



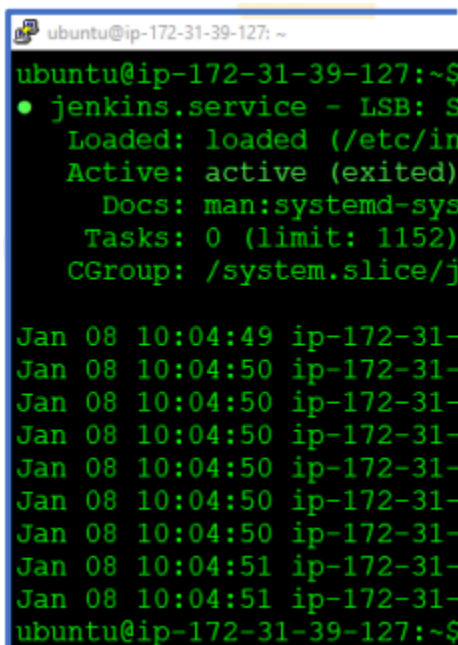
Module 6: Hands-On: Jenkins

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 Webhooks.

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 Jenkins on the Master Machine. Now Let us set up the Jenkins Master-Slave Cluster.

Step 1: Check the status of the Jenkins first

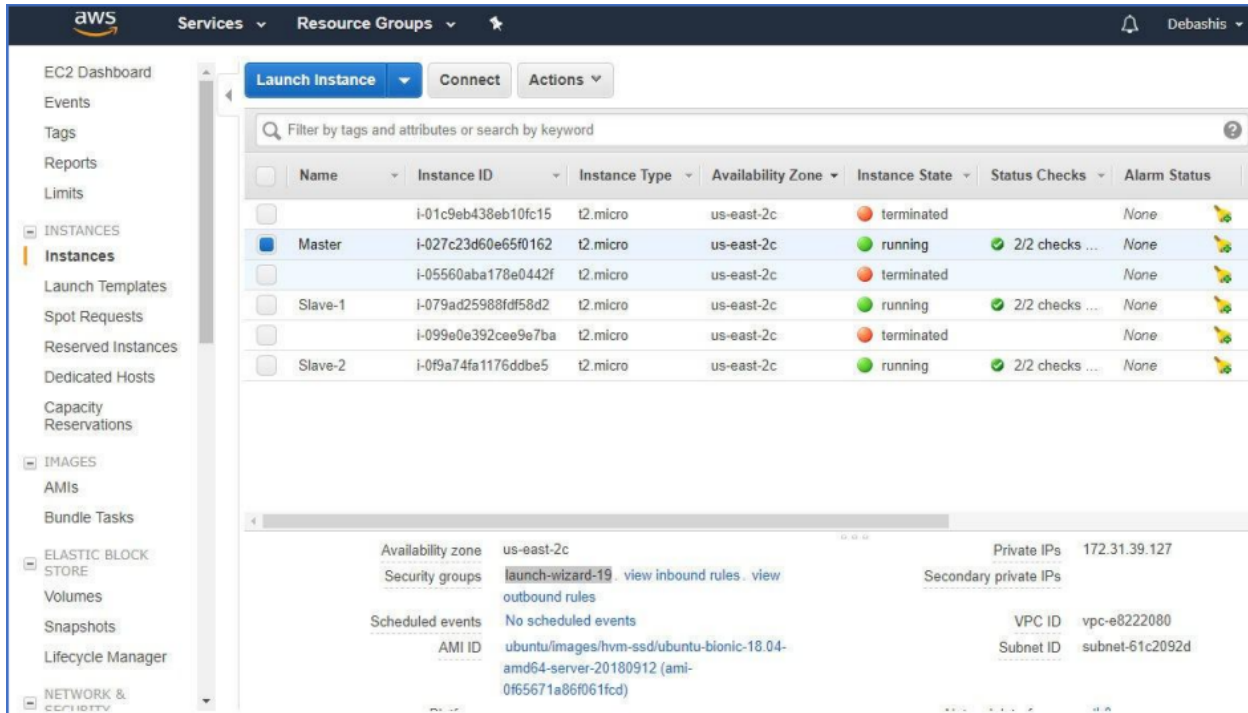
```
$ service jenkins status
```



```
ubuntu@ip-172-31-39-127: ~$
• jenkins.service - LSB: S
  Loaded: loaded (/etc/in
  Active: active (exited)
  Docs: man:systemd-sys
  Tasks: 0 (limit: 1152)
  CGroup: /system.slice/j

Jan 08 10:04:49 ip-172-31-
Jan 08 10:04:50 ip-172-31-
Jan 08 10:04:50 ip-172-31-
Jan 08 10:04:50 ip-172-31-
Jan 08 10:04:50 ip-172-31-
Jan 08 10:04:50 ip-172-31-
Jan 08 10:04:50 ip-172-31-
Jan 08 10:04:50 ip-172-31-
Jan 08 10:04:51 ip-172-31-
Jan 08 10:04:51 ip-172-31-
ubuntu@ip-172-31-39-127: ~$
```

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



| 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-079ad25988fd58d2 | t2.micro | us-east-2c | running | 2/2 checks ... | None |
| | i-099e0e392cee9e7ba | t2.micro | us-east-2c | terminated | | None |
| Slave-2 | i-0f9a74fa1176dbbe5 | t2.micro | us-east-2c | running | 2/2 checks ... | None |

Availability zone: us-east-2c
Private IPs: 172.31.39.127
Security groups: launch-wizard-19, view inbound rules, view outbound rules
Secondary private IPs
Scheduled events: No scheduled events
VPC ID: vpc-e8222080
AMI ID: ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20180912 (ami-0f65671a86f061fcd)
Subnet ID: subnet-61c2092d

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



| Type | Protocol | Port Range | Source | Description | |
|-------------|----------|------------|----------|-----------------|----------------------------|
| All traffic | All | 0 - 65535 | Anywhere | 0.0.0.0/0, ::/0 | e.g. SSH for Admin Desktop |
| All traffic | All | 0 - 65535 | Anywhere | 0.0.0.0/0, ::/0 | e.g. SSH for Admin Desktop |

Add Rule

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.

Cancel Save

Step 4: Now open browser and enter masterIP:8080

You should land on a page like this:

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](#)

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:

Password:

Confirm password:

Full name:

E-mail address:

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.

 **Jenkins**

IntelliPaat | log out

Jenkins

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Credentials
- Lockable Resources
- New View

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue


No builds in the queue.

Build Executor Status

1 Idle

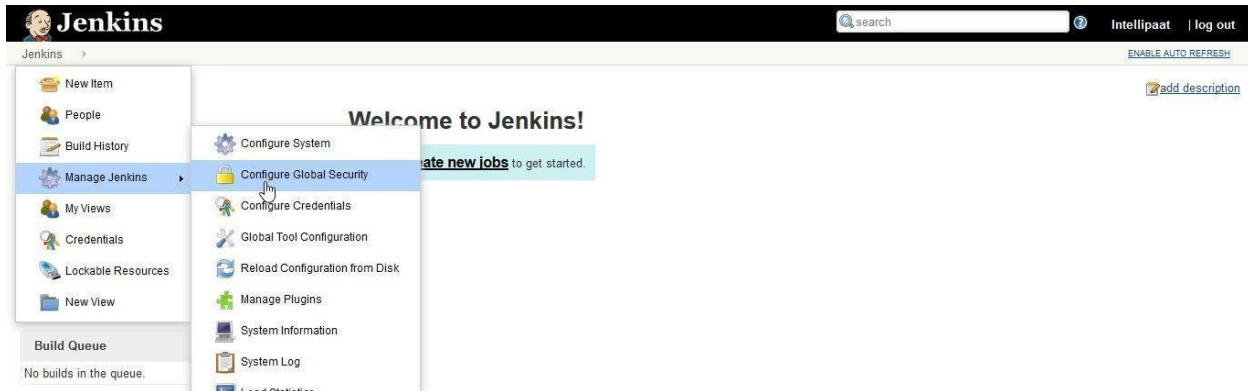
2 Idle

ENABLE AUTO REFRESH

 [add description](#)

This is our Jenkins Dashboard.

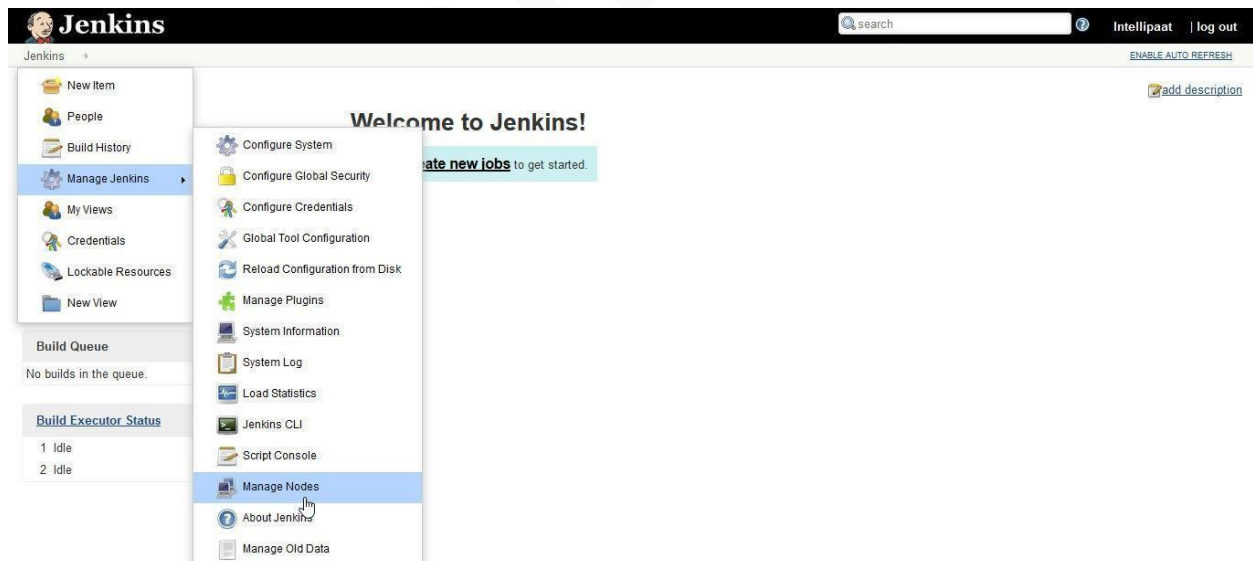
Step 7: Go to Manage Jenkins. Click on Configure Global Security.



