



CLOUD TRAIN
ACCELERATE YOUR GROWTH

JENKINS

DevOps Instructor-led Training

Contact us

TO ACCELERATE YOUR CAREER GROWTH

For questions and more details:

please call @ +91 98712 72900, or

visit <https://www.thecloudtrain.com/>, or

email at support@thecloudtrain.com, or

WhatsApp us @ +91 98712 72900

JENKINS HANDS-ON

1. Create 3 instances (Master, Slave1, Slave2) on Linux server.
2. Install Jenkins on Master. (Refer the above steps or the Jenkins installation documentation)
3. Set up a Jenkins Master-Slave Cluster
4. Create a CI CD pipeline triggered by Git Webhook.

Install Jenkins

```
sudo apt update
sudo apt install openjdk-11-jdk
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'

sudo apt update
sudo apt install jenkins
systemctl status jenkins
```

First, we have created 3 instances Master Slave1 and Slave2. 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@instance-1:~$ systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Fri 2021-02-12 12:28:52 UTC; 4min 59s ago
     Docs: man:systemd-sysv-generator(8)
   Process: 29631 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCESS)

Feb 12 12:28:50 instance-1 systemd[1]: Starting LSB: Start Jenkins at boot time...
Feb 12 12:28:51 instance-1 jenkins[29631]: Correct java version found
Feb 12 12:28:51 instance-1 jenkins[29631]: * Starting Jenkins Automation Server jenkins
Feb 12 12:28:51 instance-1 su[29682]: Successful su for jenkins by root
Feb 12 12:28:51 instance-1 su[29682]: + ??? root:jenkins
Feb 12 12:28:51 instance-1 su[29682]: pam_unix(su:session): session opened for user jenkins by (uid=0)
Feb 12 12:28:52 instance-1 jenkins[29631]: ...done.
Feb 12 12:28:52 instance-1 systemd[1]: Started LSB: Start Jenkins at boot time.
```

Step 2: 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 3: Copy the path mentioned in the page and perform cat operation in master terminal.

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

```
ubuntu@instance-1:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
79f304f38fb7467382b3601a58d75b54
ubuntu@instance-1:~$
```

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.

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.263.4

Step 4: Once done, enter the Admin User details.

Getting Started

Create First Admin User

Username:

devops

Password:

.....

Confirm password:

.....

Full name:

Devops

E-mail address:

devops@gmail.com

Jenkins 2.263.4

[Skip and continue as admin](#) [Save and Continue](#)

Then click on **Save and Continue**.

Getting Started

Instance Configuration

Jenkins URL:

http://107.178.208.230:8080/

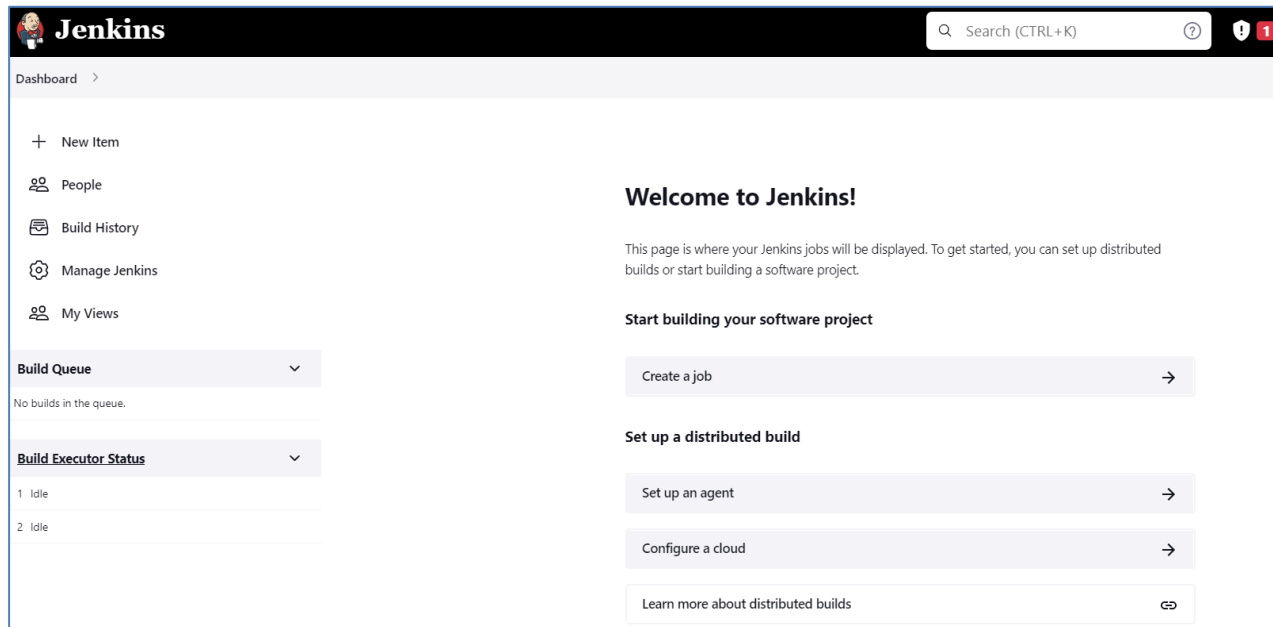
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

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 2.263.4

[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.



This is our **Jenkins Dashboard**.

Hands-On: Configure Slave nodes

Step 5: Go to **Manage Jenkins**. Click on **Configure Global Security**.

Manage Jenkins

My Views

Build Queue ▾
No builds in the queue.

Build Executor Status ▾
1 Idle
2 Idle

System Configuration

- Configure System**
Configure global settings and paths.
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

- Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials**
Configure credentials
- Manage Users**
Create/delete/modify users that can log in to this Jenkins.

Step 6: Change the **Agents** to **Random**. Then click on **Save**.

Agents

TCP port for inbound agents ?

☐ Fixed

☒ **Random**

☐ Disable

Step 7: Now go to **Manage Nodes**.

Dashboard > Manage Jenkins

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation for more information.

System Configuration

- Configure System**
Configure global settings and paths.
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

- Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials**
Configure credentials
- Manage Users**
Create/delete/modify users that can log in to this Jenkins.

Step 8: Click on **New Node**. Add **Slave1** as new node and make **Permanent Agent**. Click on **ok**.

Dashboard > Manage Jenkins > Nodes >

↑ Back to Dashboard



⚙ Manage Jenkins

+ New Node

☁ Configure Clouds

⚙ Node Monitoring

Manage nodes and clouds

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap ↓
	Built-In Node	Linux (amd64)	In sync	3.99 GB	
	Data obtained	12 min	12 min	12 min	12 min

Dashboard > Manage Jenkins > Nodes >

↑ Back to Dashboard

⚙ Manage Jenkins

+ New Node

☁ Configure Clouds

⚙ Node Monitoring

Build Queue ▼
No builds in the queue.

Build Executor Status ▼

New node

Node name

slave1

Type

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.

Create

Step 9: Go to **Launch method** change it to **Launch agent by connecting it to the controller**.

Remote root directory ?

/home/ubuntu/jenkins

Labels ?

Usage ?

Use this node as much as possible

Launch method ?

Launch agent by connecting it to the controller

Step 10: Then add the current working directory path to `/home/ubuntu/jenkins`. Then click on **Save**.

☐ Disable WorkDir ?

Custom WorkDir path ?

`/home/ubuntu/jenkins`

Internal data directory ?

remoting

Step 11: Make another node **Slave2** and copy from **Slave1** as shown below:

Back to Dashboard

Manage Jenkins

New Node

Configure Clouds

Node Monitoring

Id Queue

Builds in the queue.

Id Executor Status

Built-In Node

lle

lle

New node

Node name

`slave2`

Type

☐ 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.

☒ Copy Existing Node








Q `slave1`

Create

Step 12: Then click ok. You can see the list of nodes that we have on the Jenkins Dashboard.

Manage nodes and clouds

Refresh status

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	3.73 GB	 0 B	3.73 GB	0ms 
	slave1		N/A	N/A	N/A	N/A	N/A 
	slave2		N/A	N/A	N/A	N/A	N/A 
Data obtained		5 ms	1 ms	4 ms	1 min 12 sec	1 min 12 sec	1 min 12 sec

Step 13: Go to the Jenkins Dashboard, Click on **Slave1**. Download the **Agent.jar** file by clicking on it.

Agent slave1

[Mark this node temporarily offline](#)

Run from agent command line:

```
curl -sO http://34.125.60.191:8080/jnlpJars/agent.jar
java -jar agent.jar -jnlpUrl http://34.125.60.191:8080/computer/slave1/jenkins-agent.jnlp -secret 7fda553426f19fa3d18297a96139be0aeae22112d66ee793dc6a04d5465dbef -workDir "/home/ubuntu/jenkins"
```

Or run from agent command line, with the secret stored in a file:

```
echo 7fda553426f19fa3d18297a96139be0aeae22112d66ee793dc6a04d5465dbef > secret-file
curl -sO http://34.125.60.191:8080/jnlpJars/agent.jar
java -jar agent.jar -jnlpUrl http://34.125.60.191:8080/computer/slave1/jenkins-agent.jnlp -secret @secret-file -workDir "/home/ubuntu/jenkins"
```

Step 14: Now download **agent.jar** file and upload on the ubuntu server using scp, winscp or filezilla.

Step 15: Let us verify if the file has been transferred to **Slave1** or not. Open a new session on putty. Connect to slave1. Run **ls** command.

```
ubuntu@slave1:~$ ls -ltr
total 1488
-rw-rw-r-- 1 ubuntu ubuntu 1521553 Feb 12 12:58 agent.jar
ubuntu@slave1:~$
```


As you can see the agent.jar file appears there, which means our file has been successfully transferred to **Slave1**.

Step 16: Perform the steps for Slave2 as well. (Tip: Rename the agent.jar file of **Slave2**.

Agent slave2

[Mark this node temporarily offline](#)

Connect agent to Jenkins one of these ways:

-  Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://107.178.208.230:8080/computer/slave2/slave-agent.jnlp -secret 1e7ccf69323b5a4e50e390b5577ca559cea8175f3e1b91cf3307fa639511058d -workDir "/home/ubuntu/jenkins"
```

Run from agent command line, with the secret stored in a file:

```
echo 1e7ccf69323b5a4e50e390b5577ca559cea8175f3e1b91cf3307fa639511058d > secret-file
java -jar agent.jar -jnlpUrl http://107.178.208.230:8080/computer/slave2/slave-agent.jnlp -secret @secret-file -workDir "/home/ubuntu/jenkins"
```

Projects tied to slave2

Step 17: Again, verify by opening a new putty session for Slave2.

```
ubuntu@slave1:~$ ls -ltr
total 1488
-rw-rw-r-- 1 ubuntu ubuntu 1521553 Feb 12 12:58 agent.jar
ubuntu@slave1:~$
```

Step 18: Now before moving ahead install openjdk on both Slave1 and Slave2.

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

```
ubuntu@slave1:~$ sudo apt-get update
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 0s (583 kB/s)
Reading package lists... Done
```

```
ubuntu@slave2:~$ sudo apt-get update
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:6 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
Get:7 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
Get:8 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
Get:9 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [1884 kB]
Get:10 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [390 kB]
```

Step 19: Now install run the following installation command on both terminal.

```
$ sudo apt install openjdk-11-jdk
```

```
ubuntu@n-slave:~$ sudo apt install openjdk-11-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libnumal
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  at-spi2-core ca-certificates-java fontconfig-config fonts-dejavu-core fonts-dej
  libasound2-data libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni
  libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libf
  libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libic
```

Step 20: Now we will connect Slave1 and Slave2 to the Jenkins Server. Go to the Jenkins Dashboard, Click on Slave1, **Copy the command line** as shown.

Run the command line from Slave1 as shown below.

Agent slave1

Mark this node temporarily offline

Run from agent command line:

```
curl -sO http://34.125.60.191:8080/jnlpJars/agent.jar
java -jar agent.jar -jnlpUrl http://34.125.60.191:8080/computer/slave1/jenkins-agent.jnlp -secret 7fda553426f19fa3d18297a96139be0aee22112d66ee793dc6a04d5465dbef -workDir "/home/ubuntu/jenkins"
```

Or run from agent command line, with the secret stored in a file:

```
echo 7fda553426f19fa3d18297a96139be0aee22112d66ee793dc6a04d5465dbef > secret-file
curl -sO http://34.125.60.191:8080/jnlpJars/agent.jar
java -jar agent.jar -jnlpUrl http://34.125.60.191:8080/computer/slave1/jenkins-agent.jnlp -secret @secret-file -workDir "/home/ubuntu/jenkins"
```

```
ubuntu@slave1:~$ echo ba206675983f0546ee5fab69611cfdcf8bd969a85565207b0a881a0ca4f76582 > secret-file
ubuntu@slave1:~$ java -jar agent.jar -jnlpUrl http://107.178.208.230:8080/computer/slave1/slave-agent.jnlp -secret @secret-file -workDir "/home/ubuntu/jenkins"
Feb 12, 2021 1:08:26 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Feb 12, 2021 1:08:26 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/jenkins/remoting
Feb 12, 2021 1:08:26 PM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: slave1
Feb 12, 2021 1:08:26 PM hudson.remoting.jnlp.Main$CuiListener <init>
INFO: Jenkins agent is running in headless mode.
Feb 12, 2021 1:08:26 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 4.5
Feb 12, 2021 1:08:26 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Feb 12, 2021 1:08:26 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://107.178.208.230:8080/]
Feb 12, 2021 1:08:26 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Feb 12, 2021 1:08:26 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Agent discovery successful
    Agent address: 107.178.208.230
    Agent port: 37689
    Identity: e7:1c:03:31:93:ac:f0:16:c5:30:7d:96:34:5e:39:00
Feb 12, 2021 1:08:26 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Feb 12, 2021 1:08:26 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 107.178.208.230:37689
Feb 12, 2021 1:08:26 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP4-connect
Feb 12, 2021 1:08:27 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Remote identity confirmed: e7:1c:03:31:93:ac:f0:16:c5:30:7d:96:34:5e:39:00
Feb 12, 2021 1:08:29 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

It shows **“Connected”**.

Step 21: Perform the **Step-20** for **Slave2** as well.

Agent slave2

[Mark this node temporarily offline](#)

Run from agent command line:

```
curl -sO http://34.125.60.191:8080/jnlpJars/agent.jar
java -jar agent.jar -jnlpUrl http://34.125.60.191:8080/computer/slave2/jenkins-agent.jnlp -secret 2b276330d0eefeb5194efd402bcf5f0acfd225f0bbd14c7e3eed06991d361f5d -workDir "/home/ubuntu/jenkins"
```

Or run from agent command line, with the secret stored in a file:







```
echo 2b276330d0eefeb5194efd402bcf5f0acfd225f0bbd14c7e3eed06991d361f5d > secret-file
curl -sO http://34.125.60.191:8080/jnlpJars/agent.jar
java -jar agent.jar -jnlpUrl http://34.125.60.191:8080/computer/slave2/jenkins-agent.jnlp -secret @secret-file -workDir "/home/ubuntu/jenkins"
```

Paste the command line in the Slave2 Terminal.

