# Jenkins Notes

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

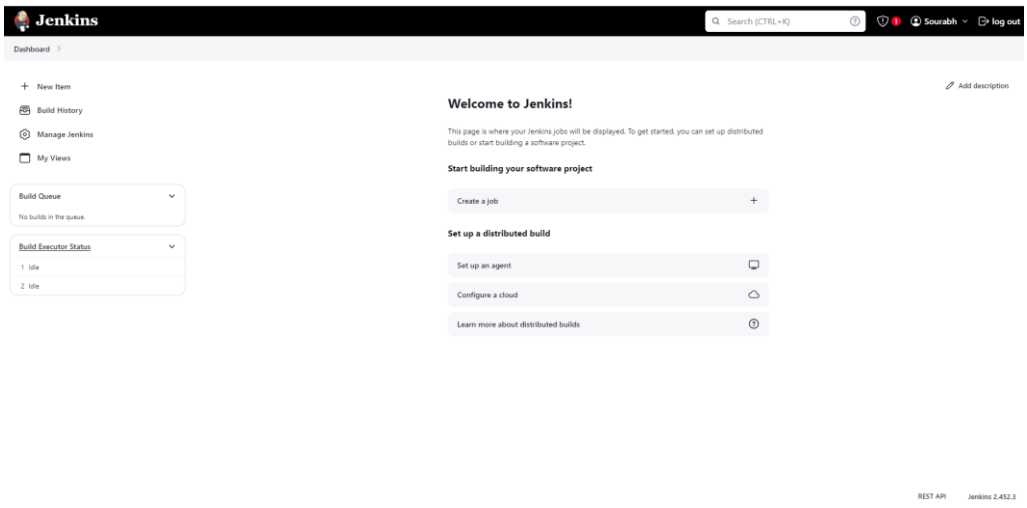
   echo "public ip_address" "jenkins_local" >> /etc/hosts

```
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

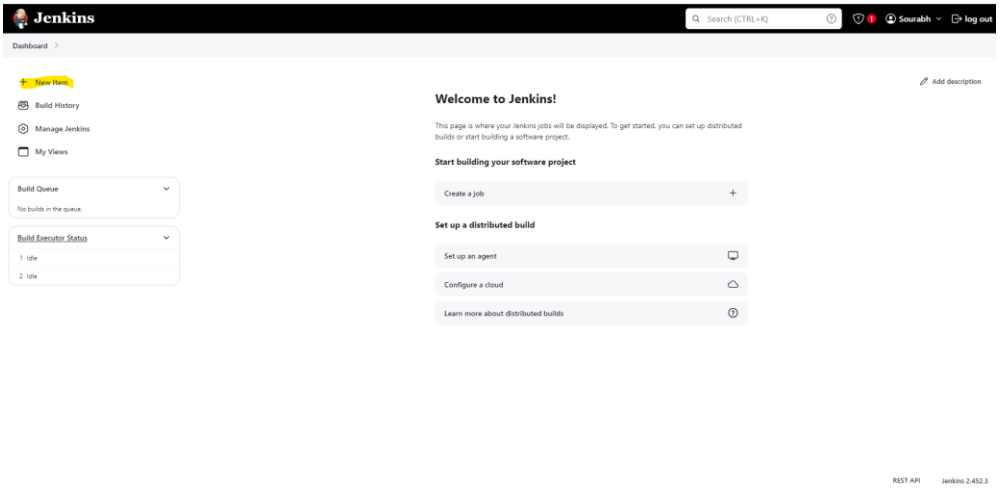
```
root@ip-172-31-94-84:/home/ubuntu# cat /etc/hosts
127.0.0.1 localhost

# 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]**

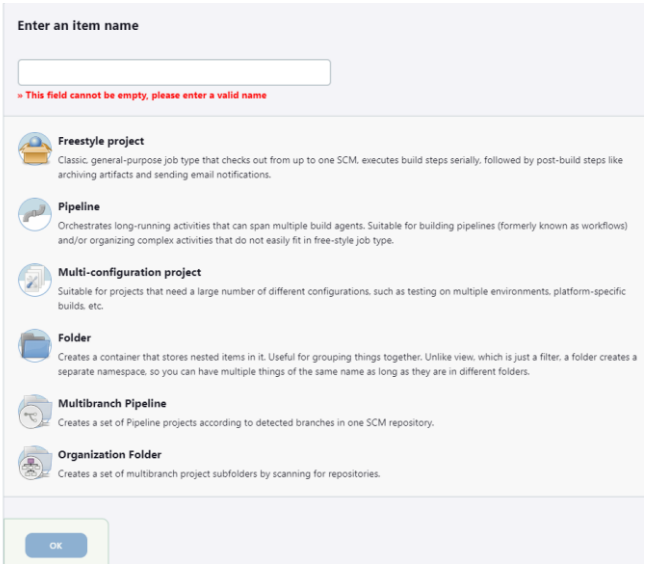



3) Create first Jenkins job **[Step 4 in Ajay Sir Notes]**

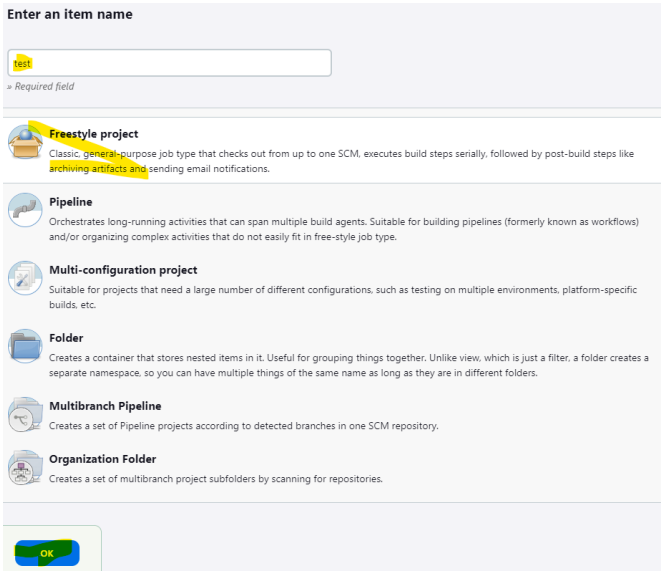
   Click on New Item

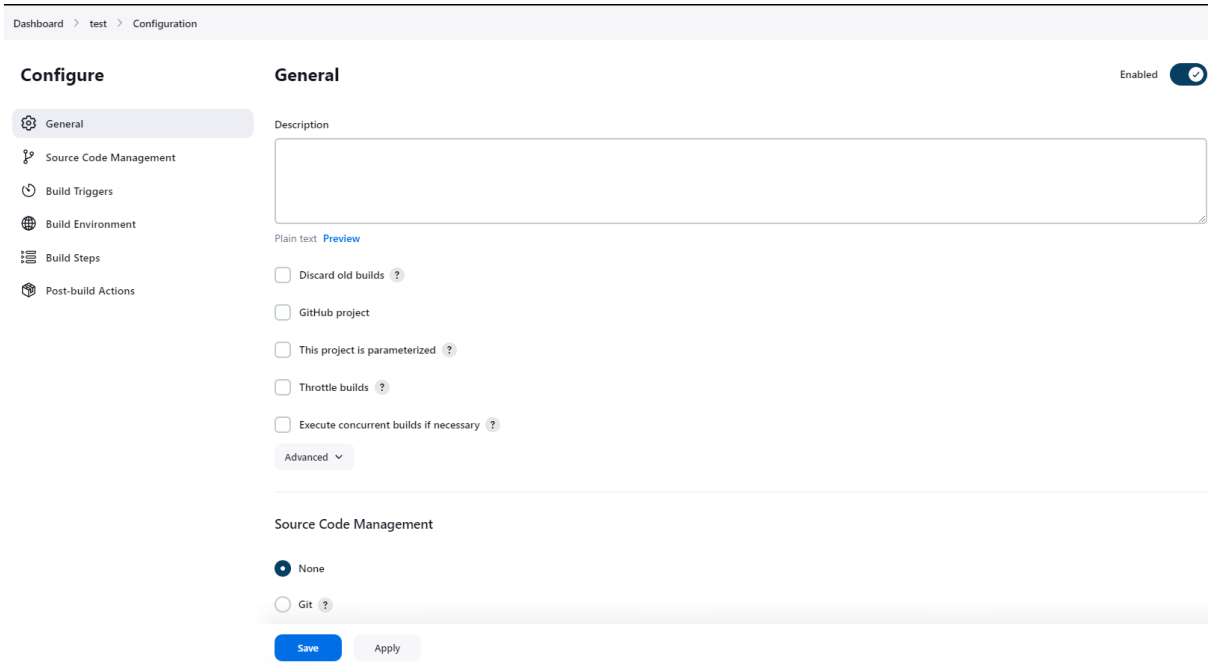


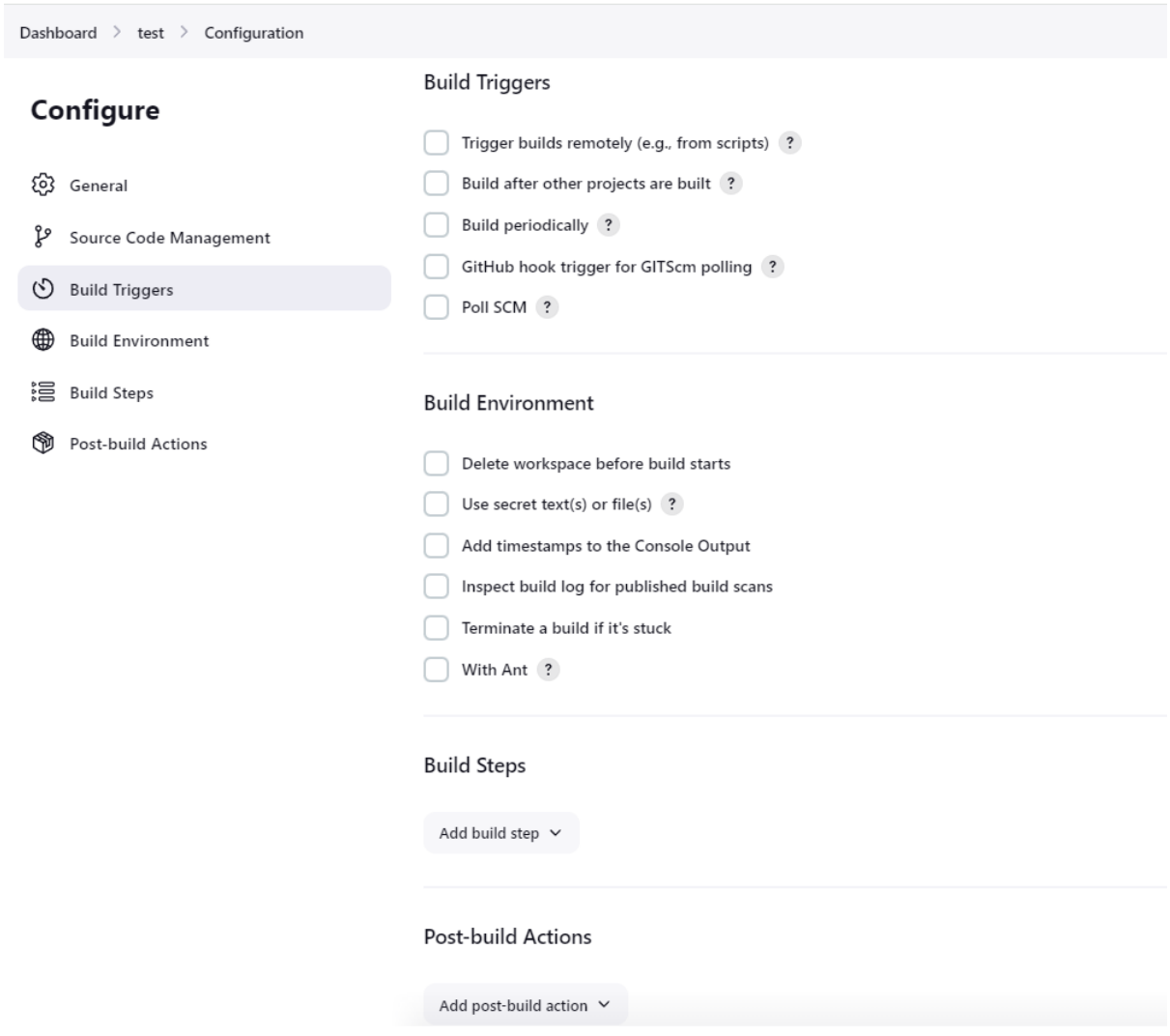

There are multiple types of jobs and their details are mentioned along with the jobs



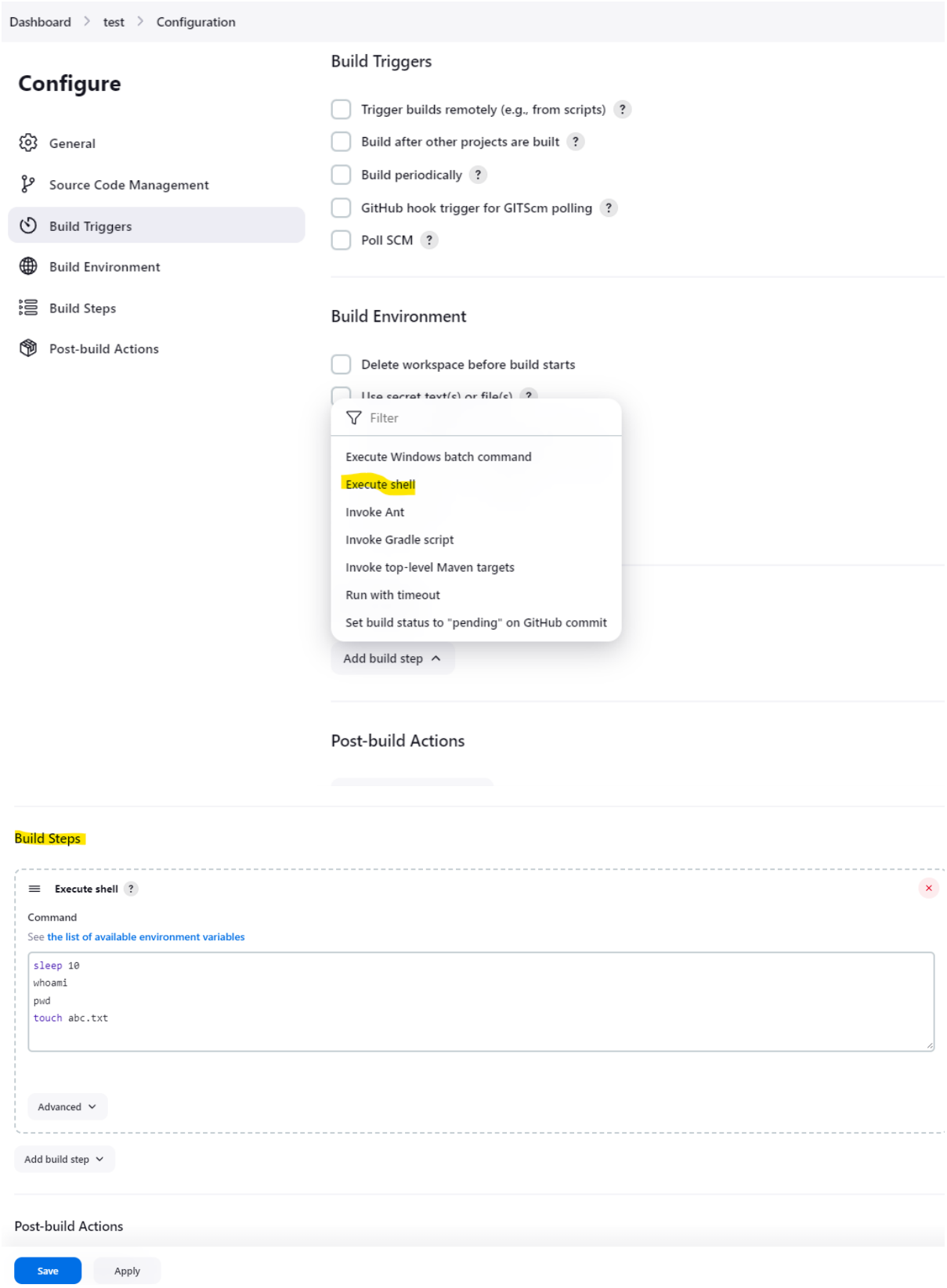

Enter the item name and select Freestyle project and hit OK



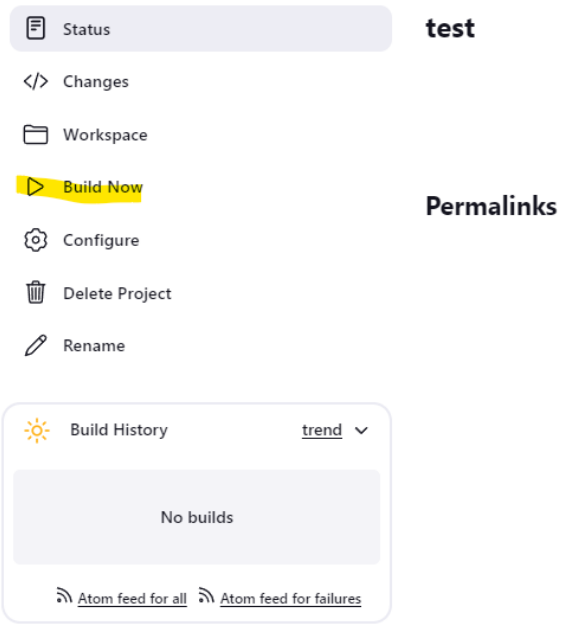

Dashboard > test > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
General
Enabled
Description
Plain text Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Advanced
Source Code Management
None
Git
Save
Apply


Dashboard > test > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
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

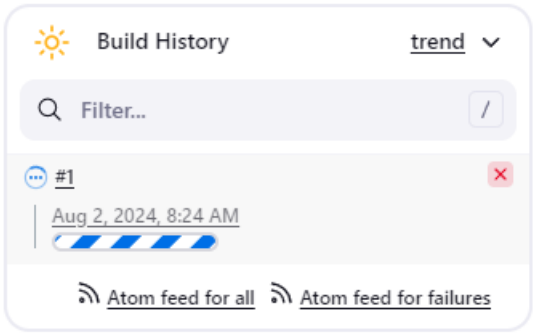
Add build Steps and select Execute Shell, Write the commands and click save



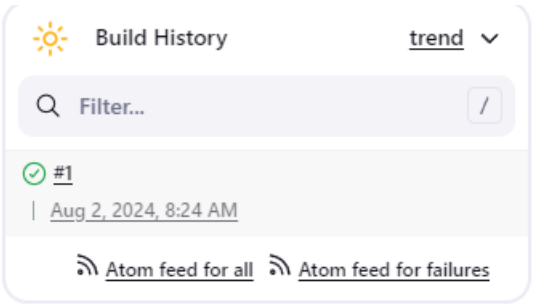

Click on build now to the Job



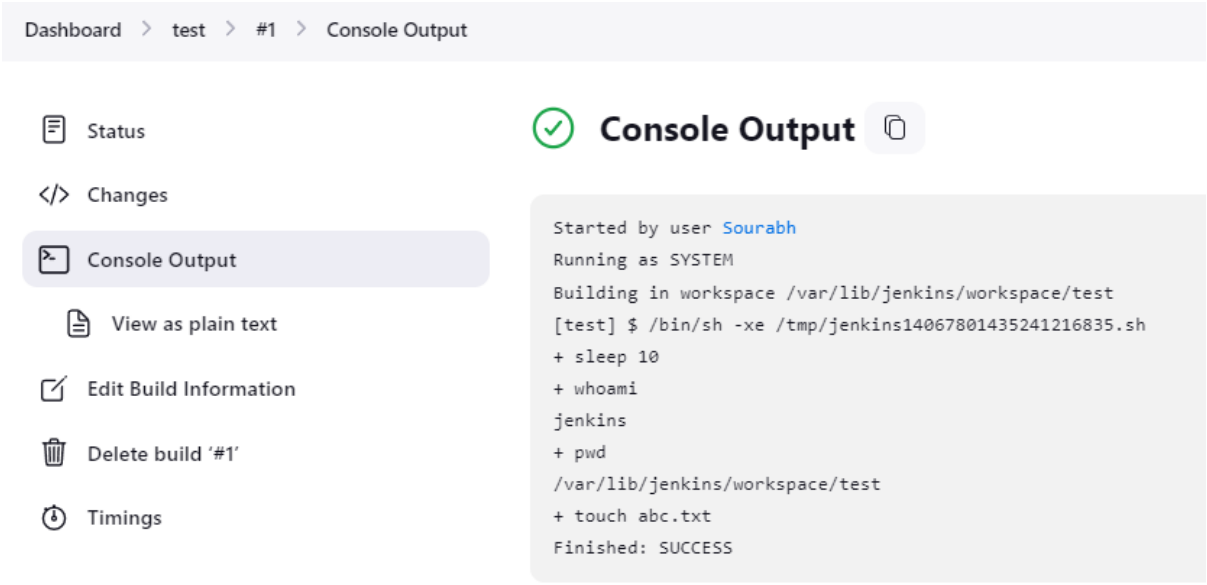

Build history will show you the Jobs that ran and are running




Below screenshot shows job completed successfully



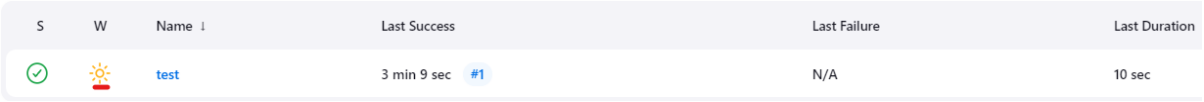

Click on the number on the above screenshot to show job details and console output to show the output of the job



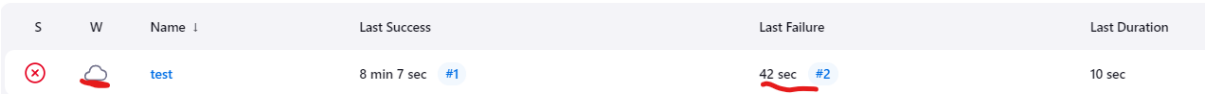

/var/lib/Jenkins/workspace/test – Job Location where it executes build.

