

# Setting up jenkins master-slave architecture for testing and production servers

## 1. Setting up master and slave servers in aws.

=>setup 3 nodes of type t2.micro of ubuntu 20.04 os with security group instructions to permit all traffic and keep all other details as default.

=>NOTE : we will be installing Jenkins only on master nodes and monitor slave nodes using master nodes.

=>On master node execute these instructions

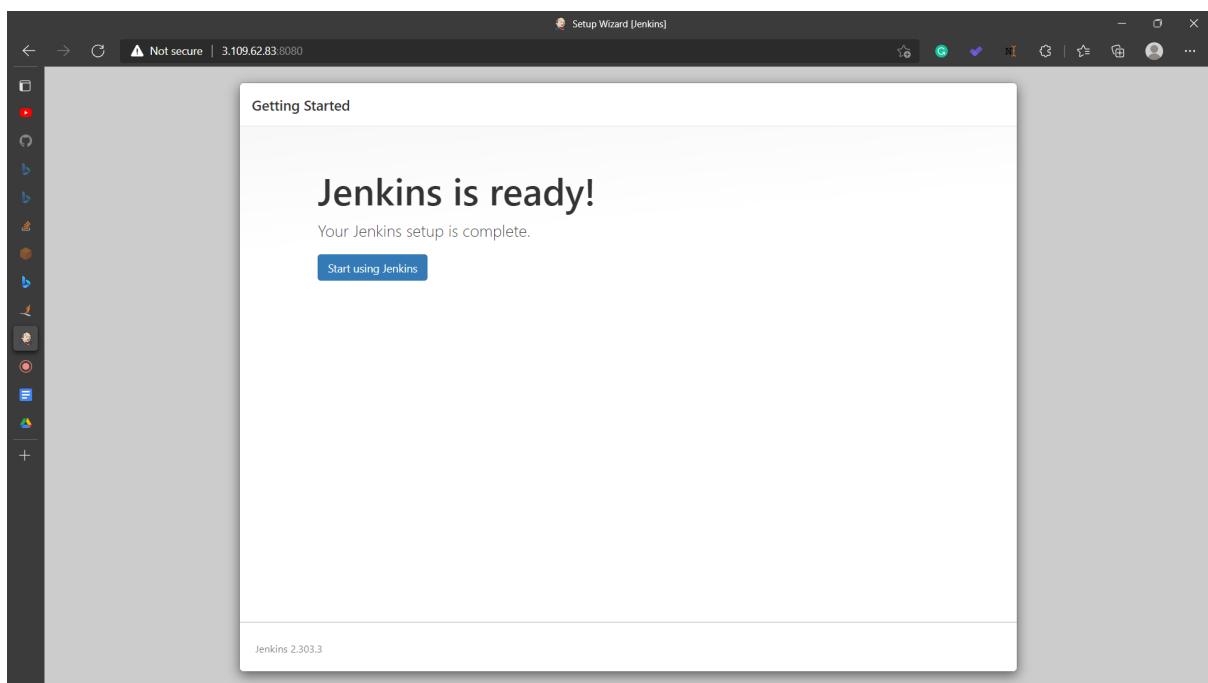
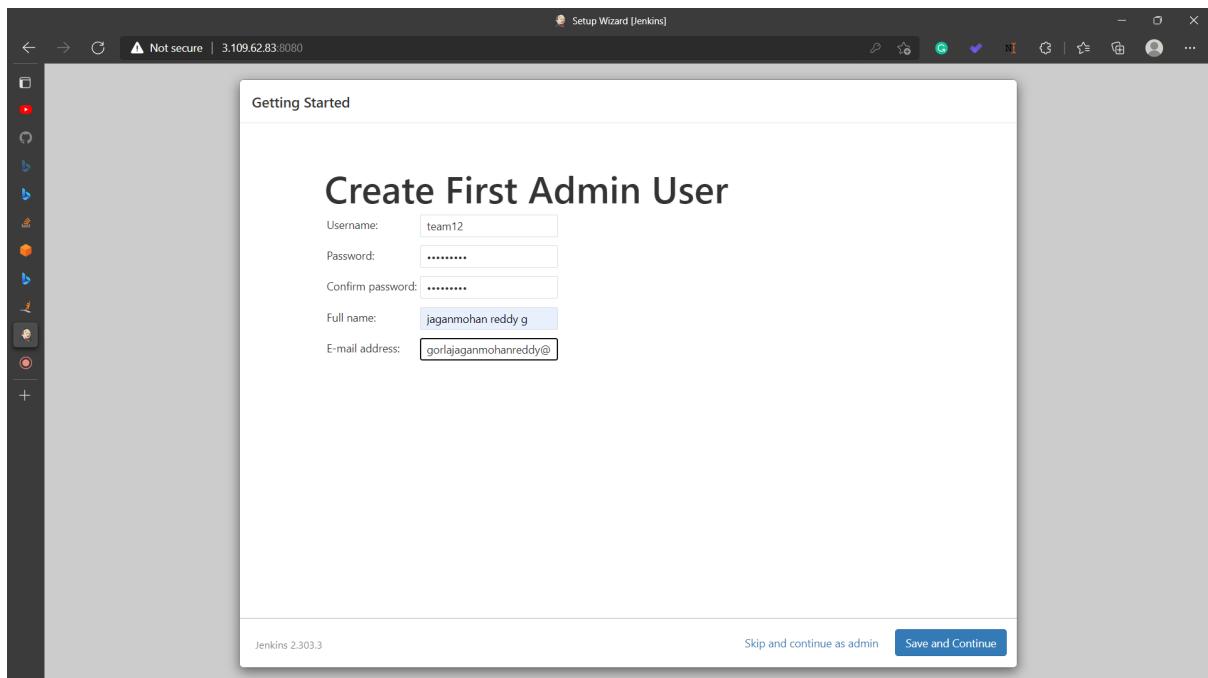
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Jenkins-Master	i-04d598d432c2b9347	Stopped	t2.micro	-	No alarms	ap-south-1a	-
Slave-1	i-001a9d68c4a14d087	Stopped	t2.micro	-	No alarms	ap-south-1a	-
Slave-2	i-082dcfe0683b1b9dd	Stopped	t2.micro	-	No alarms	ap-south-1a	-
anu-worker-n...	i-035565c3c25d9504a	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	ec2-55-1...
Jenkins-EC2	i-03dc5541407946108	Stopped	t2.medium	-	No alarms	ap-south-1b	-
anu-worker-n...	i-004203775fc012929	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1b	ec2-13-1...