```
ubuntu@slave2:~$ echo 1e7ccf69323b5a4e50e390b5577ca559caa8175f3e1b91cf3307fa639511058d > secret-file
ubuntu@slave2:~$ java -jar agent.jar -jnlpUrl http://107.178.208.230:8080/computer/slave2/slave-agent.jnlp -secret @secret-file -workDir "/home/ubuntu/jenkins"
Feb 12, 2021 1:09:33 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Feb 12, 2021 1:09:33 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/jenkins/remoting
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: slave2
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main$CuiListener <init>
INFO: Jenkins agent is running in headless mode.
Feb 12, 2021 1:09:34 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 4.5
Feb 12, 2021 1:09:34 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://107.178.208.230:8080/]
Feb 12, 2021 1:09:34 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Agent discovery successful
Agent address: 107.178.208.230
Agent port: 37689
Identity: e7:1c:03:31:93:ac:f0:16:c5:30:7d:96:34:5e:39:00
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 107.178.208.230:37689
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP4-connect
Feb 12, 2021 1:09:34 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Remote identity confirmed: e7:1c:03:31:93:ac:f0:16:c5:30:7d:96:34:5e:39:00
Feb 12, 2021 1:09:35 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

Important Note: Don't end the Sessions that we just Connected. To perform further operations on Slave1 and Slave2 duplicate the sessions.

So now that our Slave1 and Slave2 has been connected to Jenkins Server, it looks like this.

Jenkins							
Dashboard Nodes							
Back to Dashboard Manage Jenkins New Node Configure Clouds Node Monitoring							
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	6.84 GB		6.84 GB	0ms
	slave1	Linux (amd64)	In sync	7.47 GB		7.47 GB	66ms
	slave2	Linux (amd64)	In sync	7.49 GB		7.49 GB	116ms
Data obtained			15 sec	15 sec	15 sec	15 sec	15 sec
Refresh status							

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 and Run Jenkins Jobs.

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

```
https://github.com/vistasunil/devopsIQ.git
```

Step 2: Install docker on both **Slave1** and **Slave2**.

```
sudo apt install docker.io
```

```
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:~$ 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 Slave1.

Dashboard >

+ New Item

👤 People

📅 Build History

⚙️ Manage Jenkins

👤 My Views

Build Queue

No builds in the queue.

Build Executor Status

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project


Create a job




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

Enter an item name

» Required field

**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be something other than software build.

**Pipeline**

Then click on **Ok**.

You should land on a page like this.

Configuration

- ⚙️ General
- 🔑 Source Code Management
- 🕒 Build Triggers
- 🌐 Build Environment
- ☰ Build Steps
- 📁 Post-build Actions

Description

[Plain text] [Preview](#)

- ☐ Discard old builds ?
- ☐ GitHub project
- ☐ This project is parameterized ?
- ☐ Throttle builds ?
- ☐ Execute concurrent builds if necessary ?
- ☐ Restrict where this project can be run ?

Advanced...

Step 4: Place your git repository link <https://github.com/vistasunil/devopsIQ.git> as shown below.

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

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

Credentials ?

- none -

Click on **Restrict where this project can be run**. Add **Slave1** there.

☒ GitHub project

Project url ?

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

Advanced...

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

☒ Restrict where this project can be run ?

Label Expression ?

slave1

Label **slave1** matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

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

Repositories ?

Repository URL ?

https://github.com/vistasunil/devopsIQ.git

Credentials ?

- none -

+ Add

Advanced...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

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.

Dashboard > Demo >

↑ Back to Dashboard

☰ Status

</> Changes

📁 Workspace

▶ **Build Now**

⚙️ Configure

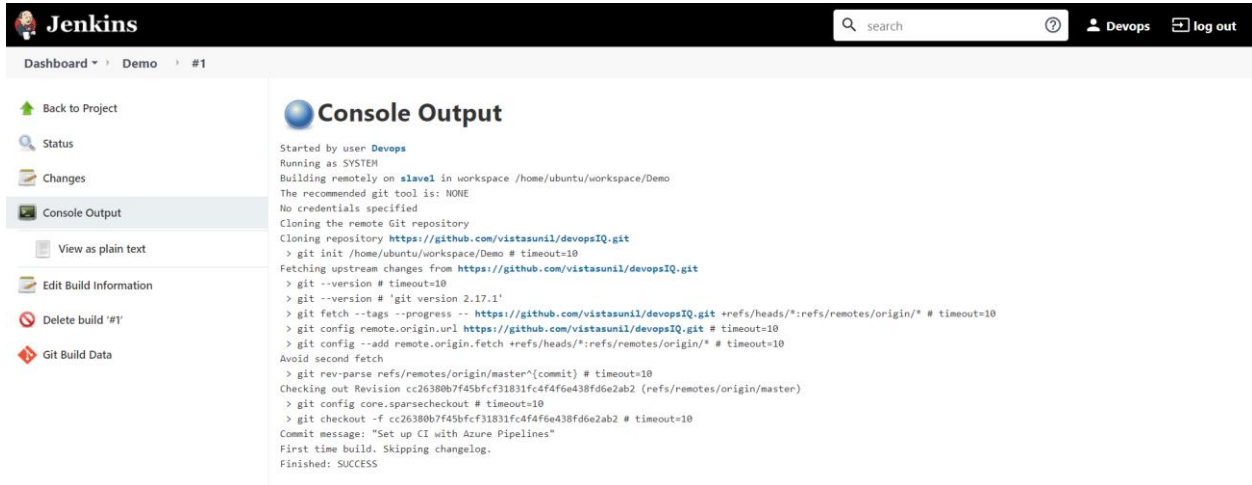
🗑️ Delete Project

🔄 GitHub

Project Demo

Permalinks

Click on the blue circle of build #1.



The image shows the Jenkins web interface. On the left is a sidebar with navigation links: Back to Project, Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete build '#1', and Git Build Data. The main area is titled 'Console Output' and shows the build log for build #1. The log starts with 'Started by user Devops' and 'Running as SYSTEM'. It shows the build is running on a slave named 'slave1' in the workspace '/home/ubuntu/workspace/Demo'. The log details the cloning of a Git repository from 'https://github.com/vistasunil/devopsIQ.git', fetching upstream changes, and checking out the master branch. The build is successful.

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

Step 6: Go to slave1.

