

# Jenkins Notes

## 1) Make Jenkins Public DNS [Step 2 in Ajay Sir Notes]

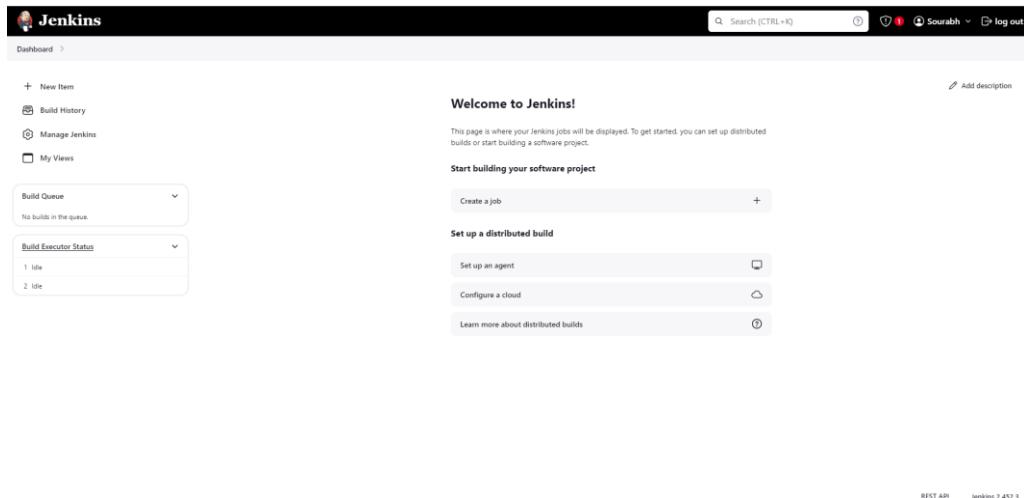
```
echo "public ip_address" "jenkins_local" >> /etc/hosts
```

```
root@ip-172-31-94-84:/home/ubuntu#  
root@ip-172-31-94-84:/home/ubuntu# echo "34.205.73.254" "jenkins_local" >> /etc/hosts
```

Cat /etc/hosts

```
root@ip-172-31-94-84:/home/ubuntu# cat /etc/hosts  
127.0.0.1 localhost  
  
# The following lines are desirable for IPv6 capable hosts  
::1 ip6-localhost ip6-loopback  
fe00::0 ip6-localnet  
ff00::0 ip6-mcastprefix  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters  
ff02::3 ip6-allhosts  
34.205.73.254 jenkins_local
```

## 2) Dashboard [Step 3 in Ajay Sir Notes]



## 3) Create first Jenkins job [Step 4 in Ajay Sir Notes]

Click on New Item

There are multiple types of jobs and their details are mentioned along with the jobs

Enter the item name and select Freestyle project and hit OK

Dashboard > test > Configuration

## Configure

### General

Enabled

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 ?

Advanced ▾

#### Source Code Management

None

Git ?

[Save](#)

[Apply](#)

Dashboard > test > Configuration

## Configure

### 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 ?

### Build Environment

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

### Build Steps

Add build step ▾

### Post-build Actions

Add post-build action ▾

Add build Steps and select Execute Shell, Write the commands and click save

The screenshot shows the Jenkins configuration interface for a job named "test". The "Build Triggers" section is visible, with several triggers listed. The "Build Environment" section follows, containing options like "Delete workspace before build starts" and "Use secret text(s) or file(s)". A dropdown menu is open over the "Execute shell" option, showing a list of available build steps: "Execute Windows batch command", "Execute shell" (which is highlighted with a yellow box), "Invoke Ant", "Invoke Gradle script", "Invoke top-level Maven targets", "Run with timeout", and "Set build status to "pending" on GitHub commit". Below this list is a button labeled "Add build step ^". The "Post-build Actions" section is also present at the bottom.

Dashboard > test > Configuration

## Configure

### 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 ?

### Build Environment

- Delete workspace before build starts
- Use secret text(s) or file(s) ?  
Filter

- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit

Add build step ^

### Post-build Actions

#### Build Steps

Execute shell ?

Command

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

```
sleep 10
whoami
pwd
touch abc.txt
```

Advanced ▾

Add build step ▾

#### Post-build Actions

**Save** **Apply**

Click on build now to the Job

The screenshot shows a project interface for 'test'. At the top, there are navigation links: 'Status', 'Changes', 'Workspace', 'Build Now' (which is highlighted in yellow), 'Configure', 'Delete Project', and 'Rename'. Below these is a section titled 'Permalinks'. A large button labeled 'Build History' is present, with a sub-section showing 'No builds' and two feed links: 'Atom feed for all' and 'Atom feed for failures'.

Build history will show you the Jobs that ran and are running

This screenshot shows the 'Build History' section with one job entry. The job is labeled '#1' and was run on 'Aug 2, 2024, 8:24 AM'. The status is indicated by a blue progress bar. Below the list are feed links: 'Atom feed for all' and 'Atom feed for failures'.

Below screenshot shows job completed successfully

This screenshot shows the 'Build History' section with one job entry. The job is labeled '#1' and was run on 'Aug 2, 2024, 8:24 AM'. A green checkmark icon is next to the job number, indicating success. Below the list are feed links: 'Atom feed for all' and 'Atom feed for failures'.

Click on the number on the above screenshot to show job details and console output to show the output of the job

Status       **Console Output** 

Changes

 **Console Output**

 View as plain text

 Edit Build Information

 Delete build '#1'

 Timings

```
Started by user Sourabh
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test
[test] $ /bin/sh -xe /tmp/jenkins14067801435241216835.sh
+ sleep 10
+ whoami
jenkins
+ pwd
/var/lib/jenkins/workspace/test
+ touch abc.txt
Finished: SUCCESS
```

/var/lib/Jenkins/workspace/test – Job Location where it executes build.

#### 4) Meaning of different Weather signs for a project/Item on the dashboard

- Sunshine – Job Success so far with no failures

S	W	Name ↴	Last Success	Last Failure	Last Duration
		test	3 min 9 sec #1	N/A	10 sec

- Cloud – Job Failed once, shows time in last failure and also tells the build number on which it failed.

S	W	Name ↴	Last Success	Last Failure	Last Duration
		test	8 min 7 sec #1	42 sec #2	10 sec

- Shower – Job failed twice, shows time in last failure and also tells the build number on which it failed.

S	W	Name ↴	Last Success	Last Failure	Last Duration
		test	9 min 42 sec #1	17 sec #3	10 sec

- Rain - Job failed thrice, shows time in last failure and also tells the build number on which it failed.

S	W	Name ↴	Last Success	Last Failure	Last Duration
		test	13 min #1	1 min 7 sec #4	10 sec

- Thunderstorm – Job failed fourth time, shows time in last failure and also tells the build number on which it failed.

S	W	Name	Last Success	Last Failure	Last Duration
		test	14 min #1	25 sec #5	10 sec

- Partly Cloudy – Status success after multiple fails, change in Last Success

		test	9 min 43 sec #7	1 hr 34 min #5
--	--	------	-----------------	----------------

This is maximum time the weather icon changes and once the build starts succeeding, icon changes back to Sunshine after multiple success.

- 5) If we want to remove old builds and keep only specific number of builds. Click configure on the job and check discard old builds and give the Max # of builds to keep mentioned in the screenshot and click save

The screenshot shows the Jenkins 'General' configuration page for a job named 'test'. On the left, there's a sidebar with links: General (selected), Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main area has a 'Description' field (empty) and a 'Plain text Preview' link. Under the 'Strategy' section, the 'Discard old builds' checkbox is checked (highlighted with a red box). Below it, the 'Log Rotation' section is collapsed. The 'Days to keep builds' field is empty. The 'Max # of builds to keep' field contains the value '2' (highlighted with a red box). At the bottom, there are 'Save' and 'Apply' buttons.

Click Build Now

**Status** **test**

</> Changes

Workspace

Build Now

Configure

Delete Project

Rename

**Permalinks**

- Last build (#5), 1 hr 18 min ago
- Last stable build (#1), 1 hr 32 min ago
- Last successful build (#1), 1 hr 32 min ago
- Last failed build (#5), 1 hr 18 min ago
- Last unsuccessful build (#5), 1 hr 18 min ago
- Last completed build (#5), 1 hr 18 min ago

Build History

Filter... /

- #5 | Aug 2, 2024, 8:38 AM
- #4 | Aug 2, 2024, 8:36 AM
- #3 | Aug 2, 2024, 8:33 AM
- #2 | Aug 2, 2024, 8:31 AM
- #1 | Aug 2, 2024, 8:24 AM

Once the build is completed, see only last 2 builds are left rest are deleted. See screenshot below

Dashboard > test >

**Status** **test**

</> Changes

Workspace

Build Now

Configure

Delete Project

Rename

**Permalinks**

- Last build (#6), 1 min 8 sec ago
- Last stable build (#6), 1 min 8 sec ago
- Last successful build (#6), 1 min 8 sec ago
- Last failed build (#5), 1 hr 20 min ago
- Last unsuccessful build (#5), 1 hr 20 min ago
- Last completed build (#6), 1 min 8 sec ago

Build History

Filter... /

- #6 | Aug 2, 2024, 9:57 AM
- #5 | Aug 2, 2024, 8:38 AM

Atom feed for all Atom feed for failures

- 6) Now if we want to delete the content of the workspace, Click configure on the job. Under Build Environment, Check the option Delete workspace before build starts and hit save.

Dashboard > test > Configuration

## Configure

**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 ?

---

**Build Environment**

Delete workspace before build starts

Advanced ▾

- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans
- Terminate a build if it's stuck
- With Ant ?

## Click Build Now

Status ✖ test

</> Changes

Workspace

Build Now Build Now

Configure

Delete Project

Rename

**Permalinks**

- Last build (#5), 1 hr 18 min ago
- Last stable build (#1), 1 hr 32 min ago
- Last successful build (#1), 1 hr 32 min ago
- Last failed build (#5), 1 hr 18 min ago
- Last unsuccessful build (#5), 1 hr 18 min ago
- Last completed build (#5), 1 hr 18 min ago

**Build History**

#	Date
5	Aug 2, 2024, 8:38 AM
4	Aug 2, 2024, 8:36 AM
3	Aug 2, 2024, 8:33 AM
2	Aug 2, 2024, 8:31 AM
1	Aug 2, 2024, 8:24 AM

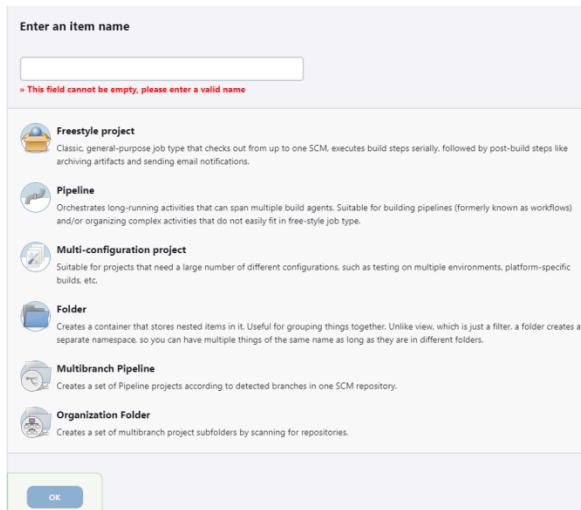
See all the contents are deleted under workspace of job test

```

Started by user Sourabh
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
[test] $ /bin/sh -xe /tmp/jenkins15190843471711515965.sh
+ sleep 10
+ whoami
jenkins
+ pwd
/var/lib/jenkins/workspace/test
+ touch abc.txt
Finished: SUCCESS

```

- 7) Create a new job and copy the contents from old eg test to test1  
 Click on New item



Enter the item name (test1) and select Freestyle project and scroll down to copy from and enter the job name where you want to copy contents from eg in our case copy contents from test and click ok

**Enter an item name**

test1  
» Required field

**Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build actions and sending email notifications.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as流水线) and/or organizing complex activities that do not easily fit in free-style job type.