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

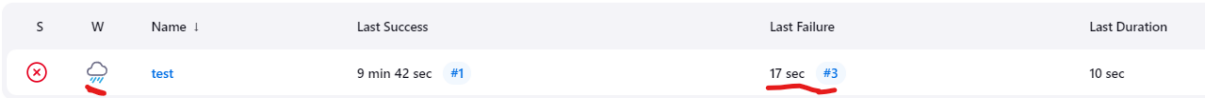
- Sunshine – Job Success so far with no failures



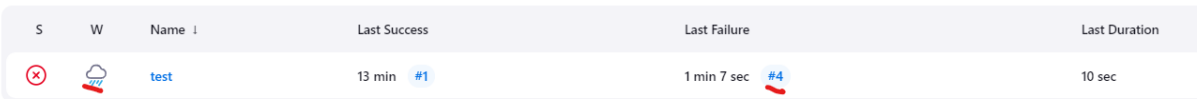

- Cloud – Job Failed once, shows time in last failure and also tells the build number on which it failed.



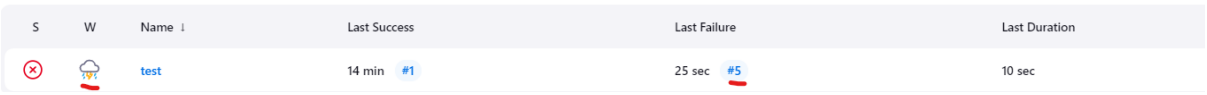

- Shower – Job failed twice, shows time in last failure and also tells the build number on which it failed.



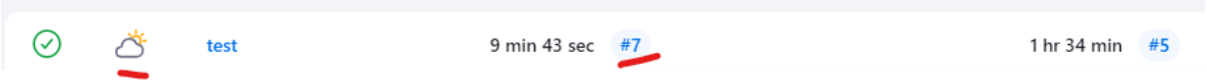

- Rain - Job failed thrice, shows time in last failure and also tells the build number on which it failed.




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

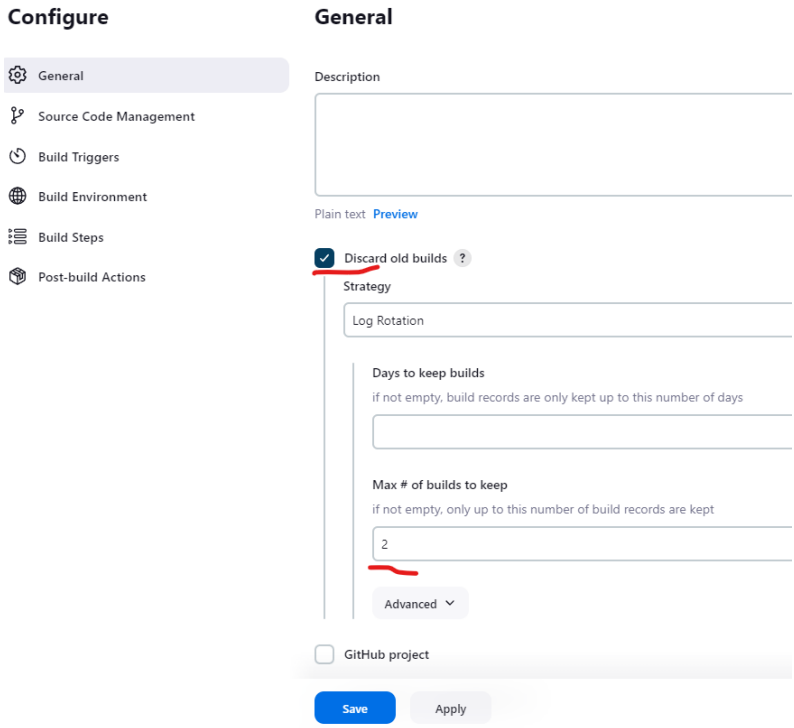



- Partly Cloudy – Status success after multiple fails, change in Last Success

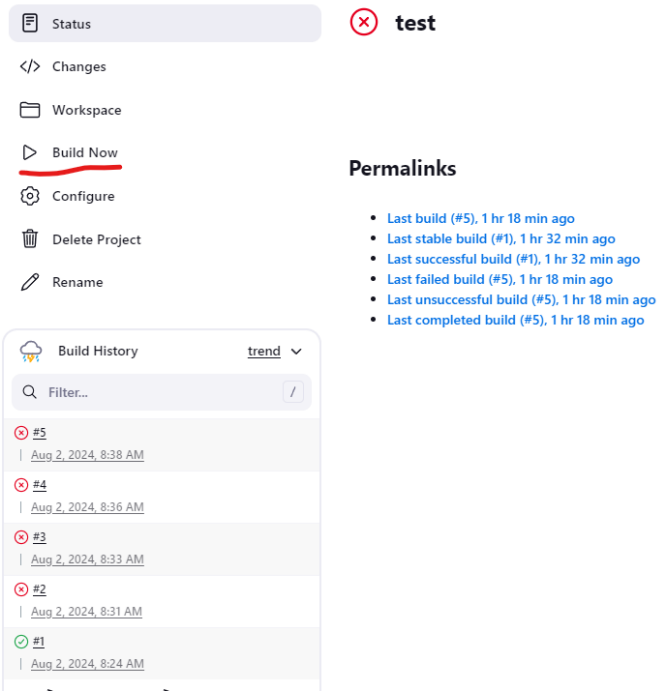



This is maximum time the weather icon changes and once the build starts succeeding, icon changes back to Sunshine after multiple success.

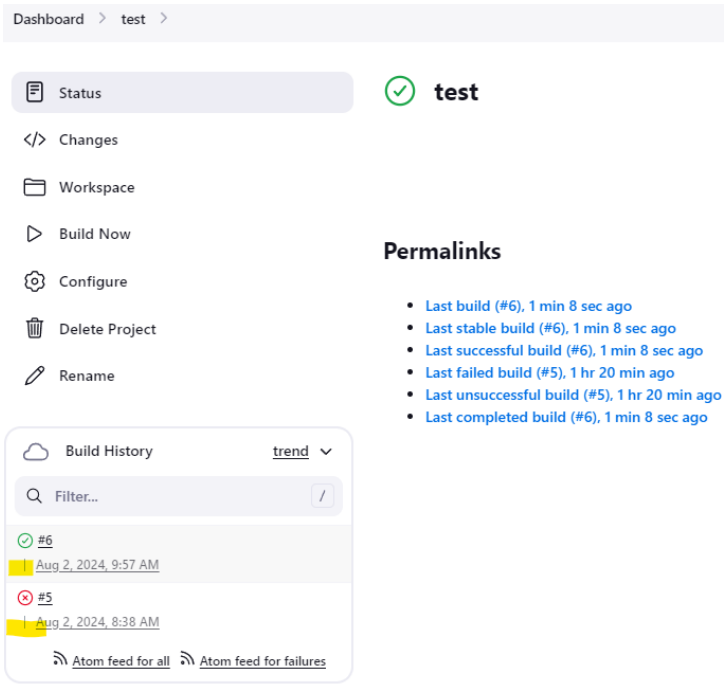
5) If we want to remove old builds and keep only specific number of builds. Click configure on the job and check discard old builds and give the Max # of builds to keep mentioned in the screenshot and click save



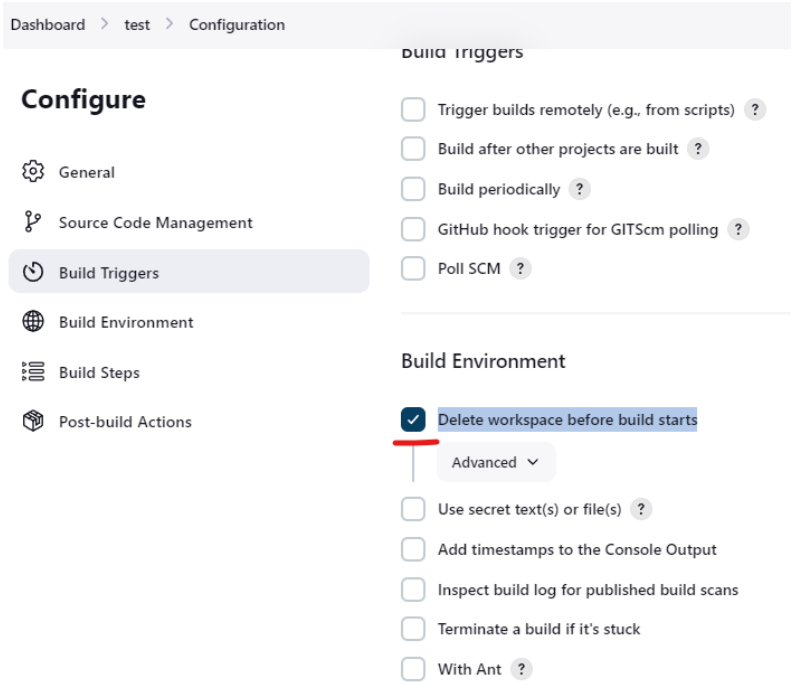

Click Build Now



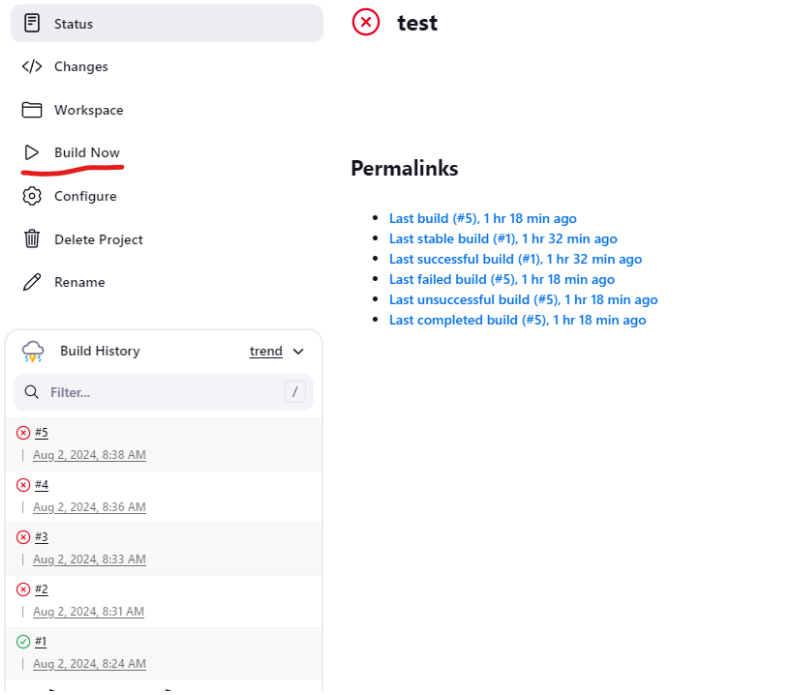

Once the build is completed, see only last 2 builds are left rest are deleted. See screenshot below



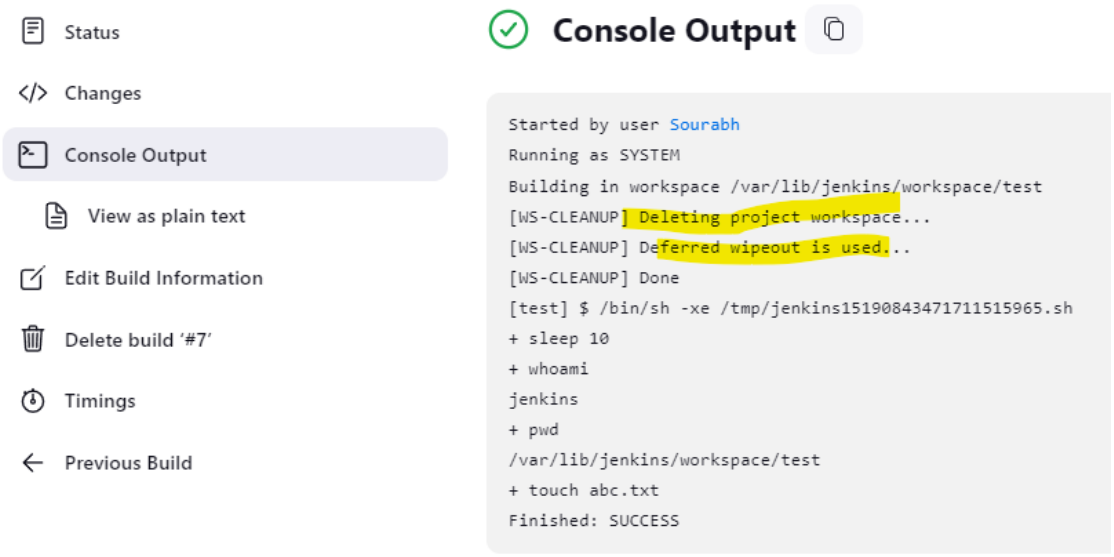

6) Now if we want to delete the content of the workspace, Click configure on the job. Under Build Environment, Check the option Delete workspace before build starts and hit save.



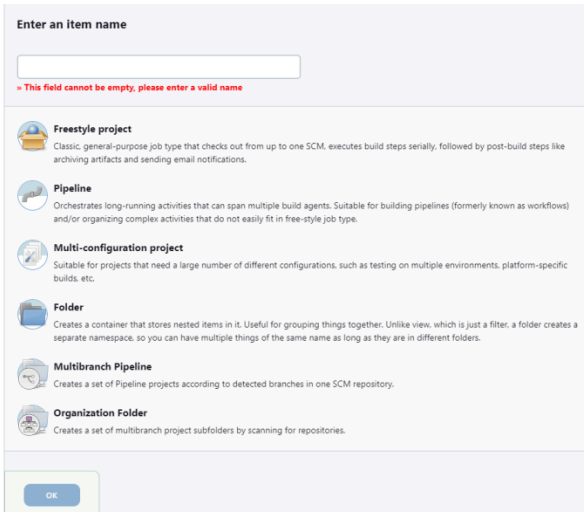

Click Build Now



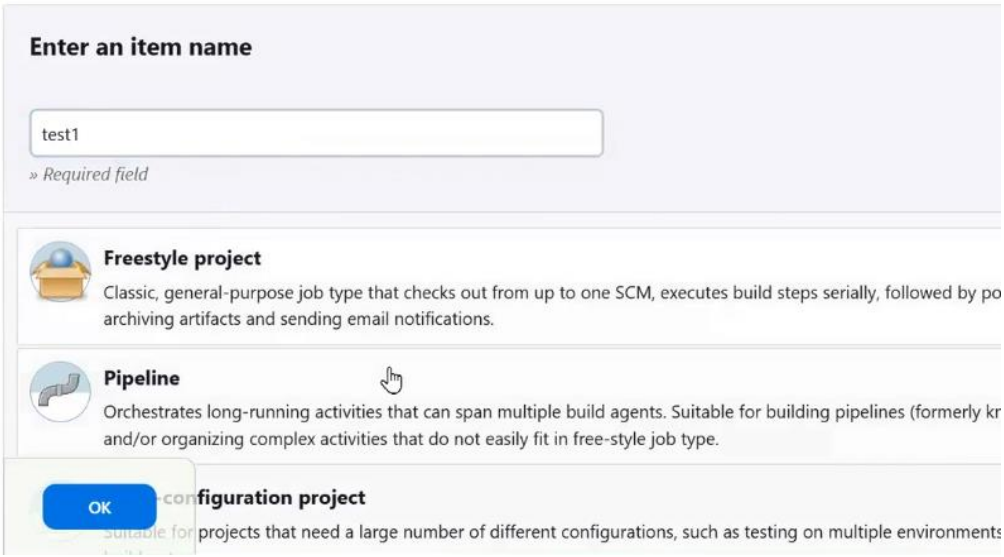

See all the contents are deleted under workspace of job test

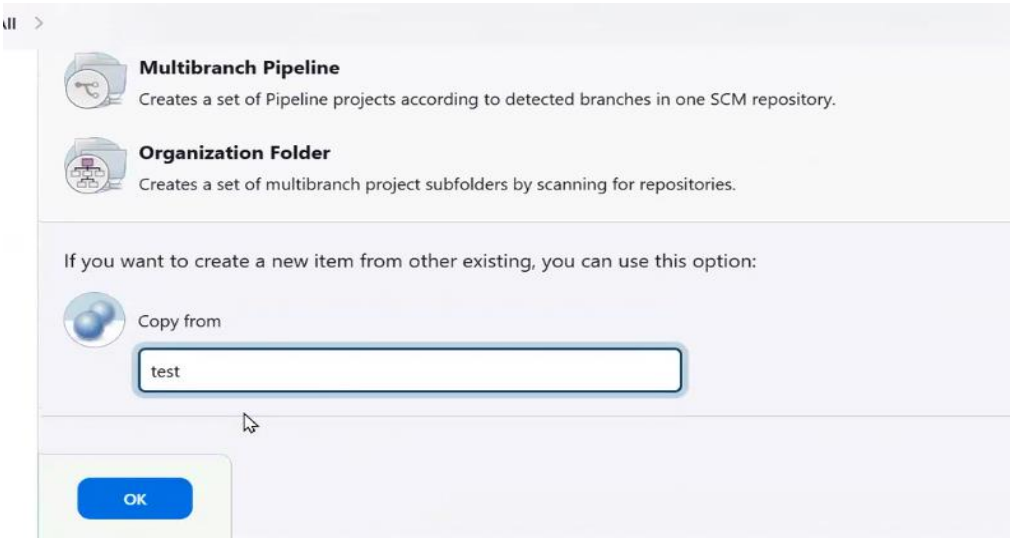



7) Create a new job and copy the contents from old eg test to test1
Click on New item

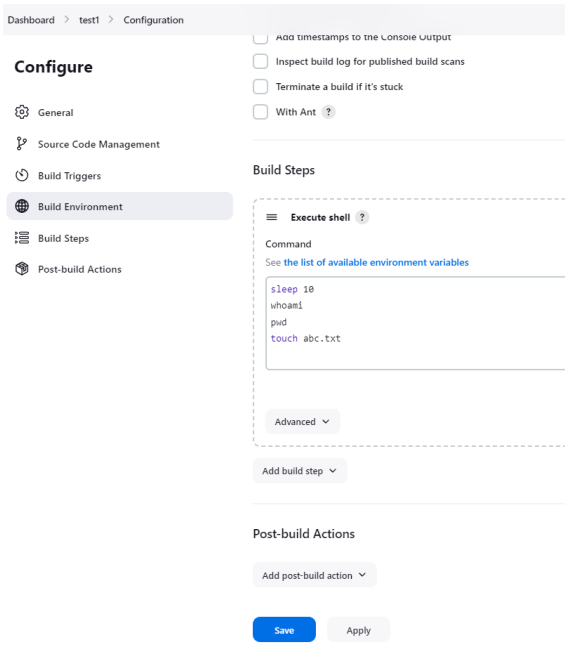



Enter the item name (test1) and select Freestyle project and scroll down to copy from and enter the job name where you want to copy contents from eg in our case copy contents from test and click ok