## COMMANDS

```
sudo apt install openjdk-8-jdk
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins
```

=>On completing the above instructions, access this server on the browser using a public ipv4 address with port number 8080 i.e ipaddress:8080.

=>after opening the above jenkins page login to jenkins to proceed further



## 2.configuring jenkins

=>Goto manage jenkins -> configure global settings -> Agents -> random -> click on save.  
=>Goto manage jenkins -> manage nodes -> new node.

The screenshot shows the Jenkins dashboard at the URL <http://3.109.62.83:8080>. The left sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. Under 'Build Queue', it says 'No builds in the queue.' Under 'Build Executor Status', there are two entries: '1 Idle' and '2 Idle'. The main content area features a 'Welcome to Jenkins!' message with a sub-section 'Start building your software project' containing a 'Create a job' button. Below that is a 'Set up a distributed build' section with 'Set up an agent' and 'Configure a cloud' buttons, along with a link to 'Learn more about distributed builds'. At the bottom right, there are links for 'REST API' and 'Jenkins 2.303.3'.

=>Create 2 slave nodes named slave-1 and slave-2 respectively with permanent agent option enabled and with the launch method as configure when master is triggered.Also add custom workdir path to be /home/ubuntu/jenkins and click on save.

The screenshot shows the 'Configure Global Security' page at the URL <http://3.109.62.83:8080/configureSecurity/>. It includes sections for 'Logged-in users can do anything' (radio button selected), 'Matrix-based security', and 'Project-based Matrix Authorization Strategy'. The 'Markup Formatter' section shows 'Plain text' selected. The 'Agents' section has 'TCP port for inbound agents' set to 'Random'. The 'CSRF Protection' section includes 'Crumb Issuer' and 'Enable proxy compatibility' options. The 'Hidden security warnings' section has 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins 'Nodes' page. On the left sidebar, there are links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', 'Configure Clouds', and 'Node Monitoring'. Below these are sections for 'Build Queue' (empty) and 'Build Executor Status'. The main table lists three nodes:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.38 GB	0 B	5.38 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

At the bottom right of the table is a 'Refresh status' button. At the very bottom of the page are links for 'REST API' and 'Jenkins 2.30.3'.

=>click on respective agents and download their agent.jar executable file.

=>Now these agent.jar files are to be sent to the respective slave nodes that were created from above points.For ubuntu os run the following command.

```
scp -i ./< pemfileofmasternode > ./agent.jar <username>@< ipv4 address of slave node >:< destination folder of slave node >
```

### 3. Connect to slave nodes from master nodes.

The screenshot shows the Jenkins 'Slave-1' page. On the left sidebar, there are links for 'Back to List', 'Status', 'Delete Agent', 'Configure', 'Build History', 'Load Statistics', and 'Log'. Below these is a section for 'Build Executor Status' (empty). The main content area is titled 'Agent Slave-1' and contains instructions for connecting the agent:

Connect agent to Jenkins one of these ways:

- [Launch](#) Launch agent from browser
- Run from agent command line:

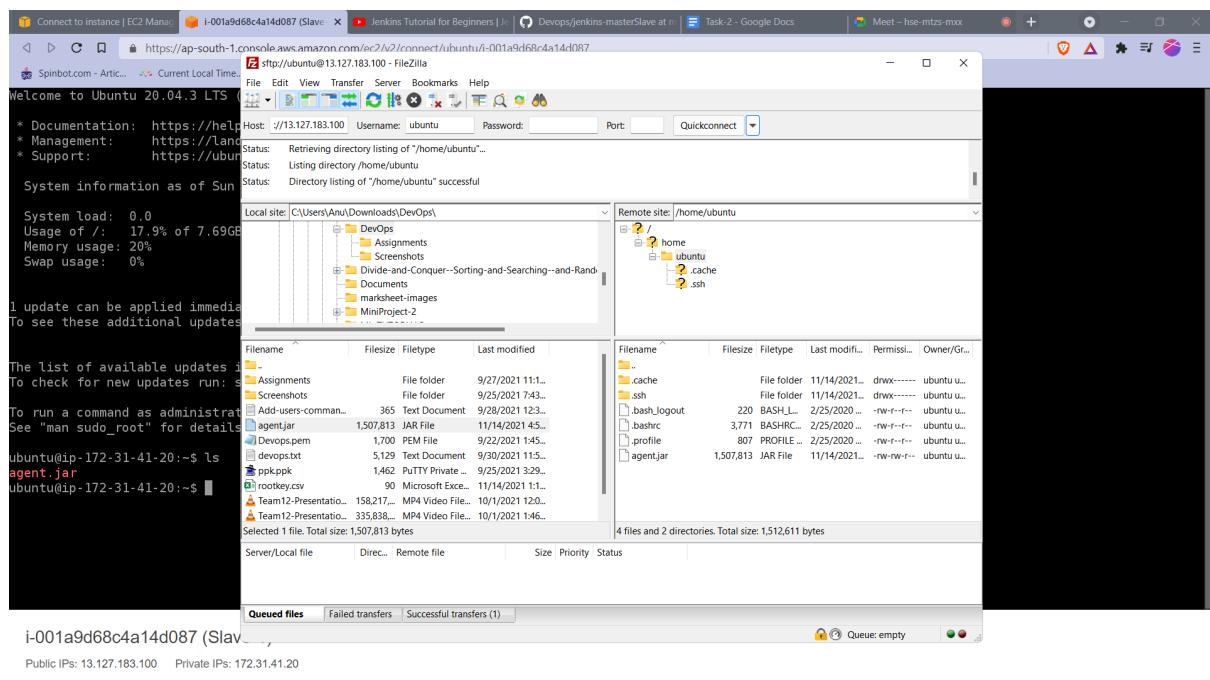
```
java -jar agent.jar -jnlpUrl http://3.109.62.83:8080/computer/Slave-1/jenkins-agent.jnlp -secret 593a1c21a72d16eaae07d58c3f42df74e94de07330437cf6666c3fdd59950b8a -workDir "/home/ubuntu/jenkins"
```

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

```
echo 593a1c21a72d16eaae07d58c3f42df74e94de07330437cf6666c3fdd59950b8a > secret-file
java -jar agent.jar -jnlpUrl http://3.109.62.83:8080/computer/Slave-1/jenkins-agent.jnlp -secret @secret-file -workDir "/home/ubuntu/jenkins"
```

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

At the bottom right of the page are links for 'REST API' and 'Jenkins 2.30.3'.



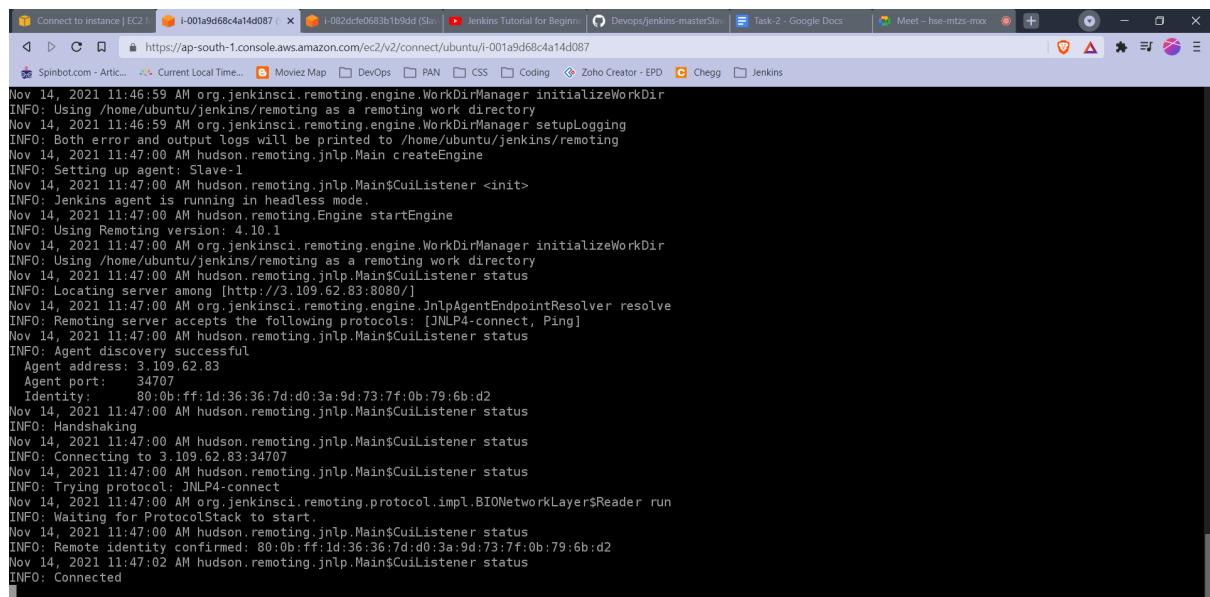
=>Goto manage jenkins -> manage nodes -> click on respective slave and copy paste the code under Run from agent command line.