```
$ ls  
  
$ cd workspace  
  
$ ls  
  
$ cd Demo  
  
$ ls
```

```
ubuntu@slave1:~$ ls -ltr  
total 1500  
-rw-rw-r-- 1 ubuntu ubuntu 1521553 Feb 12 12:58 agent.jar  
-rw-rw-r-- 1 ubuntu ubuntu 65 Feb 12 13:08 secret-file  
drwxrwxr-x 3 ubuntu ubuntu 4096 Feb 12 13:08 jenkins  
drwxrwxr-x 3 ubuntu ubuntu 4096 Feb 12 13:20 workspace  
ubuntu@slave1:~$ cd workspace/  
ubuntu@slave1:~/workspace$ ls -ltr  
total 4  
drwxrwxr-x 4 ubuntu ubuntu 4096 Feb 12 13:20 Demo  
ubuntu@slave1:~/workspace$ cd Demo/  
ubuntu@slave1:~/workspace/Demo$ ls -ltr  
total 11488  
-rw-rw-r-- 1 ubuntu ubuntu 462 Feb 12 13:20 azure-pipelines.yml  
-rw-rw-r-- 1 ubuntu ubuntu 56 Feb 12 13:20 Dockerfile  
drwxrwxr-x 3 ubuntu ubuntu 4096 Feb 12 13:20 devopsIQ  
-rwxrwxr-x 1 ubuntu ubuntu 11748168 Feb 12 13:20 docker-compose  
ubuntu@slave1:~/workspace/Demo$
```

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

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

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

```
ubuntu@slave1:~/workspace/Demo$ pwd  
/home/ubuntu/workspace/Demo
```

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/Jenkins/workspace/Demo -t  
devopsdemo
```

```
sudo docker run -it -p 82:80 -d devopsdemo
```

Build

 **Execute shell**  

Command

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

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

Advanced...

Click on save.

Before building our job again we must add one arbitrary container in slave1.

Step 9: Add container by performing the following command.



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

```
ubuntu@slave1:~/workspace/Demo$ sudo docker run -it -d ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
83ee3a23efb7: Pull complete
db98fc6f11f0: Pull complete
f611acd52c6c: Pull complete
Digest: sha256:703218c0465075f4425e58fac086e09e1de5c340b12976ab9eb8ad26615c3715
Status: Downloaded newer image for ubuntu:latest
472b62d1a51c241cfaa273b34f7abb922436bcb35b2c8c78f75853eaa42dd89c
ubuntu@slave1:~/workspace/Demo$ docker ps
```

Now we have added in a container.

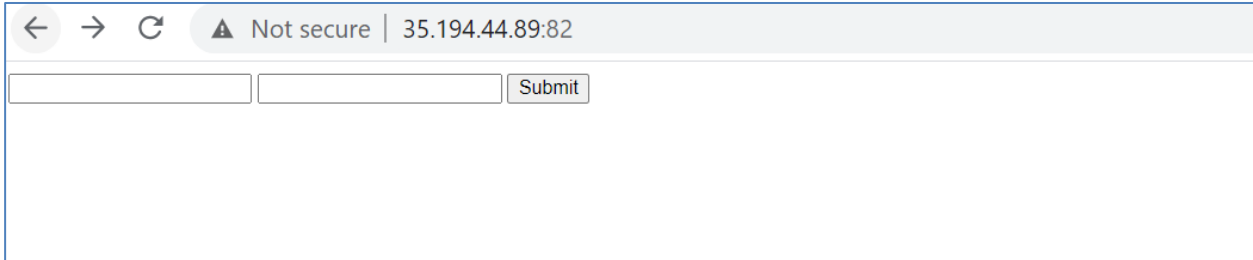
```
ubuntu@slave1:~/workspace/Demo$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
472b62d1a51c   ubuntu   "/bin/bash"             24 seconds ago Up 22 seconds          cranky_hofstadter
ubuntu@slave1:~/workspace/Demo$
```

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

The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and links for 'Devops' and 'log out'. The main content area is titled 'Project Demo' and features a sidebar with navigation options: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now' (highlighted), 'Configure', 'Delete Project', 'GitHub', and 'Rename'. The 'Build History' section shows a list of builds, with build #2 highlighted in yellow and labeled '(pending--[slave1] is offline)'. The main panel displays 'Project Demo' details, including 'Workspace' and 'Recent Changes' sections. A 'Permalinks' section lists recent build events: 'Last build (#1), 22 min ago', 'Last stable build (#1), 22 min ago', 'Last successful build (#1), 22 min ago', and 'Last completed build (#1), 22 min ago'. A 'Disable Project' button is visible in the top right corner.

Building was successful.

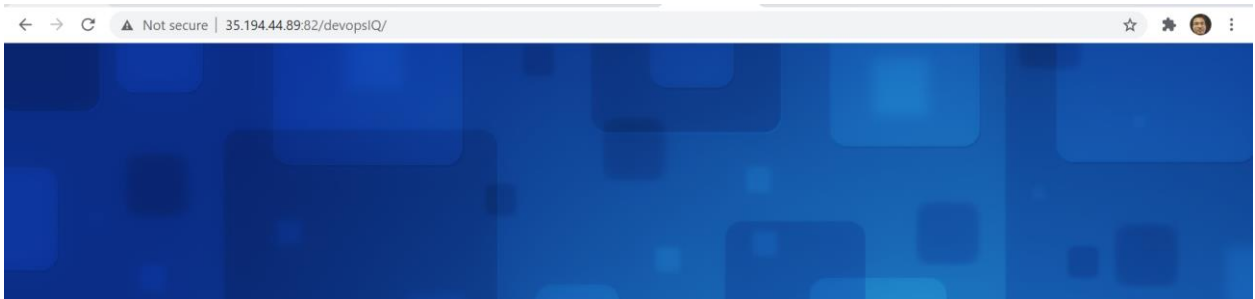
Step 11: Now open browser and enter **Slave1 IP:82**



A screenshot of a web browser window. The address bar shows "35.194.44.89:82" with a "Not secure" warning. The page content is mostly blank, with a "Submit" button visible at the top.

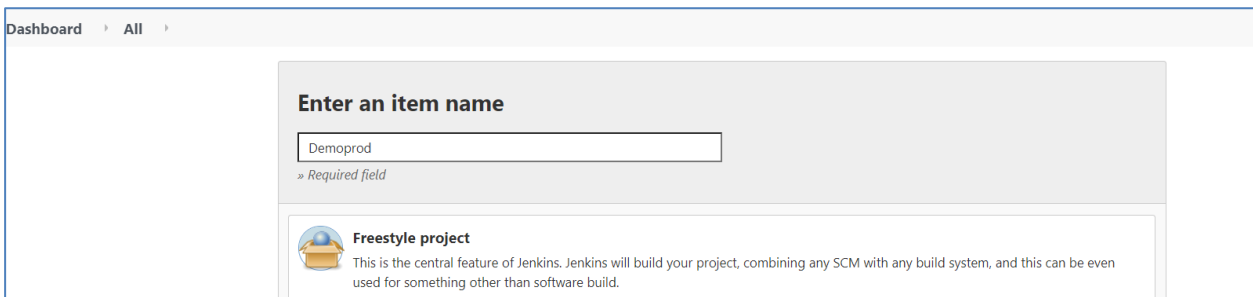
This is the apache page that means our container is working perfectly.

Step 12: Now enter **slave1 IP:82/devopsIQ/** in the browser.



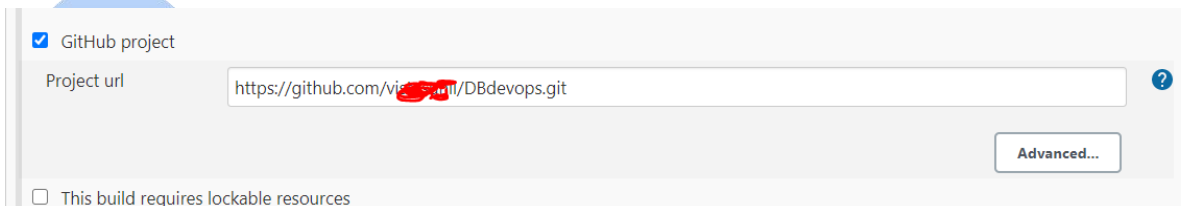
Now, we will create a new project.