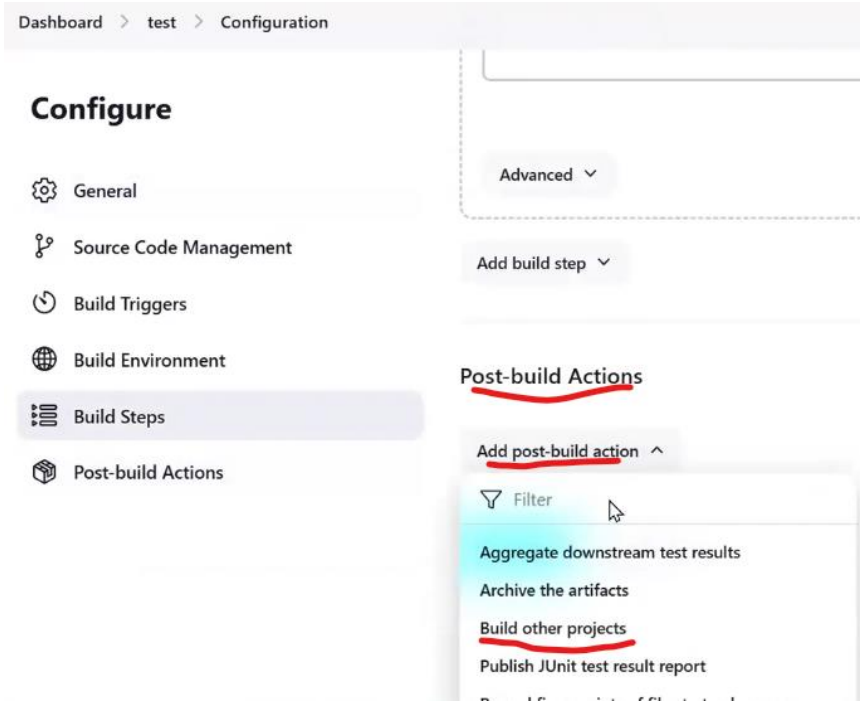




All contents are copied from test

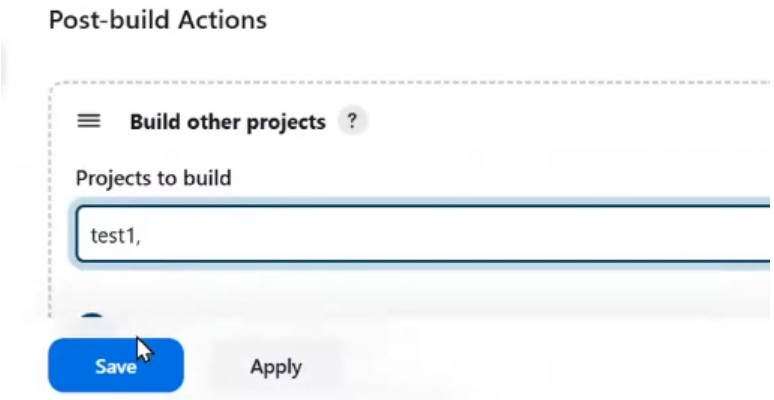


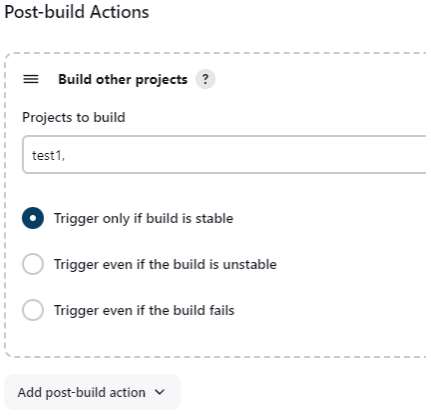
**8)** Defining second job to be triggered if the first job succeeds (Downstream) **[Step 12 in Ajay Sir Notes]**

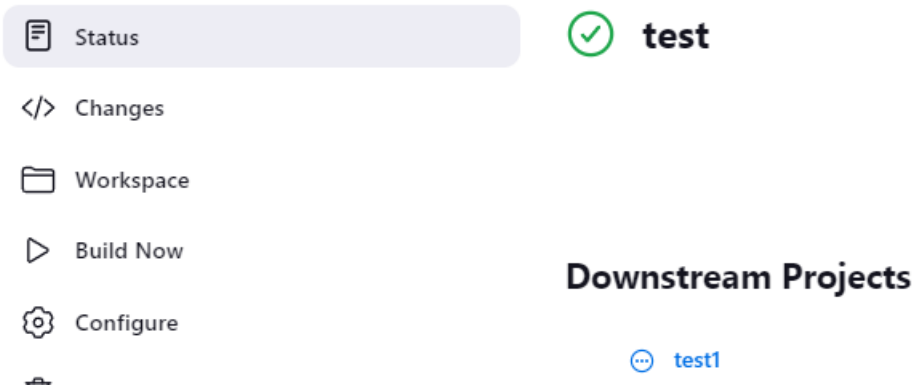
Click configure on the job. Go to post build actions and select build from other projects



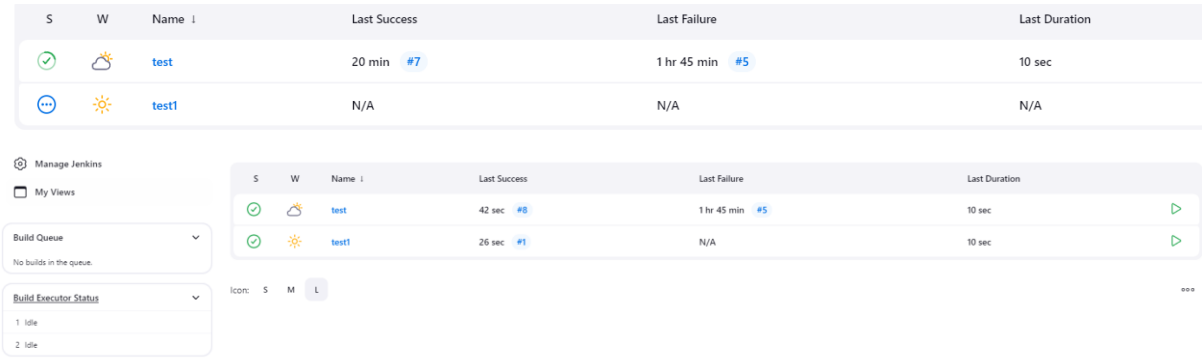

Then under Build other projects. Type the second project i.e. test1 and keep trigger only if build is stable. and click save.





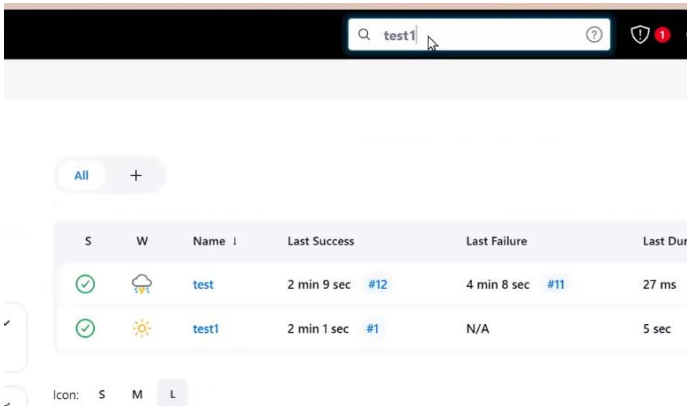

After Build now, both jobs are success in below screenshot

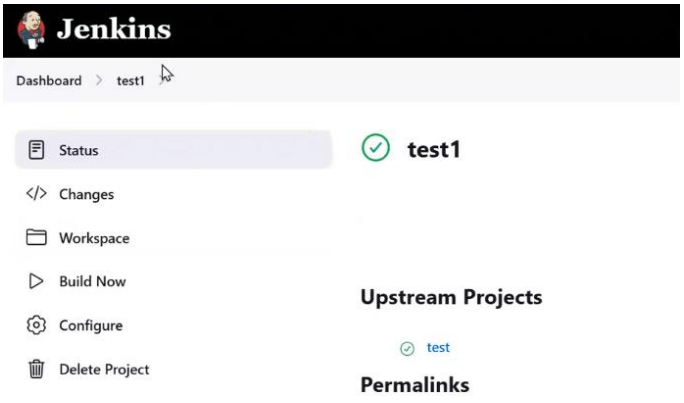



NOTE: There are two other conditions for build another project

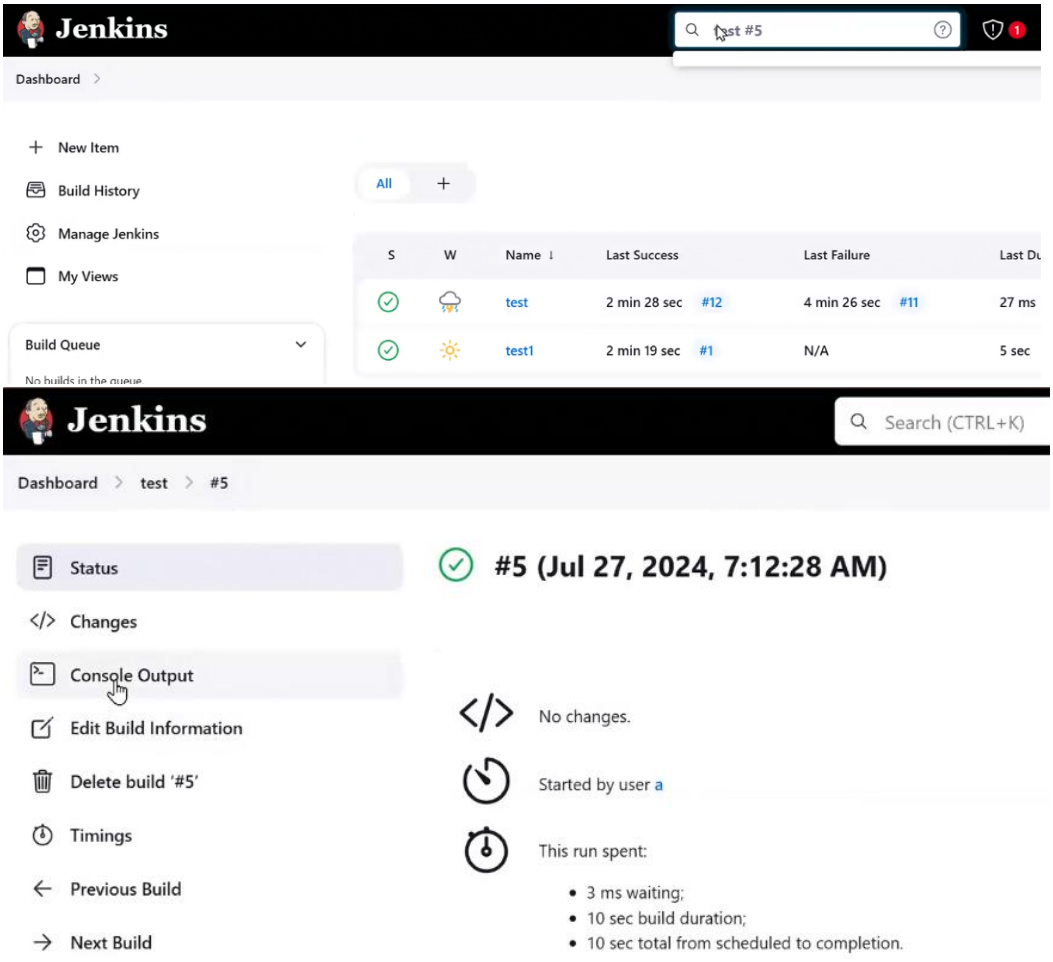
- Trigger even if the build is unstable – If this condition is selected, then the second project is executed even if there are some warnings or errors in the upstream job/project
- Trigger even if the build fails. – If this condition is selected, then the second project is executed even if the upstream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






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




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

Manage Jenkins- For admin tasks

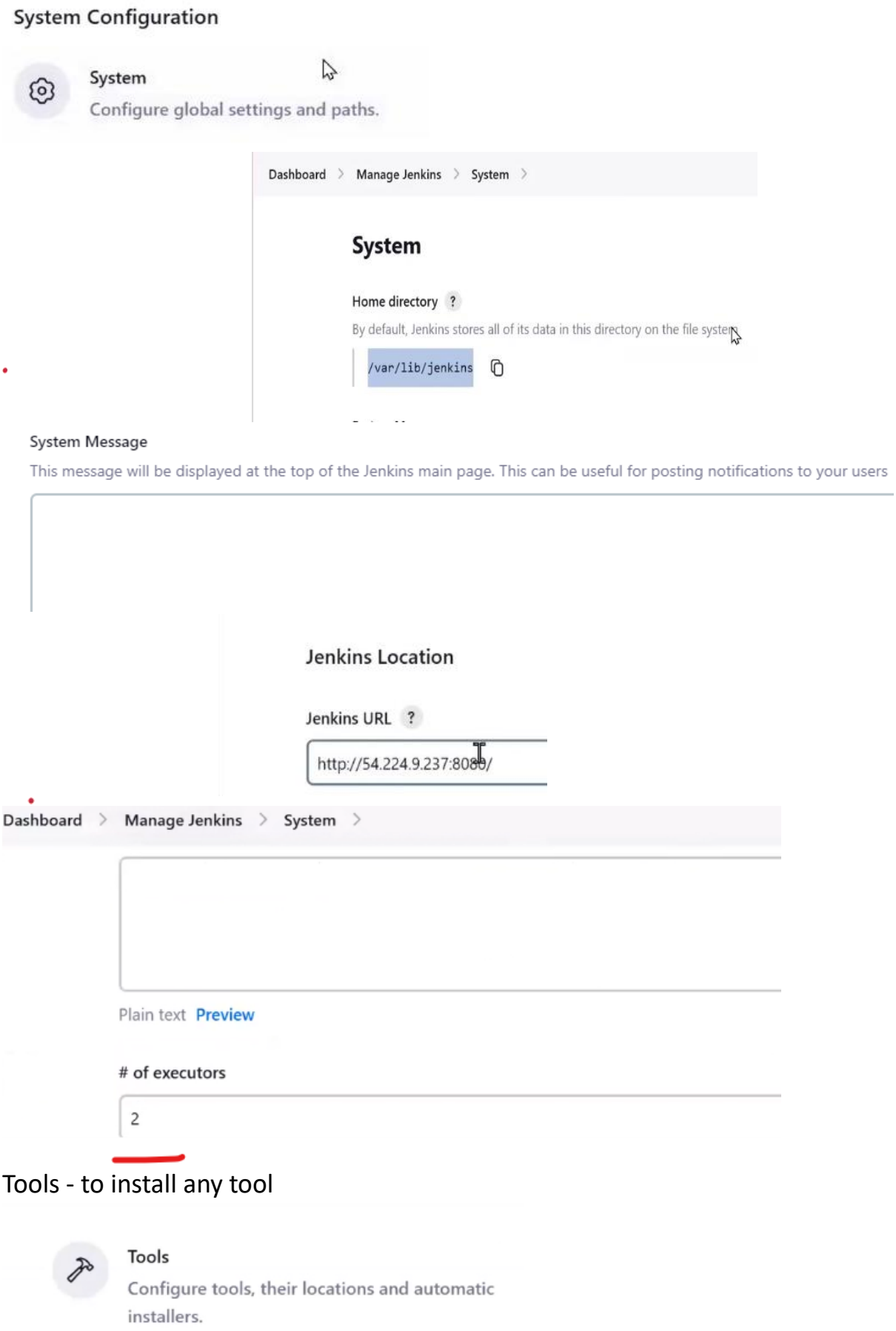
System- To change any configuration of plugins which you have installed

Home directory path

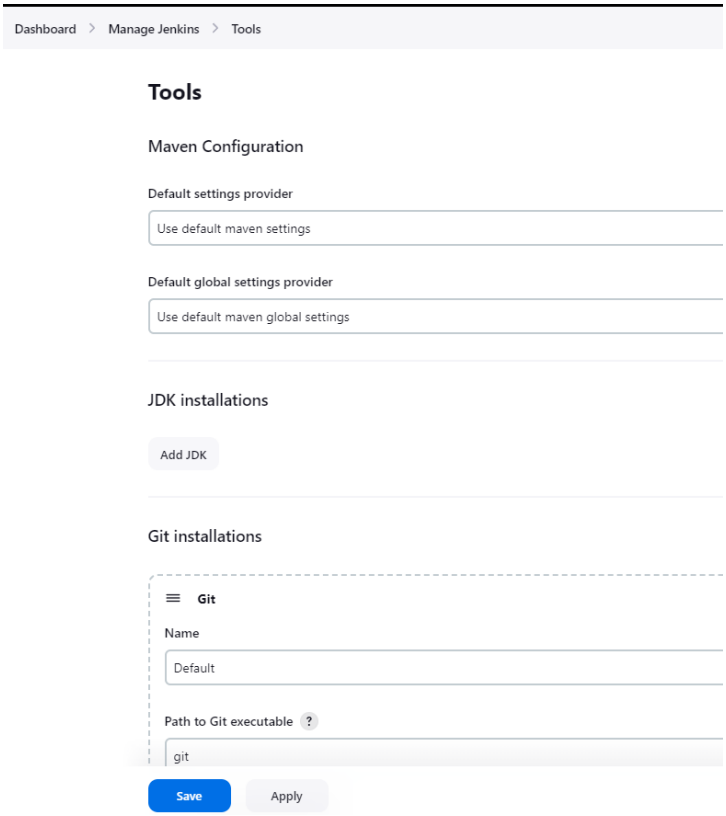
System Message

Jenkins URL

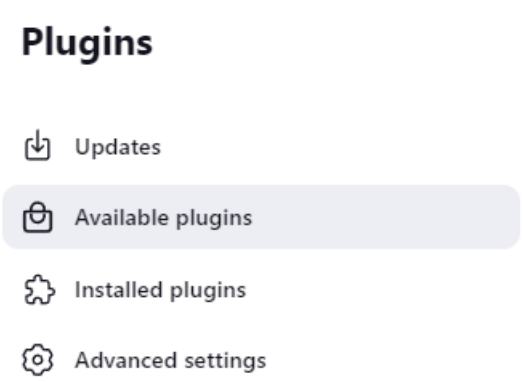
No. of executor- no of job execution together