=>Now copy paste the code in slave nodes.

### COMMANDS

**sudo apt-get update**

**sudo apt install openjdk-8-jdk**



i-001a9d68c4a14d087 (Slave-1)

Public IPs: 13.127.183.100 Private IPs: 172.31.41.20

The screenshot shows the Jenkins interface for the 'Slave-1' node. On the left, a sidebar lists various management options like Back to List, Status, Delete Agent, Configure, Build History, Load Statistics, Script Console, Log, System Information, and Disconnect. Below this is a 'Build Executor Status' section showing one idle build. The main content area is titled 'Agent Slave-1' and displays the message 'Agent is connected.' A blue button at the top right says 'Mark this node temporarily offline'. Underneath, a section titled 'Projects tied to Slave-1' shows 'None'.

The screenshot shows the Jenkins 'Nodes' page. It lists three nodes: 'master' (Linux (amd64), In sync, 5.38 GB free disk space, 0.0 ms response time), 'Slave-1' (Linux (amd64), In sync, 5.74 GB free disk space, 48ms response time), and 'Slave-2' (Linux (amd64), In sync, 5.74 GB free disk space, 44ms response time). Below the table, a 'Build Queue' section indicates 'No builds in the queue.' The 'Build Executor Status' section shows build counts for each node: master (3 sec), Slave-1 (3 sec), and Slave-2 (2.9 sec). A 'Refresh status' button is located at the bottom right of the table.

=>Now verify that the master and slave nodes are connected

## 4. Configuring Jenkins to build a project on slave-1(Test server) if successful build on slave-2(Production server).

=>NOTE : make sure to duplicate sessions in the slave node so that connection doesn't terminate.

=>Install Docker on both the slaves.

### commands

```
sudo apt-get install docker.io
```

=>Head Back to jenkins. create new job -> make it new freestyle project -> name it slave-1. Now under configure in the general section click on github project and copy paste the url of github project. Also enable the Restrict where this project can be run and type in slave-1 under label expression section. Go to source code management and enable git.

=>Under the build section click Add build step -> Execute shell. Now run the below commands.

### COMMANDS

```
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
```

=>NOTE : Make sure to run a custom container on the slave node before executing the above commands. The above steps are to be executed for both test and production server.

=>Now click on build.

=>To check the above build step head over to :/

**Test Config [Jenkins]**

Not secure | http://3.109.62.83:8080/job/Test/configure

Spinbot.com - Artic... Current Local Time... Moviez Map DevOps PAN CSS Coding Zoho Creator - EPD Chegg Jenkins

Private

Dashboard > Test >

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

Delete workspace before build starts  
 Use secret text(s) or file(s)  
 Abort the build if it's stuck  
 Add timestamps to the Console Output  
 Inspect build log for published Gradle build scans  
 With Ant

**Build**

**Execute shell**

Command

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

See the list of available environment variables

**Add build step ▾**

**Post-build Actions**

**Add post-build action ▾**

**Save** **Apply**

**Test #3 [Jenkins]**

Not secure | http://3.109.62.83:8080/job/Test/3/

Spinbot.com - Artic... Current Local Time... Moviez Map DevOps PAN CSS Coding Zoho Creator - EPD Chegg Jenkins

Jenkins

Dashboard > Test > #3

**Build #3 (Nov 14, 2021 12:38:28 PM)**

**Keep this build forever**

**add description** Started 22 sec ago took 16 sec on Slave-1

**Status**

- Changes
- Console Output
- Edit Build Information
- Delete build #3\*
- Git Build Data
- Previous Build