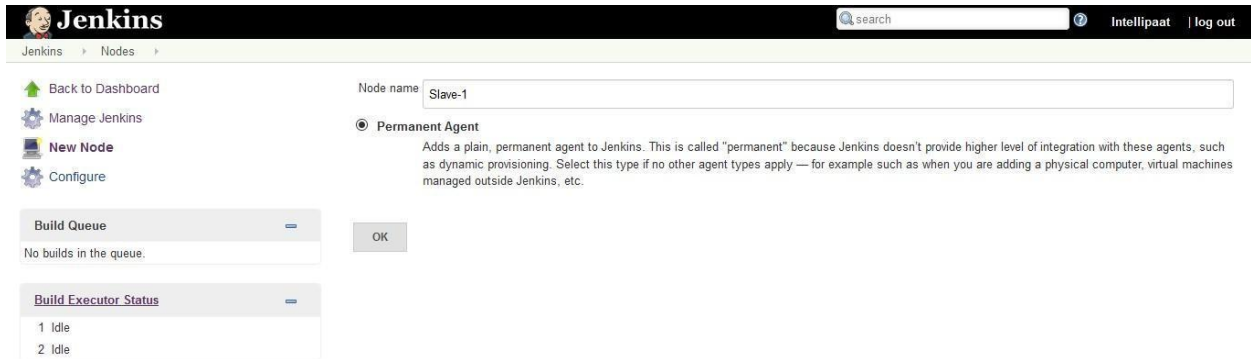
Step 8: Change the Agents to Random. Then click on Save.



Step 9: Now go to Manage Nodes.

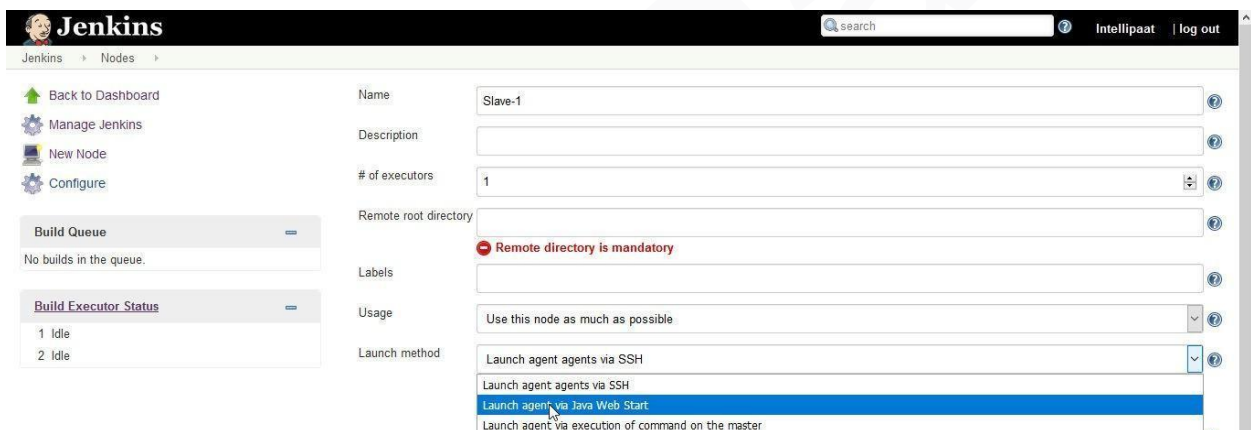


Step 10: Click on New Node. Add Slave-1 as a New Node and make Permanent Agent. Click on OK.




The screenshot shows the Jenkins 'New Node' configuration page. The 'Node name' field is set to 'Slave-1'. The 'Permanent Agent' radio button is selected. A description explains that a permanent agent is a plain agent without dynamic provisioning. On the left sidebar, the 'Build Queue' shows 'No builds in the queue' and 'Build Executor Status' shows '1 Idle' and '2 Idle'. An 'OK' button is visible at the bottom right of the configuration area.

Step 11: Go to Launch Method, change it to Launch agent via Java Web Start



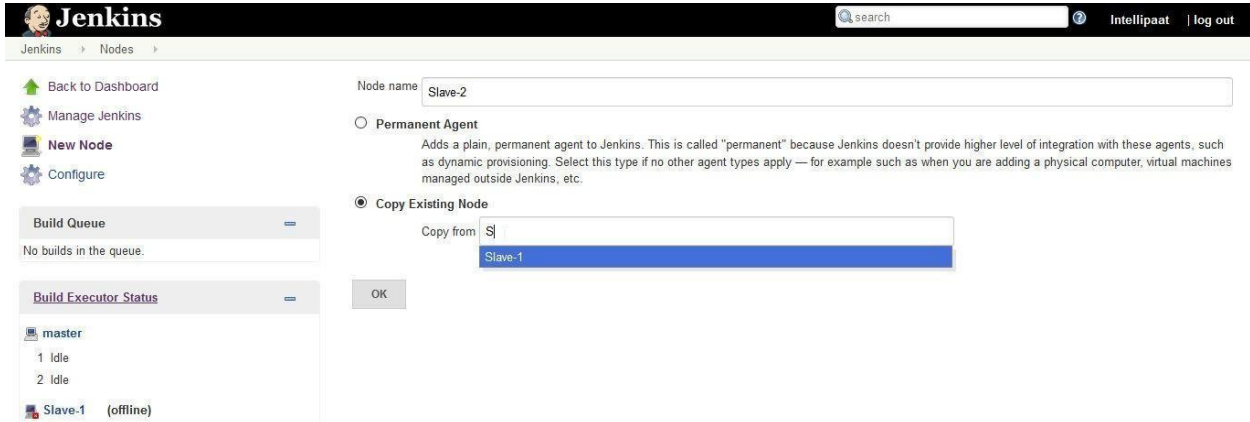
The screenshot shows the Jenkins 'Node configuration' page for 'Slave-1'. The 'Launch method' dropdown menu is open, showing options: 'Launch agent agents via SSH', 'Launch agent agents via SSH', 'Launch agent via Java Web Start' (which is highlighted), and 'Launch agent via execution of command on the master'. Other fields include 'Name' (Slave-1), 'Description', '# of executors' (1), 'Remote root directory' (with an error message 'Remote directory is mandatory'), 'Labels', 'Usage' (Use this node as much as possible), and 'Launch method'.

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



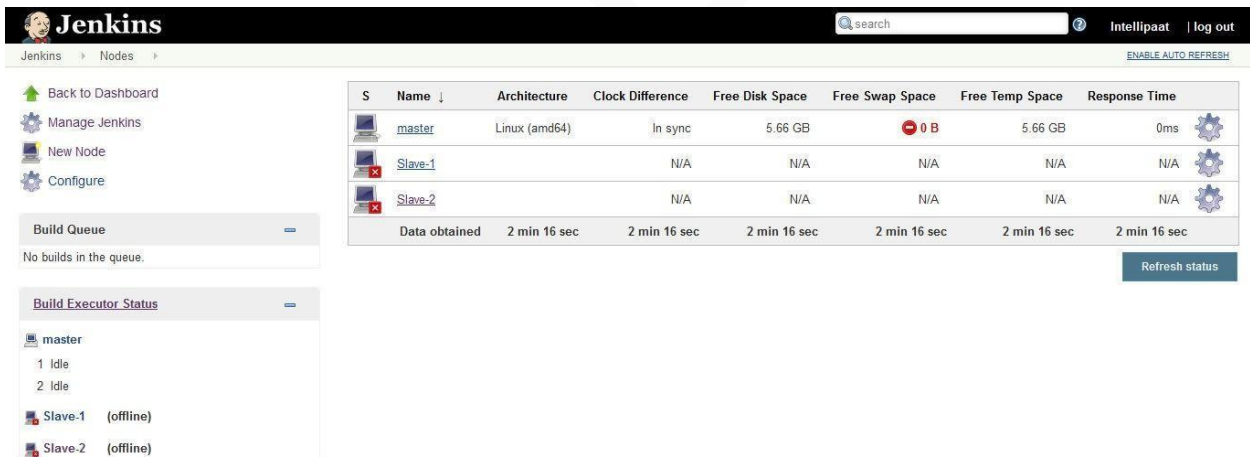
The screenshot shows the 'Launch method' dropdown set to 'Launch agent via Java Web Start'. Below it, the 'Custom WorkDir path' field is set to '/home/ubuntu/jenkins'. Other fields include 'Disable WorkDir' (unchecked), 'Internal data directory' (remoting), and 'Fail if workspace is missing' (unchecked). An 'Advanced...' button is located at the bottom right.

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



The screenshot shows the Jenkins 'New Node' configuration page. The 'Node name' field is set to 'Slave-2'. The 'Copy Existing Node' option is selected, and the 'Copy from' dropdown menu shows 'Slave-1' as the source. The 'OK' button is visible at the bottom.

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



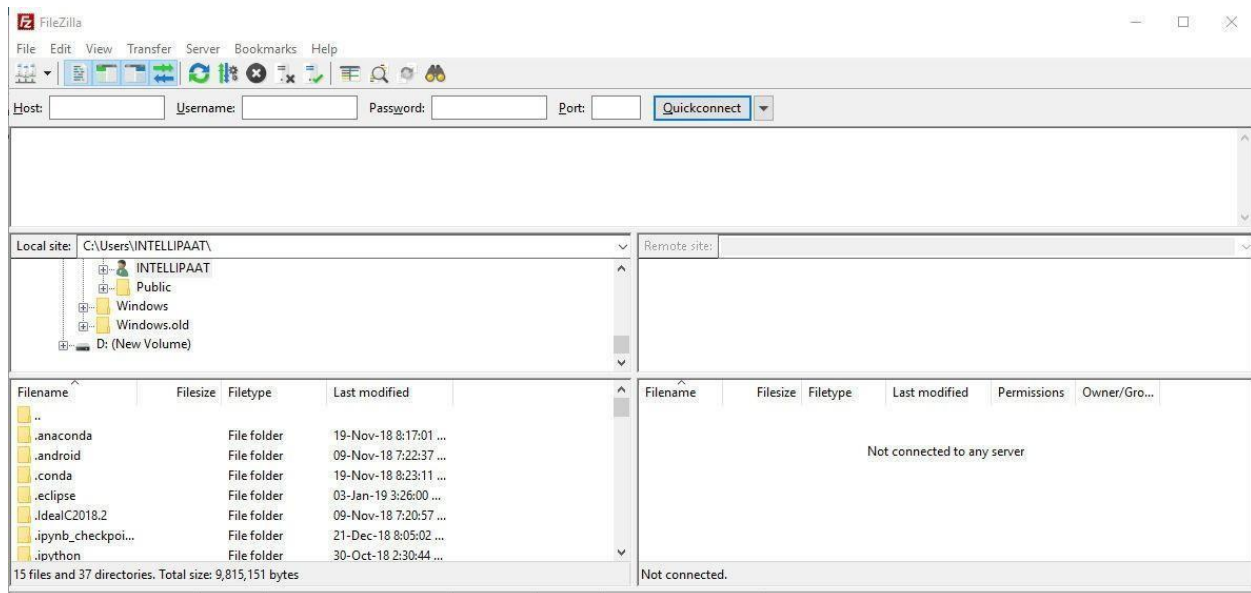
The screenshot shows the Jenkins Dashboard with a table listing the nodes. The table includes columns for Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The nodes listed are master, Slave-1, and Slave-2. Slave-1 and Slave-2 are marked as offline.

| 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 |
| | Data obtained | 2 min 16 sec | 2 min 16 sec | 2 min 16 sec | 2 min 16 sec | 2 min 16 sec | 2 min 16 sec |

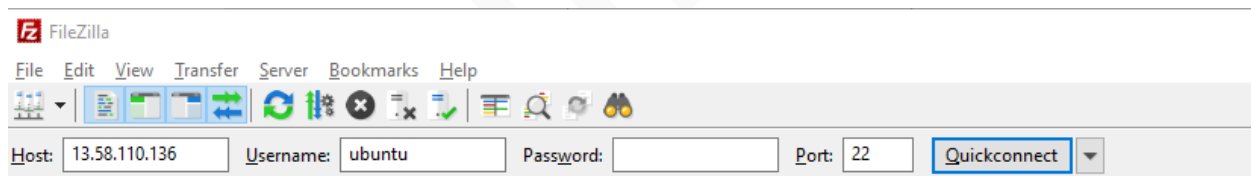
Refresh status

Step 15: Before moving ahead, download FileZilla.