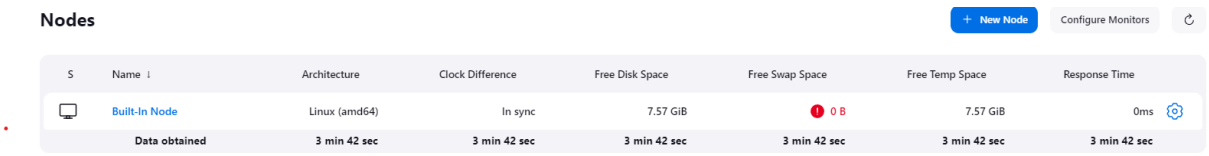

Tools - to install any tool



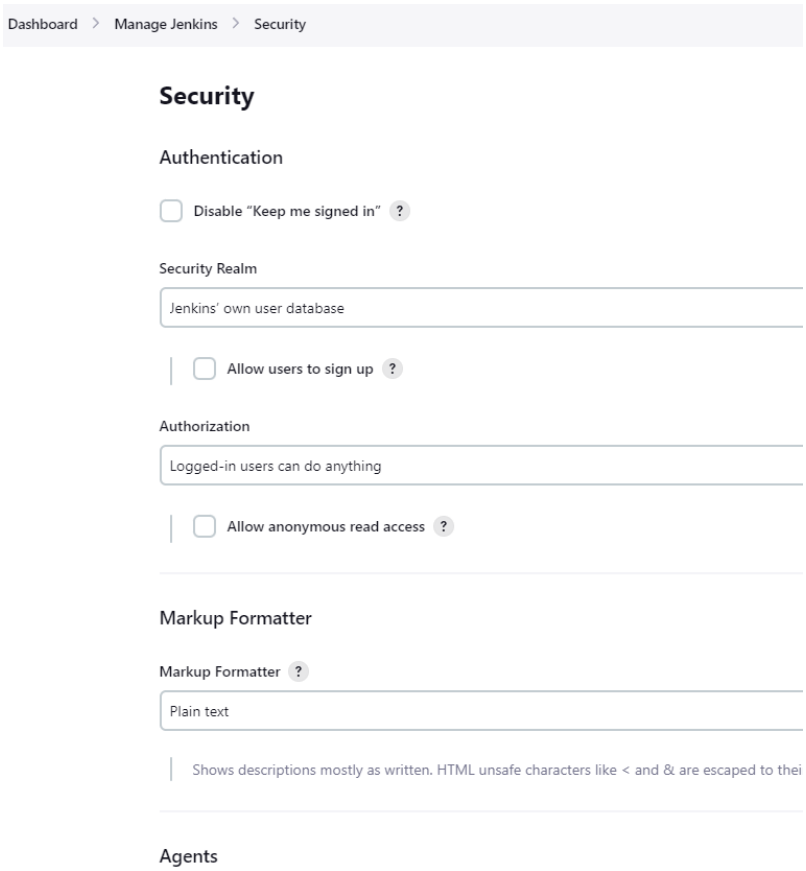

Manage plugins



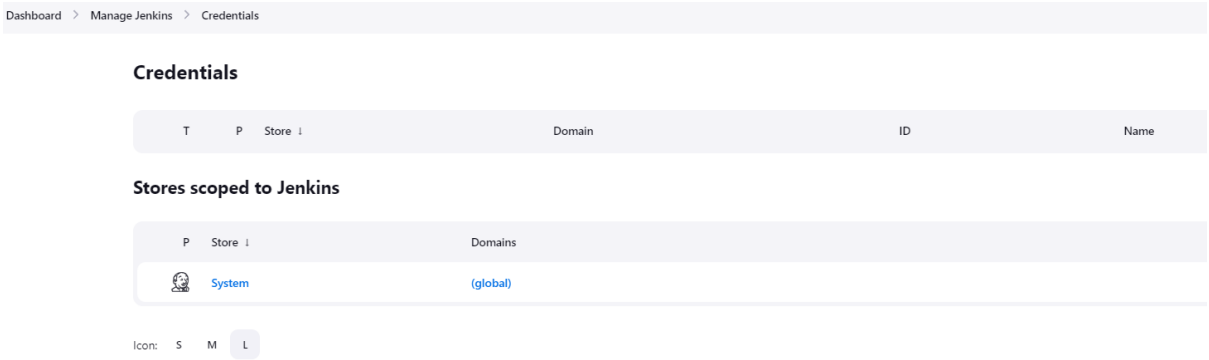

Nodes



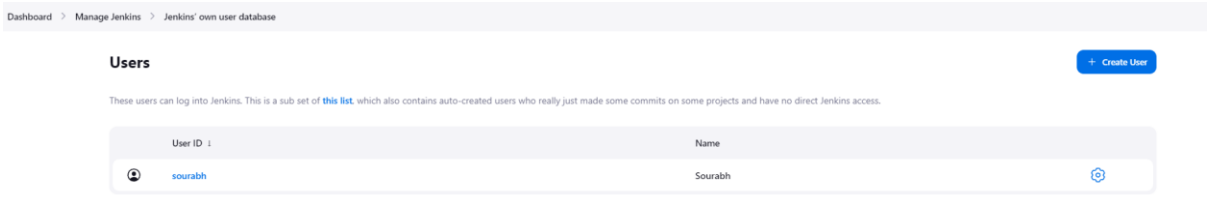

Security




## Credentials




## Users




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

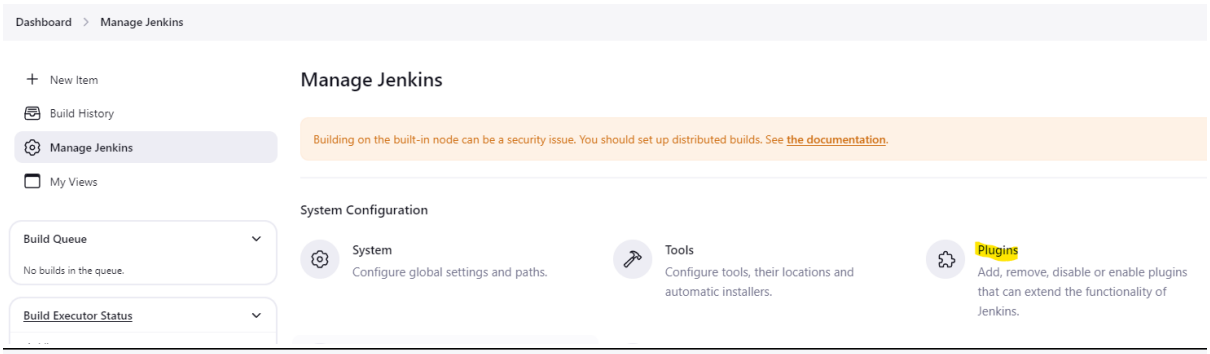
Goto to the link https://github.com/afonsof/jenkins-material-theme

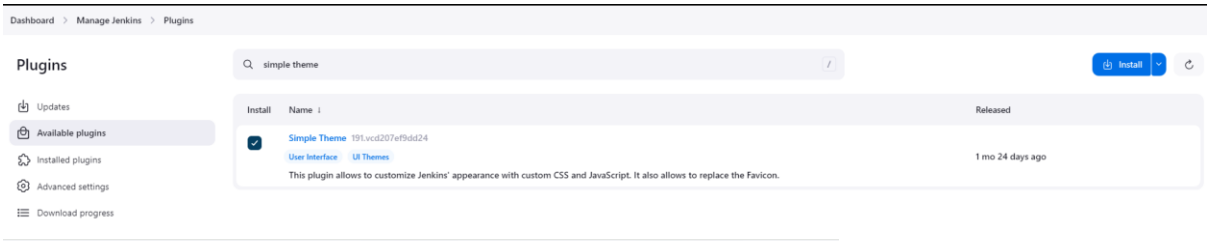
**Follow the steps GitHub page**

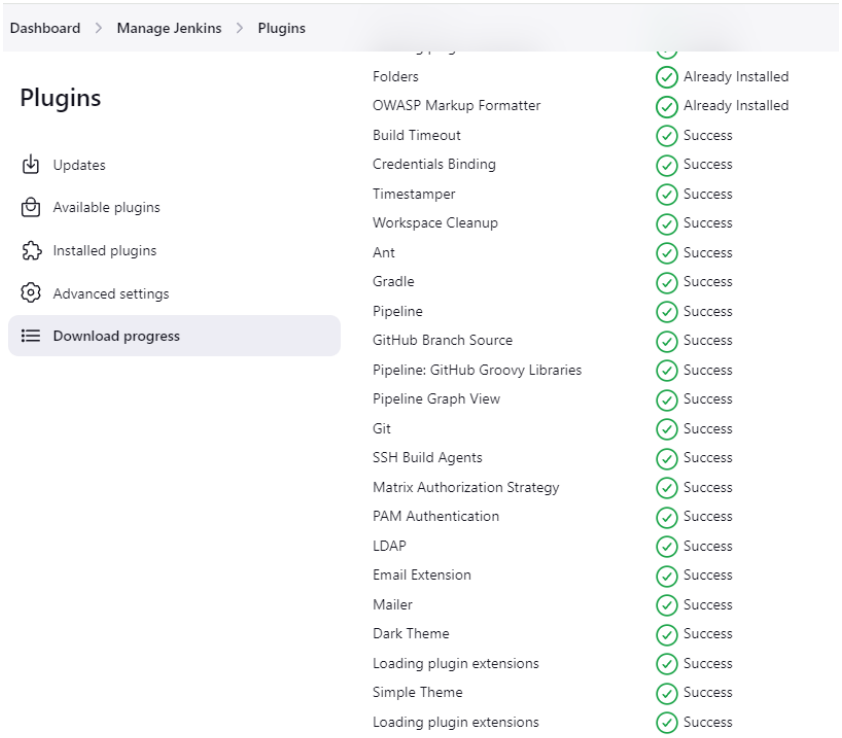
- Choose your color:

red | pink | purple | deep-purple | indigo | blue | light-blue | cyan | teal
green | light-green | lime | yellow | amber | orange | deep-orange | brown | grey
blue-grey

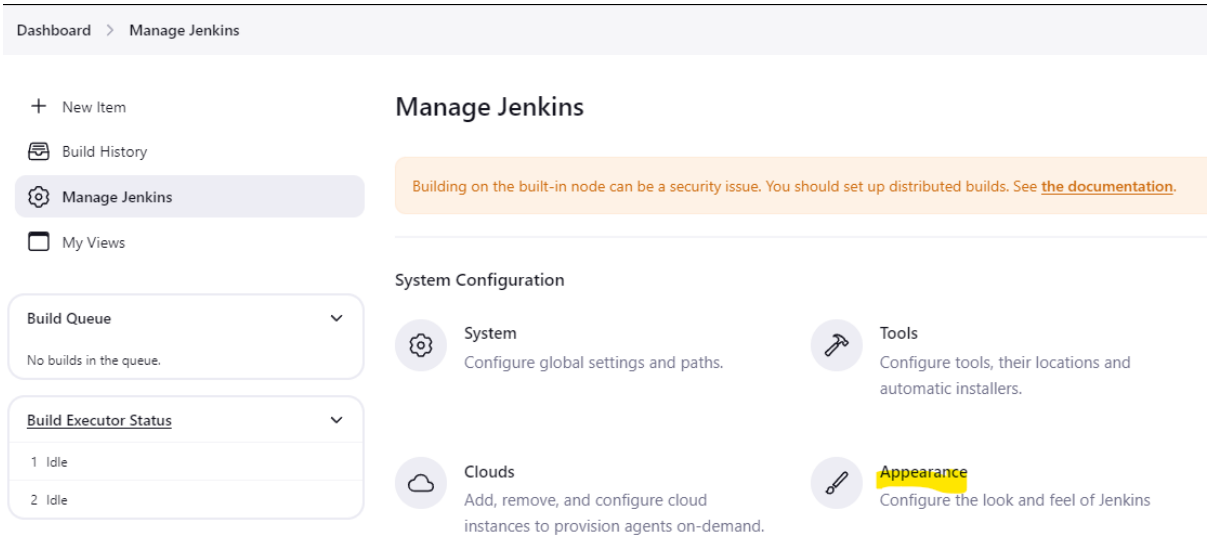
- Replace {{your-color-name}} in the URL by the chosen color: https://cdn.rawgit.com/afonsof/jenkins-material-theme/gh-pages/dist/material-purple.css
- Install Jenkins Simple Theme Plugin by going to Manage Jenkins>Plugins> Available Plugins>Simple Theme and select Install



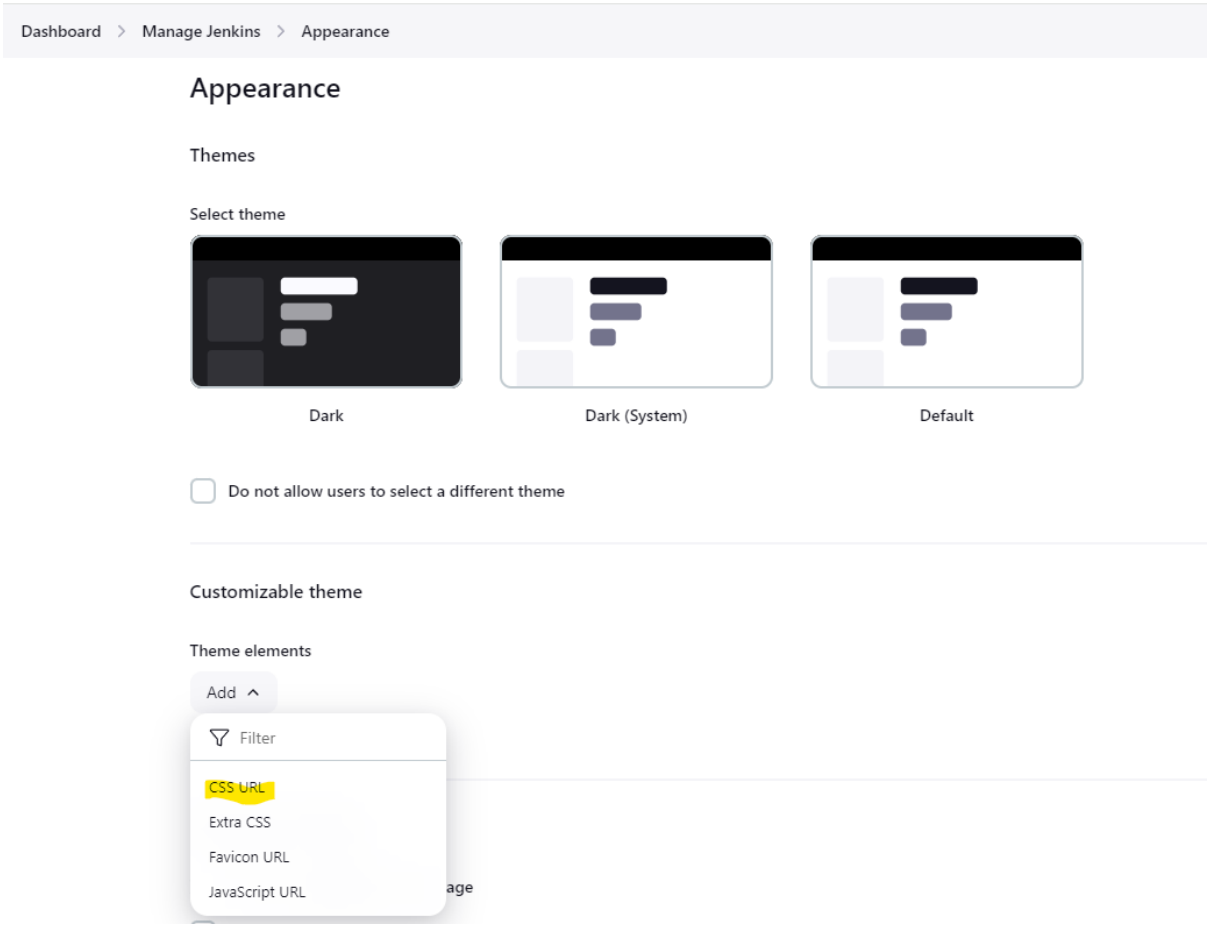



- Click Manage Jenkins > Appearance

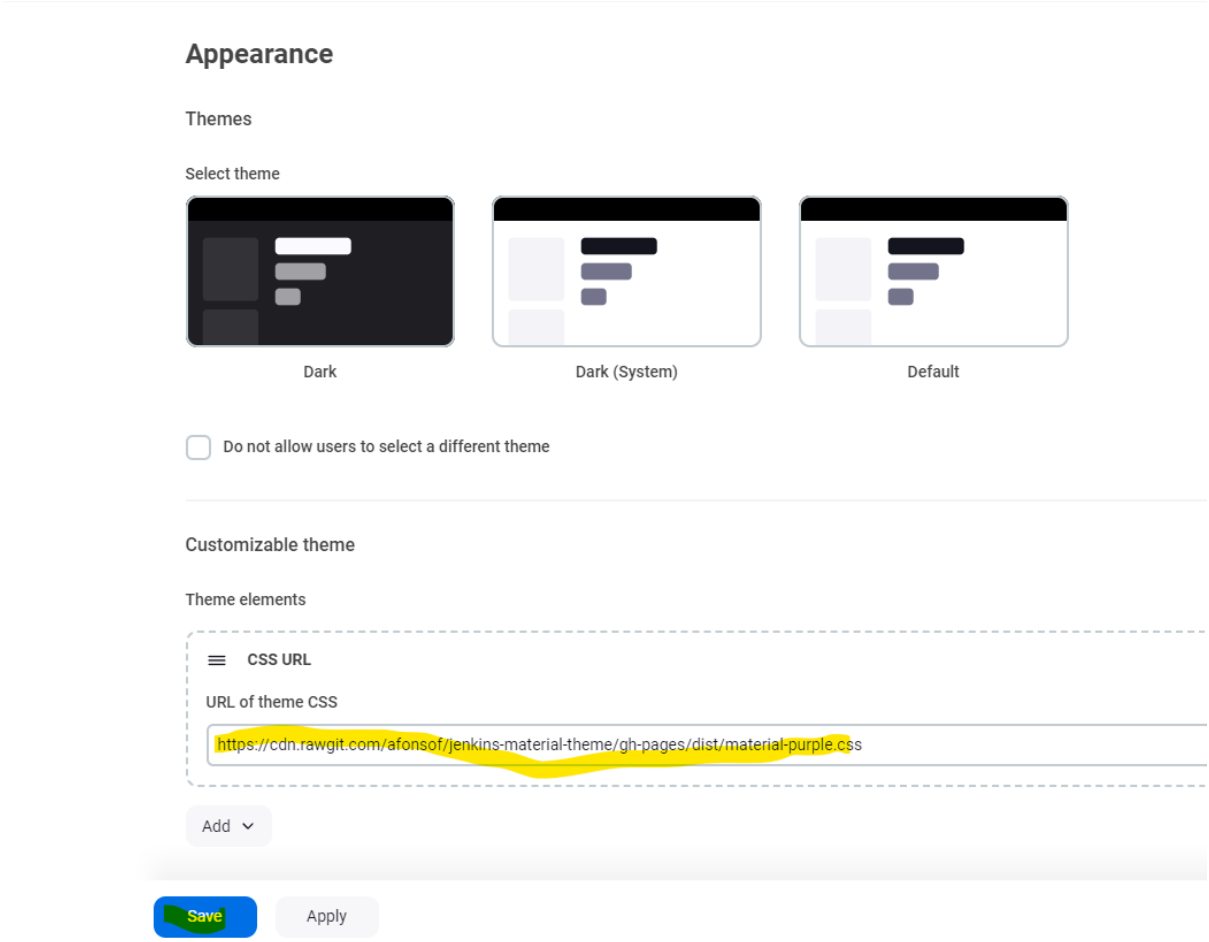



- Scroll down and under Customizable theme>Theme Elements>Drop Down on Add and select CSS URL




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

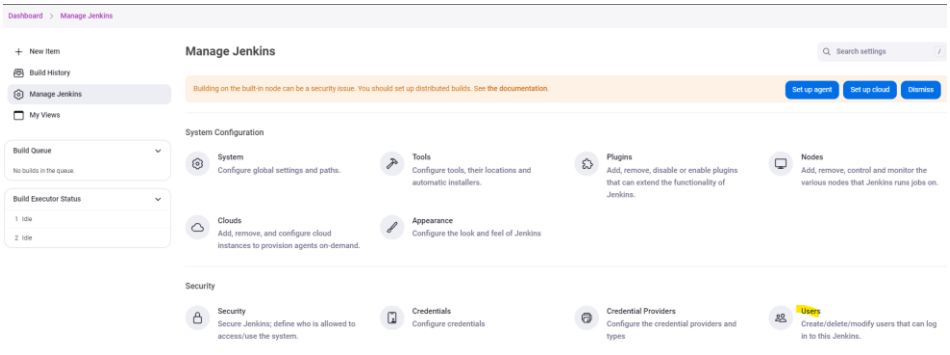
https://cdn.rawgit.com/afonsof/jenkins-material-theme/gh-pages/dist/material-purple.css



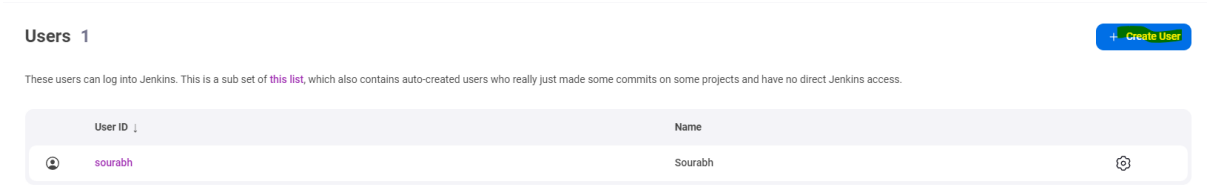

- Once we save and go back to the dashboard, we see the changes are applied.