Step 13: Create a new project.



A screenshot of the Jenkins "Create new project" form. The "Enter an item name" field contains "Demoprod". Below it, there is a "Freestyle project" option 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."

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



A screenshot of the Jenkins "Configure project" form for a GitHub project. The "Project url" field contains "https://github.com/vic-swt/DBdevops.git". There is a checkbox for "This build requires lockable resources" which is currently unchecked. An "Advanced..." button is visible on the right.

Step 15: Click on **Restrict where this project can be run** enter **Slave2**.

☒ Restrict where this project can be run ?

Label Expression ?

Label **slave2** matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Advanced...


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

Source Code Management

☐ None
☒ Git

Repositories

Repository URL ?

 Please enter Git repository. ?

Credentials ?

Branches to build

Branch Specifier (blank for 'any') X ?

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/Demoprod -t devopsprod
sudo docker run -it -p 82:80 -d devopsprod
```



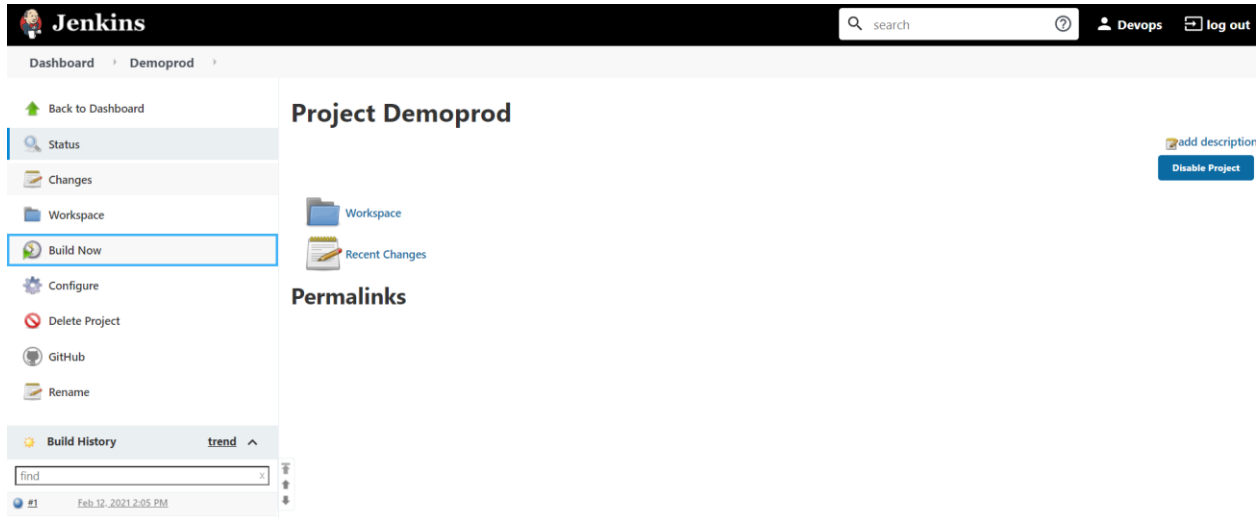
Step 18: Again, add one container to the Slave2 as shown below.

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