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

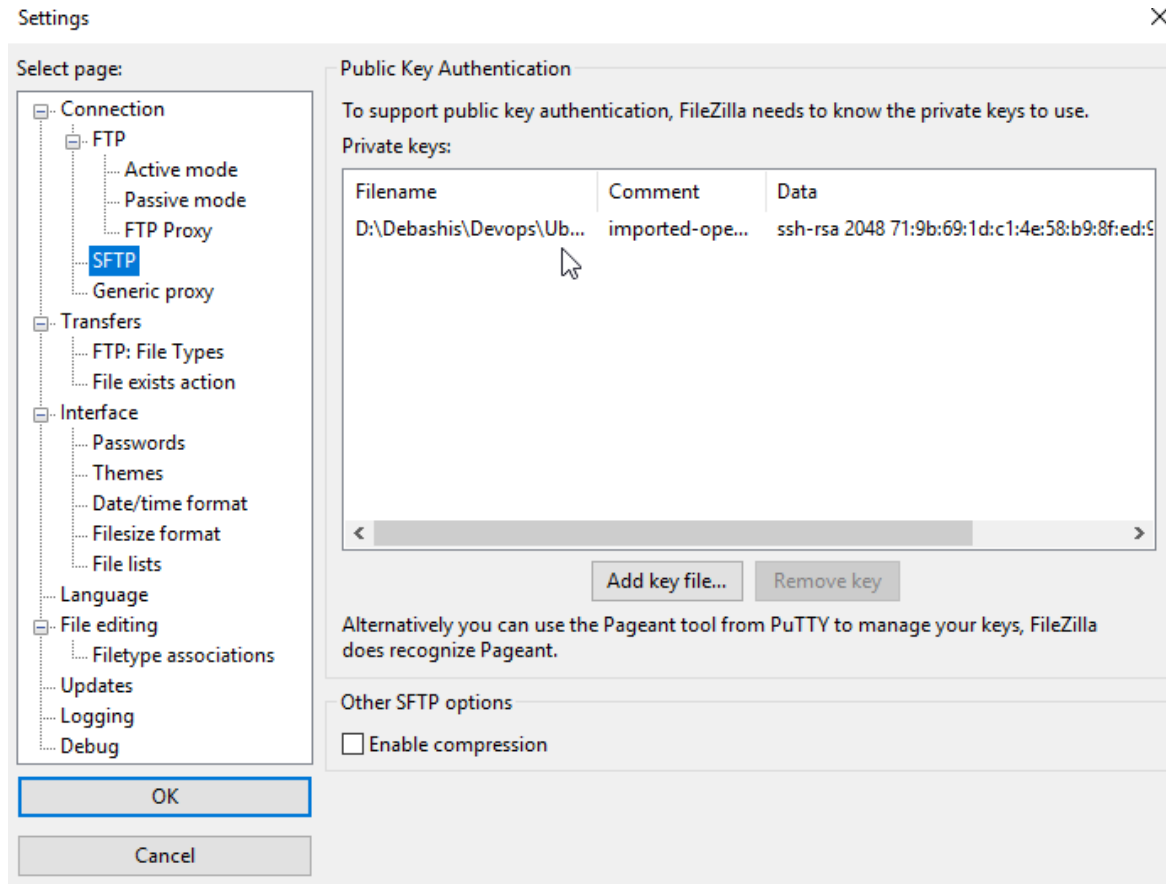


Step 17: Now copy the Slave-1 IP address. And add it as Host. Add Ubuntu as the username. Leave the password field empty. Add Port as 22.



Don't start the connection yet.

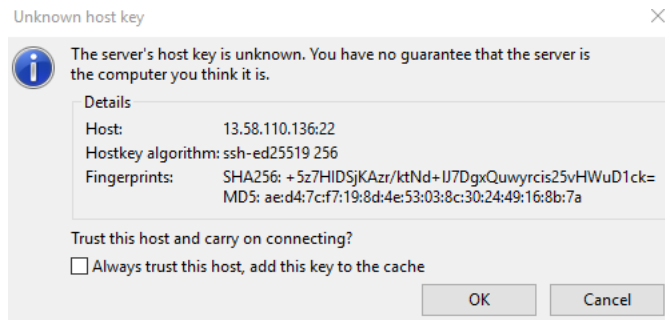
Step 18: Before we start the connection, we need to add the Private key (PPK file). Go to Edit, click on Settings. Click on SFTP. Add the PPK file there and click OK.



Step 19: Click on Quickconnect.

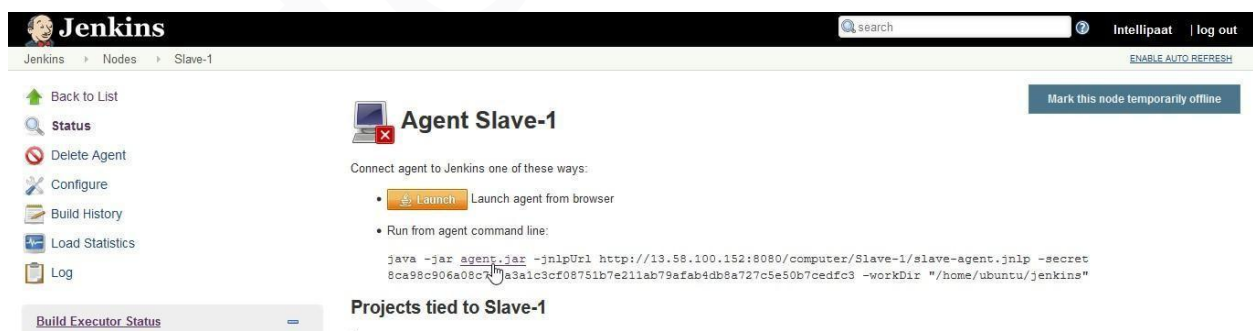


Then click on OK.

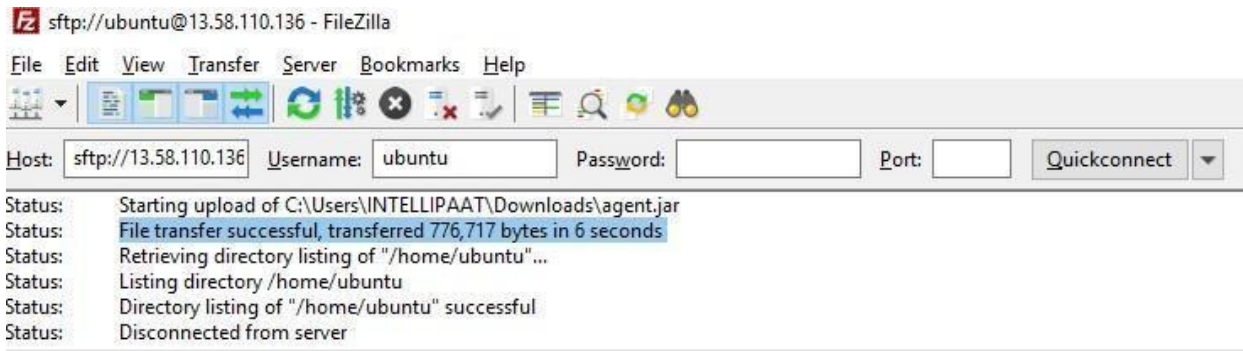


As you can see our connection is successful.

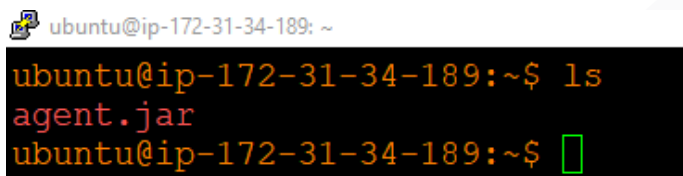
Step 20: Go to the Jenkins Dashboard, click on Slave-1. Download the Agent.jar file by clicking on it.



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.

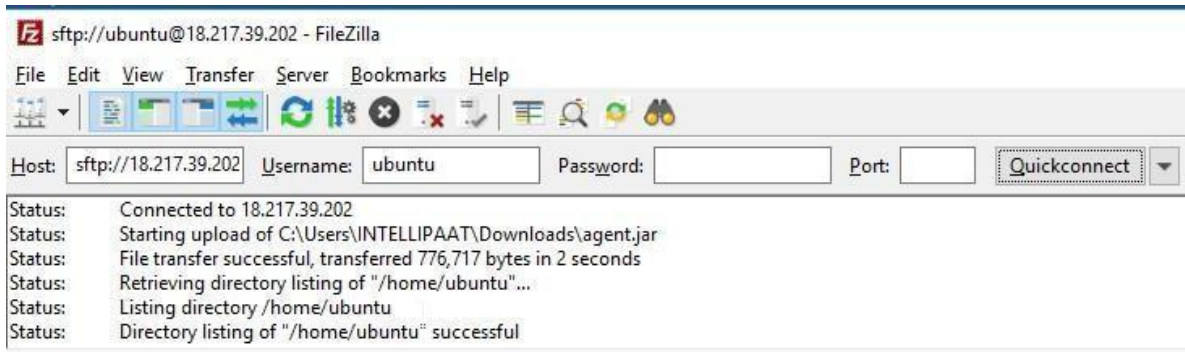


As you can see the agent.jar file appears there which means our file has been successfully transferred to Slave-1.

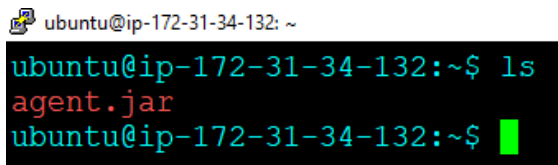
Step 23: Perform the steps 19 to 22 for Slave-2 as well. (Tip: Rename the agent.jar file of Slave-2. Before performing transfer operation in FileZilla)



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



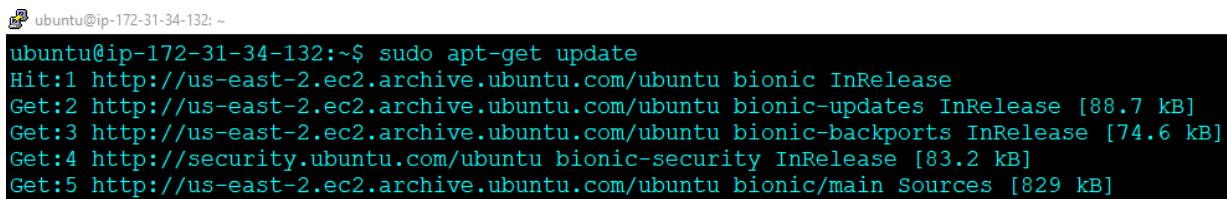
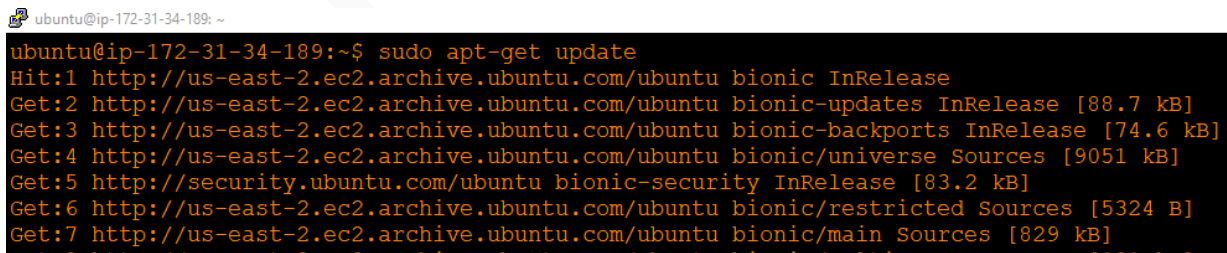
Step 24: Again, verify by opening a new PuTTY session for Slave-2.