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

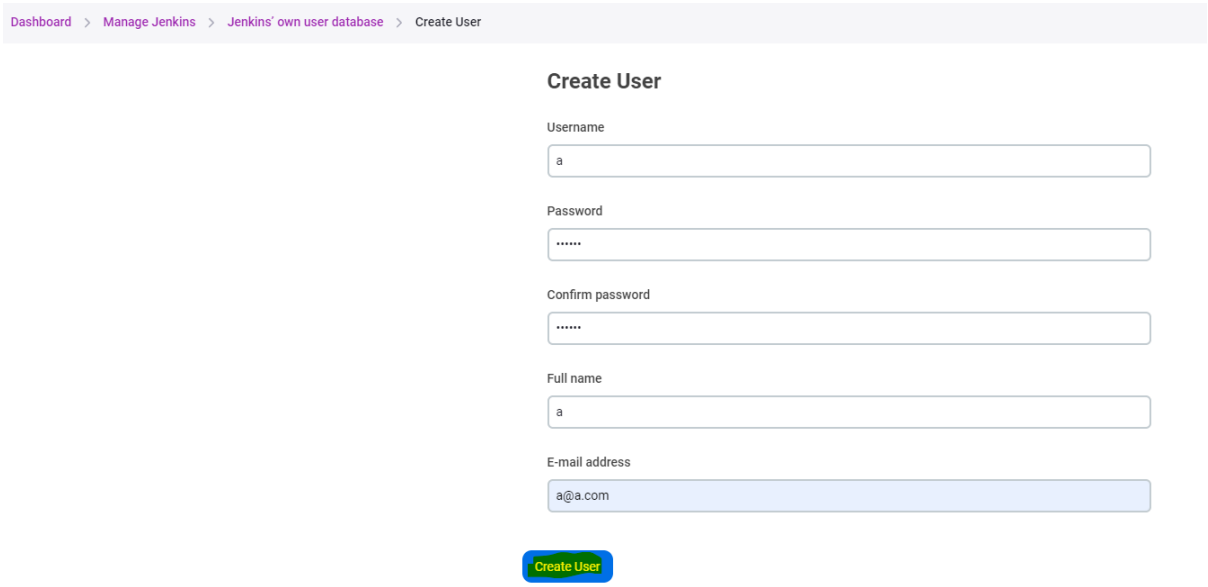
Goto Manage Jenkins > Users



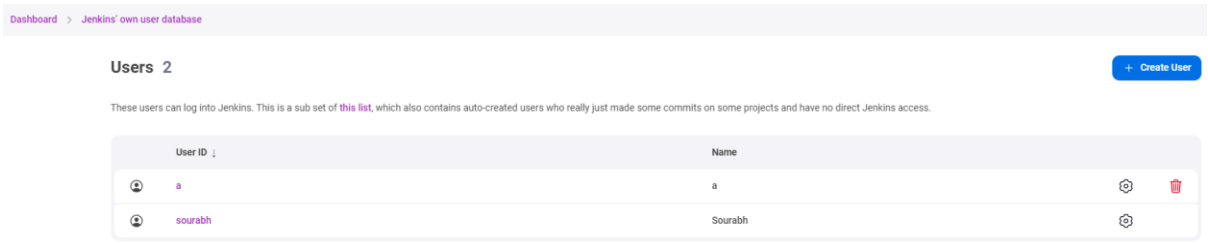
Click + Create User




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



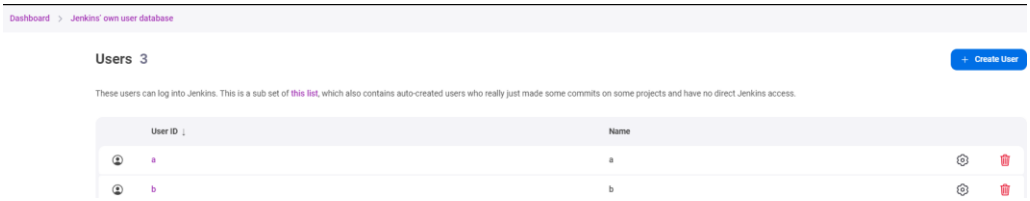

Once its created, it is displayed under Users



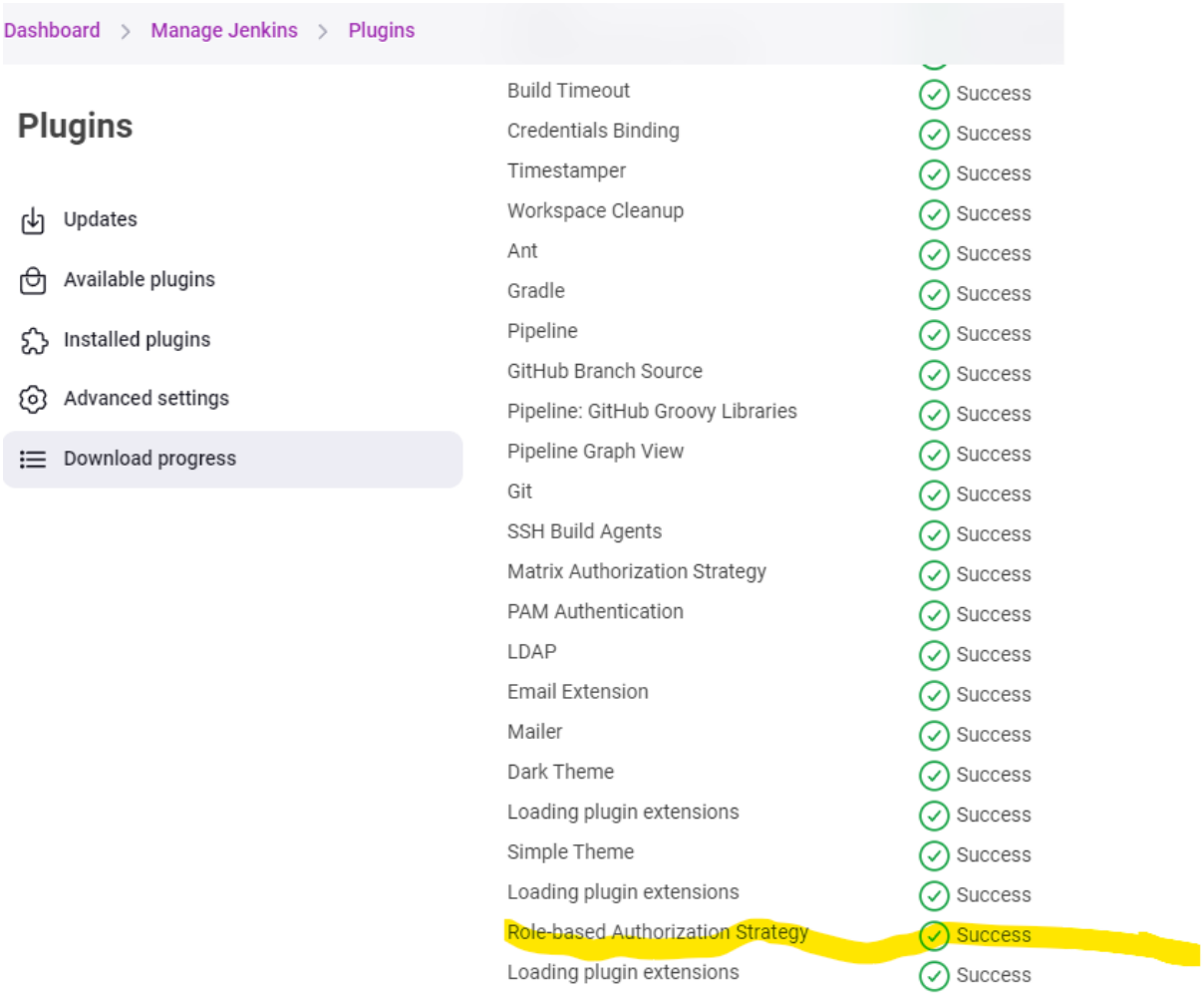

13) Jenkins Role Based Access Control **[Step 9 in Ajay Sir Notes]**

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

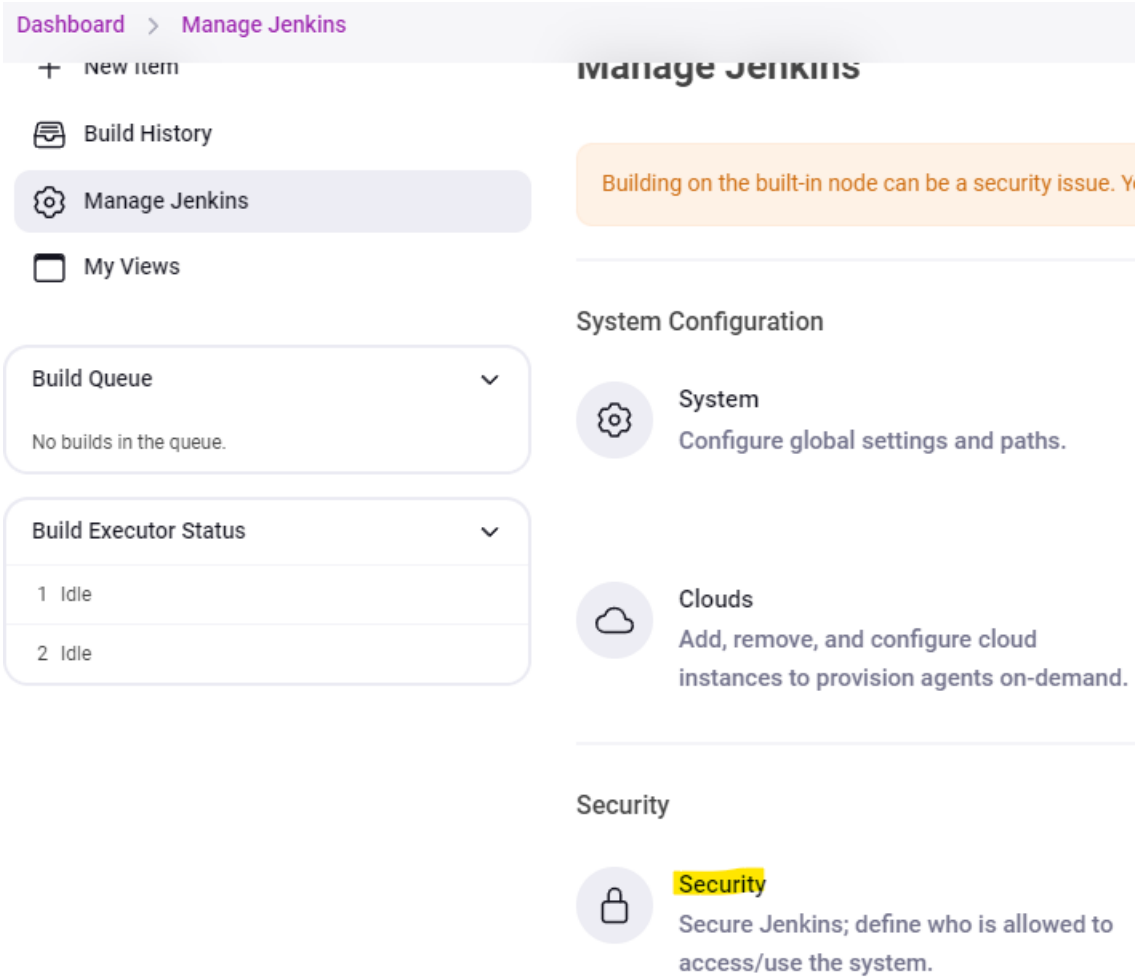
In our case, we have created user a and user b



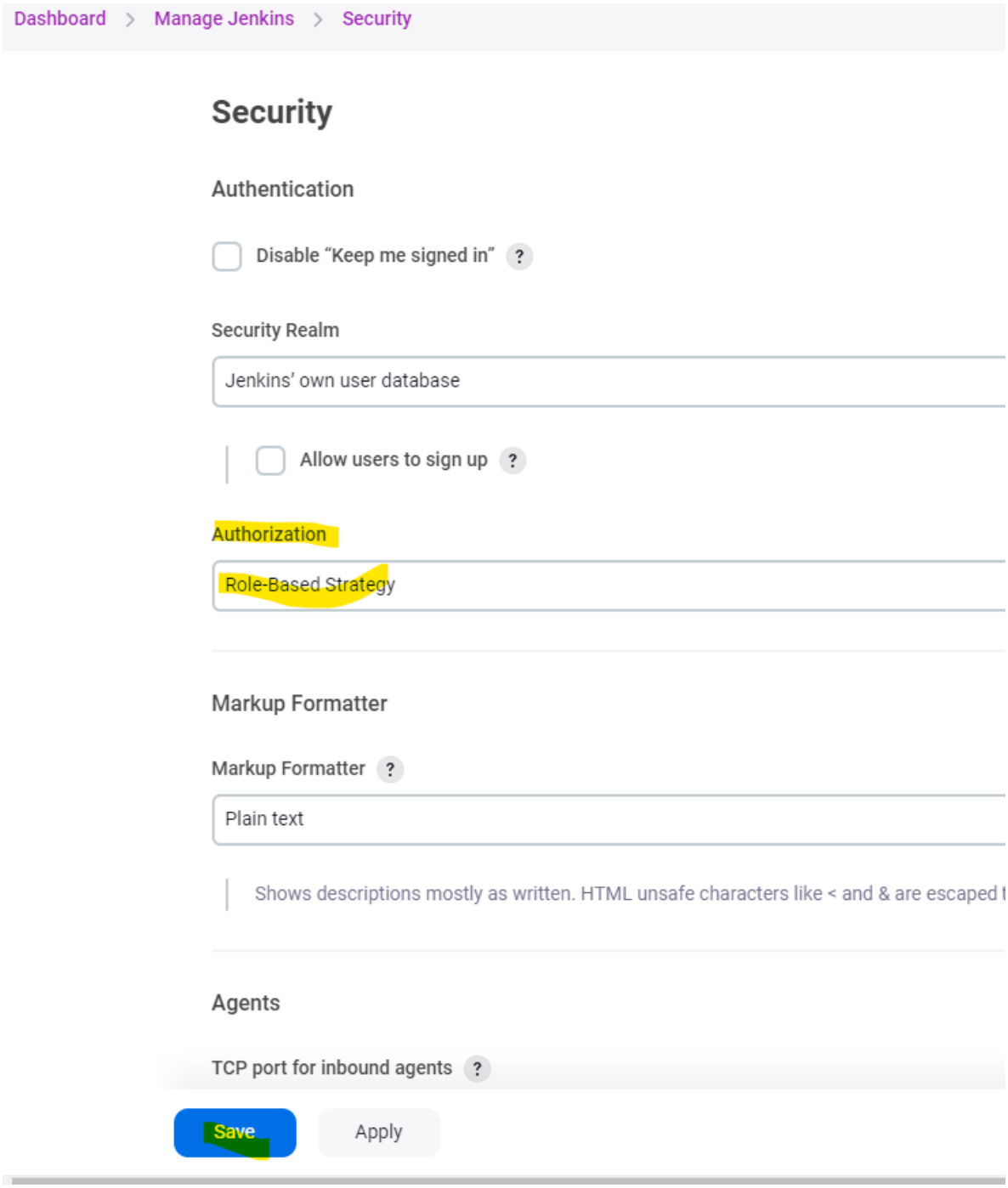

Next, we need to install Role Based Authorization Strategy plugin through available plugins.

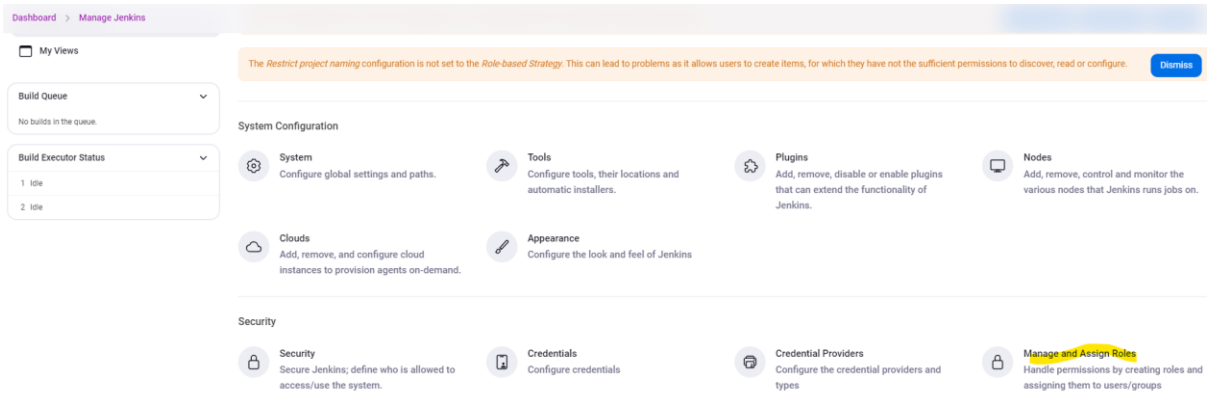

Next, we need to go to Manage Jenkins> Security



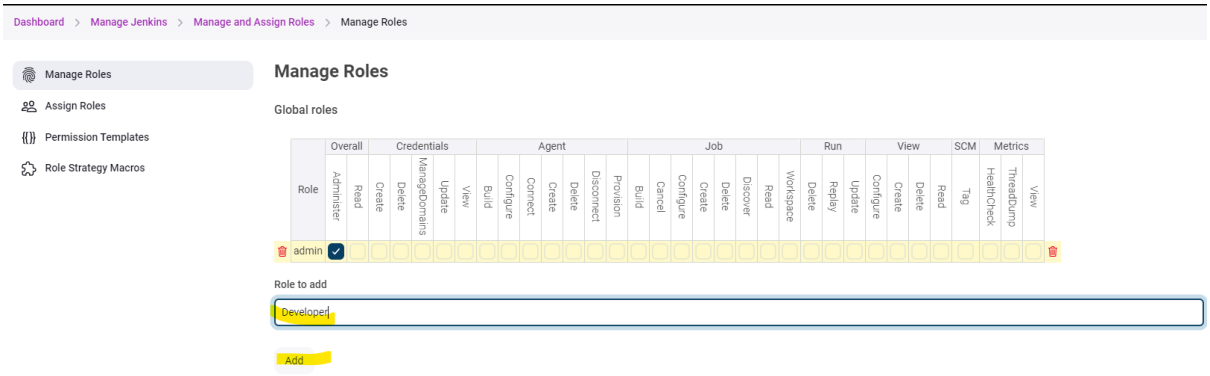

Under Authorization > Drop down and select Role-Based Strategy and click Save.



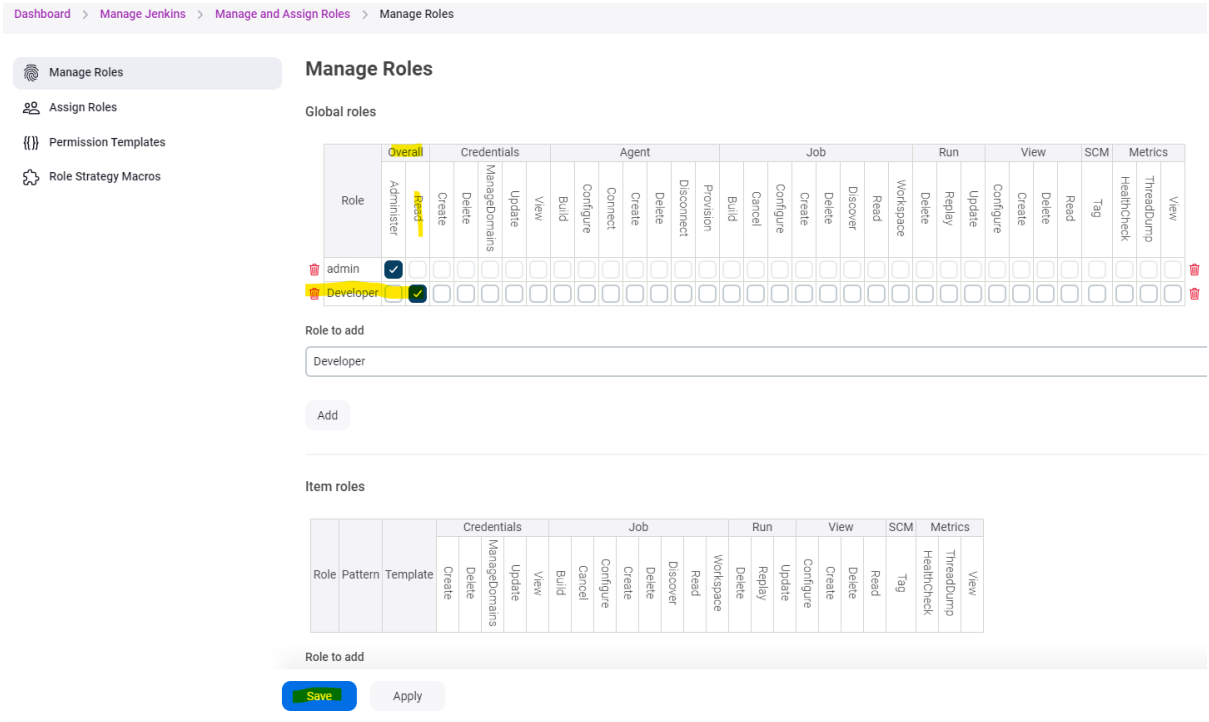

Now go to Manage Jenkins > Manage and Assign Roles.



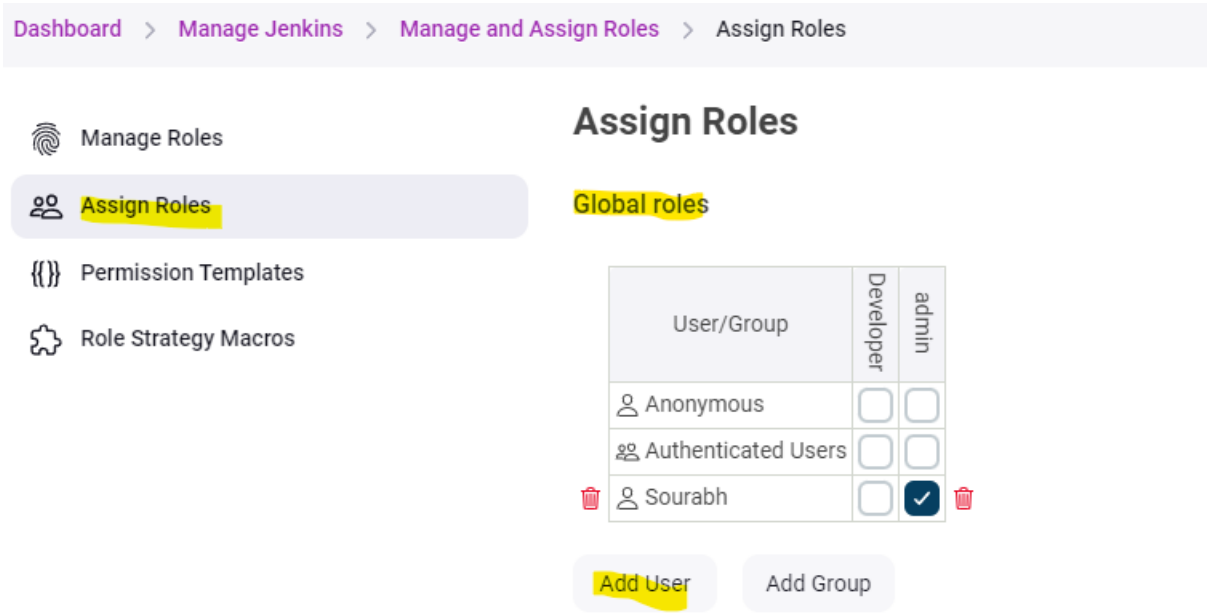

Under Manage Role > Enter “Developer” on Role to Add and click Add.