```
ubuntu@slave2:~$ sudo docker run -it -d ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
83ee3a23efb7: Pull complete
db98fc6f11f0: Pull complete
f611acd52c6c: Pull complete
Digest: sha256:703218c0465075f4425e58fac086e09e1de5c340b12976ab9eb8ad26615c3715
Status: Downloaded newer image for ubuntu:latest
db33239c567a04c84a938c4d5c41c025b31cd75bea857a08e45af19553d7468c
```

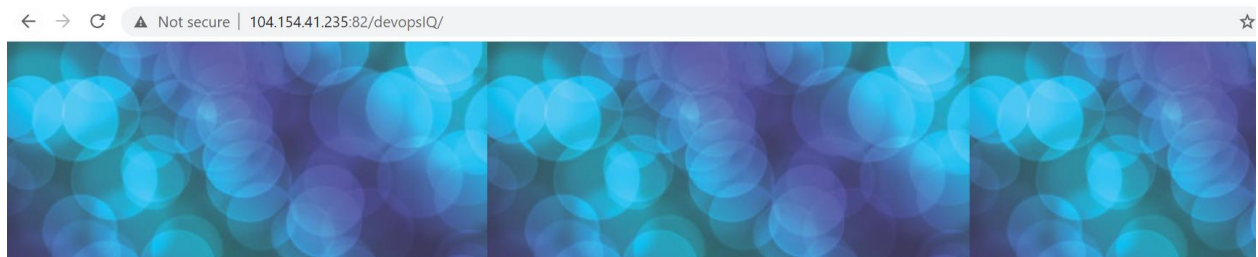
Now that we have added an arbitrary container, go to Jenkins Dashboard and build the project.

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



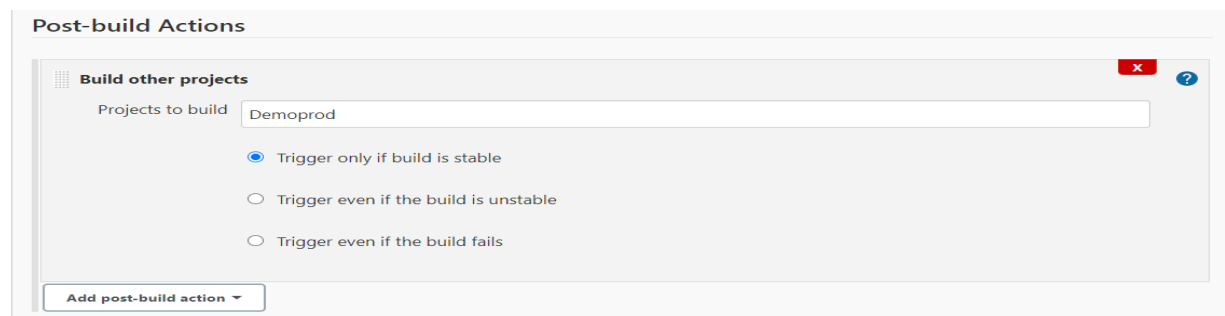
Our Project building was successful.

Step 20: Now go to the browser and enter **Slave2 IP:82/devopsIQ/**



Now trigger **Demoprod** job only when **Demo** job will be completed.

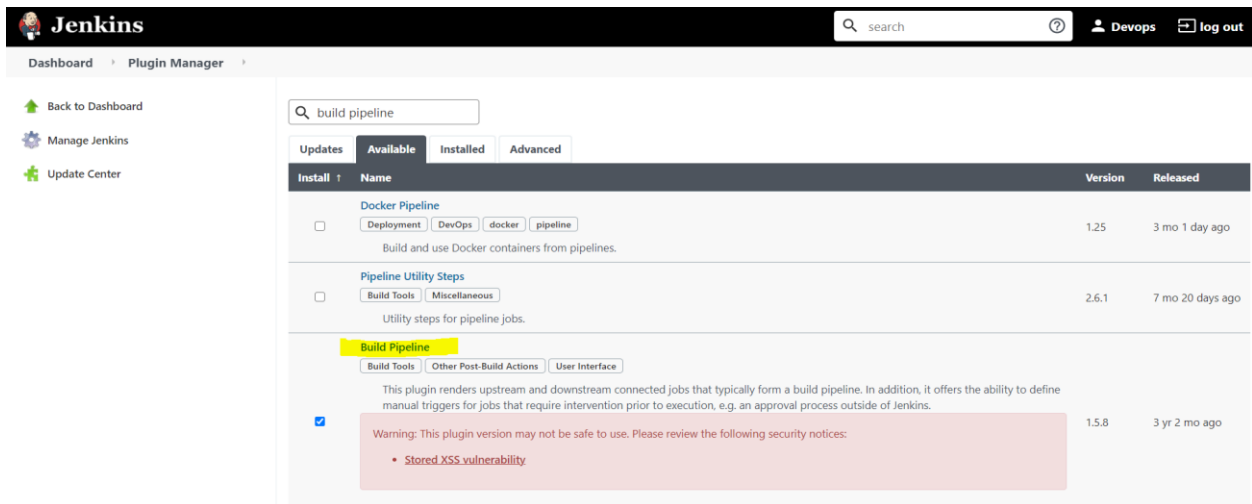
Step 21: Go to the **Demo** job, click on **Configure**. Add **Post-Build Actions**. Then go to **Build Other Projects**.



Click on Save.

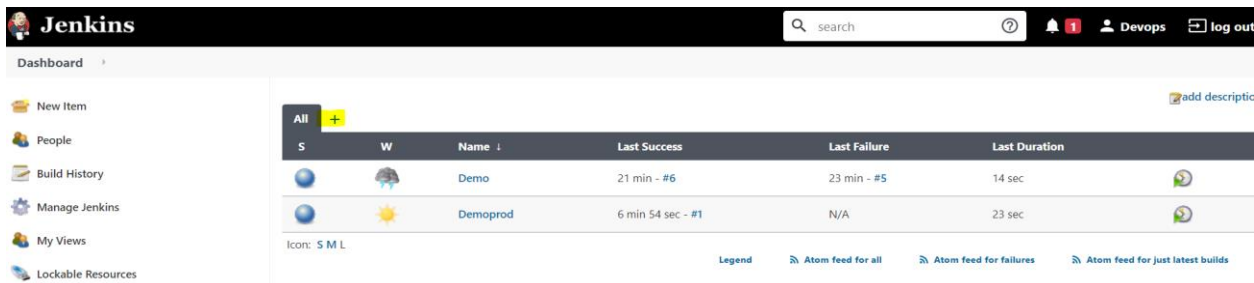
Hands-on: Create a CI CD pipeline triggered by Git Webhook.

Step 22: Go to the manage Jenkins and then Manage Plugins, click on available, search for Build Pipeline. Click on install without restart.







The screenshot shows the Jenkins Plugin Manager interface. The search bar contains "build pipeline". The "Available" tab is selected. The "Build Pipeline" plugin is highlighted with a yellow box. It has a version of 1.5.8 and was released 3 years and 2 months ago. A warning message is displayed: "Warning: This plugin version may not be safe to use. Please review the following security notices: • Stored XSS vulnerability".

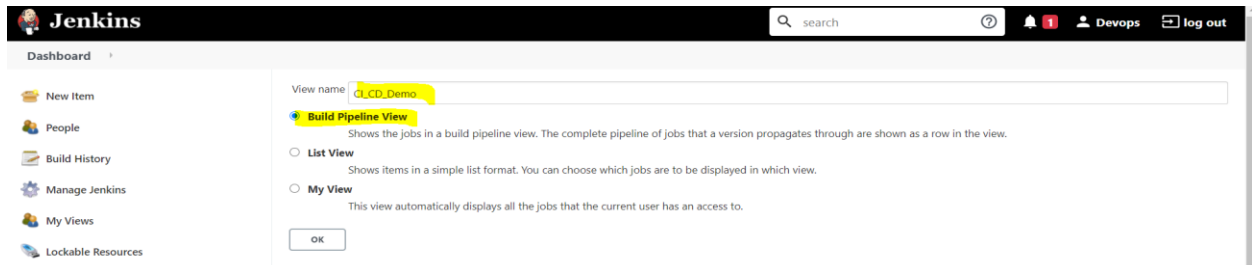
Step 23: Go to the jenkins dashboard. Click on the +.



The screenshot shows the Jenkins Dashboard. The "All" button with a "+" icon is highlighted. The dashboard displays a table of jobs with columns: S, W, Name, Last Success, Last Failure, and Last Duration. Two jobs are listed: "Demo" and "Demoprod".

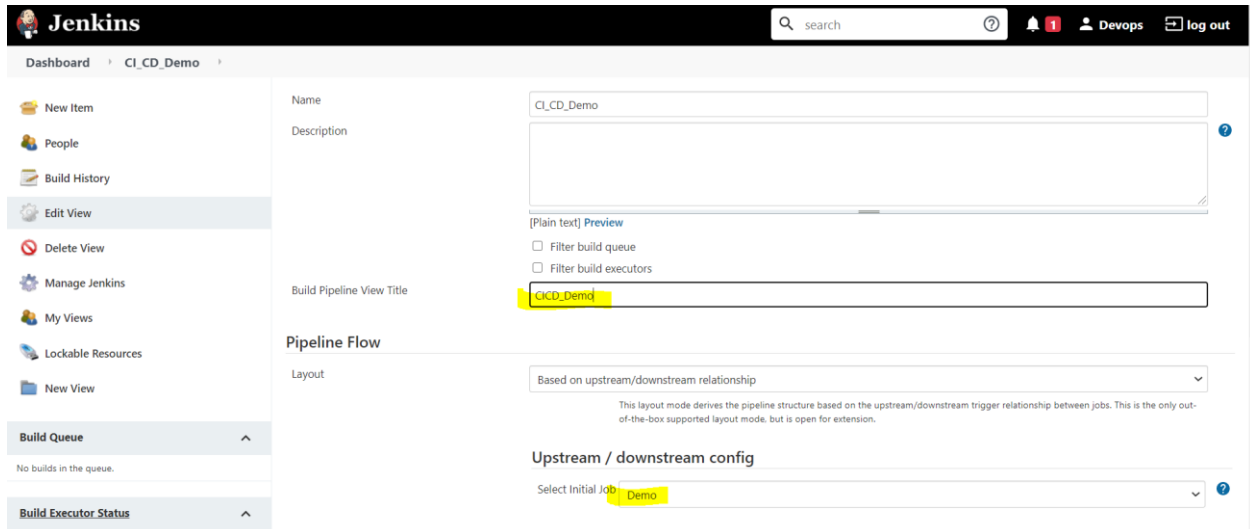
S	W	Name	Last Success	Last Failure	Last Duration
		Demo	21 min - #6	23 min - #5	14 sec
		Demoprod	6 min 54 sec - #1	N/A	23 sec

Step 24: Enter view name and click ok.



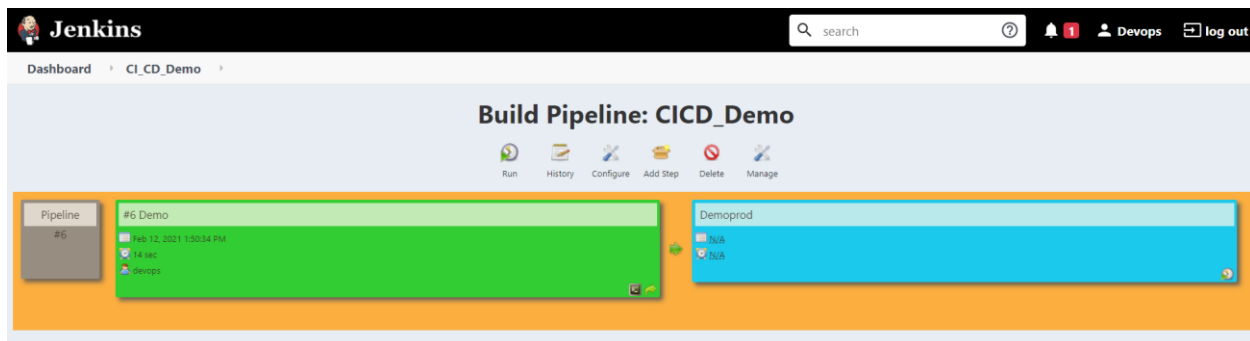
The screenshot shows the Jenkins View Configuration dialog. The "View name" field contains "CI_CD_Demo". The "Build Pipeline View" option is selected. The "List View" and "My View" options are also visible. The "OK" button is highlighted.

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

