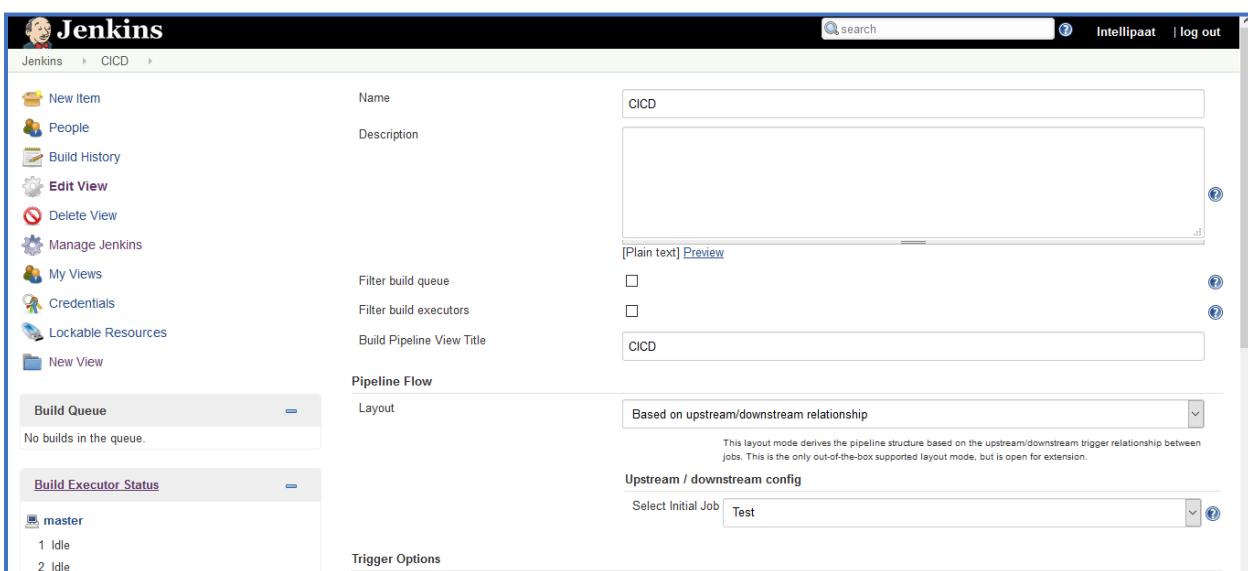
The screenshot shows the Jenkins dashboard. On the left, a sidebar menu includes 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', and 'Lockable Resources'. The main area displays a table of jobs. The first job, 'Prod', has a green status icon. The second job, 'Test', has a yellow status icon. A blue '+' icon with a cursor over it is positioned above the 'New View' button in the table header. The table columns are 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. At the bottom of the dashboard, there are links for 'Legend', 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'.

**Step 24:** Enter view name and click ok.



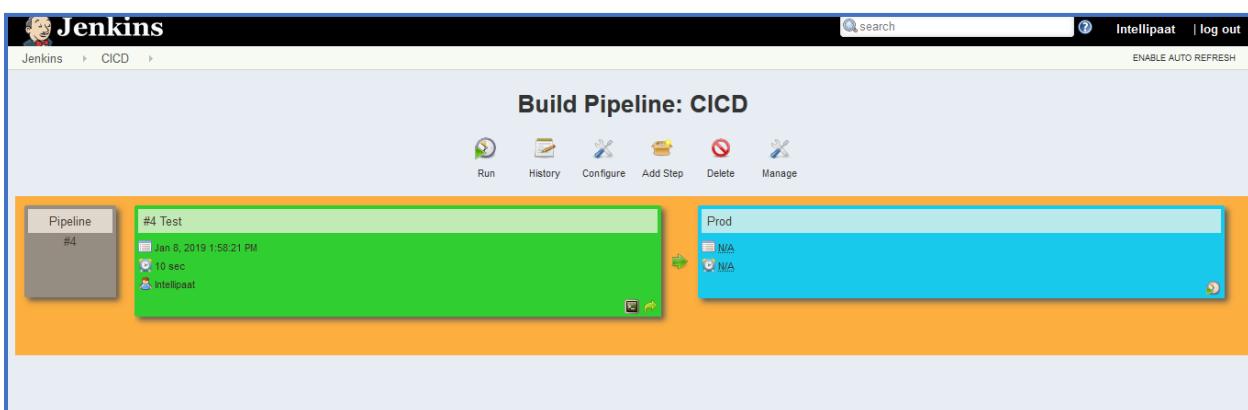
The screenshot shows the Jenkins 'New View' configuration page. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', etc. The main area has a 'View name' input field containing 'CICD'. Below it, there are three radio button options: 'Build Pipeline View' (selected), 'List View', and 'My View'. A tooltip for 'Build Pipeline View' says: 'Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.' Below these is an 'OK' button with a mouse cursor hovering over it.