Looks fine!

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

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



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

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

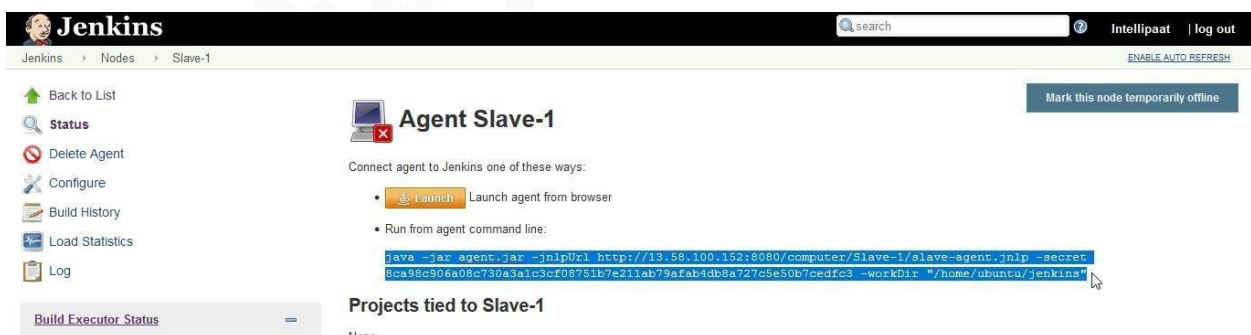
```
ubuntu@ip-172-31-34-189: ~
```

```
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: ~
```

```
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.

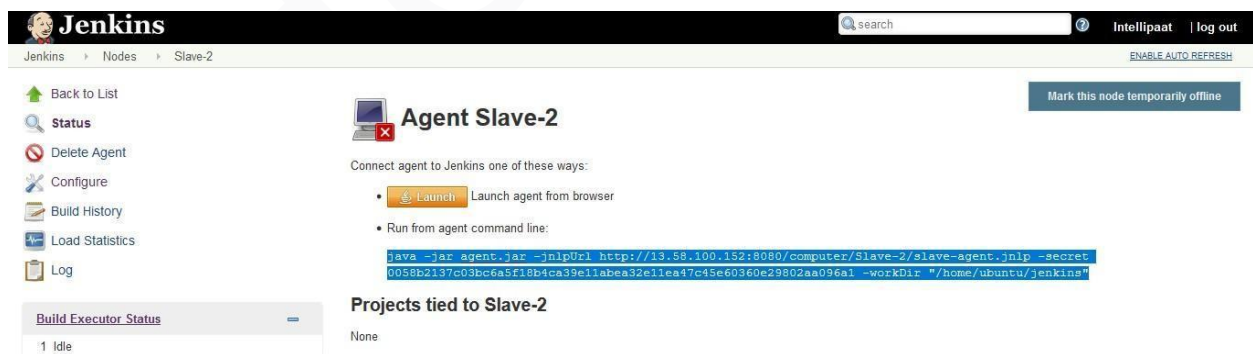


Run the command line from Slave-1 as shown below.

```
ubuntu@ip-172-31-34-189: ~
ubuntu@ip-172-31-34-189:~$ java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-1/slave
t 8ca98c906a08c730a3alc3cf08751b7e211ab79afab4db8a727c5e50b7cedfc3 -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 Step 27 for Slave-2 as well



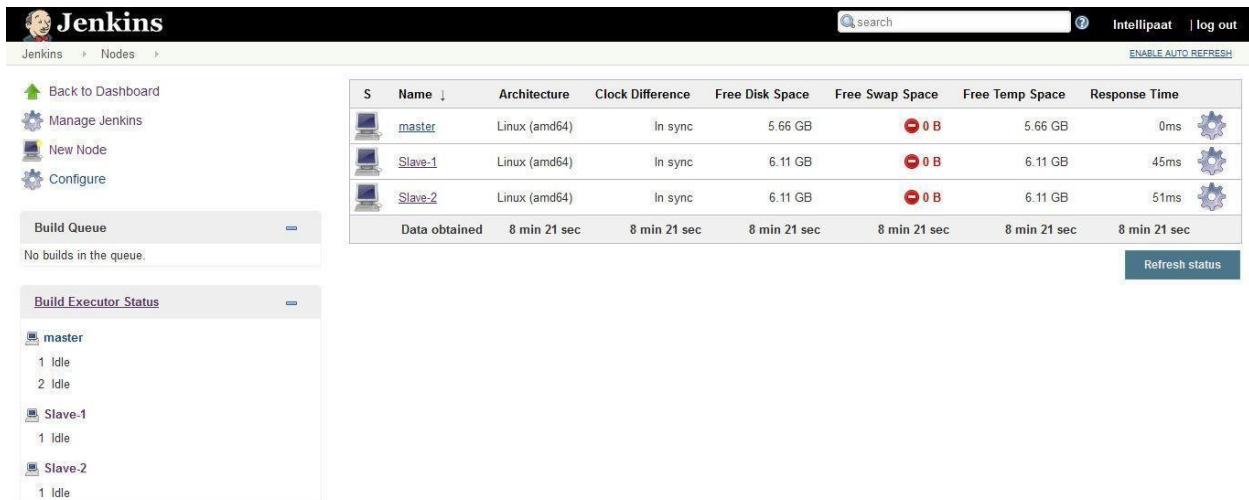
The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and links for 'IntelliPaat' and 'log out'. The breadcrumb trail indicates the current location: 'Jenkins > Nodes > Slave-2'. On the left sidebar, there are links for 'Back to List', 'Status', 'Delete Agent', 'Configure', 'Build History', 'Load Statistics', and 'Log'. The main content area is titled 'Agent Slave-2' and features a 'Mark this node temporarily offline' button. Below the title, it says 'Connect agent to Jenkins one of these ways:' and lists two options: 'Launch agent from browser' (with a 'Launch' button) and 'Run from agent command line'. The command line option is selected, showing the following command in a blue box: `java -jar agent.jar -jnlpUrl http://13.58.100.152:8080/computer/Slave-2/slave-agent.jnlp -secret 0058b2137c03bc6a5f18b4ca39e11abea32e11ea47c45e60360e29802aa096a1 -workDir "/home/ubuntu/jenkins"`. At the bottom, there is a section 'Projects tied to Slave-2' which currently shows 'None'.

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 have 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 |
| Data obtained | | | 8 min 21 sec | 8 min 21 sec | 8 min 21 sec | 8 min 21 sec | 8 min 21 sec |

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

Create a CI/CD pipeline triggered by Git Webhooks:

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

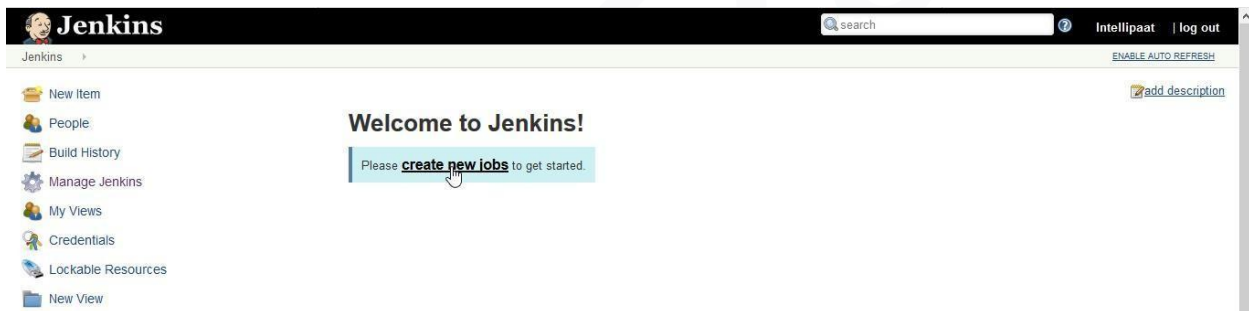
```
ubuntu@ip-172-31-34-189: ~
```

```
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.
```