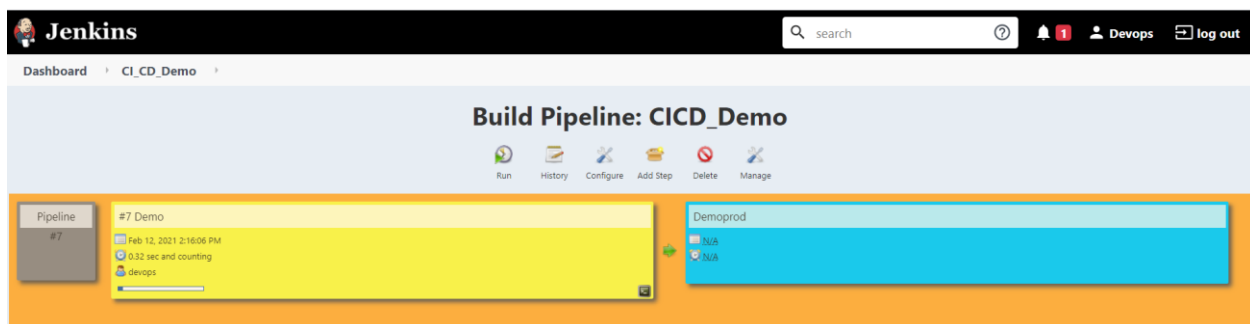


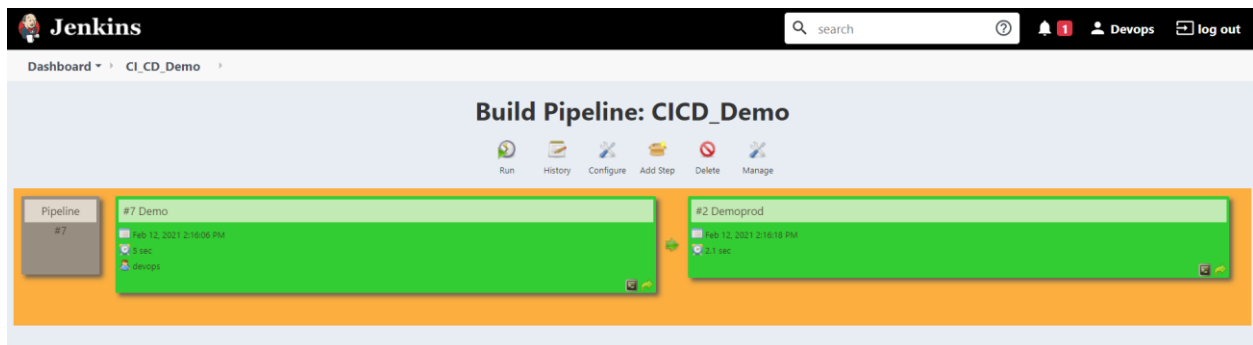
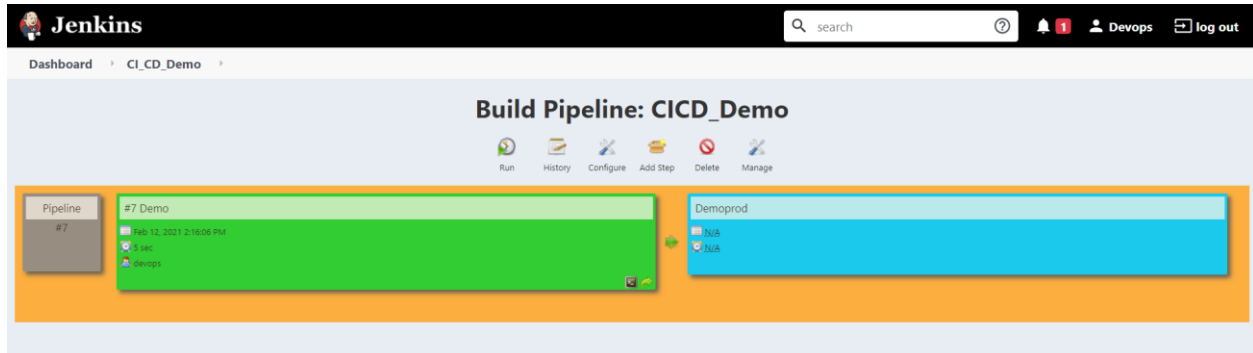
The screenshot shows the Jenkins configuration page for a new Build Pipeline named 'CI_CD_Demo'. The 'Name' field is set to 'CI_CD_Demo'. The 'Build Pipeline View Title' field is also set to 'CI_CD_Demo'. Under 'Pipeline Flow', the 'Layout' is set to 'Based on upstream/downstream relationship'. In the 'Upstream / downstream config' section, 'Select Initial Job' is set to 'Demo'.

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



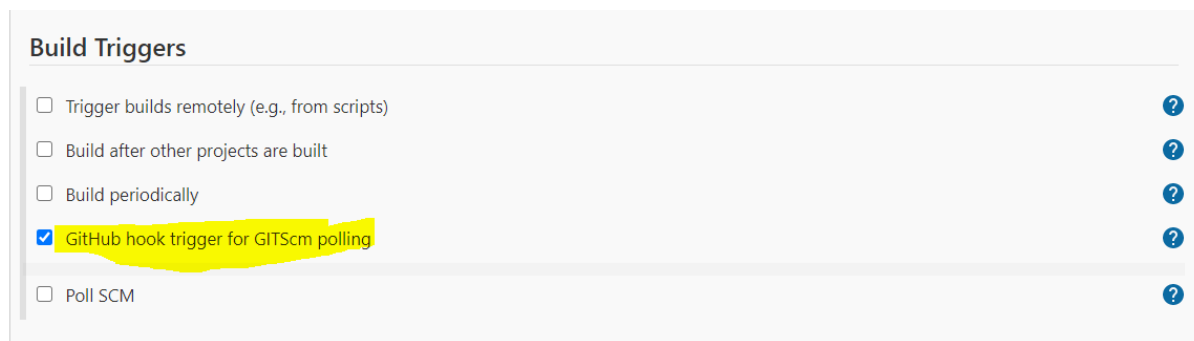
Step 26: Click on Run. Then Refresh the Page once.





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

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



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/

API Token

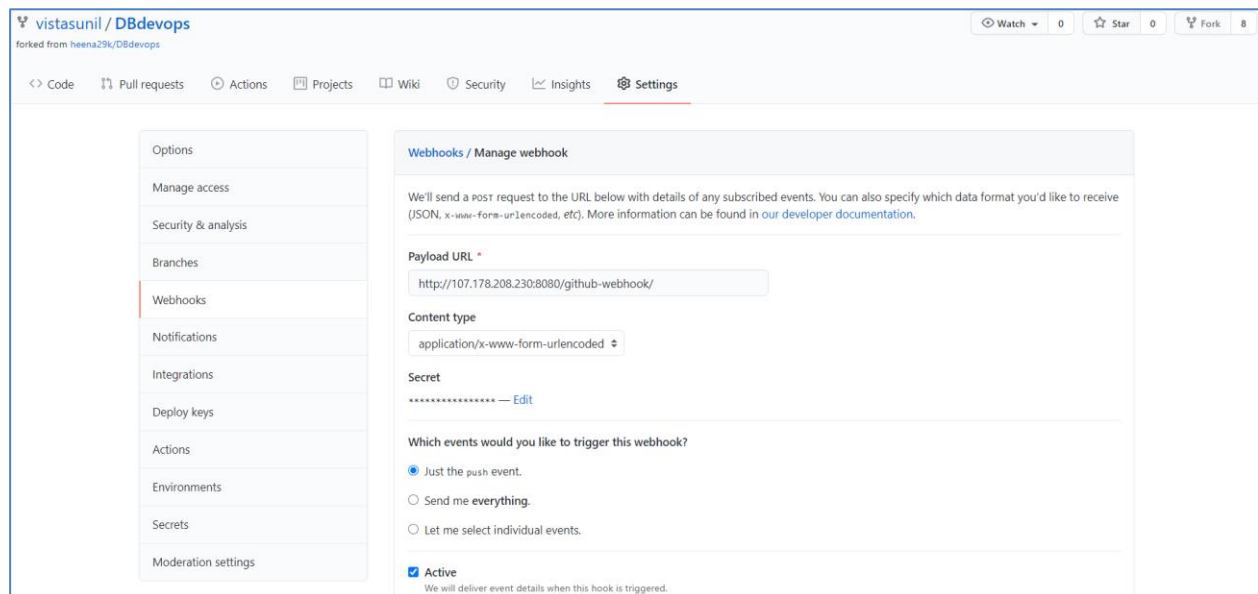
Current token(s)

Token created on 2021-02-12T14:41:55.01 **110a4228c49a5a29b513352**

⚠ Copy this token now, because it cannot be recovered in the future.

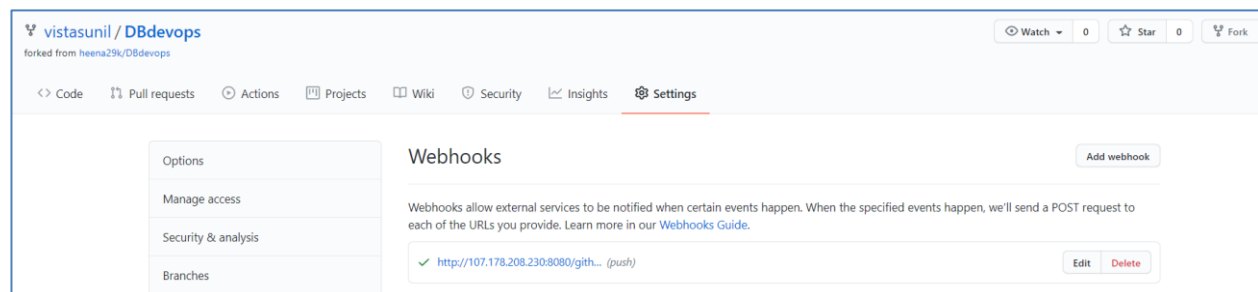
Add new Token

Credentials



The screenshot shows the GitHub repository settings for `vistasunil / DBdevops`. The left sidebar contains a menu with options like Manage access, Security & analysis, Branches, Webhooks, Notifications, Integrations, Deploy keys, Actions, Environments, Secrets, and Moderation settings. The main content area is titled "Webhooks / Manage webhook". It explains that GitHub will send a POST request to the specified URL with details of subscribed events. The "Payload URL" is set to `http://107.178.208.230:8080/github-webhook/`. The "Content type" is `application/x-www-form-urlencoded`. The "Secret" field is masked with asterisks and has an "Edit" link. Under "Which events would you like to trigger this webhook?", the "Just the push event" option is selected. The "Active" checkbox is checked, with a note: "We will deliver event details when this hook is triggered."

Click on Add webhook. You should see this.



The screenshot shows the same GitHub repository settings page, but now the "Webhooks" section displays a list of configured webhooks. A single webhook is listed with the URL `http://107.178.208.230:8080/github... (push)`. To the right of the URL are "Edit" and "Delete" buttons. An "Add webhook" button is visible in the top right corner of the Webhooks section.

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



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