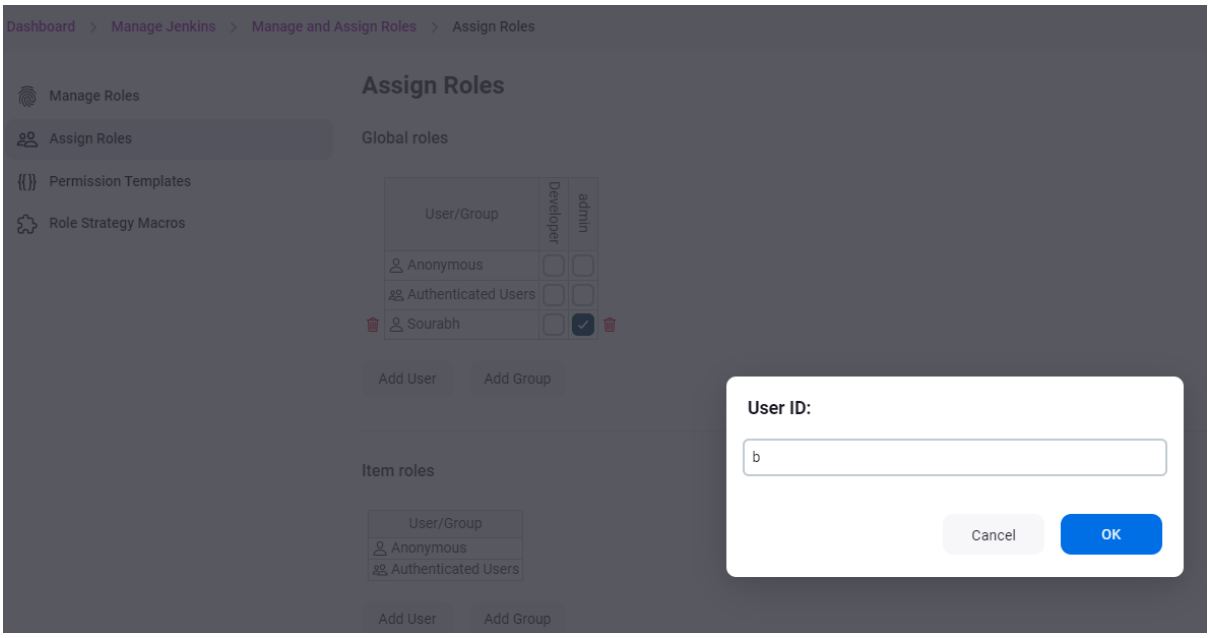

Once the role is created, check Read under Overall for the permissions to the Developer Role and click Save



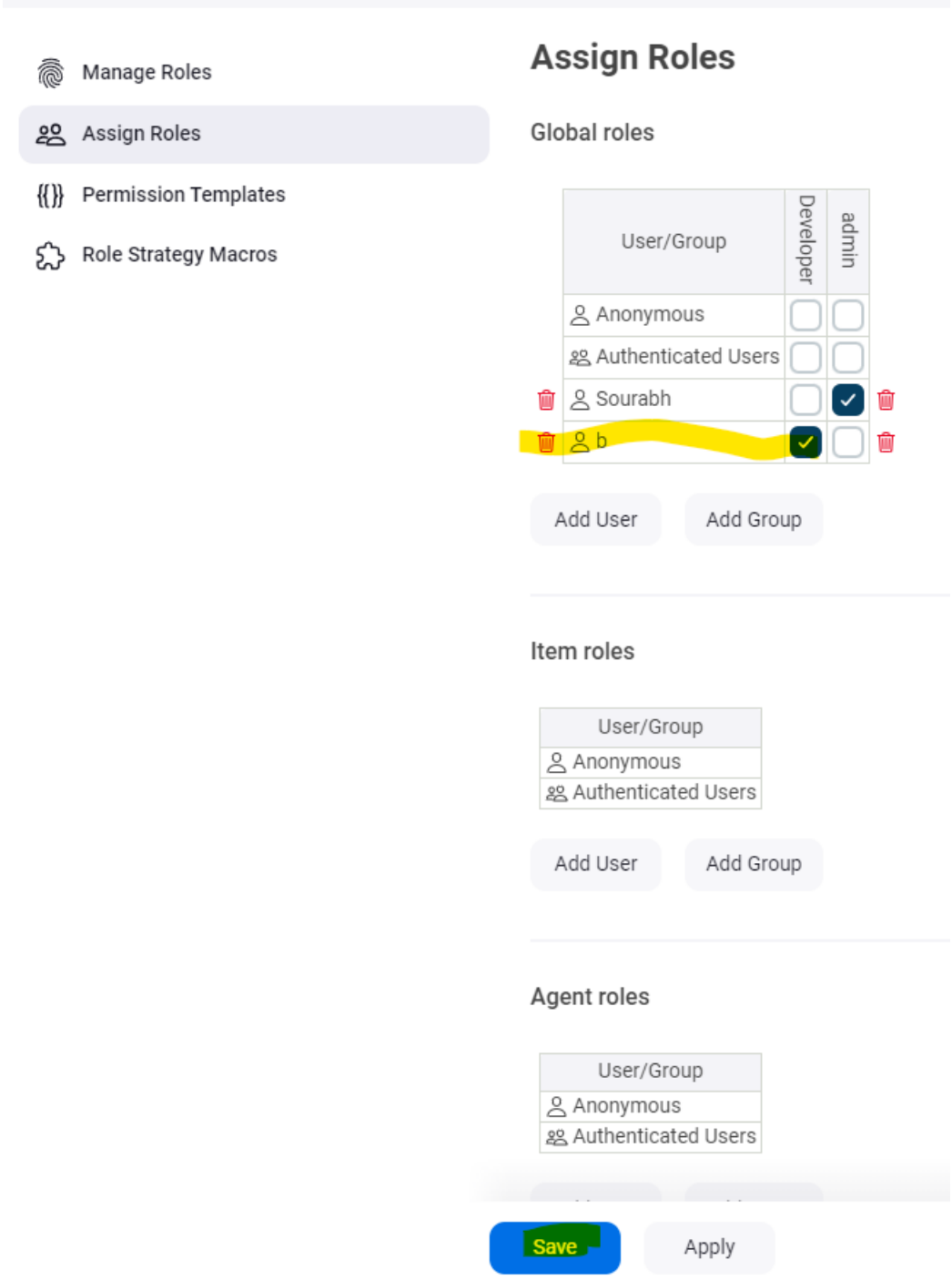
Now go to assign roles > Under Global Roles > Click on Add user



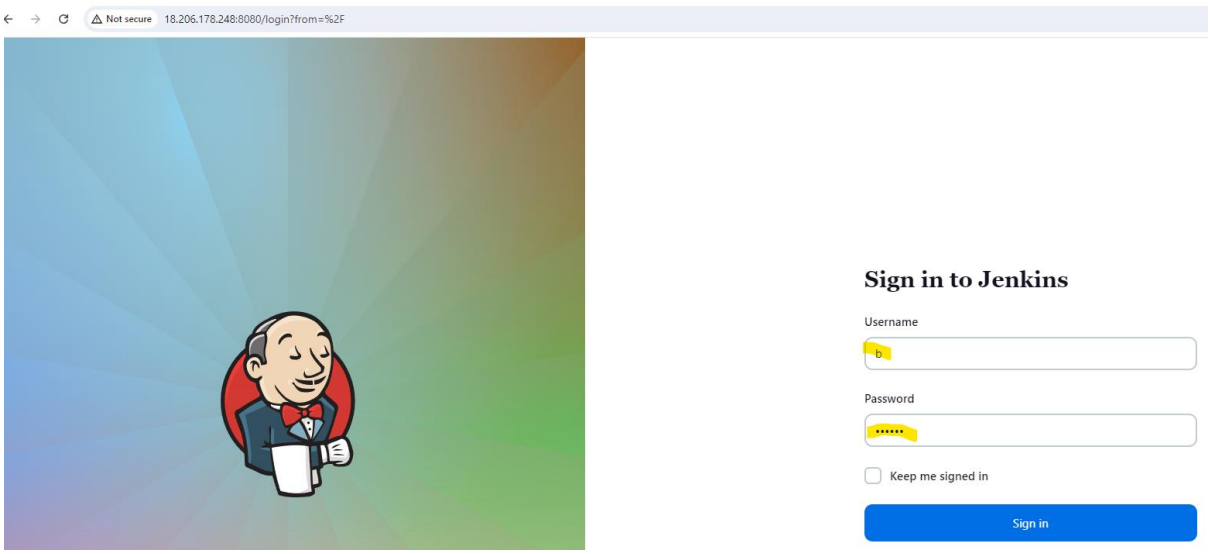

Type the User id, in our case its b and then click OK



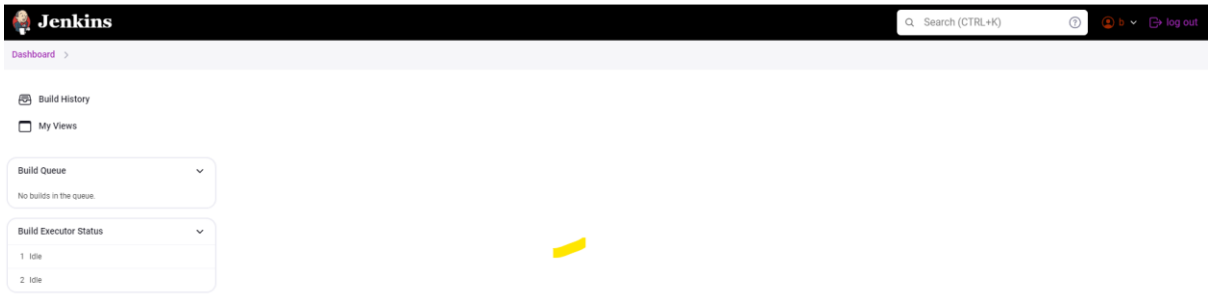

Once the user is added, we will assign developer role to it mentioned in the below screenshot and click Save.




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




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

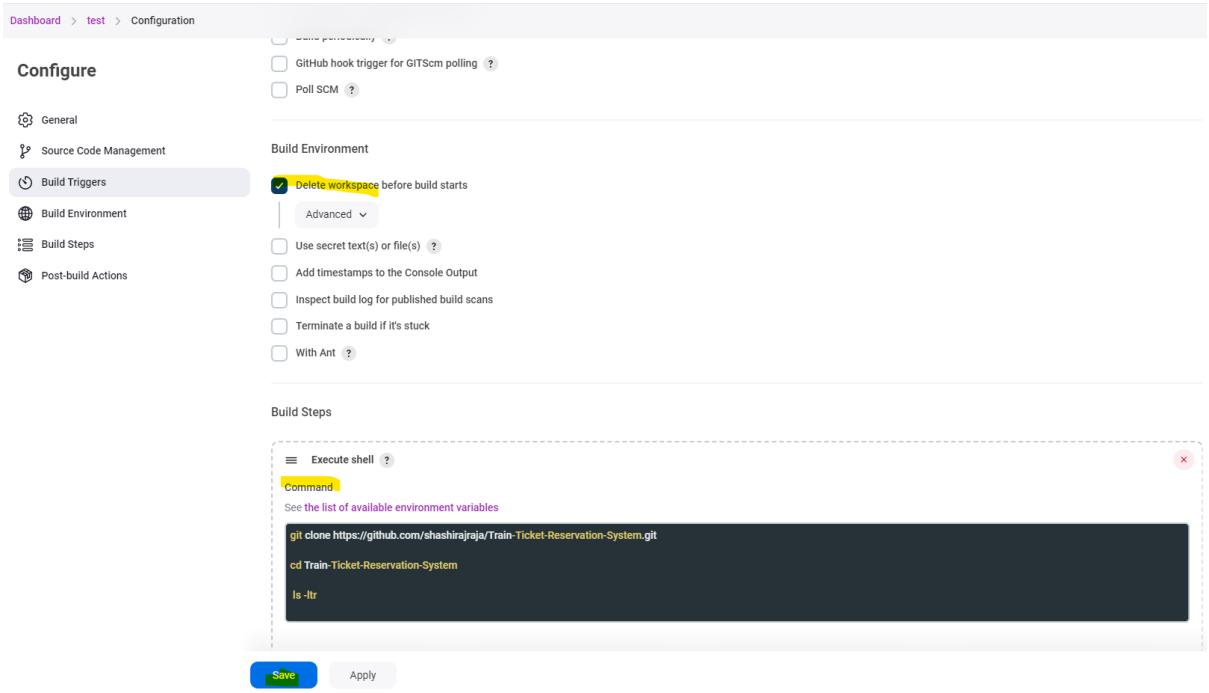
We will create a freestyle project named test and will select the below options to clone a git repo using the project and click Save.

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

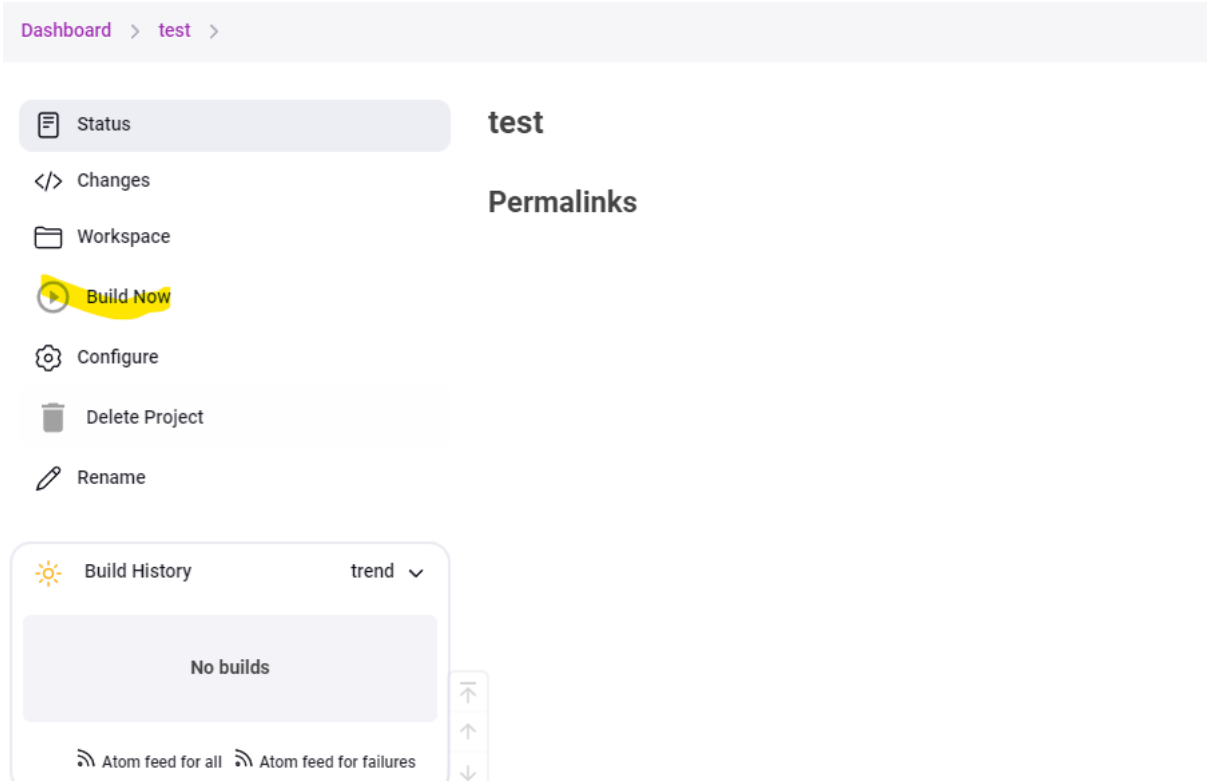
  git clone https://github.com/shashirajraja/Train-Ticket-Reservation-System.git

  cd Train-Ticket-Reservation-System

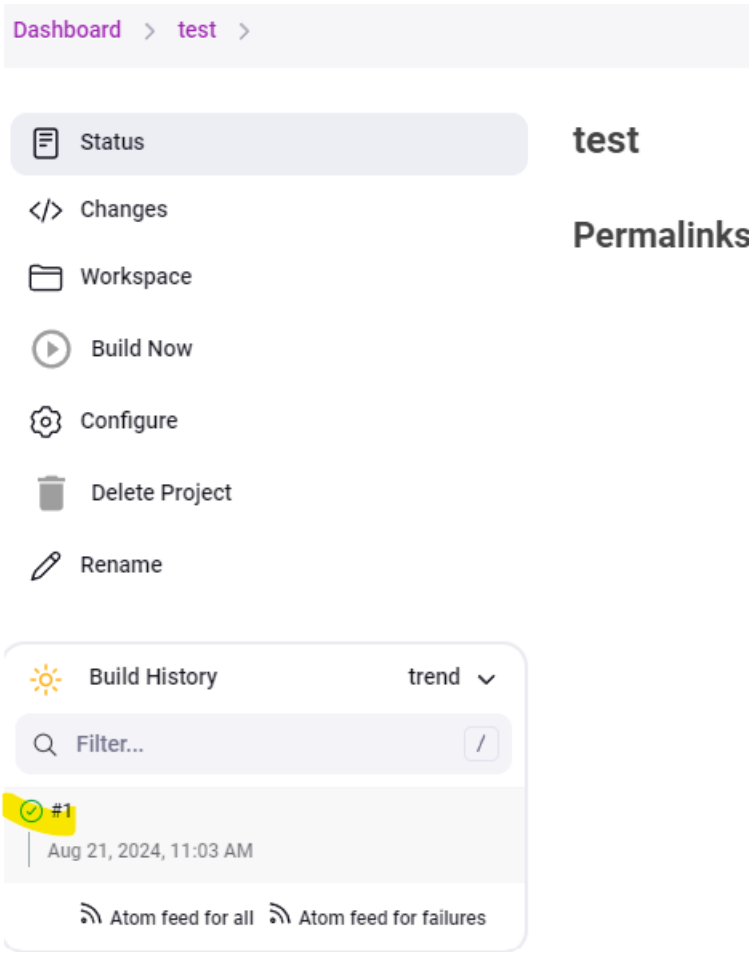
ls -ltr



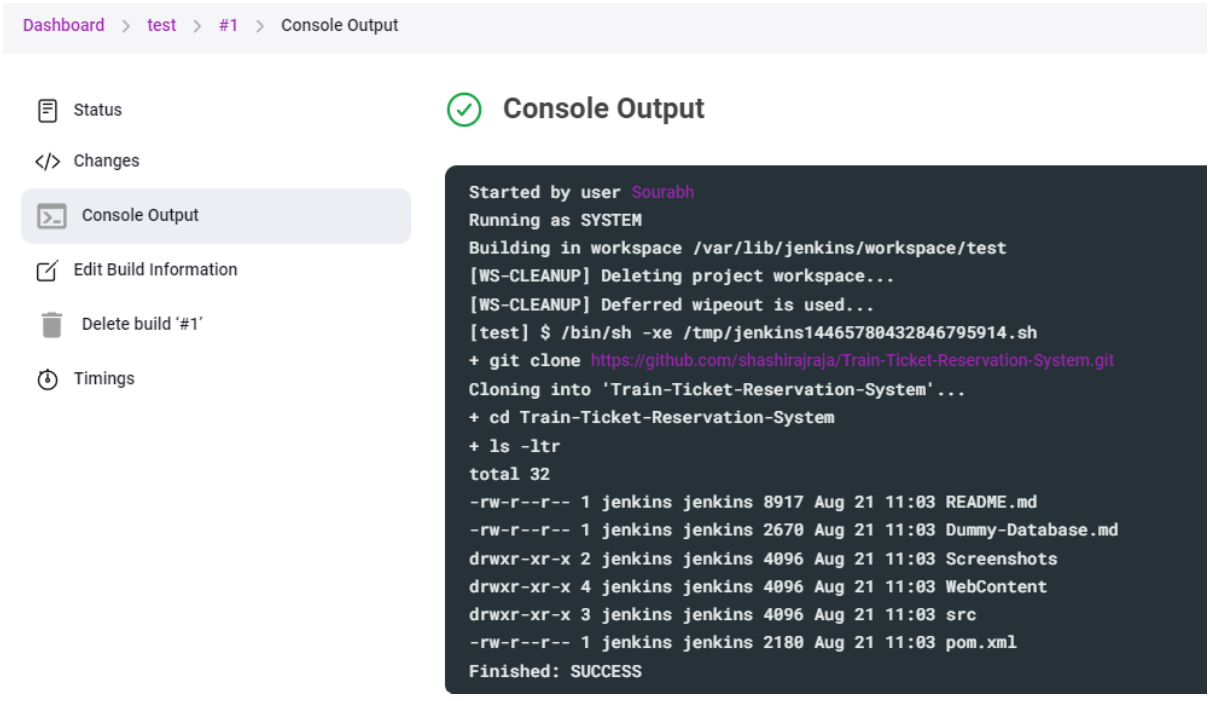

Now run Build Now on the Project “test”



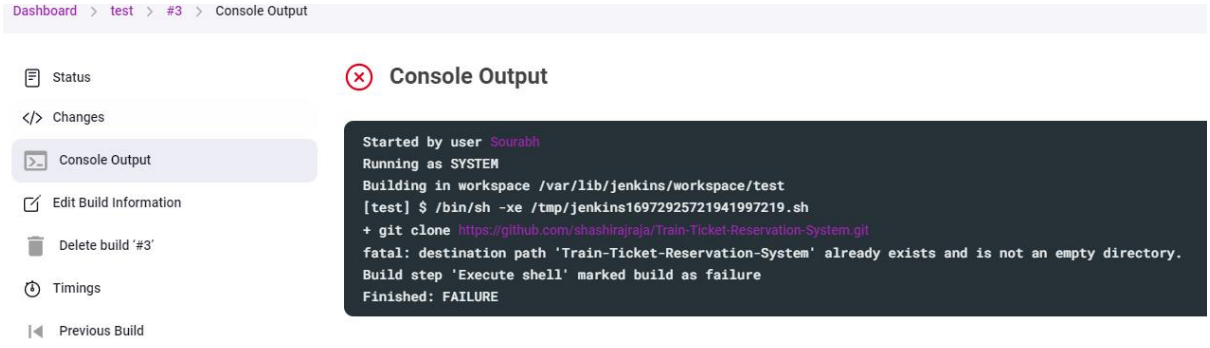

We see the build is completed successfully.