**Configuration project**  
OK  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments.

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from test OK

All contents are copied from test

Dashboard > test1 > Configuration

**Configure**

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

**Build Steps**

**Execute shell**

Command

See the list of available environment variables

```
sleep 10
whoami
pwd
touch abc.txt
```

Advanced

Add build step

**Post-build Actions**

Add post-build action

Save Apply

8) Defining second job to be triggered if the first job succeeds

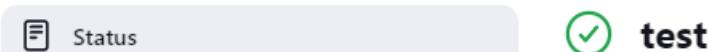
(Downstream) [Step 12 in Ajay Sir Notes]

Click configure on the job. Go to post build actions and select build from other projects

The screenshot shows the Jenkins 'Configuration' page for a job named 'test'. The left sidebar lists several configuration sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps (which is currently selected), and Post-build Actions. The 'Post-build Actions' section is expanded, showing a list of available actions: 'Aggregate downstream test results', 'Archive the artifacts', 'Build other projects' (which is highlighted with a red underline), and 'Publish JUnit test result report'. A red arrow points from the text above to the 'Build other projects' option.

Then under Build other projects. Type the second project i.e. test1 and keep trigger only if build is stable. and click save.

The top screenshot shows the 'Build other projects' configuration screen with the 'Projects to build' field containing 'test1,'. The 'Save' button is highlighted with a yellow arrow. The bottom screenshot shows the same configuration screen with the 'Trigger only if build is stable' radio button selected, while the other two options ('Trigger even if the build is unstable' and 'Trigger even if the build fails') are unselected. Both screenshots also show an 'Apply' button.



</> Changes

Workspace

Build Now

Configure

## Downstream Projects

... test1

After Build now, both jobs are success in below screenshot

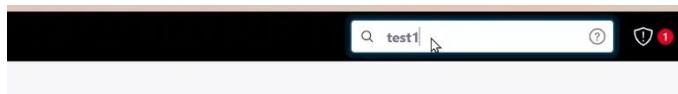
The screenshot shows the Jenkins dashboard with the 'Downstream Projects' section. It lists two projects: 'test' and 'test1'. Both projects are marked as 'Success' (green checkmark icon). The 'test' project has a 'Last Success' of 20 min #7 and a 'Last Failure' of 1 hr 45 min #5. The 'test1' project has a 'Last Success' of N/A and a 'Last Failure' of N/A. Below the main dashboard, there are navigation links for 'Manage Jenkins', 'My Views', 'Build Queue' (which is empty), and 'Build Executor Status' (which shows 1 idle and 2 idle executors).

NOTE: There are two other conditions for build another project

- Trigger even if the build is unstable – If this condition is selected, then the second project is executed even if there are some warnings or errors in the upstream job/project
- Trigger even if the build fails. – If this condition is selected, then the second project is executed even if the upstream job/project fails completely.

### 9) Search panel [Step 5 in Ajay Sir Notes]

Type Job name – It will redirect you to the job. See in screenshot



The screenshot shows the Jenkins dashboard with the search bar containing 'test1'. The search results show two projects: 'test' and 'test1'. Both projects are marked as 'Success' (green checkmark icon). The 'test' project has a 'Last Success' of 2 min 9 sec #12 and a 'Last Failure' of 4 min 8 sec #11. The 'test1' project has a 'Last Success' of 2 min 1 sec #1 and a 'Last Failure' of N/A. Below the search results, there are buttons for 'All' and '+', and a footer icon for 'Icon: S M L'.

**test1**

Status

Changes

Workspace

Build Now

Configure

Delete Project

Upstream Projects

test

Permalinks

To go to build- Type job name # Build number in search panel

Dashboard >

+ New Item

Build History

All +

Manage Jenkins

My Views

S	W	Name	Last Success	Last Failure	Last Duration
✓	rainy	test	2 min 28 sec #12	4 min 26 sec #11	27 ms
✓	sunny	test1	2 min 19 sec #1	N/A	5 sec

No builds in the queue

Dashboard > test > #5

Search (CTRL+K)

Status

#5 (Jul 27, 2024, 7:12:28 AM)

Changes

Console Output

Edit Build Information

Delete build '#5'

Timings

Previous Build

Next Build

</> No changes.

(⌚) Started by user a

(⌚) This run spent:

- 3 ms waiting;
- 10 sec build duration;
- 10 sec total from scheduled to completion.

## 10) System Configuration overview. [Step 6 in Ajay Sir Notes]

Manage Jenkins- For admin tasks

System- To change any configuration of plugins which you have installed

Home directory path

System Message

Jenkins URL

No. of executor- no of job execution together

## System Configuration

 **System**  
Configure global settings and paths.

Dashboard > Manage Jenkins > System >

### System

**Home directory** ?  
By default, Jenkins stores all of its data in this directory on the file system  
`/var/lib/jenkins` 

**System Message**  
This message will be displayed at the top of the Jenkins main page. This can be useful for posting notifications to your users

**Jenkins Location**

**Jenkins URL** ?  
`http://54.224.9.237:8080/`

Dashboard > Manage Jenkins > System >

Plain text [Preview](#)

**# of executors**  
2

**Tools - to install any tool**

 **Tools**  
Configure tools, their locations and automatic installers.

## Tools

### Maven Configuration

#### Default settings provider

Use default maven settings

#### Default global settings provider

Use default maven global settings

### JDK installations

Add JDK

### Git installations

Git

Name  
Default

Path to Git executable ?  
git

Save      Apply

## Manage plugins

## Plugins

Updates

Available plugins

Installed plugins

Advanced settings

## Nodes

### Nodes

+ New Node

Configure Monitors



S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	Built-In Node	Linux (amd64)	In sync	7.57 GiB	0 B	7.57 GiB	0ms
.	Data obtained	3 min 42 sec	3 min 42 sec	3 min 42 sec	3 min 42 sec	3 min 42 sec	3 min 42 sec

## Security

## Security

### Authentication

Disable "Keep me signed in" ?

#### Security Realm

Jenkins' own user database

Allow users to sign up ?

### Authorization

Logged-in users can do anything

Allow anonymous read access ?

### Markup Formatter

#### Markup Formatter ?

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped to thei

### Agents

## Credentials

### Credentials

T P Store ↴

Domain

ID

Name

#### Stores scoped to Jenkins

P Store ↴

Domains

 System

(global)

Icon: S M L

## Users

### Users

+ Create User

These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID ↴

Name

 sourabh

Sourabh

⋮

## 11) How to change Jenkins Theme [Step 7 in Ajay Sir Notes]

Goto to the link <https://github.com/afonsof/jenkins-material-theme>

Follow the steps GitHub page

- Choose your color:



- Replace {{your-color-name}} in the URL by the chosen color: <https://cdn.rawgit.com/afonsof/jenkins-material-theme/gh-pages/dist/material-purple.css>
- Install [Jenkins Simple Theme Plugin](#) by going to Manage Jenkins>Plugins>Available Plugins>Simple Theme and select Install

Dashboard > Manage Jenkins

## Manage Jenkins

+ New Item  
Build History  
Manage Jenkins  
My Views

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

System Configuration

Build Queue: No builds in the queue.  
Build Executor Status: No build executors available.

System: Configure global settings and paths.  
Tools: Configure tools, their locations and automatic installers.

Plugins: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Dashboard > Manage Jenkins > Plugins

Plugins: simple theme

Updates  
Available plugins (Simple Theme)  
Installed plugins  
Advanced settings  
Download progress

Install Name: Simple Theme  
User Interface UI Themes  
This plugin allows to customize Jenkins' appearance with custom CSS and JavaScript. It also allows to replace the Favicon.

Released 1 mo 24 days ago

Dashboard > Manage Jenkins > Plugins

## Plugins

Updates  
Available plugins (Simple Theme)  
Installed plugins  
Advanced settings  
Download progress

Plugin	Status
OWASP Markup Formatter	Success
Build Timeout	Success
Credentials Binding	Success
Timestamper	Success
Workspace Cleanup	Success
Ant	Success
Gradle	Success
Pipeline	Success
GitHub Branch Source	Success
Pipeline: GitHub Groovy Libraries	Success
Pipeline Graph View	Success
Git	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Dark Theme	Success
Loading plugin extensions	Success
Simple Theme	Success
Loading plugin extensions	Success

- Click Manage Jenkins > Appearance

The screenshot shows the Jenkins 'Manage Jenkins' interface. At the top, there's a navigation bar with 'Dashboard' and 'Manage Jenkins'. Below it, a sidebar has links for 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted with a yellow box), and 'My Views'. On the right, under 'System Configuration', there are four sections: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), and 'Appearance' (Configure the look and feel of Jenkins). A note at the top says: 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)'.

- Scroll down and under Customizable theme>Theme Elements>Drop Down on Add and select CSS URL

The screenshot shows the 'Appearance' configuration page. It starts with a 'Themes' section where users can choose between 'Dark', 'Dark (System)', and 'Default'. There's also a checkbox for 'Do not allow users to select a different theme'. Below this is a 'Customizable theme' section with a 'Theme elements' dropdown menu. The 'CSS URL' option is highlighted with a yellow box. Other options in the dropdown include 'Extra CSS', 'Favicon URL', and 'JavaScript URL'.

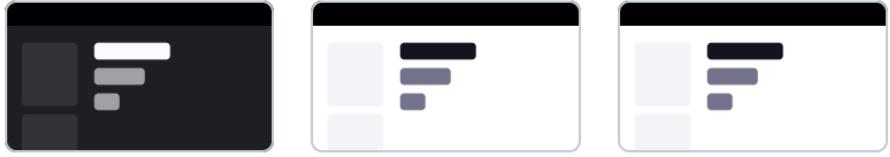
- Set the CSS URL field to the below generated URL and click Save.

<https://cdn.rawgit.com/afonsof/jenkins-material-theme/gh-pages/dist/material-purple.css>

**Appearance**

Themes

Select theme



Dark      Dark (System)      Default

Do not allow users to select a different theme

---

Customizable theme

Theme elements

CSS URL  
URL of theme CSS  
<https://cdn.rawgit.com/afonsof/jenkins-material-theme/gh-pages/dist/material-purple.css>

Add

- Once we save and go back to the dashboard, we see the changes are applied.



## 12) How to create a User in Jenkins. [Step 8 in Ajay Sir Notes]

Goto Manage Jenkins > Users

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.