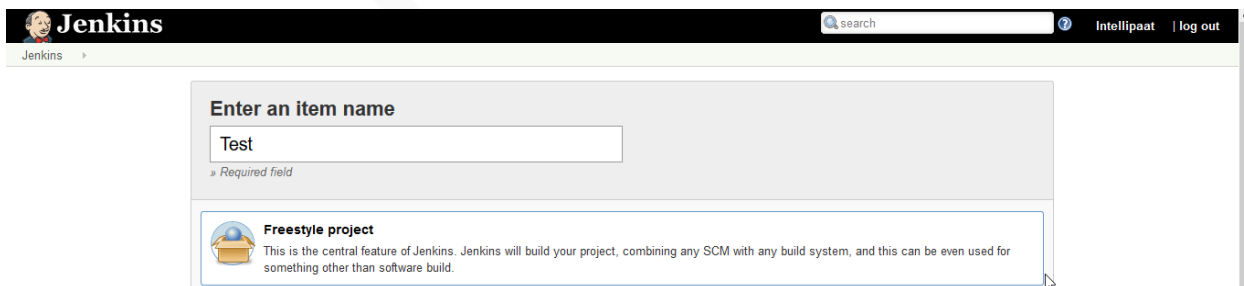
ubuntu@ip-172-31-34-132: ~

```
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

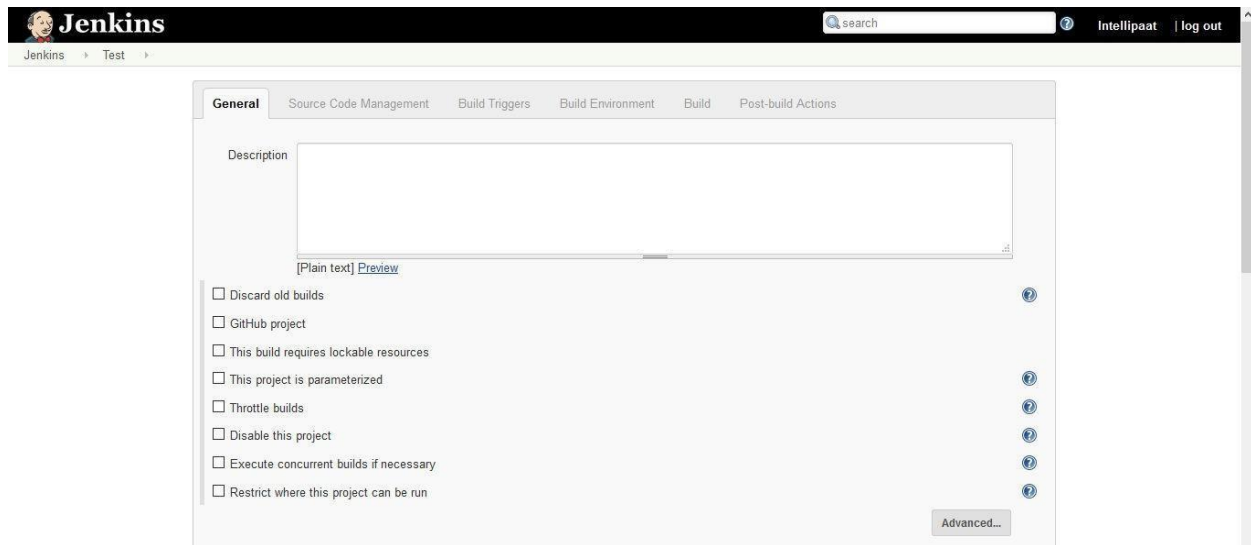


Name the Project as Test, Select Freestyle Project option.

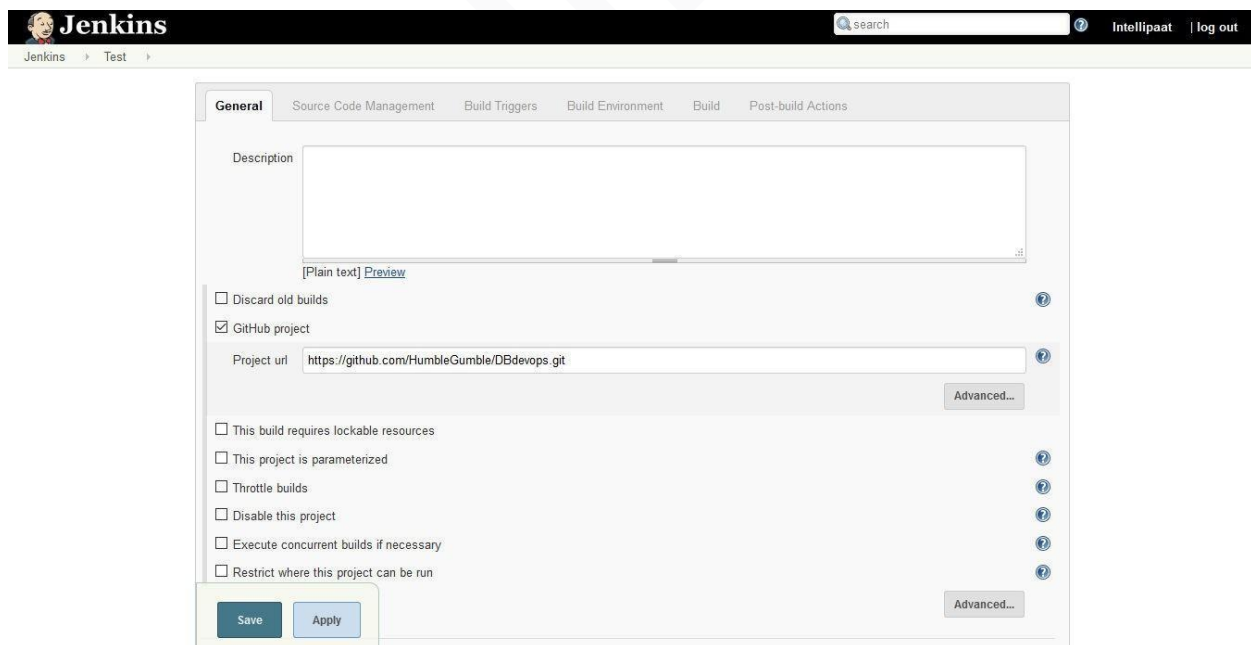


Then click on OK.

You should land on a page like this:



Step 4: Place your Git repository link as shown below:



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

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

[Plain text] [Preview](#)

☐ Discard old builds

☒ GitHub project

Project url [Advanced...](#)

☐ 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

Label Expression [Advanced...](#)

 There's no agent/cloud that matches this assignment. Did you mean 'Slave-1' instead of 'Slav'?

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


Source Code Management

☐ None

☒ Git

Repositories

Repository URL [Advanced...](#)

 Please enter Git repository.

Credentials [Add](#) [Advanced...](#) [Add Repository](#)

Branches to build

Branch Specifier (blank for 'any') [Add Branch](#)

Repository browser [Advanced...](#)

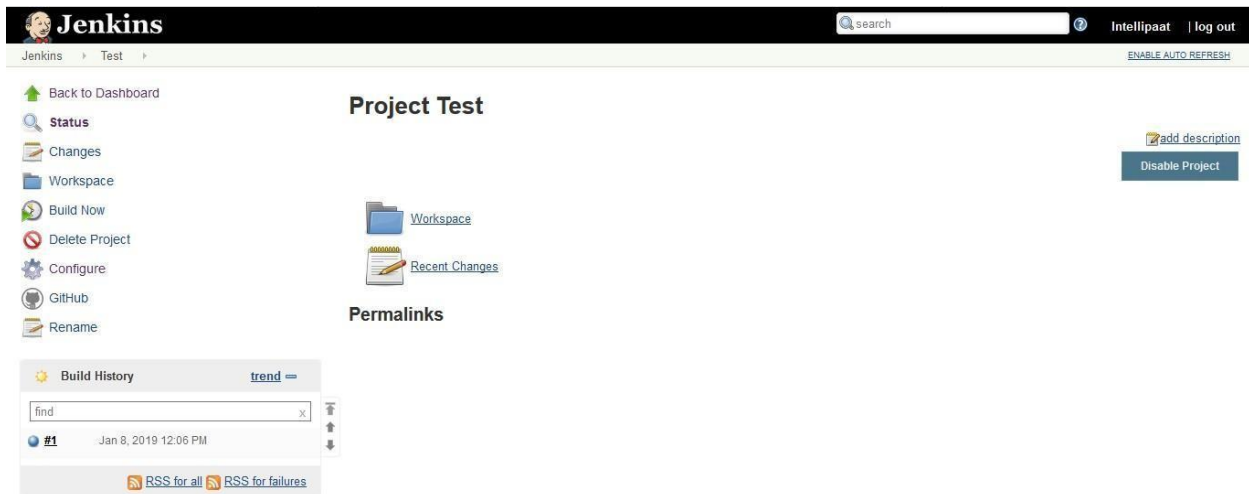
Additional Behaviours [Add](#)

☐ Subversion

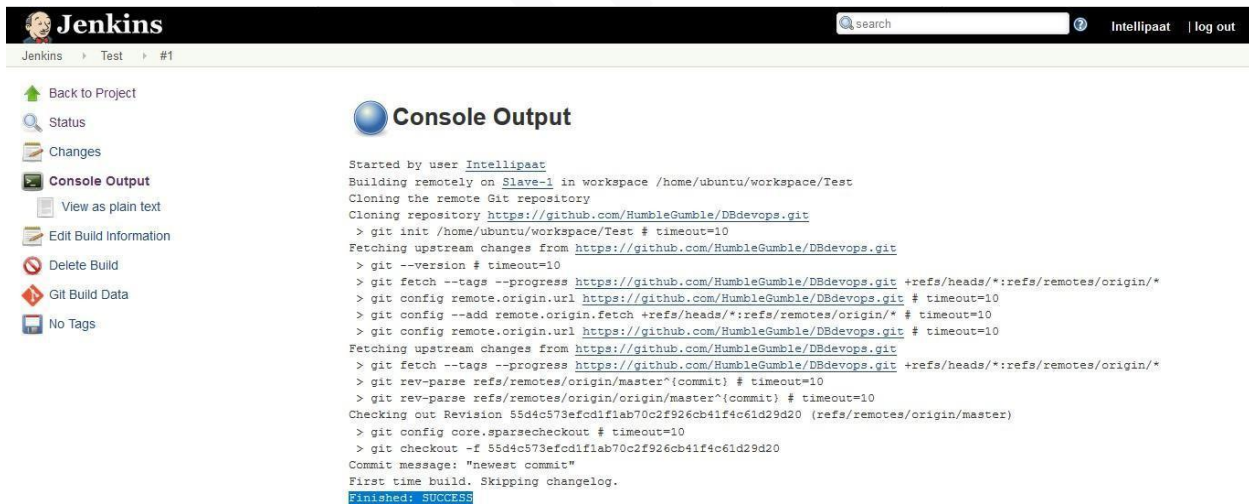
[Save](#) [Apply](#)

Click on Save.

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



Click on the blue circle of build #1



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 of 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
```



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
```

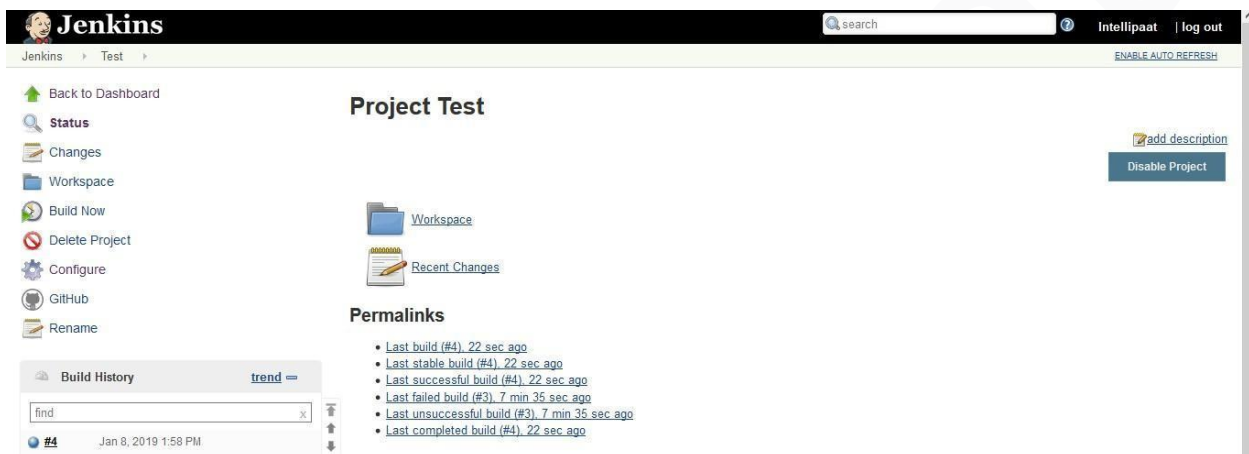
ubuntu@ip-172-31-34-189: ~/workspace/Test

```
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
be2bflc4a48d: Pull complete
a5bdc6303093: Pull complete
e9055237d68d: Pull complete
Digest: sha256:868fd30a0e47b8d8ac485df174795b5e2fe8a6c8f056cc707b232d65b8a1ab68
Status: Downloaded newer image for ubuntu:latest
e9055237d68d: Pull complete
ubuntu@ip-172-31-34-189:~/workspace/Test$
```


Now we have added it to 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
ebe701788bff        ubuntu             "/bin/bash"        About a minute ago  Up About a minute
elaxed_varahamihira
ubuntu@ip-172-31-34-189:~/workspace/Test$
```