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



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.



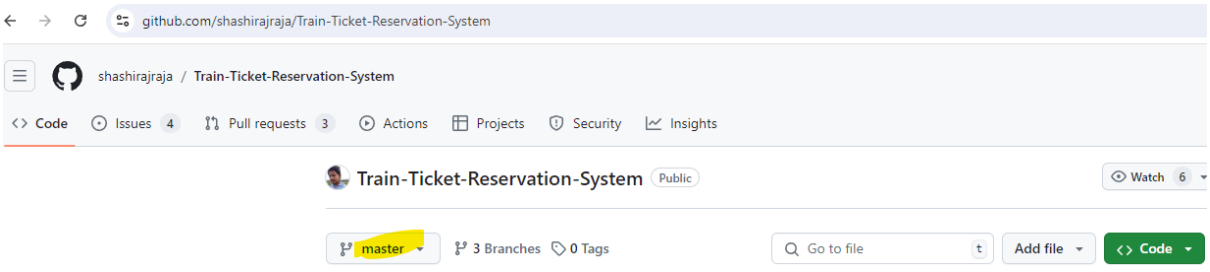

We can also use the git project path option from the source code management option. We can do it by choosing the below options

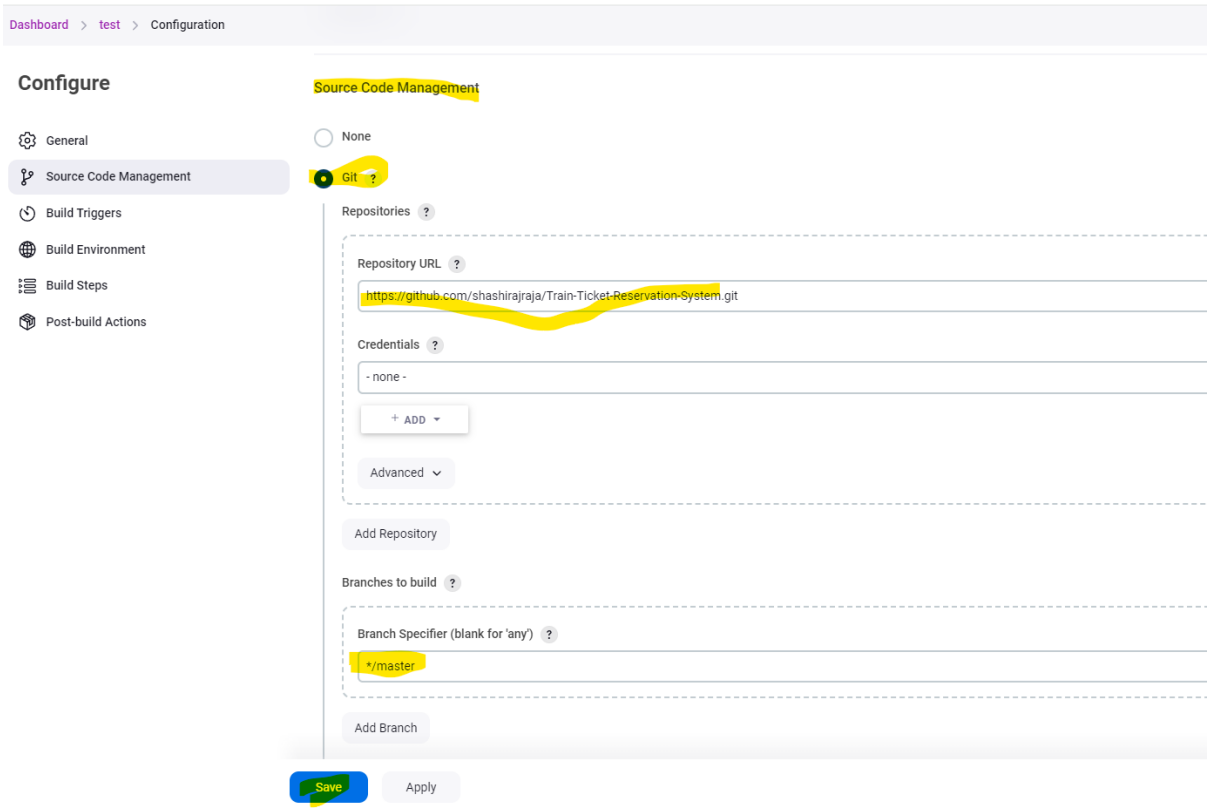
We need to select git under Source Code Management

We need to give the Repository URL under repositories

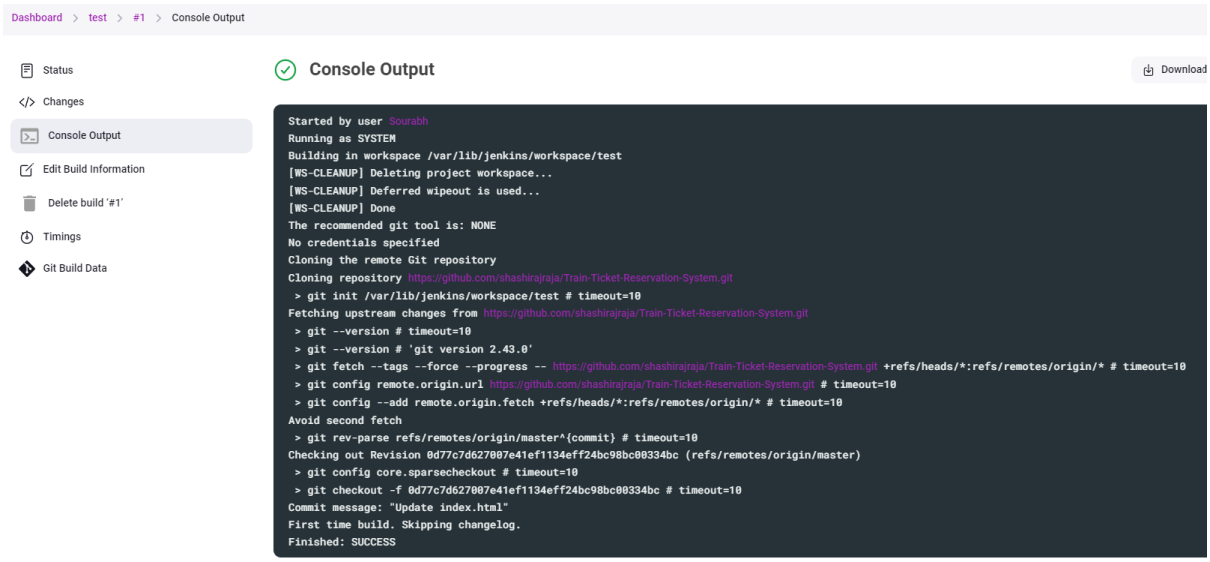
https://github.com/shashirajraja/Train-Ticket-Reservation-System.git

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






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

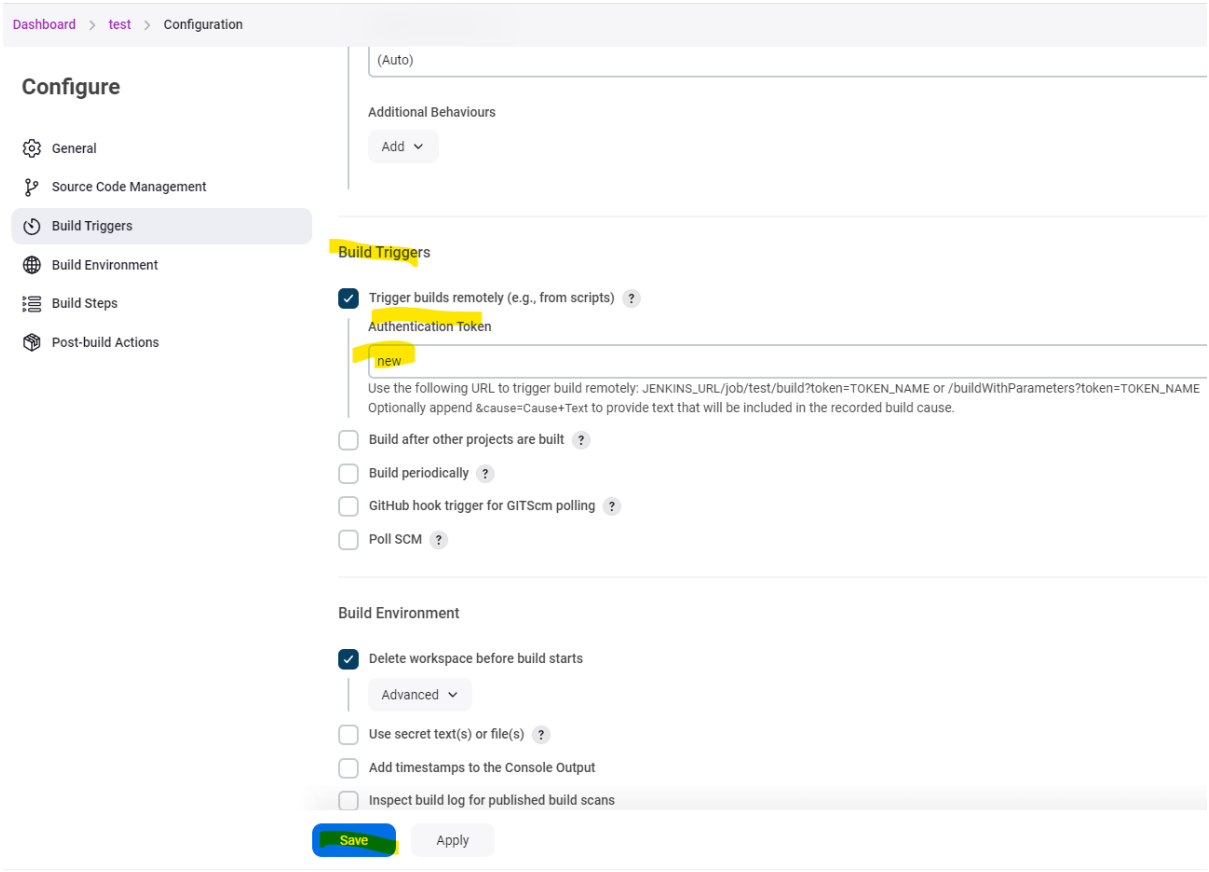



15) Trigger Build Remotely on Jenkins **[Step 11 in Ajay Sir Notes]**

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

We will follow the below steps to achieve the same.

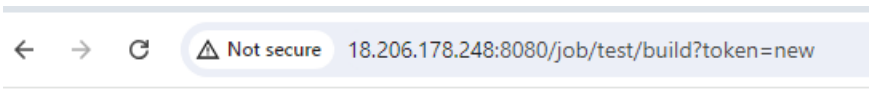
While configuring the job, we need to select Trigger builds remotely under Build Triggers, then we can give any authentication token eg new and Click Save.



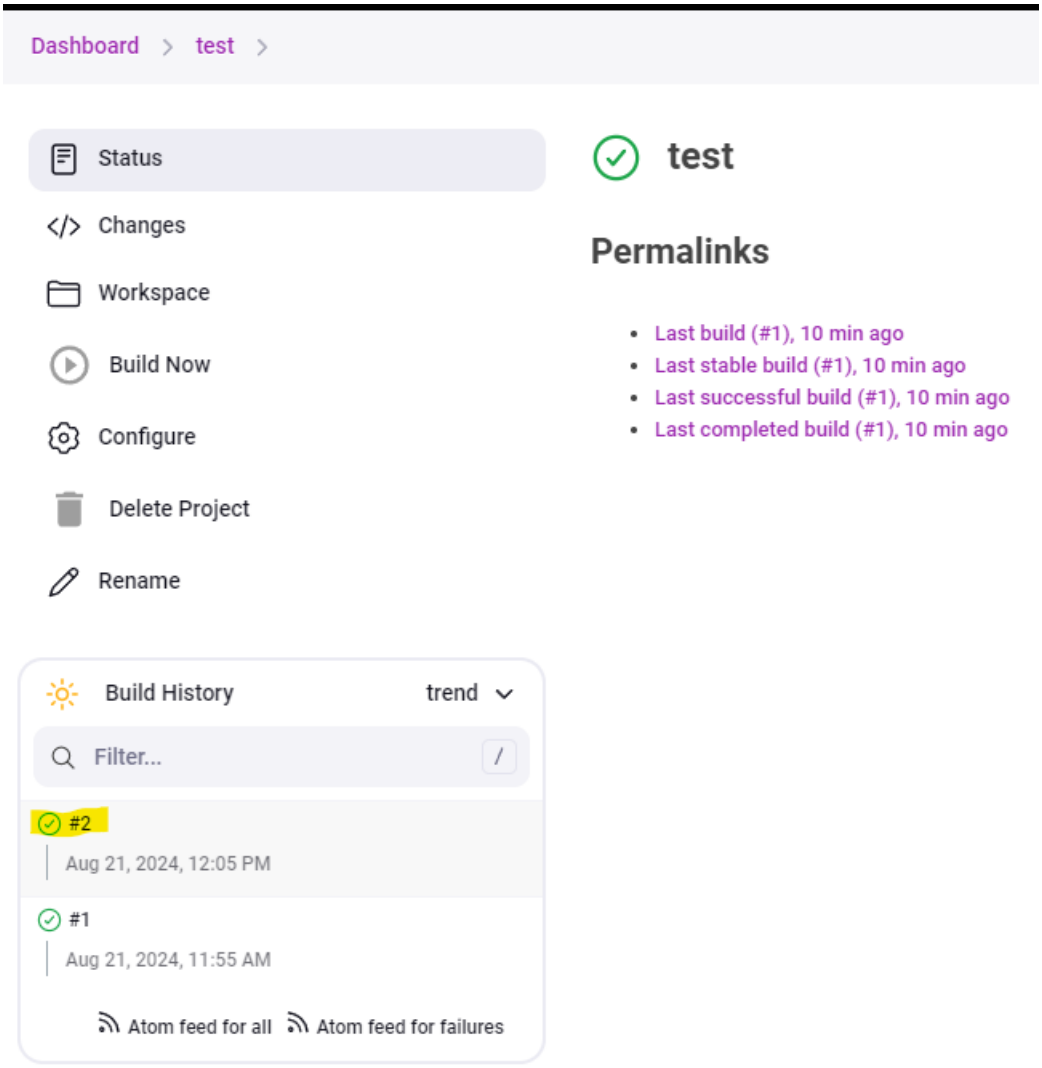

To trigger the build remotely we need use the below URL

JENKINS_URL/job/test/build?token=TOKEN_NAME or /buildWithParameters?token=TOKEN_NAME

Below screenshot is in our case

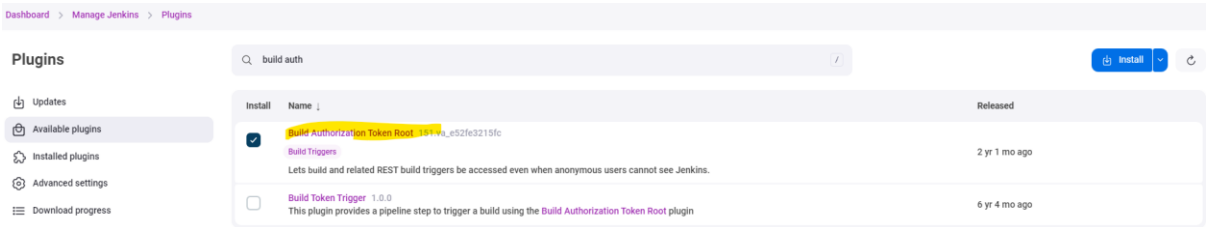


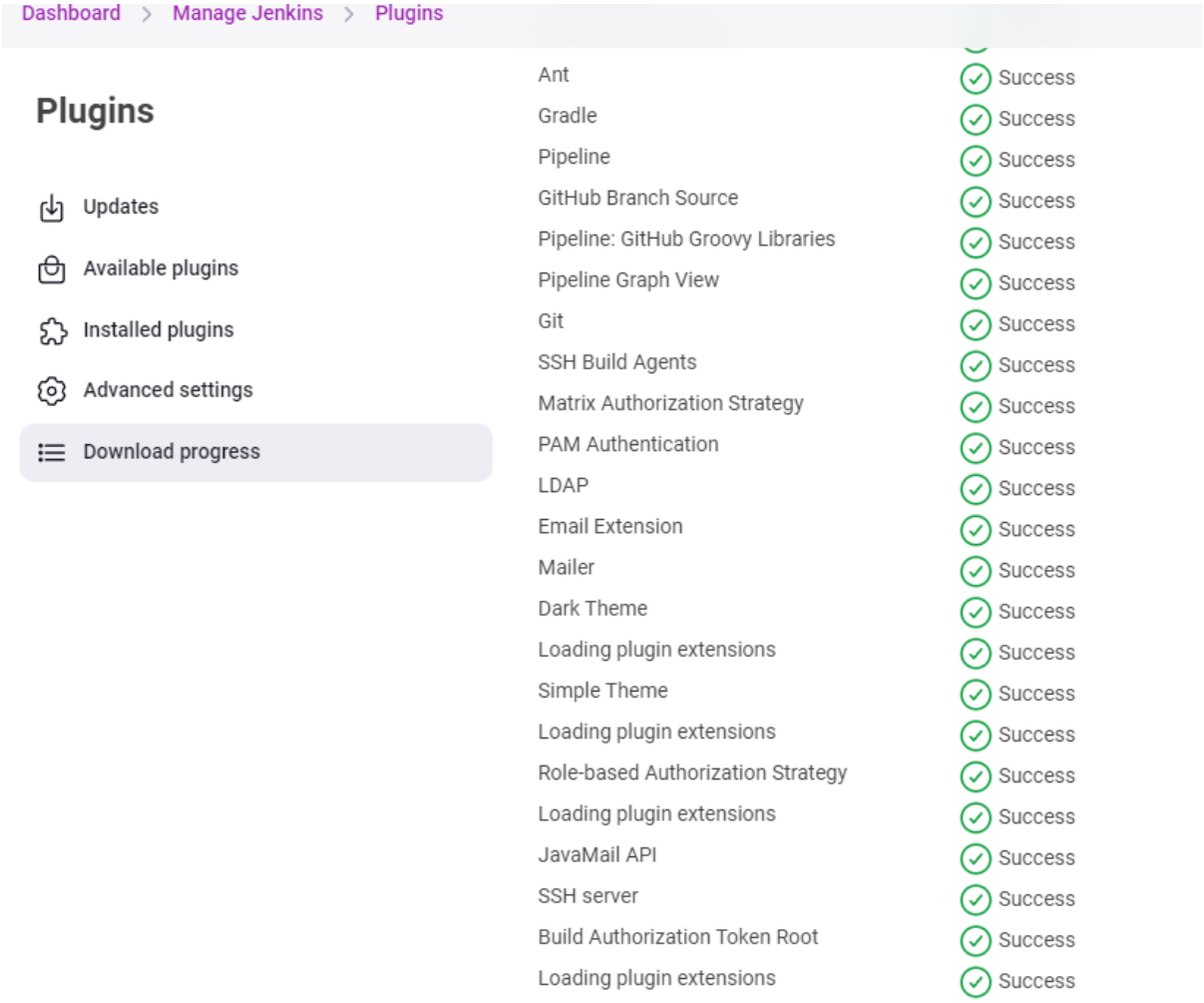

We see the 2nd Build is also executed successfully.