<b>System Configuration</b>	<b>Tools</b>	<b>Plugins</b>	<b>Nodes</b>
<input type="button" value="Build Queue"/> <small>No builds in the queue.</small>	<input type="button" value="System"/> Configure global settings and paths.	<input type="button" value="Tools"/> Configure tools, their locations and automatic installers.	<input type="button" value="Plugins"/> Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
<input type="button" value="Build Executor Status"/> <small>1 idle 2 idle</small>	<input type="button" value="Clouds"/> Add, remove, and configure cloud instances to provision agents on-demand.	<input type="button" value="Appearance"/> Configure the look and feel of Jenkins	<input type="button" value="Nodes"/> Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
<b>Security</b>	<b>Credentials</b>	<b>Credential Providers</b>	<b>Users</b>
<input type="button" value="Security"/> Secure Jenkins; define who is allowed to access/use the system.	<input type="button" value="Credentials"/> Configure credentials	<input type="button" value="Credential Providers"/> Configure the credential providers and types	<input type="button" value="Users"/> Create/delete/modify users that can log in to this Jenkins.

## Click + Create User

The screenshot shows the Jenkins 'Users' page with a single user listed. The user is named 'sourabh' with a user ID of 'sourabh'. There is a 'Create User' button at the top right.

User ID	Name
sourabh	Sourabh

Give details like username, password, Confirm Password, Full Name and Email Address and click Create User.

The screenshot shows the 'Create User' form. It includes fields for Username ('a'), Password ('.....'), Confirm password ('.....'), Full name ('a'), and E-mail address ('a@a.com'). A 'Create User' button is at the bottom.

Dashboard > Manage Jenkins > Jenkins' own user database > Create User

Create User

Username: a

Password: .....

Confirm password: .....

Full name: a

E-mail address: a@a.com

Create User

Once its created, it is displayed under Users

The screenshot shows the Jenkins 'Users' page with two users listed: 'a' and 'sourabh'. Both users have their respective user IDs. There is a 'Create User' button at the top right.

User ID	Name
a	a
sourabh	Sourabh

## 13) Jenkins Role Based Access Control [Step 9 in Ajay Sir Notes]

We need to create two users for performing the use case.

In our case, we have created user a and user b

Dashboard > Jenkins' own user database

Users 3

These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID	Name	
	a	
	b	

+ Create User

Next, we need to install Role Based Authorization Strategy plugin through available plugins.

Dashboard > Manage Jenkins > Plugins

Plugins

Updates Available plugins Installed plugins

Install Name : Role-based Authorization Strategy 743.v142ee\_b\_cdf1d3  
Security Authentication and User Management  
Enables user authorization using a Role-Based strategy. Roles can be defined globally or for particular jobs or nodes selected by regular expressions.  
Released 13 days ago

Dashboard > Manage Jenkins > Plugins

Plugins

Updates Available plugins Installed plugins Advanced settings Download progress

Build Timeout	
Credentials Binding	
Timestamper	
Workspace Cleanup	
Ant	
Gradle	
Pipeline	
GitHub Branch Source	
Pipeline: GitHub Groovy Libraries	
Pipeline Graph View	
Git	
SSH Build Agents	
Matrix Authorization Strategy	
PAM Authentication	
LDAP	
Email Extension	
Mailer	
Dark Theme	
Loading plugin extensions	
Simple Theme	
Loading plugin extensions	
<b>Role-based Authorization Strategy</b>	
Loading plugin extensions	

Next, we need to go to Manage Jenkins> Security

Dashboard > Manage Jenkins

+ New item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

Idle
1 Idle
2 Idle

System Configuration

System

Configure global settings and paths.

Clouds

Add, remove, and configure cloud instances to provision agents on-demand.

Security

Security

Secure Jenkins; define who is allowed to access/use the system.

The screenshot shows the Jenkins Manage Jenkins dashboard. At the top left, there's a navigation bar with 'Dashboard' and 'Manage Jenkins'. Below it are links for 'New item', 'Build History', 'Manage Jenkins' (which is highlighted), and 'My Views'. On the left, there are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (listing 1 Idle and 2 Idle). The main area is titled 'System Configuration' and contains three items: 'System' (with a gear icon), 'Clouds' (with a cloud icon), and 'Security' (with a lock icon). A prominent orange warning box at the top right states: 'Building on the built-in node can be a security issue. You should consider using a separate node for your builds.' The 'Security' section is highlighted with a yellow box around its title.

Under Authorization > Drop down and select Role-Based Strategy and click Save.

## Security

### Authentication

Disable "Keep me signed in" [?](#)

### Security Realm

Jenkins' own user database

Allow users to sign up [?](#)

### Authorization

Role-Based Strategy

### Markup Formatter

#### Markup Formatter [?](#)

Plain text

Shows descriptions mostly as written. HTML unsafe characters like < and & are escaped

### Agents

#### TCP port for inbound agents [?](#)

Save

Apply

Now go to Manage Jenkins > Manage and Assign Roles.

The Restrict project naming configuration is not set to the Role-based Strategy. This can lead to problems as it allows users to create items, for which they have not the sufficient permissions to discover, read or configure. [\[Dismiss\]](#)

Under Manage Role > Enter “Developer” on Role to Add and click Add.

Role	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>						
Developer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Role to add  
Developer  
Add

Once the role is created, check Read under Overall for the permissions to the Developer Role and click Save

Role	Overall	Credentials	Agent	Job	Run	View	SCM	Metrics
admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>						
Developer	<input checked="" type="checkbox"/>	<input type="checkbox"/>						

Role to add  
Developer  
Add

Item roles

Role Pattern	Template	Credentials	Job	Run	View	SCM	Metrics
<input type="checkbox"/>							
<input type="checkbox"/>							

Role to add  
[Save](#) [Apply](#)

Now go to assign roles > Under Global Roles > Click on Add user

The screenshot shows the Jenkins 'Assign Roles' page. At the top, there's a breadcrumb navigation: Dashboard > Manage Jenkins > Manage and Assign Roles > Assign Roles. Below the breadcrumb, there are four main links: Manage Roles, Assign Roles (which is highlighted with a yellow box), Permission Templates, and Role Strategy Macros. On the right side, under 'Global roles', there's a table where rows represent User/Groups and columns represent roles (admin, Developer). The table includes rows for Anonymous, Authenticated Users, and a specific user named Sourabh. Sourabh has the 'Developer' role assigned (indicated by a checked checkbox). There are also 'Add User' and 'Add Group' buttons at the bottom.

Type the User id, in our case its b and then click OK

This screenshot shows the same 'Assign Roles' page as above, but with a modal dialog box overlaid. The dialog is titled 'User ID:' and contains a single input field with the value 'b'. At the bottom of the dialog are 'Cancel' and 'OK' buttons. The background of the page is dimmed, and the 'Add User' button is visible at the bottom of the main content area.

Once the user is added, we will assign developer role to it mentioned in the below screenshot and click Save.

 Manage Roles

 Assign Roles

 Permission Templates

 Role Strategy Macros

## Assign Roles

### Global roles

User/Group	Developer	admin
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>
Sourabh	<input type="checkbox"/>	<input checked="" type="checkbox"/>
b	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Add User

Add Group

### Item roles

User/Group
Anonymous
Authenticated Users

Add User

Add Group

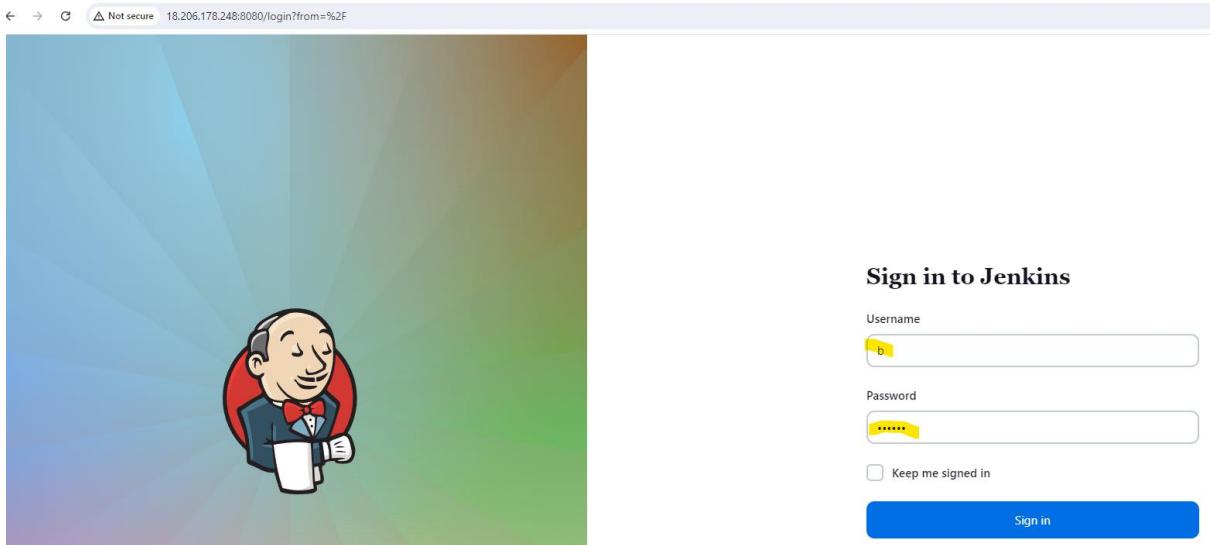
### Agent roles

User/Group
Anonymous
Authenticated Users

 Save

Apply

Now we will see if the role is applied properly on user b by login to Jenkins with user b credentials.



The role is applied successfully as nothing visible on the dashboard for user b

Jenkins

Dashboard >

Build History

My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

#### 14) Use of git plugin and Clean Workspace in Jenkins. [Step 10 in Ajay Sir Notes]

We will create a freestyle project named test and will select the below options to clone a git repo using the project and click Save.

- Select Delete workspace before build starts
- Under Build Steps> Select Execute Shell and write steps below:

```
git clone https://github.com/shashirajraja/Train-Ticket-Reservation-System.git
```

```
cd Train-Ticket-Reservation-System
```

**ls -ltr**

The screenshot shows the Jenkins project configuration for a project named "test". In the "Build Steps" section, there is a single step defined:

```
git clone https://github.com/shashirajraja/Train-Ticket-Reservation-System.git
cd Train-Ticket-Reservation-System
ls -ltr
```

At the bottom of the configuration page, there are "Save" and "Apply" buttons.

Now run Build Now on the Project “test”

The screenshot shows the Jenkins dashboard for the "test" project. The "Build Now" button is highlighted with a yellow box.

**Permalinks**

- Status
- </> Changes
- Workspace
- Build Now
- Configure
- Delete Project
- Rename

**Build History** trend ▾

No builds

Atom feed for all Atom feed for failures

We see the build is completed successfully.

Status      **test**

</> Changes

Workspace

Build Now

Configure

Delete Project

Rename

**Build History**      trend ▾

Filter... /

#1      Aug 21, 2024, 11:03 AM

Atom feed for all Atom feed for failures

We will go into the build and check the console output to see the status.  
All the steps have been executed successfully.