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



The screenshot shows the Jenkins 'Edit View' configuration page for a view named 'CICD'. The left sidebar includes 'Edit View' and 'New View' options. The main configuration area has a 'Name' field set to 'CICD' and a 'Description' field below it. Under 'Pipeline Flow', there's a 'Layout' dropdown set to 'Based on upstream/downstream relationship'. In the 'Upstream / downstream config' section, 'Select Initial Job' is set to 'Test'. The page also displays sections for 'Build Queue' (empty) and 'Build Executor Status' (showing 1 idle and 2 idle executors).

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



The screenshot shows the Jenkins 'Build Pipeline: CICD' page. At the top, there are buttons for 'Run', 'History', 'Configure', 'Add Step', 'Delete', and 'Manage'. Below this is a pipeline card for 'Pipeline #4'. The card shows a green status bar with the text '#4 Test' and 'Jan 8, 2019 1:58:21 PM'. It also lists '10 sec' and 'intellipaat' under the 'Duration' and 'Last Run' sections respectively. To the right of the pipeline card is a 'Prod' section showing 'N/A' for both 'Status' and 'Last Run'. The background features a large orange horizontal bar.

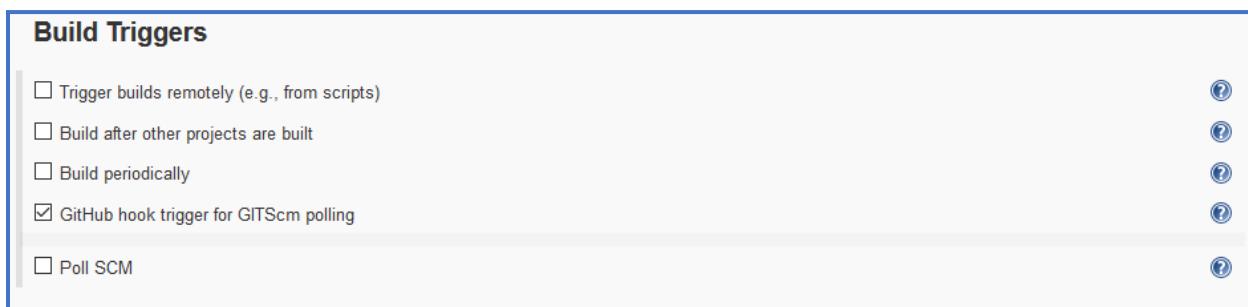
**Step 26:** Click on Run. Then Refresh the Page once.



The screenshot shows the Jenkins dashboard for a 'Build Pipeline: CICD' project. At the top, there are navigation links: Jenkins, CICD, search bar, and Intellipaat log out. Below the header, there's a toolbar with icons for Run, History, Configure, Add Step, Delete, and Manage. The main area displays two build cards. The first card, 'Pipeline #5', is green and labeled '#5 Test'. It shows a timestamp of Jan 8, 2019 2:55:07 PM and a duration of 2.1 sec. The second card, 'Pipeline #7', is also green and labeled '#7 Prod'. It shows a timestamp of Jan 8, 2019 2:55:16 PM and a duration of 1.4 sec. Both cards have small icons for more details and a trash bin for deletion.

**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.

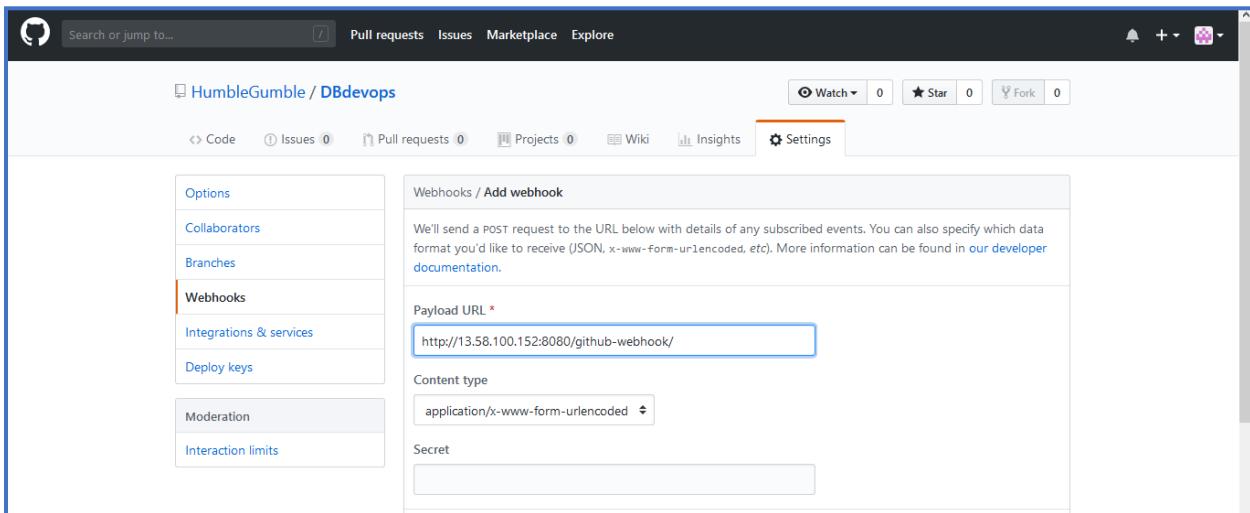


The screenshot shows the 'Build Triggers' configuration page. It lists several options with checkboxes and help icons. The 'GitHub hook trigger for GITScm polling' option is checked. Other options include 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', and 'Poll SCM'.