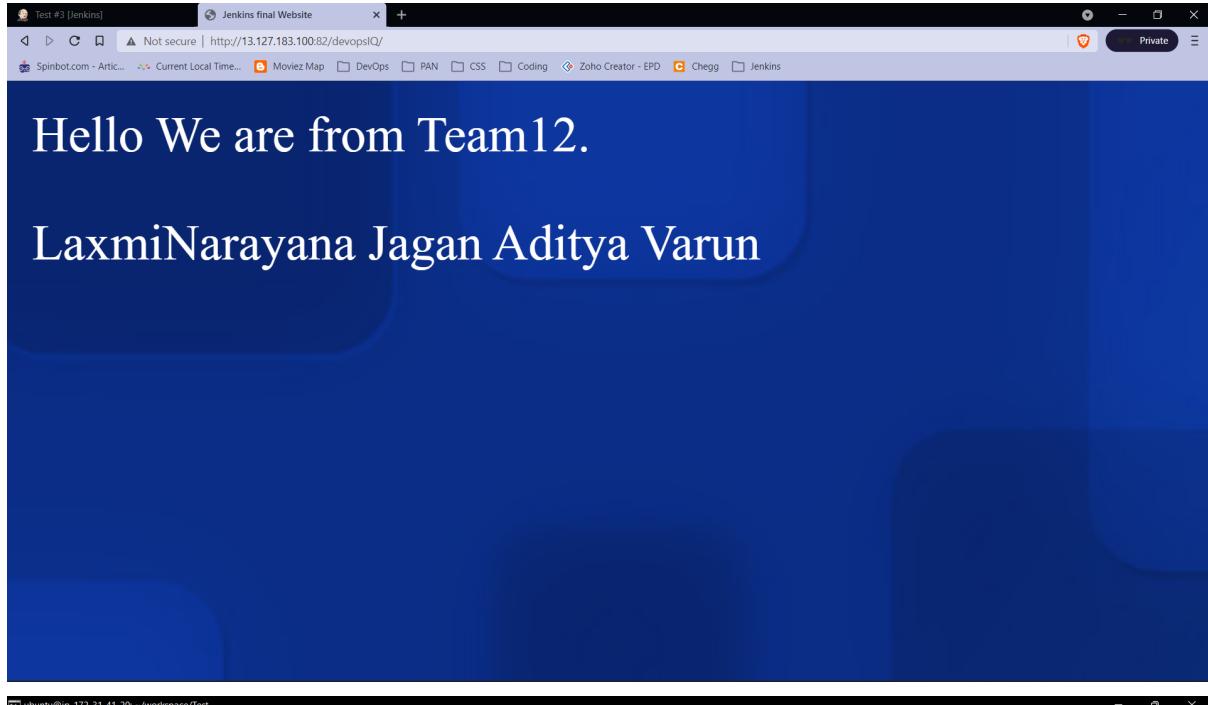
No changes.

Started by user **jaganmohan reddy.g**

**git** Revision: 6be43b10a8dafde9e2f0da9f6746f82e1e12968c  
Repository: <https://github.com/LaxmiNarayanaK/web-app-test.git>

- refs/remotes/origin/master

REST API Jenkins 2.303.3



```
ubuntu@ip-172-31-41-20:~/workspace/Test
-bash: cd: test: No such file or directory
ubuntu@ip-172-31-41-20:~/workspace$ cd Test
ubuntu@ip-172-31-41-20:~/workspace/Test$ ls
Dockerfile README.md azure-pipelines.yml devopsIQ docker-compose
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker run -it -p 82:80 -d test
Unable to find image 'test:latest' locally
docker: Error response from daemon: pull access denied for test, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See docker run --help.
ubuntu@ip-172-31-41-20:~/workspace/Test$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: anu33317
Password:
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.24/auth: dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: anu33317
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker run -it -p 82:80 -d test
Unable to find image 'test:latest' locally
docker: Error response from daemon: pull access denied for test, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See docker run --help.
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker run -it
'docker run' requires at least 1 argument.
See 'docker run --help'.

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker run -it -d ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
7blaab02e44d: Pull complete
Digest: sha256:628ffr6856a00254b438eb7e0f3b11d4da9675088f4781a50ae288f3322
Status: Downloaded newer image for ubuntu:latest
391a997b4467c2eaa1a3c0937faf78f267703120a6e904056ce31746938fac
ubuntu@ip-172-31-41-20:~/workspace/Test$ ls
Dockerfile README.md azure-pipelines.yml devopsIQ docker-compose
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
5e03fd170486        test                "/bin/sh -c 'apachec..."   2 minutes ago      Up 2 minutes       0.0.0.0:82->80/tcp, ::82->80/tcp   distracted_heyrovsky
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker rm -f 5e03fd170486
5e03fd170486
ubuntu@ip-172-31-41-20:~/workspace/Test$ sudo docker run -it -d ubuntu
ac011524fb360069f044a12ae88300fbffcc07add2c80a7245c2273e9358ecf
ubuntu@ip-172-31-41-20:~/workspace/Test$
```

Screenshot of a web browser showing Jenkins Slave-2 #2 build details and a terminal window displaying a Jenkins slave log.

**Jenkins Slave-2 #2 [Jenkins]** | **Jenkins final Website**

Not secure | http://3.109.62.83:8080/job/Slave-2/2/

Spinbot.com - Article... Current Local Time... Moviez Map DevOps PAN CSS Coding Zoho Creator - EPD Chegg Jenkins

**Jenkins** search Private jaganmohan reddy g log out

Dashboard > Slave-2 > #2

**Build #2 (Nov 14, 2021 12:57:48 PM)**

Keep this build forever

No changes.