Status      **Console Output**

</> Changes

Console Output

Edit Build Information

Delete build '#1'

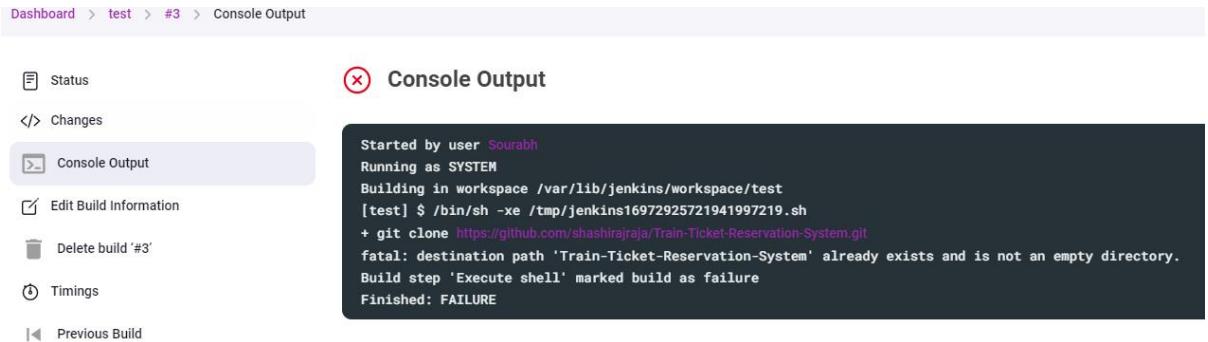
Timings

```

Started by user Sourabh
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[test] $ /bin/sh -xe /tmp/jenkins14465780432846795914.sh
+ git clone https://github.com/shashirajraja/Train-Ticket-Reservation-System.git
Cloning into 'Train-Ticket-Reservation-System'...
+ cd Train-Ticket-Reservation-System
+ ls -ltr
total 32
-rw-r--r-- 1 jenkins jenkins 8917 Aug 21 11:03 README.md
-rw-r--r-- 1 jenkins jenkins 2670 Aug 21 11:03 Dummy-Database.md
drwxr-xr-x 2 jenkins jenkins 4096 Aug 21 11:03 Screenshots
drwxr-xr-x 4 jenkins jenkins 4096 Aug 21 11:03 WebContent
drwxr-xr-x 3 jenkins jenkins 4096 Aug 21 11:03 src
-rw-r--r-- 1 jenkins jenkins 2180 Aug 21 11:03 pom.xml
Finished: SUCCESS

```

If we run the build again without selecting the Delete Workspace before build starts then it will give an error that project is already there. So we have to select it in order run the build successfully.



The screenshot shows the Jenkins interface for a build named 'test' under job '#3'. The 'Console Output' tab is selected. The output log shows the following error:

```
Started by user Sourabh
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test
[test] $ /bin/sh -xe /tmp/jenkins16972925721941997219.sh
+ git clone https://github.com/shashirajraja/Train-Ticket-Reservation-System.git
fatal: destination path 'Train-Ticket-Reservation-System' already exists and is not an empty directory.
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

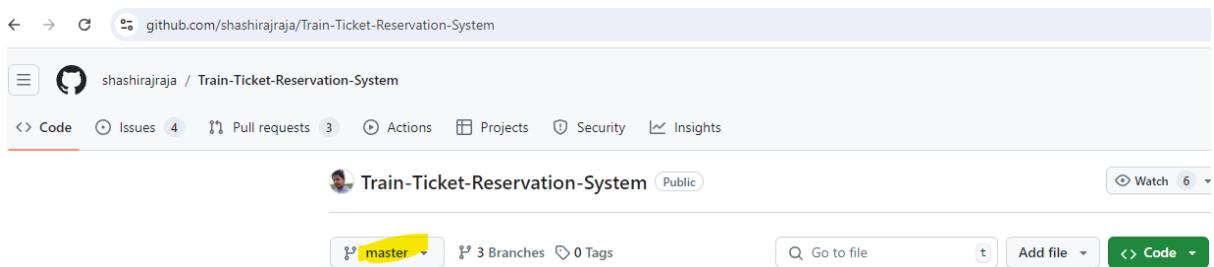
We can also use the git project path option from the source code management option. We can do it by choosing the below options

We need to select git under Source Code Management

We need to give the Repository URL under repositories

<https://github.com/shashirajraja/Train-Ticket-Reservation-System.git>

Lastly, we need to specify the branch name same as it is on the git hub link, in our case its \*/master in the git link also so we need to specify the same in the job also. Then Click Save



The screenshot shows the Jenkins configuration interface for a job named 'test'. The 'Source Code Management' section is selected. It shows a 'Repository URL' field containing 'https://github.com/shashirajraja/Train-Ticket-Reservation-System.git' and a 'Branch Specifier' field containing '+/master'. There are 'Save' and 'Apply' buttons at the bottom.

We can see that git repo is cloned in the build job.

The screenshot shows the Jenkins console output for a build. It includes a log of commands run by Jenkins, such as 'git init', 'git fetch', and 'git config', which show the cloning of the 'Train-Ticket-Reservation-System' repository from GitHub.

```

Started by user Sourabh
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/shashirajraja/Train-Ticket-Reservation-System.git
> git init /var/lib/jenkins/workspace/test # timeout=10
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/shashirajraja/Train-Ticket-Reservation-System.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/shashirajraja/Train-Ticket-Reservation-System.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 0d77c7d627007e41ef1134eff24bc98bc00334bc (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0d77c7d627007e41ef1134eff24bc98bc00334bc # timeout=10
Commit message: "Update index.html"
First time build. Skipping changelog.
Finished: SUCCESS

```

## 15) Trigger Build Remotely on Jenkins [Step 11 in Ajay Sir Notes]

We can trigger a job remotely through a Web URL in Jenkins also.

We will follow the below steps to achieve the same.

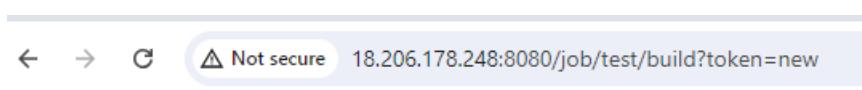
While configuring the job, we need to select Trigger builds remotely under Build Triggers, then we can give any authentication token eg new and Click Save.

The screenshot shows the Jenkins job configuration page for a job named 'test'. The 'Build Triggers' section is highlighted with a yellow box. Under 'Build Triggers', the 'Trigger builds remotely (e.g., from scripts)' option is checked, and the 'Authentication Token' field contains the value 'new'. Other trigger options like 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM' are shown but not selected. Below the triggers, the 'Build Environment' section is partially visible, showing options like 'Delete workspace before build starts' (which is checked) and 'Advanced'. At the bottom of the configuration page are 'Save' and 'Apply' buttons.

To trigger the build remotely we need use the below URL

JENKINS\_URL/job/test/build?token=TOKEN\_NAME or  
/buildWithParameters?token=TOKEN\_NAME

Below screenshot is in our case



We see the 2<sup>nd</sup> Build is also executed successfully.

The screenshot shows a Jenkins project named 'test'. The top navigation bar includes 'Dashboard > test >'. On the left, there's a sidebar with links: 'Status' (selected), 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', 'Rename', and a 'Build History' section. The 'Build History' section displays two builds: '#2' (Aug 21, 2024, 12:05 PM) and '#1' (Aug 21, 2024, 11:55 AM). Below the history are links for 'Atom feed for all' and 'Atom feed for failures'. To the right of the sidebar, the main content area has a green checkmark icon and the word 'test'. Below it is a 'Permalinks' section with a list of recent builds.

test

Permalinks

- Last build (#1), 10 min ago
- Last stable build (#1), 10 min ago
- Last successful build (#1), 10 min ago
- Last completed build (#1), 10 min ago

Alternate Way to run the job through remotely is by using a plugin

We need to install the plugin Build Authorization Token Root

The screenshot shows the Jenkins 'Manage Jenkins > Plugins' page. The left sidebar has links for 'Updates', 'Available plugins' (selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area shows a search bar with 'build auth' and a list of plugins. The 'Build Authorization Token Root' plugin is highlighted with a yellow box. It was released 2 yr 1 mo ago and provides REST build triggers for anonymous users. Below it is the 'Build Token Trigger' plugin, released 6 yr 4 mo ago.

## Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Ant		Success
Gradle		Success
Pipeline		Success
Github Branch Source		Success
Pipeline: GitHub Groovy Libraries		Success
Pipeline Graph View		Success
Git		Success
SSH Build Agents		Success
Matrix Authorization Strategy		Success
PAM Authentication		Success
LDAP		Success
Email Extension		Success
Mailer		Success
Dark Theme		Success
Loading plugin extensions		Success
Simple Theme		Success
Loading plugin extensions		Success
Role-based Authorization Strategy		Success
Loading plugin extensions		Success
JavaMail API		Success
SSH server		Success
Build Authorization Token Root		Success
Loading plugin extensions		Success

Now we will run the below URL to see if build runs.

<Jenkins\_URL>buildByToken/build?job=<Job\_Name>&token=<token Name>

18.206.178.248:8080/buildByToken/build?job=test&token=new

See the Build 3 is also completed successfully.

Dashboard > test >

Status
test

Changes
Permalinks

- Workspace
- Build Now
- Configure
- Delete Project
- Rename

Build History trend ▾

Filter...	
#3	Aug 21, 2024, 12:14 PM
#2	Aug 21, 2024, 12:05 PM
#1	Aug 21, 2024, 11:55 AM

[Atom feed for all](#) [Atom feed for failures](#)

We can also run it on the Ubuntu terminal using the below command.

Just need to add “\” before the & and then run.