Trigger Type	Status	Help
Trigger builds remotely (e.g., from scripts)	<input type="checkbox"/>	
Build after other projects are built	<input type="checkbox"/>	
Build periodically	<input type="checkbox"/>	
GitHub hook trigger for GITScm polling	<input checked="" type="checkbox"/>	
Poll SCM	<input type="checkbox"/>	

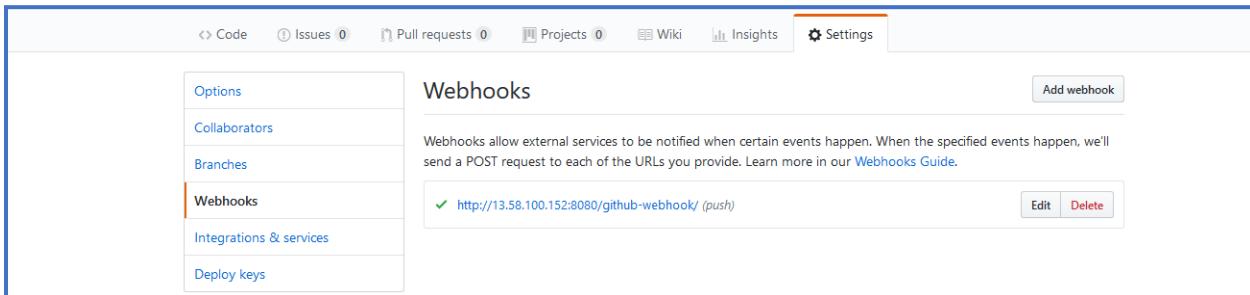
**Step 28:** Now configure GitHub Webhook. Go to settings, then click on Webhooks, then add webhooks. There insert the Jenkins Server Address as shown.

\$ JenkinsServer Address/github-webhook/



A screenshot of the GitHub Settings interface, specifically the Webhooks section. On the left, there's a sidebar with options like Options, Collaborators, Branches, Webhooks (which is selected and highlighted in orange), Integrations & services, Deploy keys, Moderation, and Interaction limits. The main content area has a title 'Webhooks / Add webhook'. It contains a note about sending POST requests to a URL with event details. Below that is a 'Payload URL' input field containing 'http://13.58.100.152:8080/github-webhook/'. There are dropdowns for 'Content type' (set to 'application/x-www-form-urlencoded') and a 'Secret' input field. At the top of the content area, there are 'Watch' (0), 'Star' (0), 'Fork' (0) buttons, and a 'Settings' tab.

Click on Add webhook. You should see this.



A screenshot of the GitHub Settings interface, specifically the Webhooks section. The sidebar shows the same options as the previous screenshot. The main content area has a title 'Webhooks' and a 'Add webhook' button. Below it is a note about external services being notified for events. A table lists the added webhook: a green checkmark next to 'http://13.58.100.152:8080/github-webhook/' (push). To the right of the URL are 'Edit' and 'Delete' buttons.

**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/devopsIQ
ubuntu@ip-172-31-39-127:~$ ls
DBdevops
ubuntu@ip-172-31-39-127:~$ cd DBdevops
ubuntu@ip-172-31-39-127:~/DBdevops$ ls
Dockerfile devopsIQ docker-compose
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$ nano index.html[ ]
```

**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
GNU nano 2.9.3                                         index.html

<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

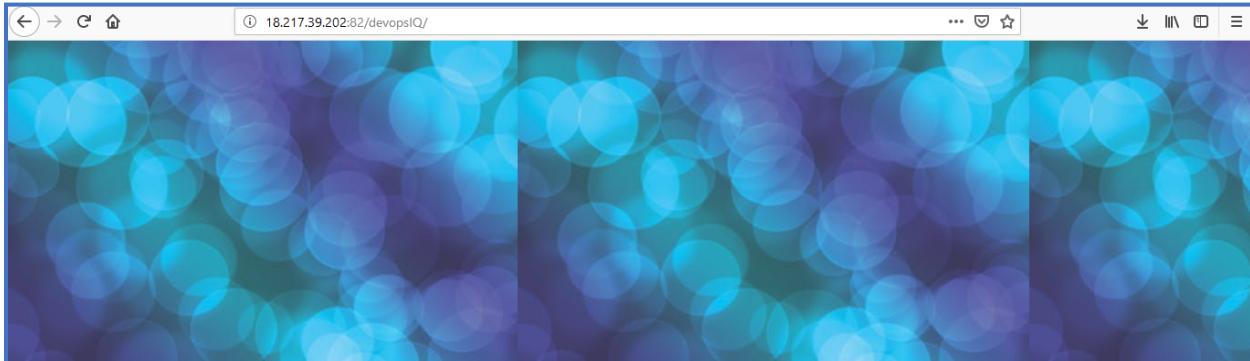
1 file changed, 2 insertions(+), 2 deletions(-)
```

**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.