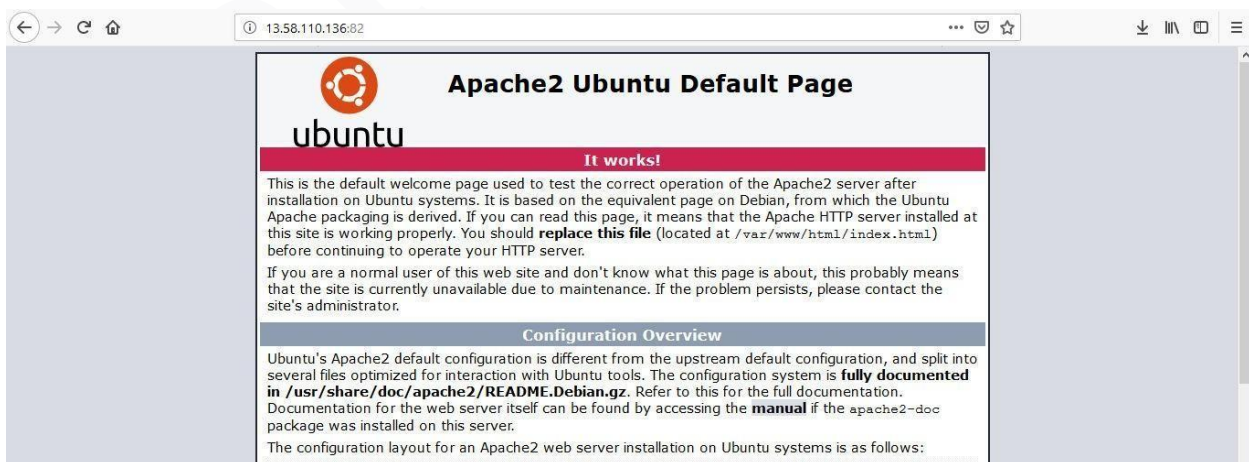
Step 10: Now open Jenkins Dashboard and build the project



The screenshot shows the Jenkins web interface for a project named 'Test'. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, GitHub, and Rename. The main area displays 'Project Test' with a 'Workspace' icon and 'Recent Changes' link. Below this is a 'Permalinks' section listing build history: Last build (#4) 22 sec ago, Last stable build (#4) 22 sec ago, Last successful build (#4) 22 sec ago, Last failed build (#3) 7 min 35 sec ago, Last unsuccessful build (#3) 7 min 35 sec ago, and Last completed build (#4) 22 sec ago. A 'Build History' table at the bottom shows a single entry for build #4 on Jan 8, 2019 at 1:58 PM.

Building was successful.

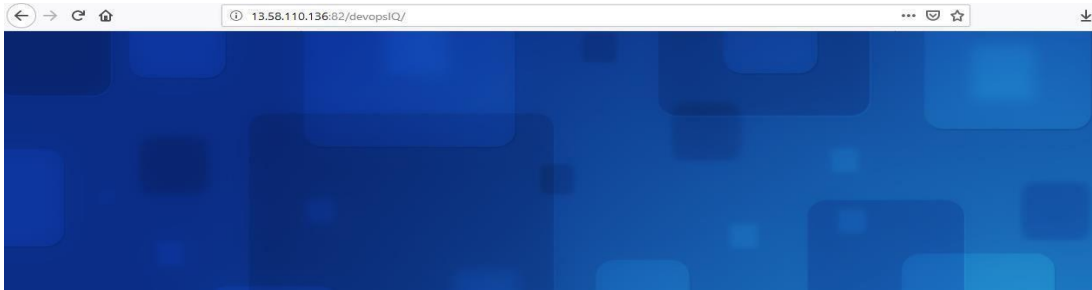
Step 11: Now open browser and enter Slave-1 IP:82



The screenshot shows a web browser displaying the 'Apache2 Ubuntu Default Page'. The page features the Ubuntu logo and the text 'It works!'. Below this, a paragraph explains that this is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It also provides instructions for replacing the file at /var/www/html/index.html and mentions that the configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. A 'Configuration Overview' section follows, detailing the differences between the upstream default configuration and the Ubuntu configuration, and providing a link to the manual for the apache2-doc package. The page concludes with the statement: 'The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:'.

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

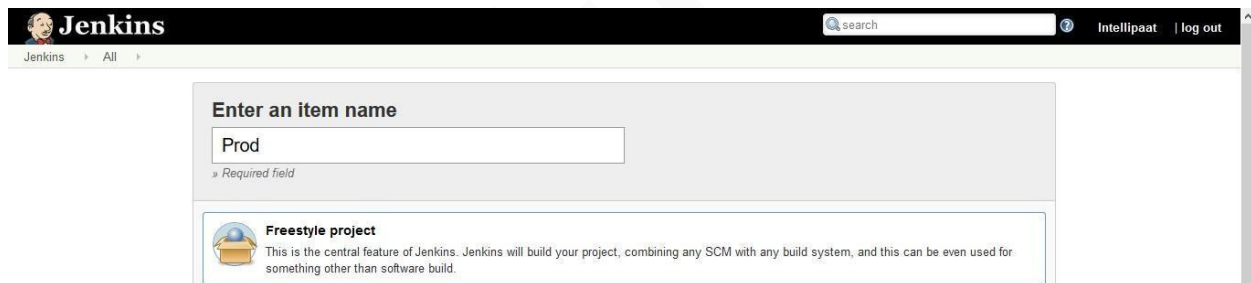
Step 12: Now enter slave-1 IP:82/devopsIQ/ in the browser



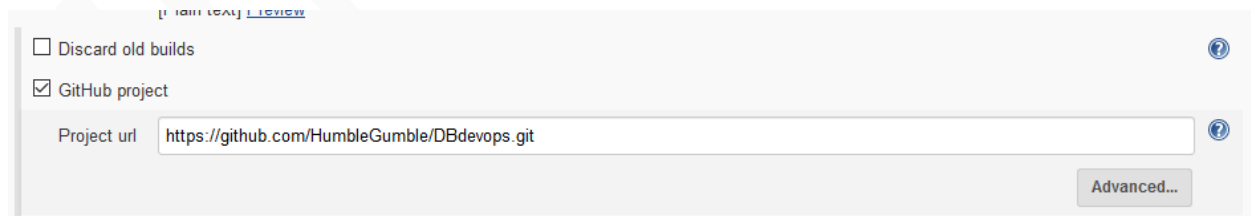
This looks fine.

Now, we will create a new project.

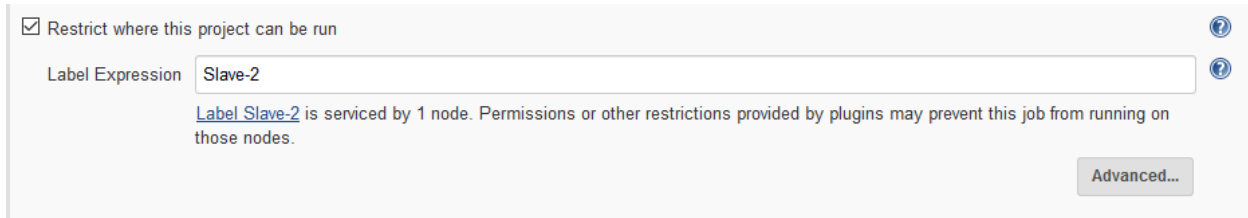
Step 13: Create a new project.



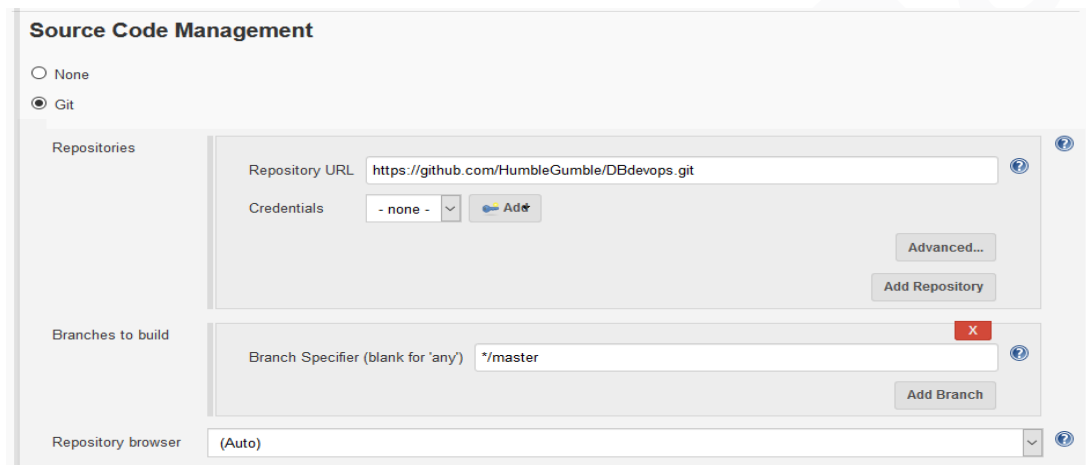
Step 14: Click on Git project. Enter the GitHub repository URL.



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

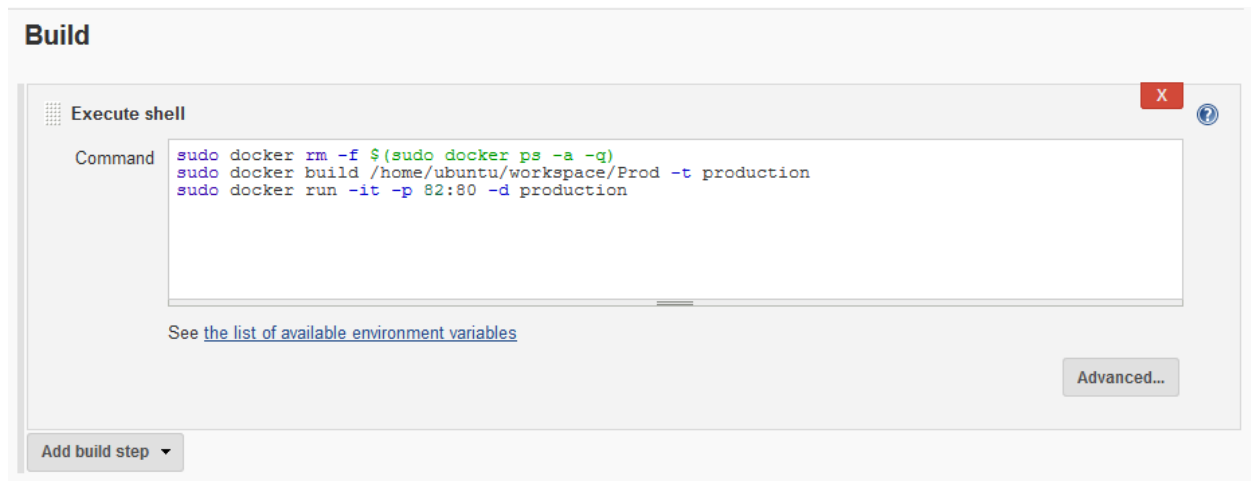


Step 16: Go to Source code management and enter the Git repository URL there as well.