```
ubuntu@instance-1:~$ git clone https://github.com/vistasunil/DBdevops.git
Cloning into 'DBdevops'...
remote: Enumerating objects: 87, done.
remote: Total 87 (delta 0), reused 0 (delta 0), pack-reused 87
Receiving objects: 100% (87/87), 11.46 MiB | 26.79 MiB/s, done.
Resolving deltas: 100% (15/15), done.
```

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

```
$ vim index.html
```

```
ubuntu@instance-1:~$ cd DBdevops/
ubuntu@instance-1:~/DBdevops$ cd devopsIQ/
ubuntu@instance-1:~/DBdevops/devopsIQ$ vim index.html
```

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

```
<html>
<title>Jenkins Updated Website</title>
<body background="images/2.jpeg">
<h1>This is webhook triggered website!!</h1>
</body>
</html>
```

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

```
$ git add
```

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

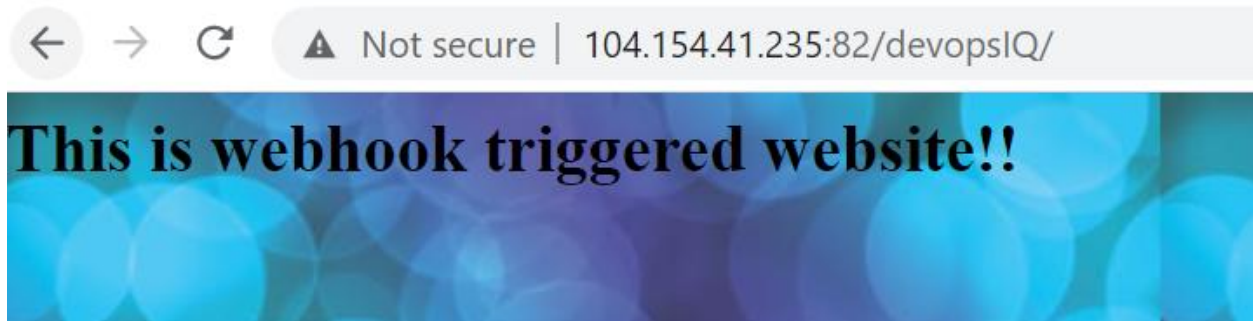
```
ubuntu@instance-1:~/DBdevops$ git add .
ubuntu@instance-1:~/DBdevops$ git commit -m "Webhook commit"
[master bc55baa] Webhook commit
1 file changed, 2 insertions(+), 1 deletion(-)
ubuntu@instance-1:~/DBdevops$
```

Step 33: Perform git push.

```
$ git push origin master
```

```
ubuntu@instance-1:~/DBdevops$ git push origin master
```

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.