```
<Jenkins_URL>buildByToken/build?job=<Job_Name>\&token=<token Name>
```

```
root@ip-172-31-33-31:/var/lib/jenkins/workspace/test# curl http://18.206.178.248:8080/buildByToken/build?job=test\&token=new
root@ip-172-31-33-31:/var/lib/jenkins/workspace/test#
```

Build 4 is completed successfully

The screenshot shows the Jenkins interface for the 'test' project. At the top, there's a navigation bar with a Jenkins logo and the word 'Jenkins'. Below it, a breadcrumb navigation shows 'Dashboard > test >'. On the left, a sidebar lists project management options: Status (highlighted), Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main content area has a green checkmark icon next to the project name 'test'. Below it, a section titled 'Permalinks' lists four recent builds: Last build (#3), Last stable build (#3), Last successful build (#3), and Last completed build (#3), all from 5 min 22 sec ago. To the right of the sidebar is a 'Build History' card. It shows a table with four rows, each representing a build: #4 (Aug 21, 2024, 12:20 PM), #3 (Aug 21, 2024, 12:14 PM), #2 (Aug 21, 2024, 12:05 PM), and #1 (Aug 21, 2024, 11:55 AM). Each row has a green circular icon with a checkmark. At the bottom of the card, there are links for 'Atom feed for all' and 'Atom feed for failures'.

Build	Date
#4	Aug 21, 2024, 12:20 PM
#3	Aug 21, 2024, 12:14 PM
#2	Aug 21, 2024, 12:05 PM
#1	Aug 21, 2024, 11:55 AM

## 16) Build jobs periodically in Jenkins [Step 13 in Ajay Sir Notes]

We can set the builds to run on periodic basis. To perform that we need to follow below steps.

While configuring the job, we need to set the schedule under Build Triggers and then check Build Periodically and define the schedule and then click Save.

In our e.g., we are scheduled the job to run every 3 minutes, so whenever we save the configuration on the job, from there onwards, it should run every minute until we remove the settings. So, we have used the same job “test” for our example

The screenshot shows the Jenkins 'Configure' screen for a job named 'test'. In the left sidebar, 'Build Triggers' is selected. Under 'Build Triggers', the 'Build periodically' checkbox is checked, and the schedule is set to 'H/3 \* \* \* \*'. A tooltip indicates the job would last have run at Wednesday, August 21, 2024 at 12:48:22 PM Coordinated Universal Time and would next run at Wednesday, August 21, 2024 at 12:46:22 PM Coordinated Universal Time. Other trigger options like 'Trigger builds remotely' and 'GitHub hook trigger for GITScm polling' are also listed. Below the triggers, the 'Build Environment' section is shown with the 'Delete workspace before build starts' checkbox checked. At the bottom, there are 'Save' and 'Apply' buttons.

Now you can see the job runs as per the schedule after every 3 minutes.

The screenshot shows the Jenkins 'Dashboard' for the 'test' project. It displays the build history with two entries: '#9 Aug 21, 2024, 12:52 PM' and '#8 Aug 21, 2024, 12:49 PM'. The most recent build (#9) is highlighted with a yellow box. On the left, there are links for 'Build Now', 'Configure', 'Delete Project', and 'Rename'. To the right, a list shows the last three builds: 'Last stable build (#8), 34 sec ago', 'Last successful build (#8), 34 sec ago', and 'Last completed build (#8), 34 sec ago'.

## 17) Poll SCM through Jenkins [Step 14 in Ajay Sir Notes]

This option will only check in every given time if any change happened in GIT and if there is any change in GIT, it will execute the build accordingly.

We can schedule it while configuring the build under the Build Triggers option then check Poll SCM and give the schedule on which it will check for any SCM change. Then click save

The screenshot shows the Jenkins 'Configure' screen for a project named 'test'. Under the 'Build Triggers' section, the 'Poll SCM' checkbox is checked, and the 'Schedule' dropdown is set to 'H/3 \* \* \* \*'. Below the schedule, a note states: 'Would last have run at Wednesday, August 21, 2024 at 1:19:57 PM Coordinated Universal Time; would next run at Wednesday, August 21, 2024 at 1:19:57 PM Coordinated Universal Time.' There is also an unchecked checkbox for 'Ignore post-commit hooks'. The 'Build Environment' section is partially visible below, with the 'Delete workspace before build starts' checkbox checked. At the bottom, there are 'Save' and 'Apply' buttons, with 'Save' being highlighted.

Any kind of changes which happens in git, it will be logged into Git Polling log. Below example shows no changes.

The screenshot shows the Jenkins 'Git Polling Log' page. The log output is as follows:

```
Started on Aug 21, 2024, 1:22:00 PM
Using strategy: Default
[poll] Last Built Revision: Revision 8d77c7d627007e41ef1134eff24bc98bc00334bc (refs/remotes/origin/master)
The recommended git tool is: NONE
No credentials specified
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git ls-remote -h -- https://github.com/shashirajraja/Train-Ticket-Reservation-System.git # timeout=10
Found 3 remote heads on https://github.com/shashirajraja/Train-Ticket-Reservation-System.git
[poll] Latest remote head revision on refs/heads/master is: 8d77c7d627007e41ef1134eff24bc98bc00334bc - already built by 14
Done. Took 0.11 sec
No changes.
```

**18) GitHub hook trigger for GITScm polling through Jenkins [Step 15 in Ajay Sir Notes]**

This is used when a job needs to be triggered when there is any change in the GitHub repo.

Steps to followed are below:

While configuring the job, Check the option of GitHub hook trigger for GITScm polling under Build Triggers and then click Save.

The screenshot shows the Jenkins 'Configure' screen for a job named 'test'. The left sidebar lists configuration sections: General, Source Code Management, Build Triggers (which is selected and highlighted in grey), Build Environment, Build Steps, and Post-build Actions. The 'Build Triggers' section contains several options: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (which is checked and highlighted in yellow), and 'Poll SCM'. Below the triggers is the 'Build Environment' section, which includes options like 'Delete workspace before build starts' (checked), 'Advanced' (with 'Use secret text(s) or file(s)', 'Add timestamps to the Console Output', 'Inspect build log for published build scans', 'Terminate a build if it's stuck', and 'With Ant'). At the bottom are 'Save' and 'Apply' buttons.

Dashboard > test > Configuration

(Auto)

**Configure**

General

Source Code Management

**Build Triggers**

Build Environment

Build Steps

Post-build Actions

Additional Behaviours

Add ▾

**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 ?

**Build Environment**

Delete workspace before build starts

Advanced ▾

Use secret text(s) or file(s) ?

Add timestamps to the Console Output

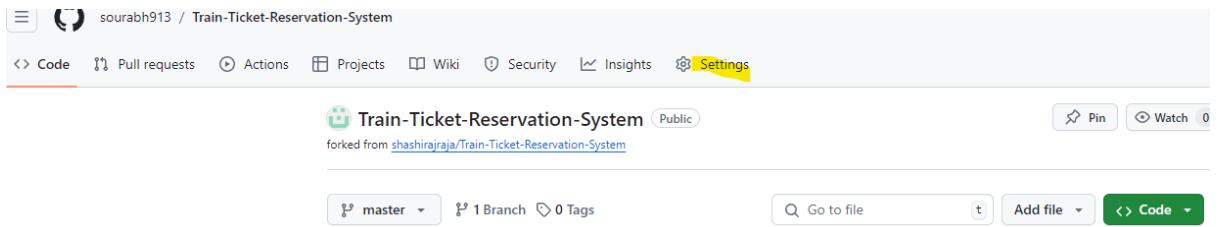
Inspect build log for published build scans

Terminate a build if it's stuck

With Ant ?

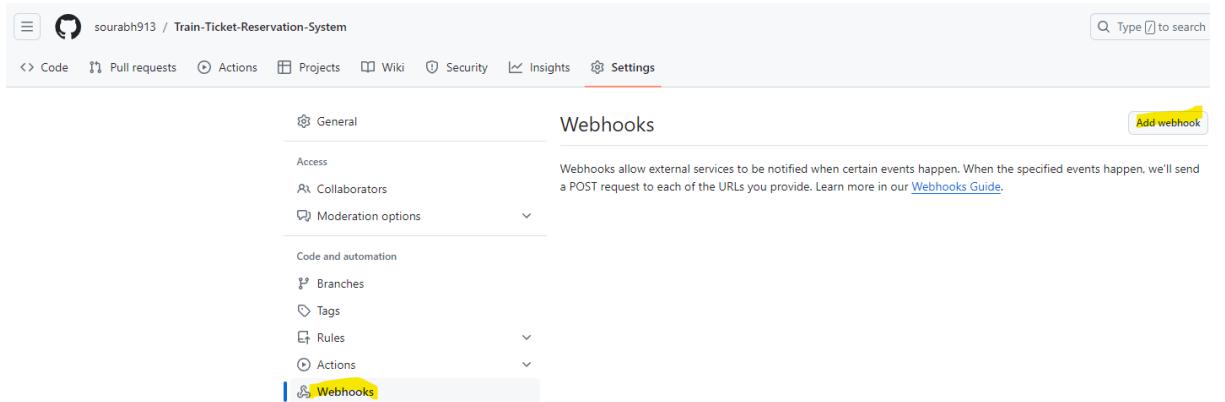
**Save** Apply

Then on the GitHub side, Go to Settings



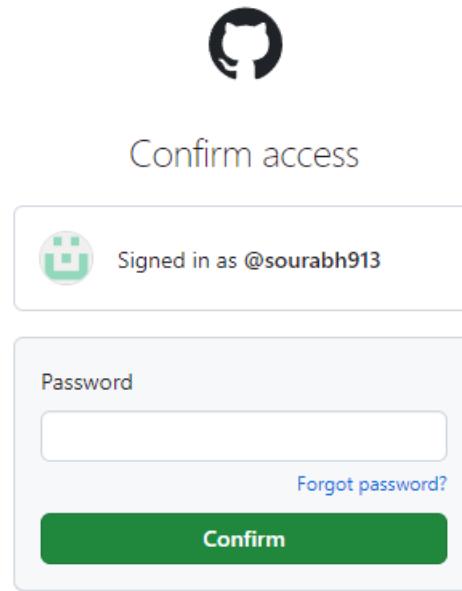
A screenshot of a GitHub repository settings page. The repository is named "Train-Ticket-Reservation-System". The "Settings" tab is highlighted with a yellow box. The page shows basic repository statistics: master branch, 1 branch, 0 tags. There are buttons for "Go to file", "Add file", and "Code".

Select Webhooks > Add webhook



A screenshot of the GitHub settings page, specifically the "Webhooks" section. The "Webhooks" tab is highlighted with a yellow box. It shows a brief description of what webhooks are and a link to the "Webhooks Guide". Below this, there's a list of sections: General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, and Webhooks. The "Webhooks" section is currently selected.

Need to put in password for your GitHub account.



The image shows a GitHub sign-in form. At the top is the GitHub logo. Below it is the text "Confirm access". Inside a box, it says "Signed in as @sourabh913". Below this is a "Password" input field with a "Forgot password?" link next to it. A large green "Confirm" button is at the bottom. A tip at the bottom of the form reads: "Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity."

Then under the Payload URL, give the Jenkins URL /github webhook

And under Content type > Select application/json.

Click Add Webhook

The screenshot shows the GitHub settings interface for managing webhooks. On the left, a sidebar lists various configuration sections: General, Access, Collaborators, Moderation options, Code and automation (with Branches, Tags, Rules, Actions), Webhooks (which is selected and highlighted in blue), Environments, Codespaces, Pages, Security (Code security and analysis, Deploy keys, Secrets and variables), Integrations (GitHub Apps, Email notifications), and Integrations (GitHub Apps, Email notifications). The main content area is titled "Webhooks / Add webhook". It contains instructions about sending POST requests to a URL with event details. The "Payload URL" field is set to "http://18.206.178.248:8080/github-webhook". The "Content type" dropdown is set to "application/json". There is a "Secret" input field which is empty. Under "SSL verification", there is a note that SSL certificates are verified by default, and a radio button group where "Enable SSL verification" is selected. Below that, it asks "Which events would you like to trigger this webhook?", with three options: "Just the push event." (selected), "Send me everything.", and "Let me select individual events.". A checkbox labeled "Active" is checked, with a note below stating "We will deliver event details when this hook is triggered". At the bottom is a green "Add webhook" button.

The screenshot shows the GitHub settings interface for managing webhooks. The sidebar is identical to the previous screenshot. The main content area is titled "Webhooks" and contains a single entry: "http://18.206.178.248:8080/github-... (all events)". To the right of this entry are "Edit" and "Delete" buttons. Above the list, there is a note: "Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#)".

For Example, in my case I made a change under my repo > screenshots > rename viewprofil.png to viewprofile.png

Before Change

The image consists of two screenshots from GitHub illustrating a file rename operation.

**Screenshot 1: File List**

This screenshot shows a list of files in the 'Screenshots' directory of the 'Train-Ticket-Reservation-System' repository. The file 'viewprofile.png' has been renamed to 'viewprofilा.png'. The commit message is 'Rename viewprofile.png to viewprofilा.png'. The file was committed 2 minutes ago by user 'sourabh913'.

Name	Last commit message	Last commit date
..		
Availability.png	Screenshots of WebPages for this project	5 years ago
Search.png	Screenshots of WebPages for this project	5 years ago
TicketBook.png	Screenshots of WebPages for this project	5 years ago
addtrains.png	Screenshots of WebPages for this project	5 years ago
booknow.png	Screenshots of WebPages for this project	5 years ago
fare result.png	Screenshots of WebPages for this project	5 years ago
fareenquiry.png	Screenshots of WebPages for this project	5 years ago
login.png	Screenshots of WebPages for this project	5 years ago
passwordchange.png	Screenshots of WebPages for this project	5 years ago
registeruser.png	Screenshots of WebPages for this project	5 years ago
usehome.png	Screenshots of WebPages for this project	5 years ago
viewprofile.png	Rename viewprofile.png to viewprofilा.png	2 minutes ago

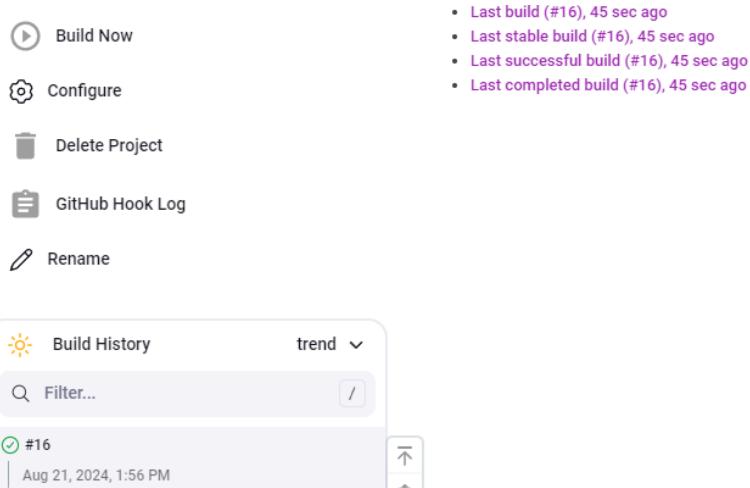
**Screenshot 2: Commit Changes Dialog**

This screenshot shows the 'Commit changes' dialog for the file 'viewprofile.png'. The commit message is 'Rename viewprofilा.png to viewprofile.png'. The extended description field contains the placeholder 'Add an optional extended description..'. The 'Commit directly to the master branch' radio button is selected. The 'Commit changes' button is highlighted in green.

After Change

Train-Ticket-Reservation-System / Screenshots /		
		Add file ⚙️ · ...
 sourabh913 · Rename viewprofile.png to viewprofile.png		064c16f · now · History
This branch is 2 commits ahead of shashiraJraJa/Train-Ticket-Reservation-System:master ·		I Contribute · Sync fork ·
Name	Last commit message	Last commit date
Availability.png	Screenshots of WebPages for this project	5 years ago
Search.png	Screenshots of WebPages for this project	5 years ago
TicketBook.png	Screenshots of WebPages for this project	5 years ago
addtrains.png	Screenshots of WebPages for this project	5 years ago
booknow.png	Screenshots of WebPages for this project	5 years ago
fare result.png	Screenshots of WebPages for this project	5 years ago
fareenquiry.png	Screenshots of WebPages for this project	5 years ago
login.png	Screenshots of WebPages for this project	5 years ago
passwordchange.png	Screenshots of WebPages for this project	5 years ago
registeruser.png	Screenshots of WebPages for this project	5 years ago
userhome.png	Screenshots of WebPages for this project	5 years ago
 viewprofile.png	Rename viewprofile.png to viewprofile.png	now

Immediately after the change, we see that build ran in Jenkins.



Build Now

- Last build (#16), 45 sec ago
- Last stable build (#16), 45 sec ago
- Last successful build (#16), 45 sec ago
- Last completed build (#16), 45 sec ago

Configure

Delete Project

GitHub Hook Log

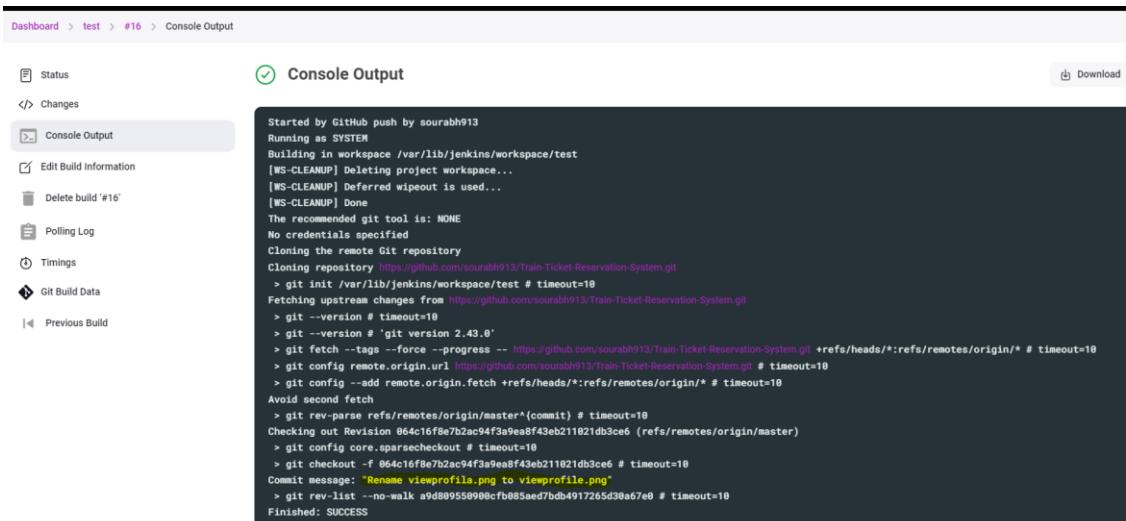
Rename

Build History

#16

Aug 21, 2024, 1:56 PM

We can see the changes in the console output reflected successfully



Status · test · #16 · Console Output

Console Output