Started by user **jaganmohan reddy g**

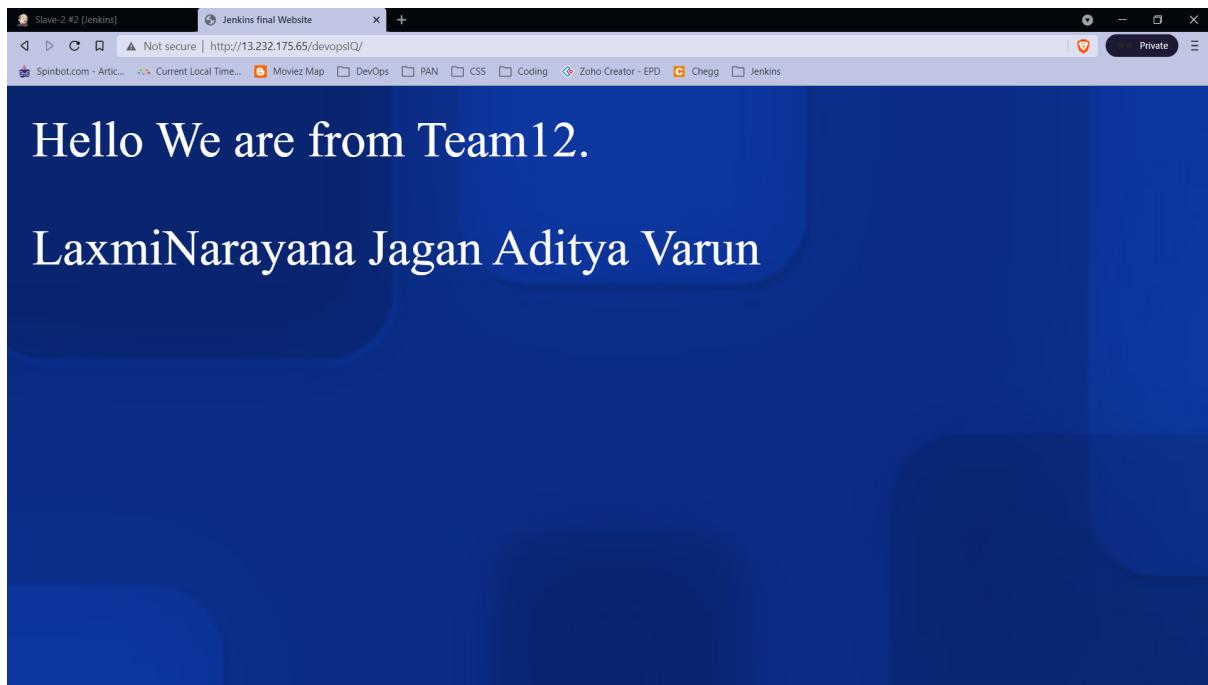
**git** Revision: 6be43b10a8dafde9e2f0da9f674682e1e12968c  
Repository: <https://github.com/LaxmiNarayanaK/web-app-test.git>  
refs/remotes/origin/master

REST API Jenkins 2.303.3

```
ubuntu@ip-172-31-45-216:~jenkins
* Support: https://ubuntu.com/advantage
System information as of Sun Nov 14 12:50:41 UTC 2021
System load: 0.1 Processes: 108
Usage of /: 30.5% of 7.69GB Users logged in: 1
Memory usage: 38% IPv4 address for docker0: 172.17.0.1
Swap usage: 0% IPv4 address for eth0: 172.31.45.216
* Ubuntu Pro delivers the most comprehensive open source security and
compliance features.
https://ubuntu.com/pro

26 updates can be applied immediately.
15 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Sun Nov 14 12:19:26 2021 from 13.233.177.0
ubuntu@ip-172-31-45-216:~$ ls
agent.jar jenkins
ubuntu@ip-172-31-45-216:~$ cd jenkins
ubuntu@ip-172-31-45-216:~/jenkins$ ls
remoting
ubuntu@ip-172-31-45-216:~/jenkins$ cd ..
ubuntu@ip-172-31-45-216:~$ ls
agent.jar jenkins
ubuntu@ip-172-31-45-216:~$ sudo docker run -it -d ubuntu
ubuntu@ip-172-31-45-216:~$ sudo docker pull ubuntu:latest
ubuntu@ip-172-31-45-216:~$ ls
agent.jar jenkins
ubuntu@ip-172-31-45-216:~$ cd jenkins
ubuntu@ip-172-31-45-216:~/jenkins$ ls
remoting workspace
ubuntu@ip-172-31-45-216:~/jenkins$ cd ..
ubuntu@ip-172-31-45-216:~$ cd jenkins
ubuntu@ip-172-31-45-216:~/jenkins$ sudo docker run -it -d ubuntu
ae9dcad2283b33a6dd530614ff6fc9d3cc730ebbe45dbd69ccdc32406a9e5feff1
ubuntu@ip-172-31-45-216:~/jenkins$
```



=>Now we need to configure the servers in such a way that after testing production is built.To enable this Goto the test(freestyle project of slave-1) and under the post build actions click on build other projects and enter production(freestyle project of slave-2).