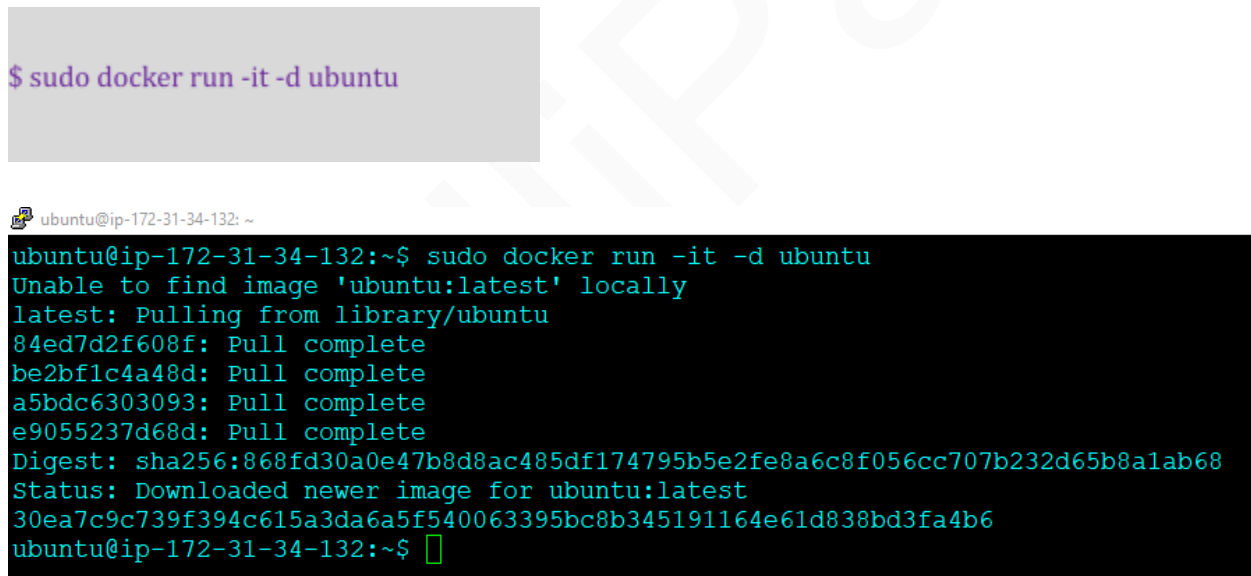


Step 17: Now enter the following command in the Execution shell

```
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
```

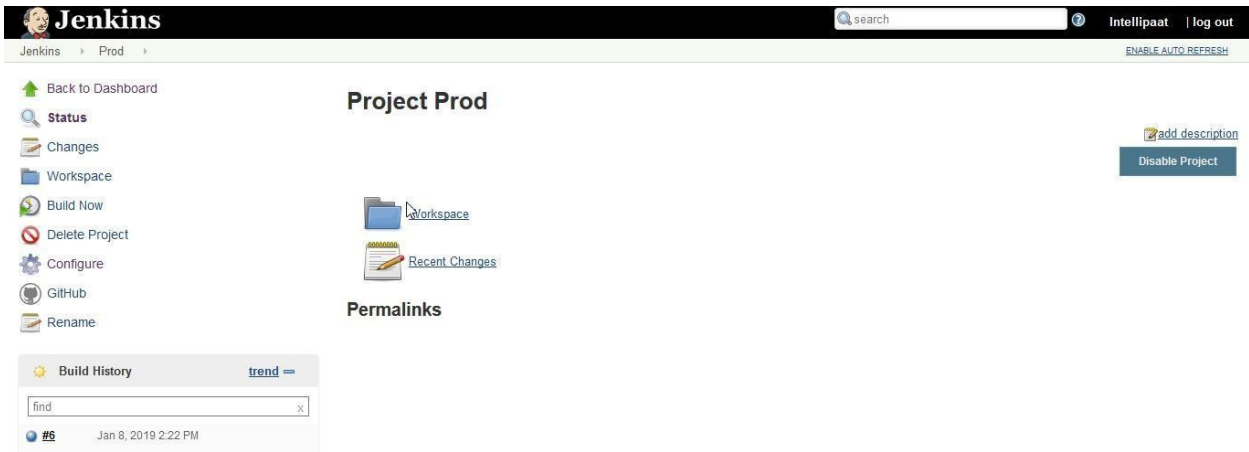


Step 18: Again, add one container to the Slave-2 as shown below:



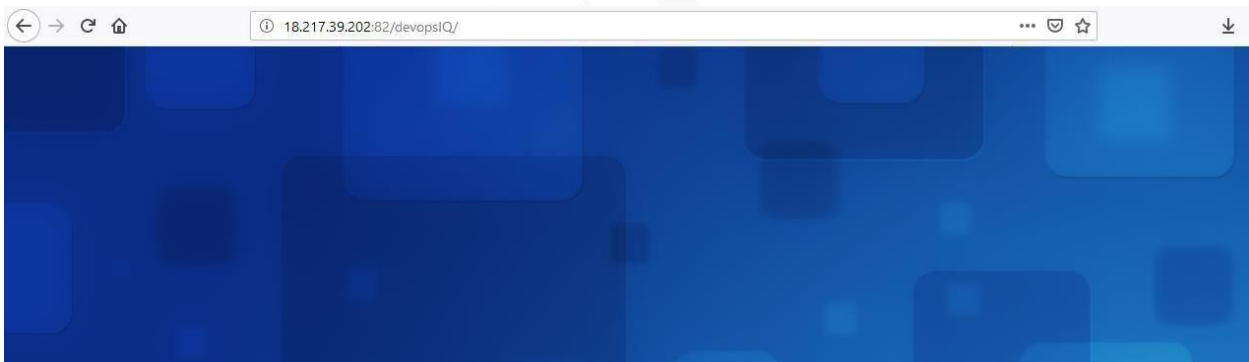
Now that we have added an arbitrary container, go to Jenkins Dashboard and build the project.

Step 19: Build the project Prod.



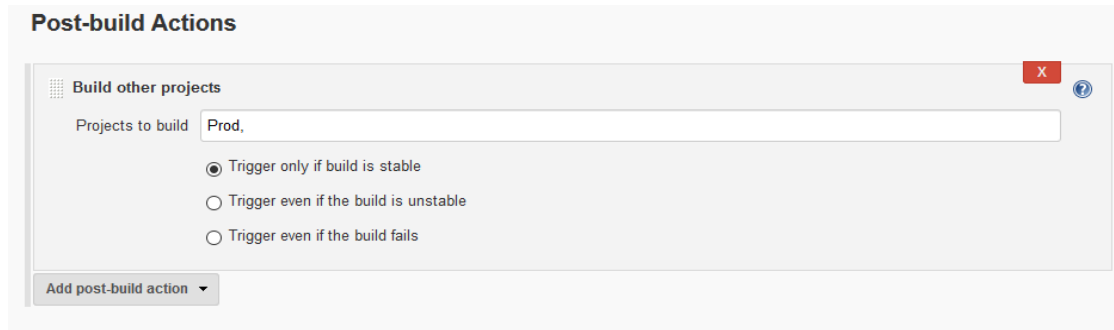
Our Project building was successful.

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 the 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.

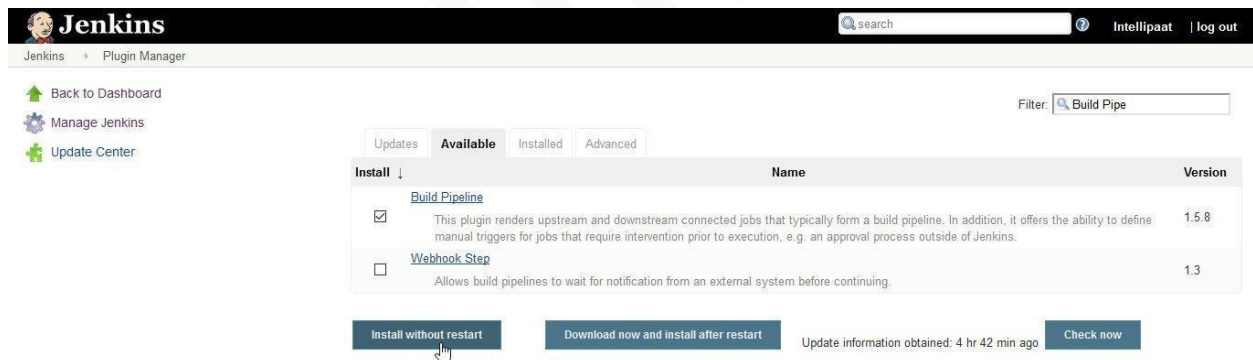


The image shows the 'Post-build Actions' configuration page in Jenkins. Under the 'Build other projects' section, the 'Projects to build' field contains 'Prod,'. Below this, three radio buttons are visible: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom of this section 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 image shows the Jenkins Plugin Manager interface. The 'Available' tab is selected, and a search filter 'Build Pipe' is applied. A table lists available plugins:

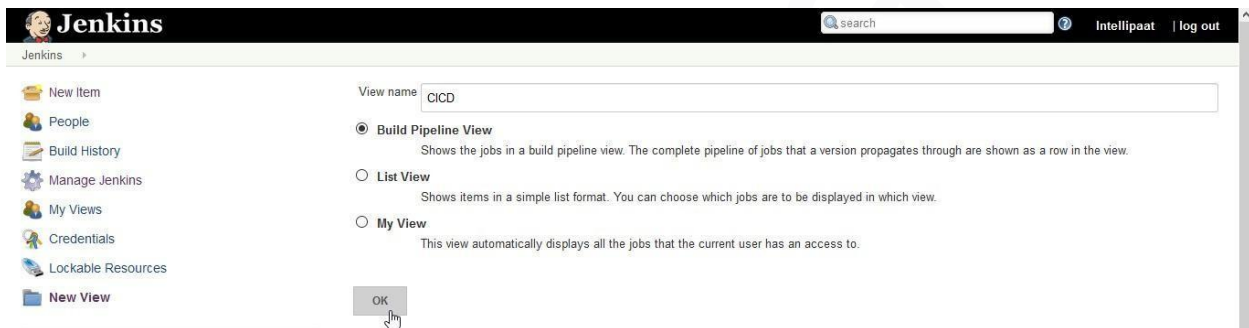
| Install | Name | Version |
|-------------------------------------|---|---------|
| <input checked="" type="checkbox"/> | Build Pipeline This plugin renders upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins. | 1.5.8 |
| <input type="checkbox"/> | Webhook Step Allows build pipelines to wait for notification from an external system before continuing. | 1.3 |

At the bottom, there are two buttons: 'Install without restart' (which is highlighted with a mouse cursor) and 'Download now and install after restart'. To the right, it says 'Update information obtained: 4 hr 42 min ago' and a 'Check now' button.