```

Started by GitHub push by sourabh913
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/sourabh913/Train-Ticket-Reservation-System.git
> git init /var/lib/jenkins/workspace/test # timeout=10
Fetching upstream changes from https://github.com/sourabh913/Train-Ticket-Reservation-System.git
> git --version # timeout=10
> git --version # git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/sourabh913/Train-Ticket-Reservation-System.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/sourabh913/Train-Ticket-Reservation-System.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 064c16f8e7b2ac94f3a9ea8f43eb211021db3ce6 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 064c16f8e7b2ac94f3a9ea8f43eb211021db3ce6 # timeout=10
Commit message: "Rename viewprofile.png to viewprofile.png"
> git rev-list --no-walk a9d809558900cfb085aed7bdb4917265d30a67e0 # timeout=10
Finished: SUCCESS

```

## 19) Setting Environment Variables in Jenkins. [Step 16 in Ajay Sir Notes]

We can set the Custom Variables under Build Steps > Execute Shell then click Save

For Example -

name=Sourabh

echo "My name is \${name}"

The screenshot shows the Jenkins configuration interface for a job named 'test1'. Under the 'Build Steps' section, there is an 'Execute shell' step. The command entered is:

```
name=Sourabh
echo "My name is ${name}"
```

Below the command, there is an 'Advanced' dropdown and a 'Post-build Actions' section. At the bottom, there are 'Save' and 'Apply' buttons.

Output of the above build

The screenshot shows the Jenkins console output for build #1. The 'Console Output' tab is selected. The log shows:

```
Started by user Sourabh
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/test1
[test1] $ /bin/sh -xe /tmp/jenkins2631533352821369486.sh
+ name=Sourabh
+ echo My name is Sourabh
My name is Sourabh
Finished: SUCCESS
```

System Variables can be setup by using reference on the below link in Jenkins

## Jenkins URL/env-vars.html

20) Setting Global Environment variables in Jenkins [Step 17 in Ajay Sir Notes]

Global Variables are the variables which can be used in any or all jobs. Also, they do not need to define in each job.

Follow the Steps to setup Global Variables

## Goto Manage Jenkins > System

[+ New Item](#)

[Build History](#)

[Manage Jenkins](#)

[My Views](#)

Build Queue

No builds in the queue.

Build Executor Status

**Manage Jenkins**

Building on the built-in node can be a security issue.

**System Configuration**

**System**

Configure global settings and paths.

Under System > Global Properties and then check Environment Variables. Then Click Save Example

Name > class

Value > echo "My name is \${name}"

Dashboard > Manage Jenkins > **System** >

Without a resource root URL, resources will be served from the Jenkins URL with C

**Global properties**

Disable deferred wipeout on this node ?

Disk Space Monitoring Thresholds

**Environment variables**

List of variables ?

Name	<b>class</b>
Value	<b>echo "My name is \${name}"</b>

Add

Tool Locations

**Metrics**

Access keys ?

Add new access key

**Save** **Apply**

## 21) Parameterized Jobs in Jenkins [Step 18 in Ajay Sir Notes]

In this we can define a value/parameter in job.

Steps on how we can configure parameterized job in Jenkins using example

Configure the job and Under General check the option of “This project is parameterized” then select drop down from Add parameter and we can select from a list of below parameters and then Click Save

Boolean Parameter

Choice Parameter

multi-line string parameter

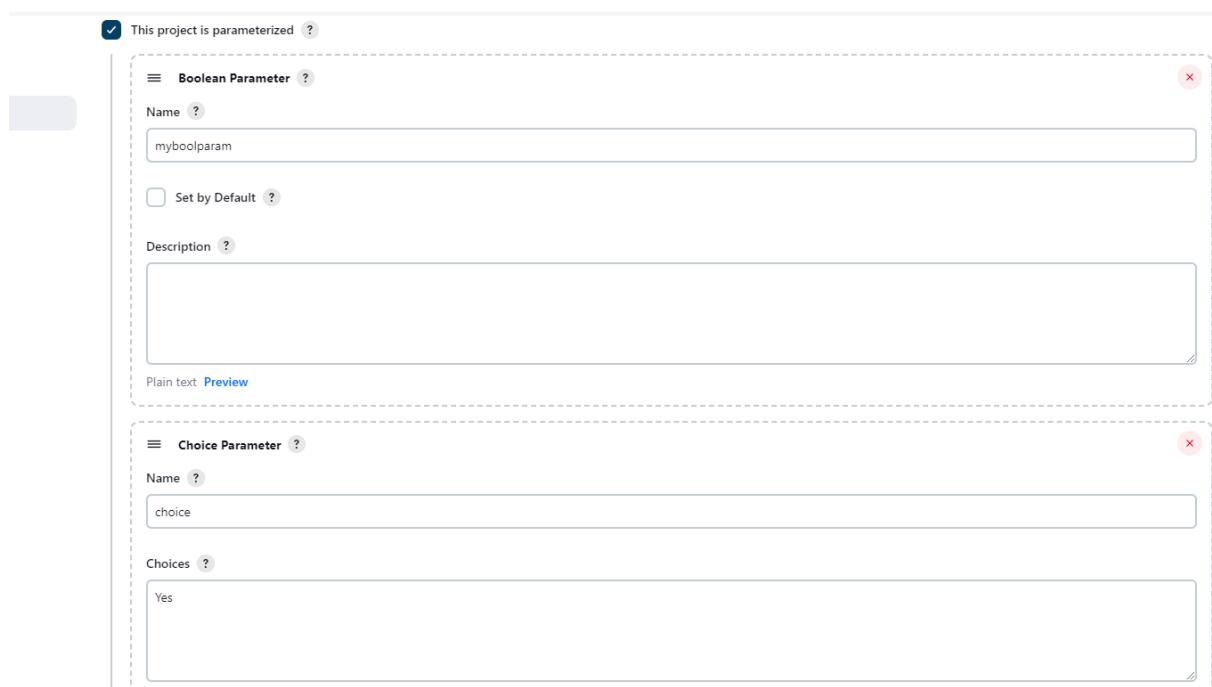
Credentials Parameter

File Parameter

Password Parameter

String Parameter

Run Parameter



## 22) How to create a pipeline in Jenkins [Step 19 in Ajay Sir Notes]

Now we will see, how to create a pipeline in Jenkins

## Steps to create a pipeline

We need to create two jobs at least and create a dependency on each other (Upstream/Downstream)

### Example - Dev, Test, Prod

The screenshot shows the Jenkins dashboard. On the left, there are links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are sections for 'Build Queue' (empty) and 'Build Executor Status' (1 idle, 2 idle). The main area displays a table of jobs:

S	W	Name	Last Success	Last Failure	Last Duration
...	...	Dev	N/A	N/A	N/A
...	...	prod	N/A	N/A	N/A
...	...	test	N/A	N/A	N/A

At the bottom, there are icons for S, M, L and a link to 'Add description'.

## Now Install a plugin build pipeline

The screenshot shows the 'Manage Jenkins > Plugins' page. The left sidebar has links for 'Updates', 'Available plugins' (selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The right panel shows the 'Available plugins' section with a search bar for 'build pi'. A specific plugin is highlighted:

Install	Name	Released
<input checked="" type="checkbox"/>	Build Pipeline 2.0.2	3 mo 5 days ago

Details for the 'Build Pipeline' plugin:

- User Interface
- Build Tools
- Other Post-Build Actions

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.

Warning: This plugin version may not be safe to use. Please review the following security notices:  
• Stored XSS vulnerability

This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

The screenshot shows the 'Manage Jenkins > Plugins' page. The left sidebar has links for 'Updates', 'Available plugins' (selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The right panel lists the 'Installed plugins' with their status:

Plugin Name	Status
Pipeline Graph Analysis	Success
Metrics	Success
Pipeline Graph View	Success
Git	Success
EDDSA API	Success
Trilead API	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Theme Manager	Success
Dark Theme	Success
Loading plugin extensions	Success
Parameterized Trigger	Success
JavaMail API	Success
Oracle Java SE Development Kit Installer	Success
SSH server	Success
Command Agent Launcher	Success
jQuery	Success
Build Pipeline	Success
Loading plugin extensions	Success

set project dependency on test to dev and prod to test

Dashboard > test >

Status      **test**

</> Changes      Upstream Projects

Workspace      Dev

Build Now      Downstream Projects

Configure      prod

Delete Project      Permalinks

Rename

Dashboard > prod >

Status      **prod**

</> Changes      Upstream Projects

Workspace      test

Build Now      Permalinks

Configure

Delete Project

Rename

Now click on + sign on dashboard

Jenkins

Dashboard >

+ New Item

All      +

Build History

Manage Jenkins

## Give Pipeline view a name and select Build Pipeline View

Dashboard > New view

New view

Name: Pipe

Type:

- Build Pipeline View**  
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.
- List View  
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.
- My View  
This view automatically displays all the jobs that the current user has an access to.

Create

Select Initial Job as Dev under Pipeline Flow and Click OK.

Pipeline Flow

Layout

Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only supported layout mode for pipelines.

Upstream / downstream config

Select Initial Job ?

Dev

Trigger Options

Build Cards

Standard build card

Use the default build cards

Restrict triggers to most recent successful builds ?

Yes

No

Always allow manual trigger on pipeline steps ?

Yes

OK Apply

Now will run the Build Pipeline and see the output below

Dashboard > Pipe >

Build Pipeline

Run History Configure Add Step Delete Manage

Pipeline #1

#1 Dev Aug 21, 2014 14:27 PM 5:14 sec Source

#1 test Aug 21, 2014 14:44:47 PM 24 ms

#1 prod Aug 21, 2014 14:44:47 PM 1 ms

### 23) Running job on Jenkins Worker/Slave node [Step 21 in Ajay Sir Notes]

In this, we will setup a Jenkins Master and Worker Nodes and will establish connection between the two and run the job on Worker Node

Steps that needed to be followed

- We need to create tow EC2 instances (Jenkins Master and Jenkins Worker) Jenkins master should have Jenkins installed on it.  
Nothing needs to be done on Worker Node apart from OS update and upgrade and install open-jdk also

	Name ↴	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
□	JenkinsMaster	i-0b28c37e4874489cf	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1e	ec2-52-201-160-152.co...	52.201.160.152
□	JenkinsWorker	i-0883df89b961b19f6	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a> +	us-east-1e	ec2-54-86-43-62.com...	54.86.43.62

- Once the Instance are created, we need to setup password less authentication between Master and Worker. We need to follow the below steps for the same.

Run the commands in Jenkins Master to generate a public and private key.

ssh-keygen

```
root@JenkinsMaster:/home/ubuntu# ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:VJPDTHSsr205HXxI78w/50JzSwxzCNySumk8u8XJGR0 root@JenkinsMaster
The key's randomart image is:
+--[ED25519 256]--+
|      =*o+   |
|      .=B..   |
|     . .ooE.   |
|     . . .+.+..|
|    S oo..=   |
|     *oo=ooo.   |
|     . o0+.+o.   |
|     .*.*.o.   |
|     ooo. ++=|
+---[SHA256]---+
root@JenkinsMaster:/home/ubuntu#
```

Now we need to copy the contents of the public key (e.g. id\_ed25519.pub) from the path /root/.ssh/ to worker node /root/.ssh/authorized\_keys file

```
cd /root/.ssh
cat id_ed25519.pub (copy the contents and append in the worker
machine authorized_keys file)
```

```
root@JenkinsMaster:/home/ubuntu# cd /root/.ssh/
root@JenkinsMaster:~/ssh#
root@JenkinsMaster:~/ssh#
root@JenkinsMaster:~/ssh#
root@JenkinsMaster:~/ssh# ls -ltr
total 16
-rw----- 1 root root 552 Aug 22 07:39 authorized_keys
-rw-r--r-- 1 root root 100 Aug 22 08:12 id_ed25519.pub
-rw----- 1 root root 411 Aug 22 08:12 id_ed25519
-rw----- 1 root root 142 Aug 22 10:01 known_hosts
root@JenkinsMaster:~/ssh# cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAQIGQU9GF6tuLBLGofjUG4u100bGXMKYaTl5Ux+mav7s1 root@JenkinsMaster
root@JenkinsMaster:~/ssh#
root@JenkinsMaster:~/ssh#
root@JenkinsMaster:~/ssh#
root@JenkinsMaster:~/ssh#
```

On worker Node

Cd /root/.ssh

```
root@JenkinsWorker:/home/ubuntu# vi /etc/hosts
root@JenkinsWorker:/home/ubuntu#
root@JenkinsWorker:/home/ubuntu#
root@JenkinsWorker:/home/ubuntu#
root@JenkinsWorker:/home/ubuntu#
root@JenkinsWorker:/home/ubuntu# cd /root/.ssh
root@JenkinsWorker:~/ssh#
root@JenkinsWorker:~/ssh#
root@JenkinsWorker:~/ssh# ls -ltr
total 4
-rw----- 1 root root 552 Aug 22 07:39 authorized_keys
root@JenkinsWorker:~/ssh#
```

cat >> authorized\_keys

add the copied content from master node

ctrl+C

Verify once by doing

cat authorized\_keys (entry is made or not)

```
root@JenkinsWorker:~/ssh# cat >> authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAQIGQU9GF6tuLBLGofjUG4u100bGXMKYaTl5Ux+mav7s1 root@JenkinsMaster
^C
root@JenkinsWorker:~/ssh#
root@JenkinsWorker:~/ssh#
root@JenkinsWorker:~/ssh# cat authorized_keys
no-port-forwarding,no-agent-forwarding,no-X11-forwarding,command="echo 'Please login as the user \"ubuntu\" rather than the user \"root\".';echo;sleep 10;exit 142" ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQAC/wR6ChuFgnuxj0z/aoAv0ARTzOMJEU7iMHRCP01IE5NFVE4hXPQfETnwaC1sonQYruh+s3d3uRHVA8SEZ4zpqrxvZNaGw5k6R6/G2h1lU3EBYjP6kwAuGN92zNy9wdmUqfEL8ANaC-l+K66Iulg4of65bw/vcVmuvja+i6dAztXarShd2A1dJ12zEyLgApJvu63akps8rTUdC7NovXPecsEnISLviQHd/5xweVER5fmFKjvPuod6n9LAp63CczbULth/1kcuj99Wmpa4hw2y8vR3s/v0MG+UnkuhGB97c14p1c4KyNz9hQ/8mRAQvrLb5ng
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAQIGQU9GF6tuLBLGofjUG4u100bGXMKYaTl5Ux+mav7s1 root@JenkinsMaster
root@JenkinsWorker:~/ssh#
```

- Now we will test whether password less ssh is working or not doing ssh from Jenkins Master to Jenkins Worker. Below screenshot shows it worked.