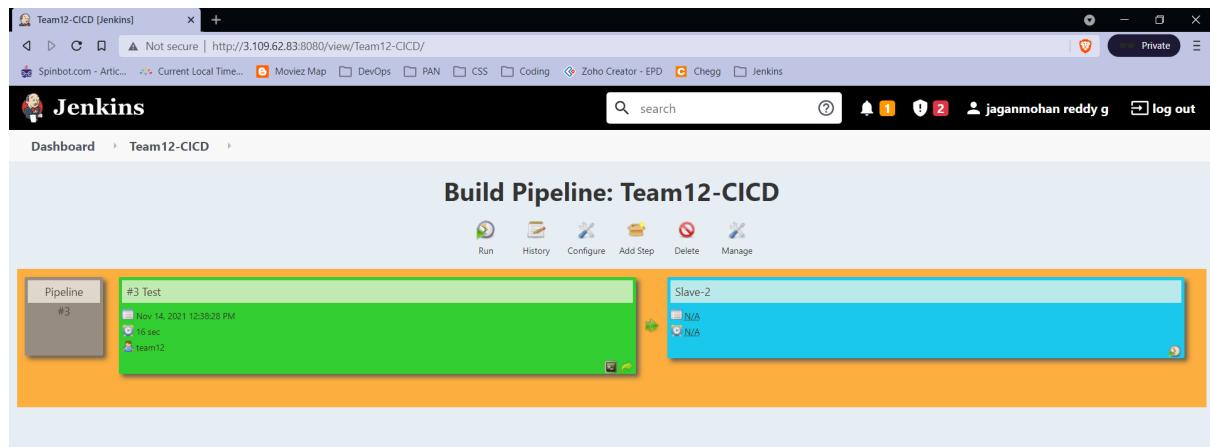
## 5. Build in the ci-cd pipeline for test and production servers.

=>Headover to manage jenkins -> manage plugins -> available.Now search for build pipeline,install the plugin.

=>In the jenkins home page click on the + sign near the All.

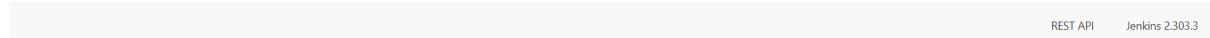
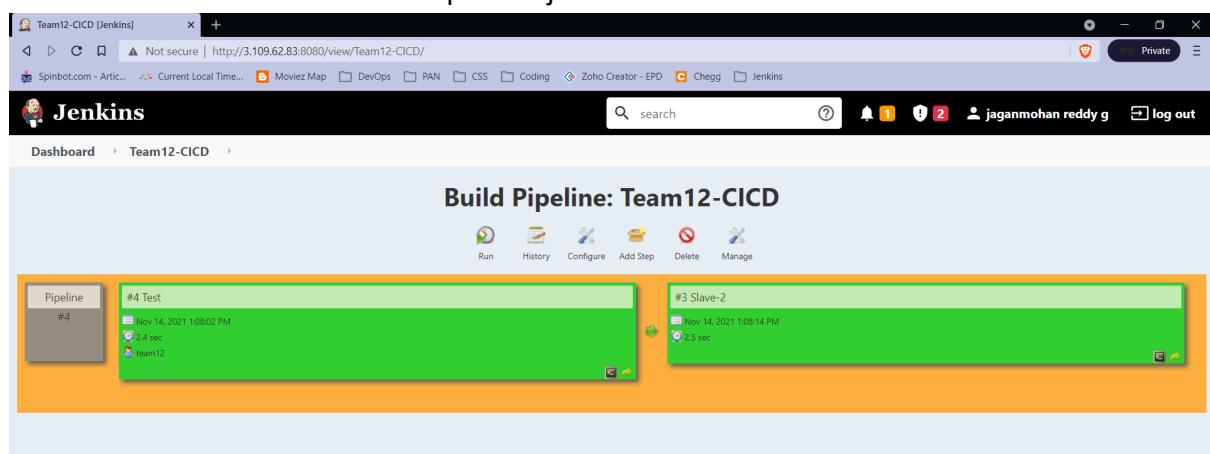
=>Now click the Build pipeline view, name the view as CiCd.Also under Build pipeline view title give CiCd and with other options as default and click save.

A screenshot of the Jenkins 'Edit View' configuration page. The left sidebar shows navigation links like 'Dashboard', 'New Item', 'People', 'Build History', 'Edit View' (which is selected), 'Delete View', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main content area has a form for 'Team12-CICD'. It includes fields for 'Name' (set to 'Team12-CICD'), 'Description' (empty), 'Plain text' preview (empty), 'Filter build queue' (unchecked), 'Filter build executors' (unchecked), 'Build Pipeline View Title' (set to 'Team12-CICD'), and a 'Pipeline Flow' section. In the 'Pipeline Flow' section, 'Layout' is set to 'Based on upstream/downstream relationship'. A note states: 'This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.' Below this is the 'Upstream / downstream config' section with 'Select Initial Job' set to 'Test'. At the bottom are 'OK' and 'Apply' buttons.