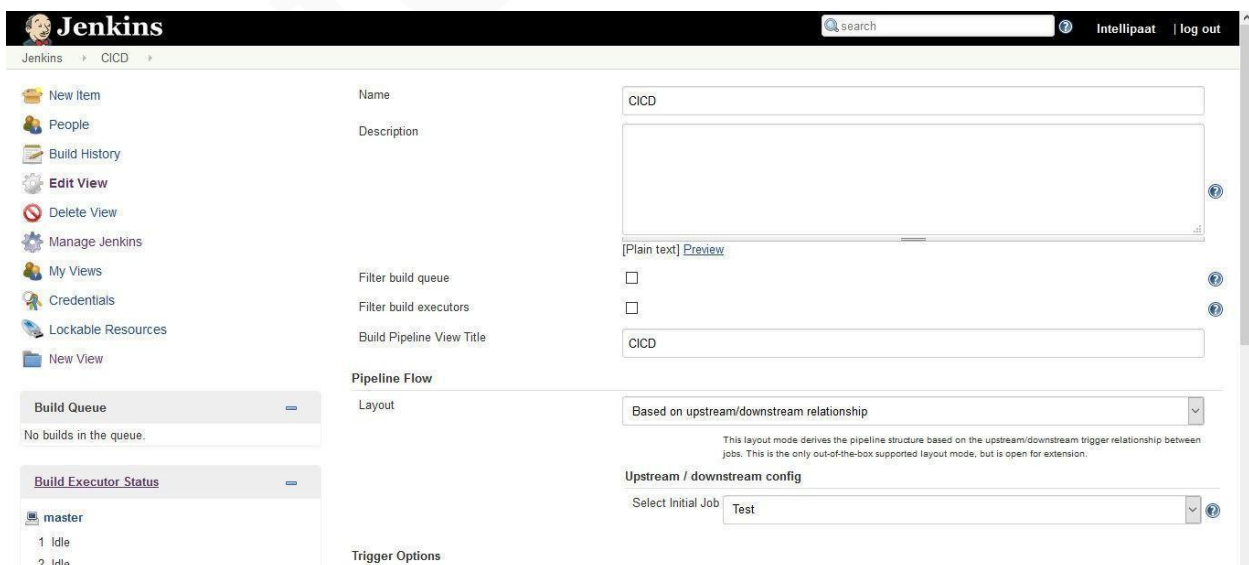
Step 23: Go to the Jenkins dashboard. Click on the +



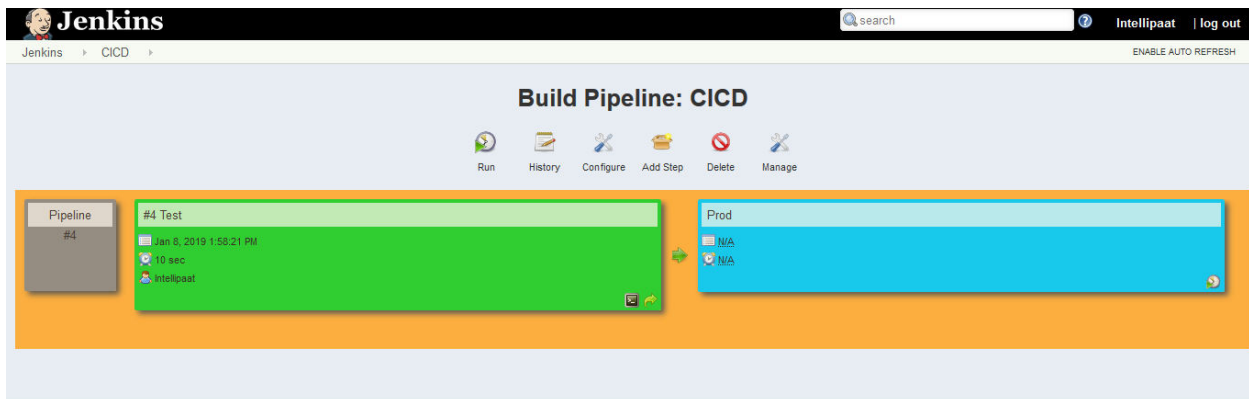
Step 24: Enter view name and click OK.



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



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

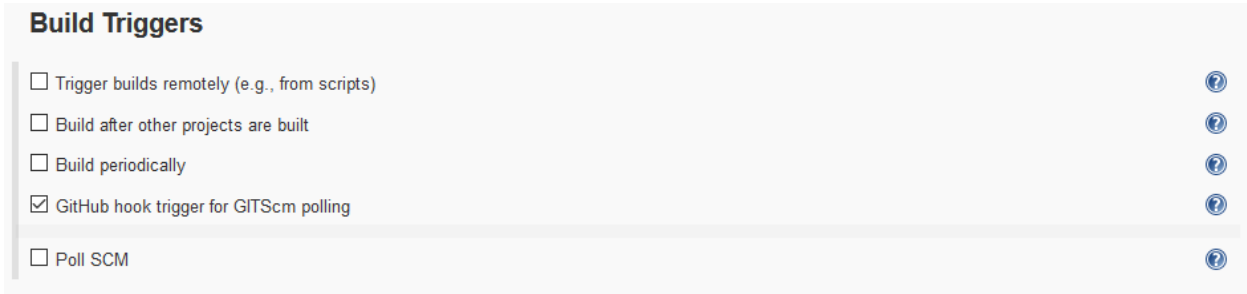


Step 26: Click on Run. Then Refresh the Page once.



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.

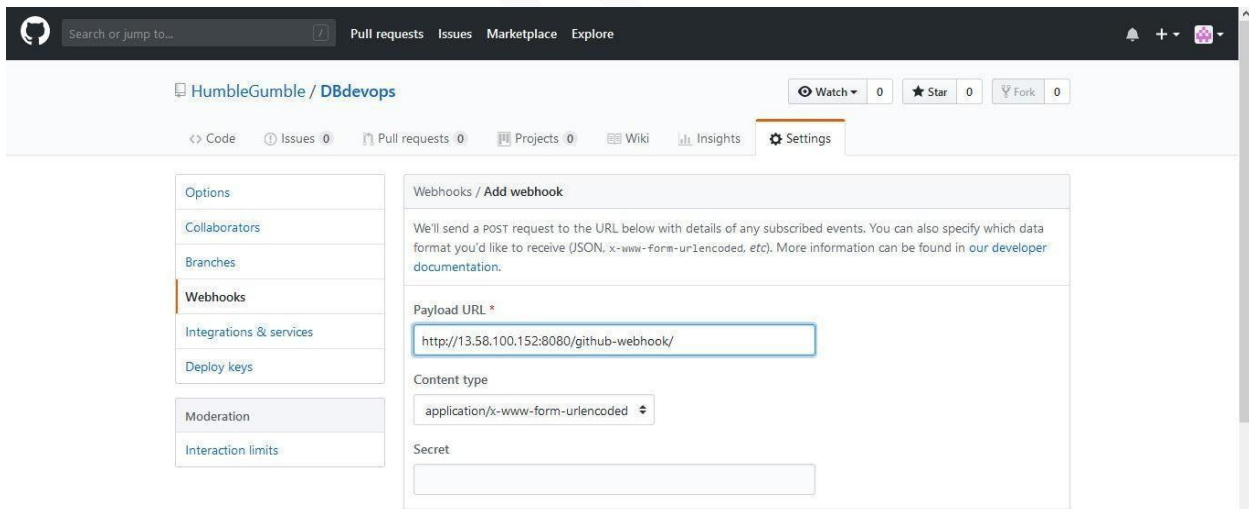


Build Triggers

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

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

`$ JenkinsServer Address/github-webhook/`



GitHub repository: HumbleGumble / DBdevops

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our [developer documentation](#).

Payload URL *

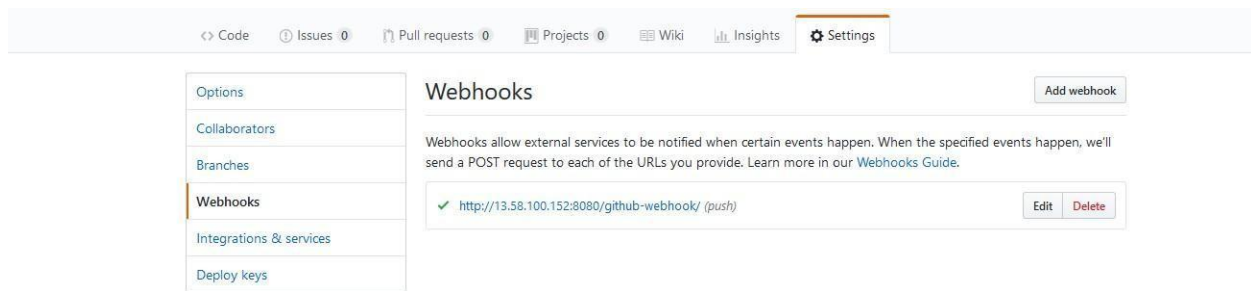
`http://13.58.100.152:8080/github-webhook/`

Content type

`application/x-www-form-urlencoded`

Secret

Click on Add webhook. You should see this.



Step 29: Go to the mater terminal to trigger a built

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

ubuntu@ip-172-31-39-127: ~

```
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 the index.html file. Open it for modification.

```
$ nano index.html
```

```
ubuntu@ip-172-31-39-127: ~/DBdevops
ubuntu@ip-172-31-39-127:~/DBdevops$ ls
Dockerfile  devops1
ubuntu@ip-172-31-39-127:~/DBdevops$ cd devops1
ubuntu@ip-172-31-39-127:~/DBdevops/devops1$ ls
images  index.html
ubuntu@ip-172-31-39-127:~/DBdevops/devops1$
```

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

```
ubuntu@ip-172-31-39-127: ~/DBdevops/devops1Q
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

```
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(-)
```

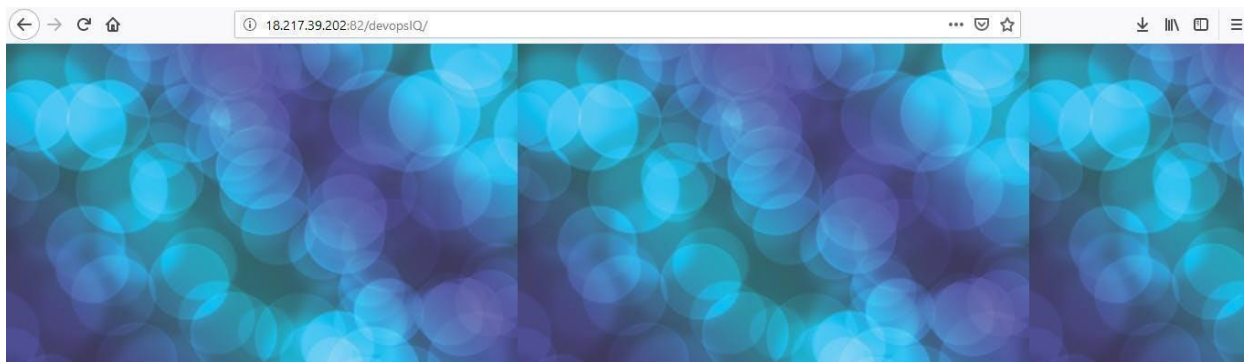
Step 33: Perform git push

```
$ git push origin master
```

ubuntu@ip-172-31-39-127: ~/DBdevops/devopsIQ

```
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 has changed.



Congratulations! You have successfully completed Jenkins hands-on.