```
root@JenkinsMaster:/home/ubuntu# ssh 54.86.43.62
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Thu Aug 22 10:37:34 UTC 2024

System load:  0.0          Processes:           116
Usage of /:   22.6% of 8.65GB  Users logged in:    1
Memory usage: 25%          IPv4 address for enX0: 172.31.52.78
Swap usage:   0%

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.
root@JenkinsWorker:~#
```

- Now we will configure Jenkins to add the node as Worker Node.

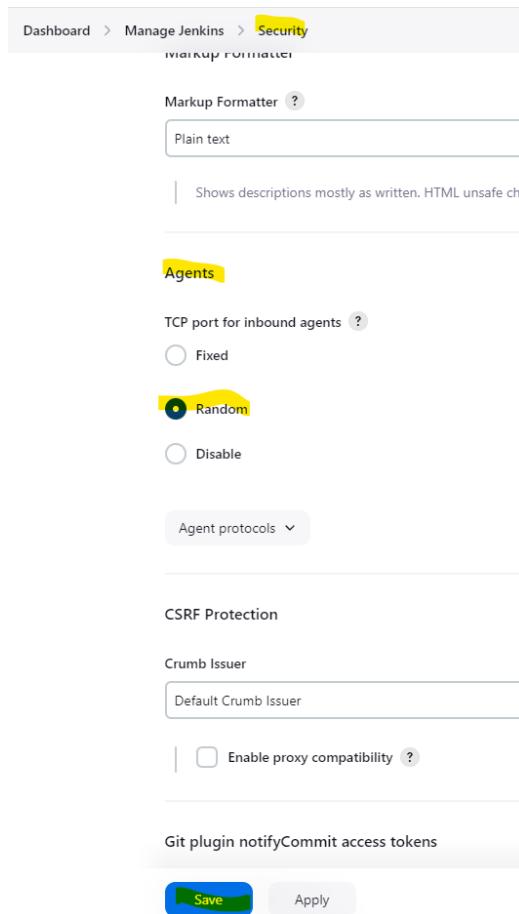
From Jenkins dashboard, go to Manage Jenkins > Security

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links: '+ New Item', 'Build History', 'Manage Jenkins' (which is highlighted with a yellow background), and 'My Views'. The main area is titled 'Manage Jenkins' and contains several sections:

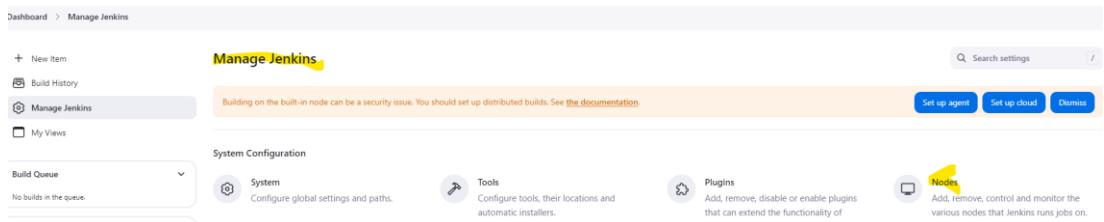
- System Configuration**: Includes 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle).
- Security**: Includes 'System' (Configure global settings and paths), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), and 'Security' (Secure Jenkins; define who is allowed to access/use the system).

A yellow box highlights the 'Security' link under System Configuration, indicating it's the active section.

- Under Agents change the TCP port for inbound agents to Random and click Save



- Now go to Manage Jenkins > Nodes.



- Click on + New Node.



- Give a Name to the node e.g. Worker and select permanent agent and click create.

The screenshot shows the Jenkins 'New node' configuration page. At the top, there's a navigation bar with 'Dashboard > Manage Jenkins > Nodes > New node'. The main title is 'New node'. Below it, there's a 'Node name' field containing 'Worker'. Under 'Type', there's a radio button for 'Permanent Agent' which is selected. A tooltip for this type says: '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.' At the bottom is a blue 'Create' button.

- Now give the Remote root directory (/var/lib/Jenkins) and under Labels we can give any name e.g. Worker and click Save.

The screenshot shows the Jenkins 'Nodes' configuration page for the 'Worker' node. The 'Description' field is empty. The 'Number of executors' is set to 1. The 'Remote root directory' is set to '/var/lib/jenkins'. The 'Labels' field contains 'Worker'. Under 'Usage', it says 'Use this node as much as possible'. Under 'Launch method', it says 'Launch agent by connecting it to the controller'. Under 'Availability', it says 'Keep this agent online as much as possible'. At the bottom is a blue 'Save' button.

- Now you will see the Node showing added in Nodes dashboard.

Nodes							
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.65 GB	0 B	5.65 GB	0ms
	Worker	Linux (amd64)	N/A	N/A	N/A	N/A	N/A
		Data obtained	4 ms	2 ms	0.12 sec	2 ms	0.12 sec
							2 ms

- Now we will click on Worker node.

Agent Worker							
Status		Run from agent command line: (Unix) ⓘ					
	Delete Agent						
	Configure						
	Build History						
	Load Statistics						

- We will copy the commands in the screenshot and run them on the worker node ubuntu terminal.

Agent Worker							
Status		Run from agent command line: (Unix) ⓘ					
	Delete Agent						
	Configure						
	Build History						
	Load Statistics						

- The commands are run on the worker ubuntu terminal successfully. In the below output it shows connected, which confirms that worker is online.

```
root@jenkinsWorker:~# curl -sU http://52.201.160.152:8080/jnlpJars/agent.jar
root@jenkinsWorker:~#
root@jenkinsWorker:~#
root@jenkinsWorker:~# curl -sU http://52.201.160.152:8080/jnlpJars/agent.jar -secret 63f01fdb2f17b98a8ef1455153c8b8cfab1a491fb54ee3b8725c576f0677d -name Worker -workDir "/var/lib/jenkins"
Aug 22, 2024 11:19:47 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /var/lib/jenkins/remoting as a remoting work directory
Aug 22, 2024 11:19:48 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /var/lib/jenkins/remoting
Aug 22, 2024 11:19:48 AM hudson.remoting.Launcher createEngine
INFO: Creating engine for worker
Aug 22, 2024 11:19:48 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3248.3258.v3277a.8e8c9b
Aug 22, 2024 11:19:48 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /var/lib/jenkins/remoting as a remoting work directory
Aug 22, 2024 11:19:48 AM hudson.remoting.Launcher$CullListener status
INFO: Locating servers among [http://52.201.160.152:8080]
Aug 22, 2024 11:19:48 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remote server accepts the following protocols: [JNLP4-connect, Ping]
Aug 22, 2024 11:19:48 AM hudson.remoting.Launcher$CullListener status
INFO: Agent discovery successful
Agent address: 52.201.160.152
Agent port: 35273
Identity: e6:f6:44:99:85:a6:ae:1c:5c:77:50:16:47:a8:05:19
Aug 22, 2024 11:19:48 AM hudson.remoting.Launcher$CullListener status
INFO: Remotely making connection to 52.201.160.152:35273
Aug 22, 2024 11:19:48 AM hudson.remoting.Launcher$CullListener status
INFO: Connecting to 52.201.160.152:35273
Aug 22, 2024 11:19:48 AM hudson.remoting.Launcher$CullListener status
INFO: Server reports protocol JNLP4-connect-proxy not supported, skipping
Aug 22, 2024 11:19:48 AM hudson.remoting.Launcher$CullListener status
INFO: Remotely pinged
Aug 22, 2024 11:19:49 AM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start
Aug 22, 2024 11:19:49 AM hudson.remoting.Launcher$CullListener status
INFO: Remote identity confirmed: e6:f6:44:99:85:a6:ae:1c:5c:77:50:16:47:a8:05:19
Aug 22, 2024 11:19:49 AM hudson.remoting.Launcher$CullListener status
INFO: Connected
```

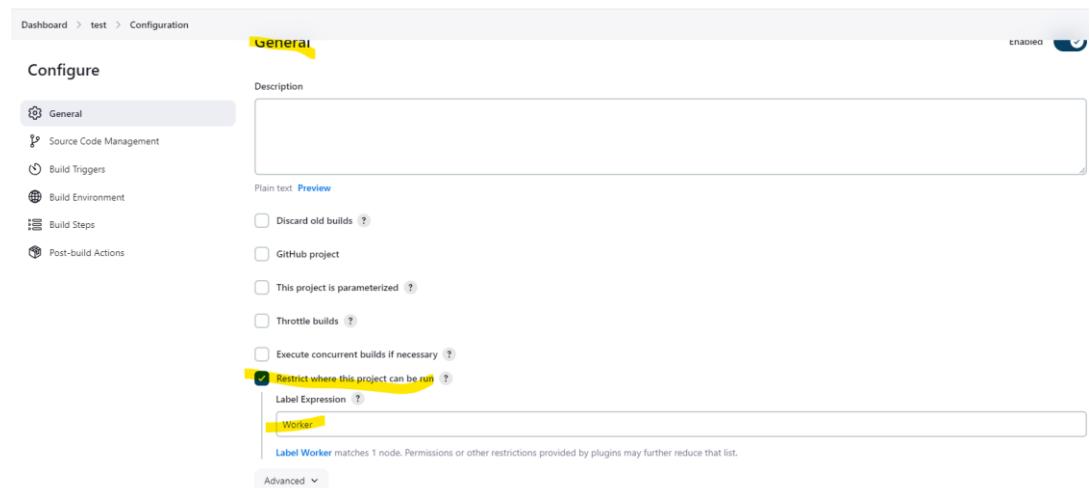
- We can see in the node dashboard that Worker is Online now as a node.

Nodes							
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.65 GB	0 B	5.65 GB	0ms
	Worker	Linux (amd64)	In sync	4.08 GB	0 B	4.08 GB	7ms
		Data obtained	3.5 sec	3.5 sec	3.4 sec	3.4 sec	3.4 sec

- Now we will run the job on the worker node, for that we will create a freestyle project and do the following configuration.

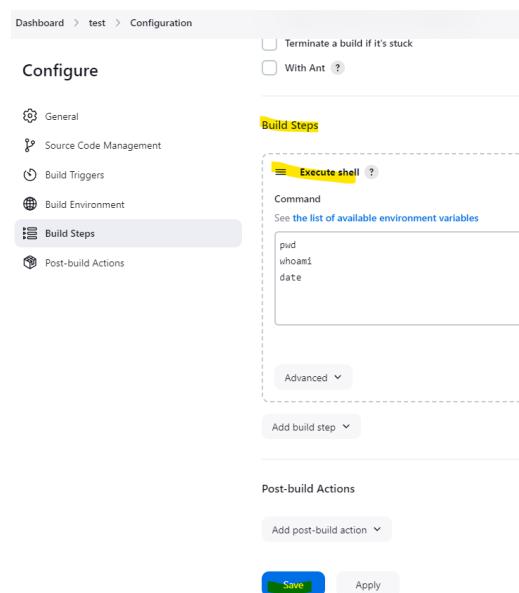
While configuring, we will select Restriction where this project can be run under General tab.

Then we will select our worker node from the Label Expression e.g. in our case its Worker



Then we give some commands which we want to run as an example on the worker node under Build Steps > Execute shell

Then Click Save



- Once we run Build Now, it should us the build number and console output should confirm that it ran on the worker node.

The screenshot shows the Jenkins interface for a project named "test".

**Dashboard > test >**

**Status** (highlighted) | **test** (green checkmark)

**Permalinks**

- Last build (#1), 9 min 33 sec ago
- Last stable build (#1), 9 min 33 sec ago
- Last successful build (#1), 9 min 33 sec ago
- Last completed build (#1), 9 min 33 sec ago

**Actions:**

- </> Changes
- Workspace
- Build Now (highlighted)
- Configure
- Delete Project
- Rename

**Build History**

Build #1 (Aug 22, 2024, 11:19 AM)

Atom feed for all | Atom feed for failures

**Console Output**

```

Started by user Sourabh
Running as SYSTEM
Building remotely on Worker in workspace /var/lib/jenkins/workspace/test
[test] $ /bin/sh -xe /tmp/jenkins11611333456846808263.sh
+ pwd
/var/lib/jenkins/workspace/test
+ whoami
root
+ date
Thu Aug 22 11:19:55 UTC 2024
Finished: SUCCESS
  
```

- The console output confirms that the build ran on Worker Node.