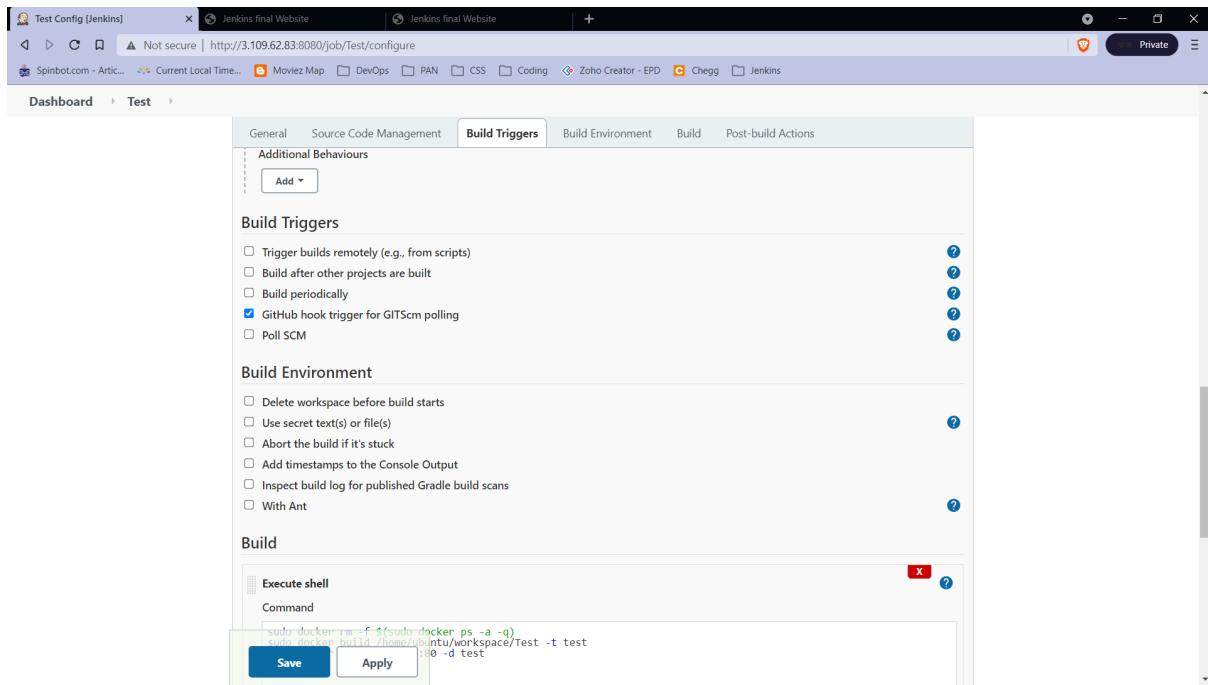


=>Under the Configure section of the CiCd pipeline,make Pipeline flow -> select initial job -> test.

=>Now click on Run to run the respective jobs.



## 6. Using Github-Webhook to reflect changes made to github projects in the test and production servers.



=>Goto jenkins dashboard -> test -> configure -> build triggers. And enable the github hook trigger for GIT-Scm polling option and click save.

=>Headover to your github webapp project and goto settings -> Webhooks -> Add webhook. Now copy paste the ip address of your jenkins server(master node) under the payload url section and click add webhook.

=>NOTE : Verify that you get a tick mark under the webhook section for the url that you copy pasted.

=>Now lets trigger a build by committing port changes to the github webapp project and see the changes made to the test and production server.

### COMMANDS

```
git clone <github url>
cd <projectfolder>
ls
nano index.html // make changes to this file
git add .
git commit -m "message"
git push origin master
```

The screenshot shows the GitHub settings interface for a repository named 'LaxmiNarayanaK/web-app-test'. The left sidebar has 'Webhooks' selected. The main area is titled 'Webhooks / Add webhook' and contains the following fields:

- Payload URL \***: `http://3.109.62.83:8080/github-webhook/`
- Content type**: `application/x-www-form-urlencoded`
- Secret**: (empty input field)
- Which events would you like to trigger this webhook?**
  - Just the `push` event.
  - Send me **everything**.
  - Let me select individual events.
- Active**:  (with explanatory text: "We will deliver event details when this hook is triggered.")

At the bottom right is a green **Add webhook** button.

The screenshot shows the GitHub settings interface for the same repository. The left sidebar has 'Webhooks' selected. The main area is titled 'Webhooks' and lists the previously added webhook:

<code>http://3.109.62.83:8080/github-w... (push)</code>	<a href="#">Edit</a>	<a href="#">Delete</a>
---------------------------------------------------------	----------------------	------------------------