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

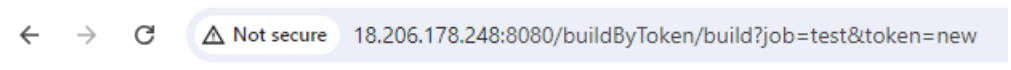
We need to install the plugin Build Authorization Token Root



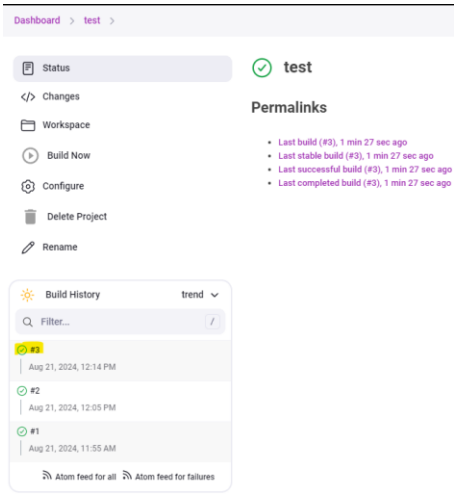



Now we will run the below URL to see if build runs.

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




See the Build 3 is also completed successfully.



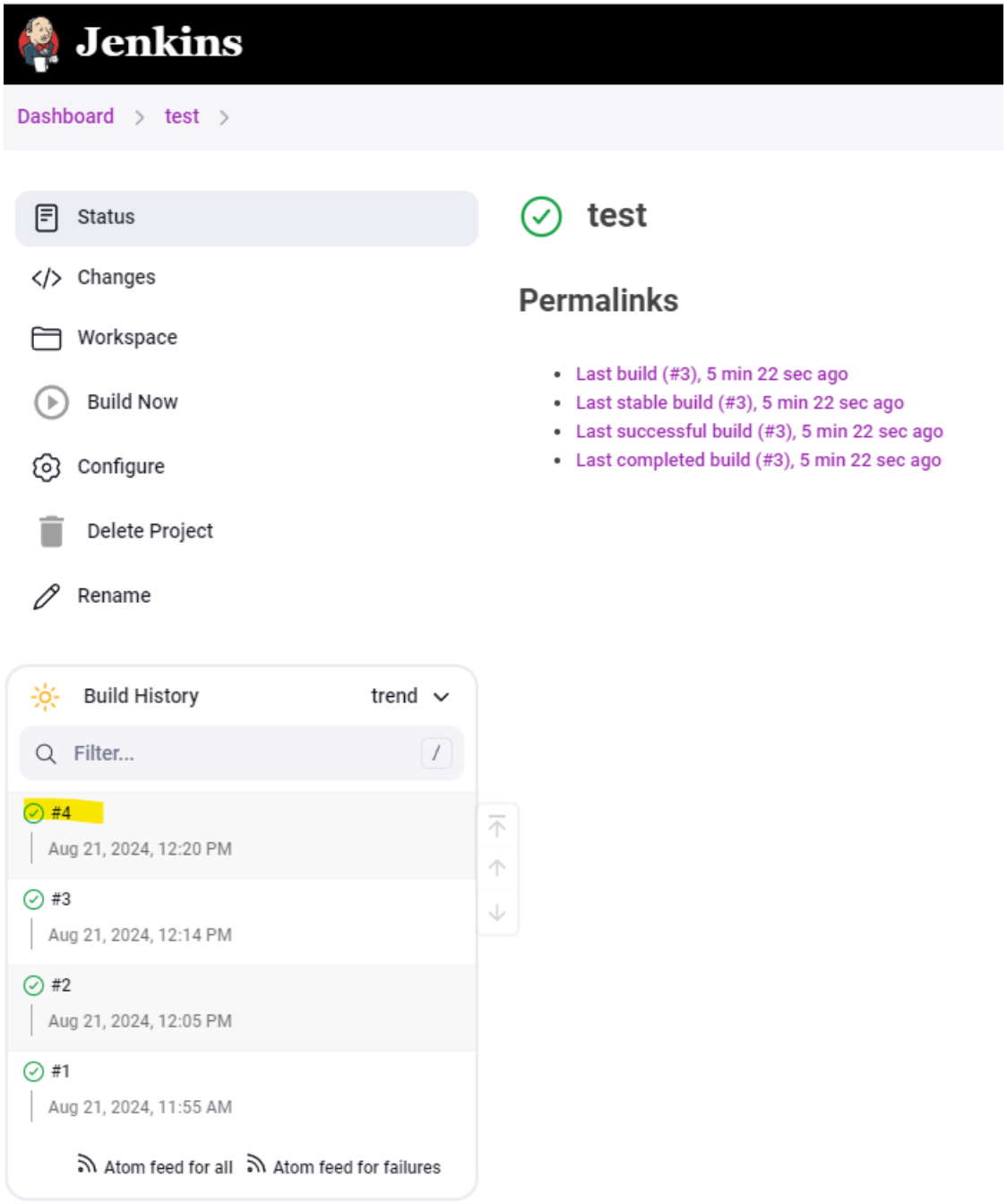
We can also run it on the Ubuntu terminal using the below command.

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

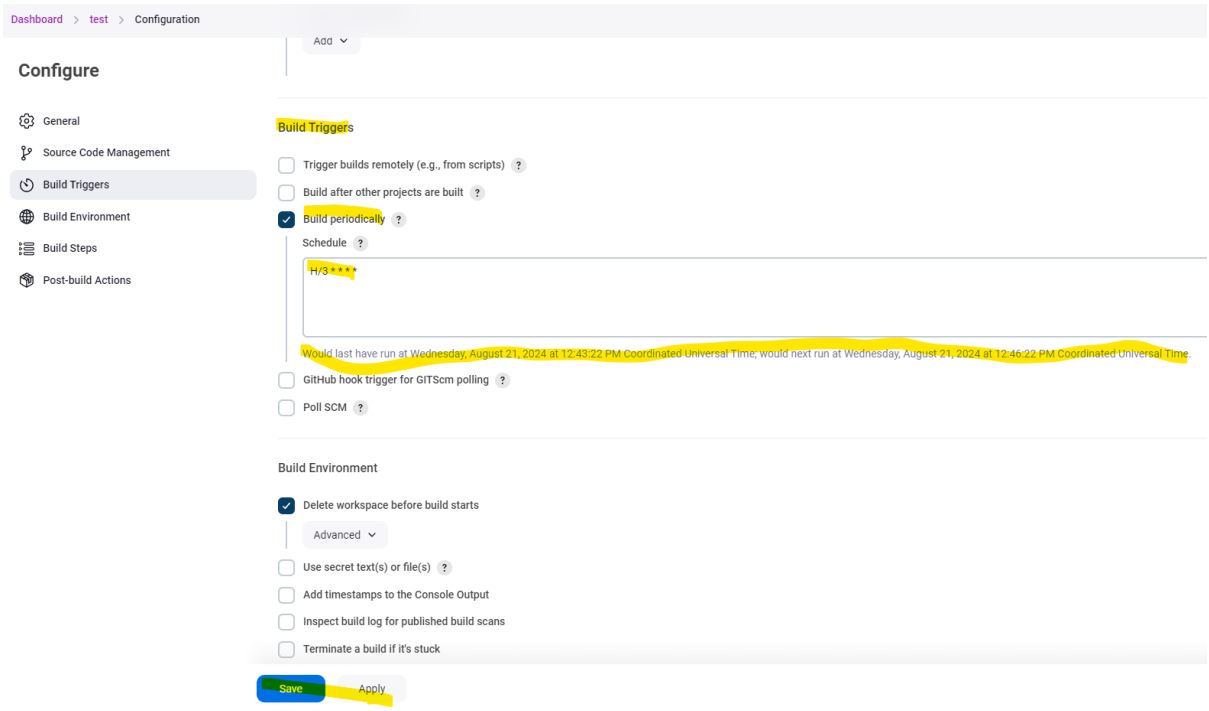


16) Build jobs periodically in Jenkins **[Step 13 in Ajay Sir Notes]**

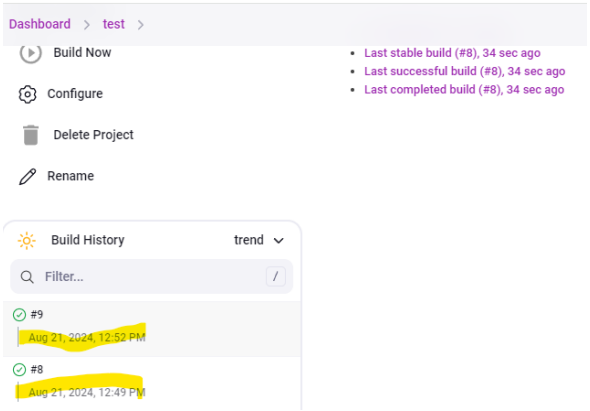
We can set the builds to run on periodic basis. To perform that we need to follow below steps.

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



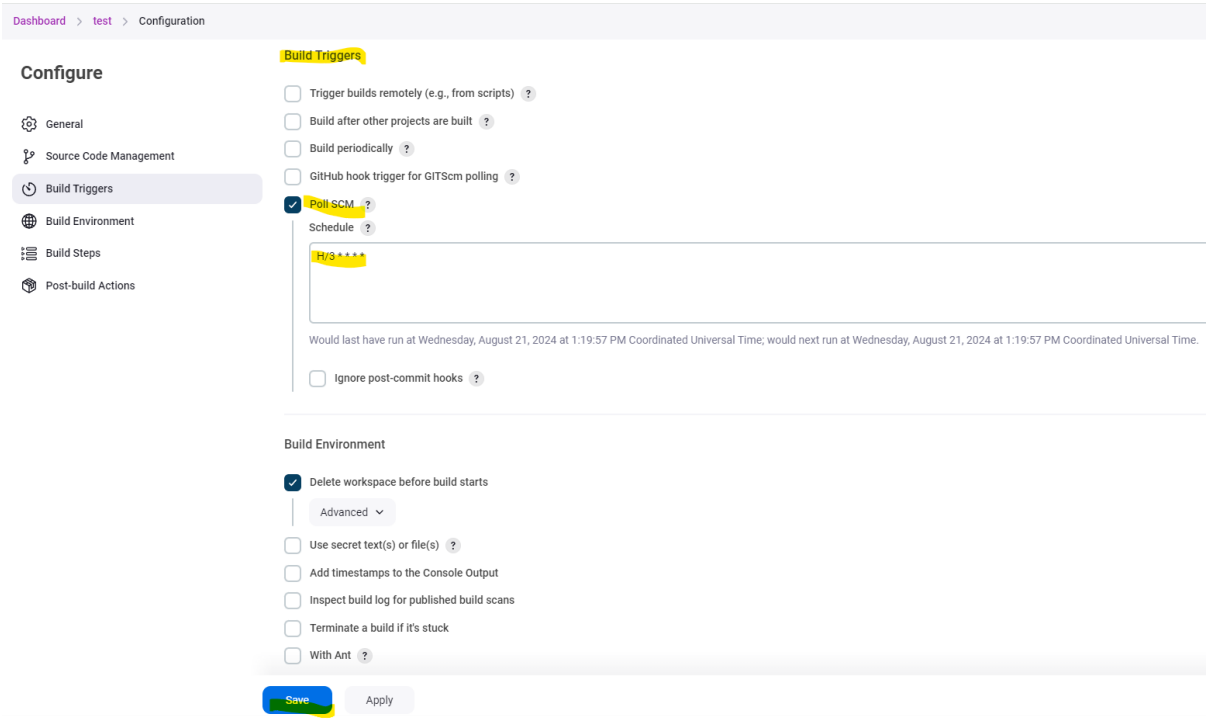

Now you can see the job runs as per the schedule after every 3 minutes.



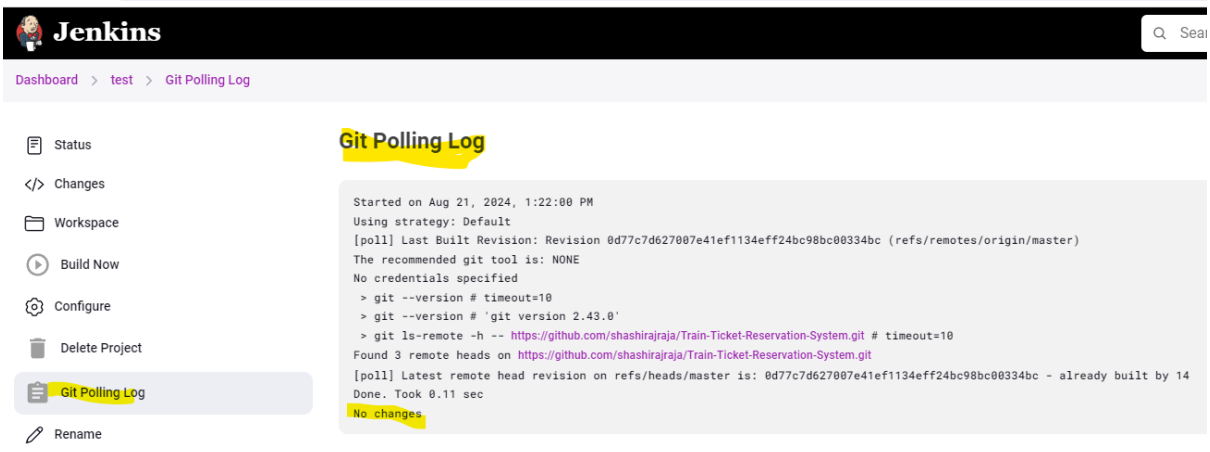
17) Poll SCM through Jenkins **[Step 14 in Ajay Sir Notes]**

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




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

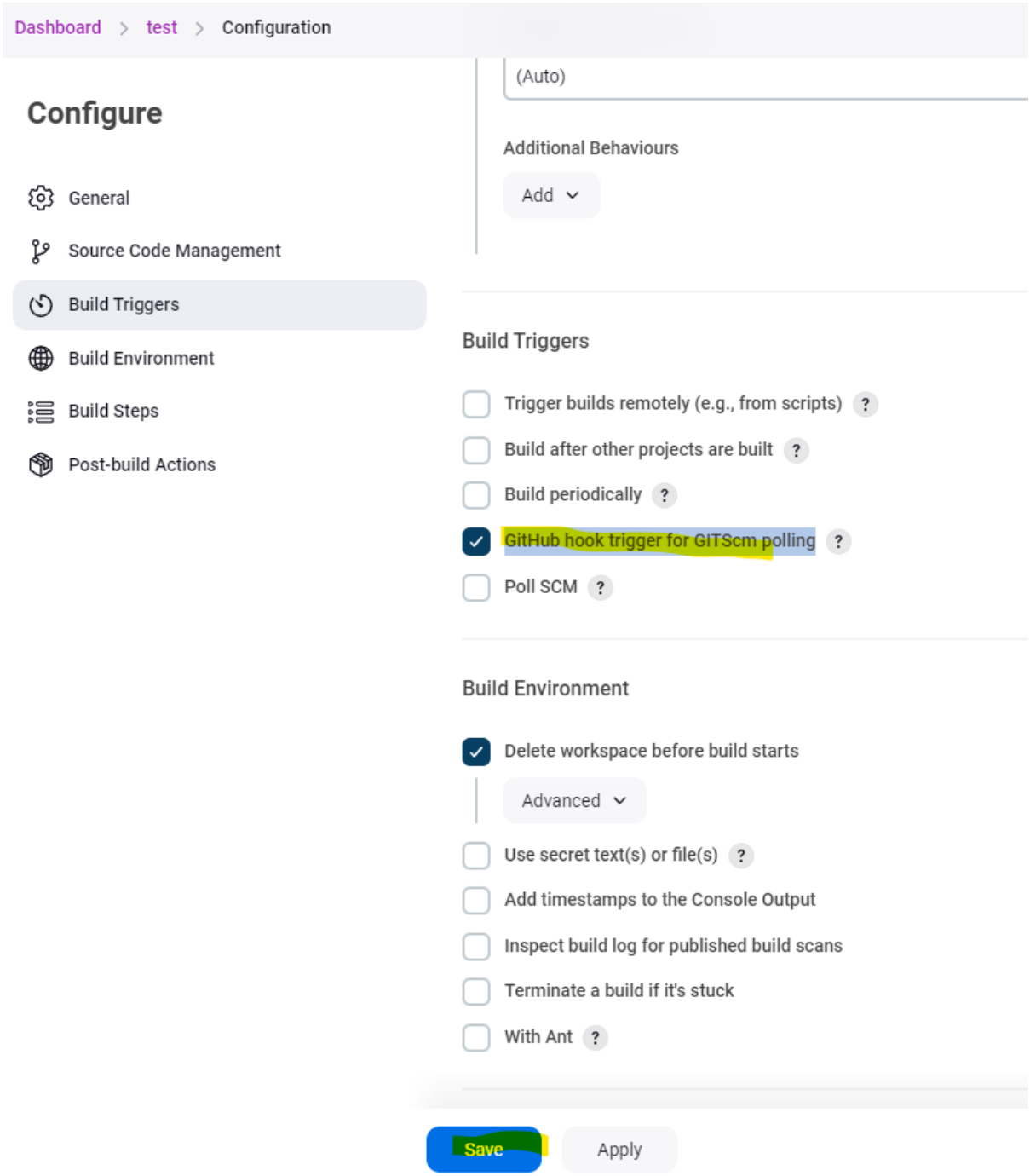


18) GitHub hook trigger for GITScm polling through Jenkins **[Step 15 in Ajay Sir Notes]**

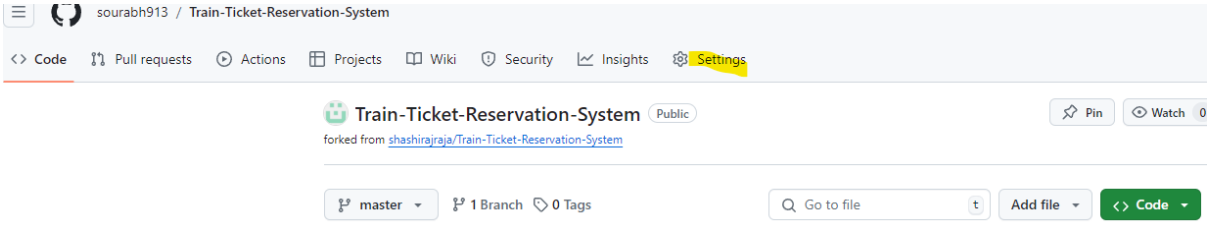
This is used when a job needs to be triggered when there is any change in the GitHub repo.

Steps to followed are below:

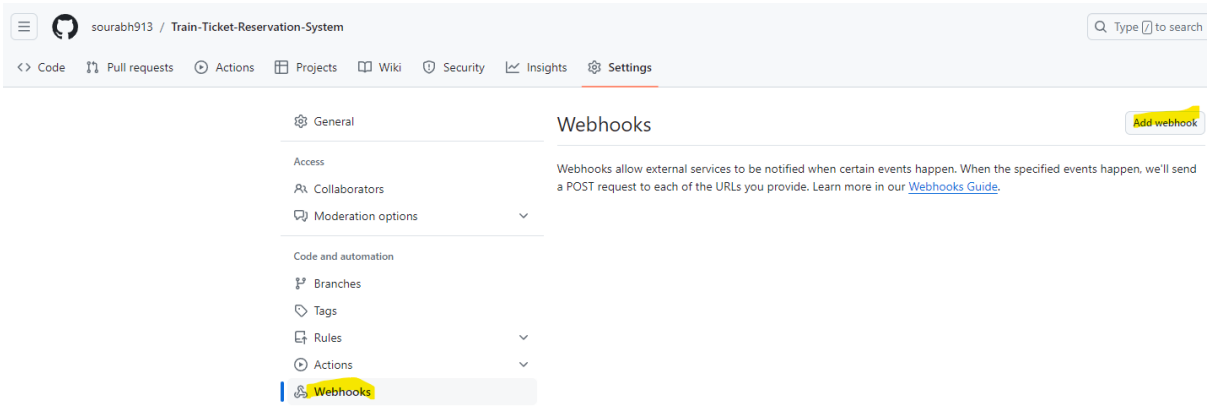
While configuring the job, Check the option of GitHub hook trigger for GITScm polling under Build Triggers and then click Save.



Then on the GitHub side, Go to Settings

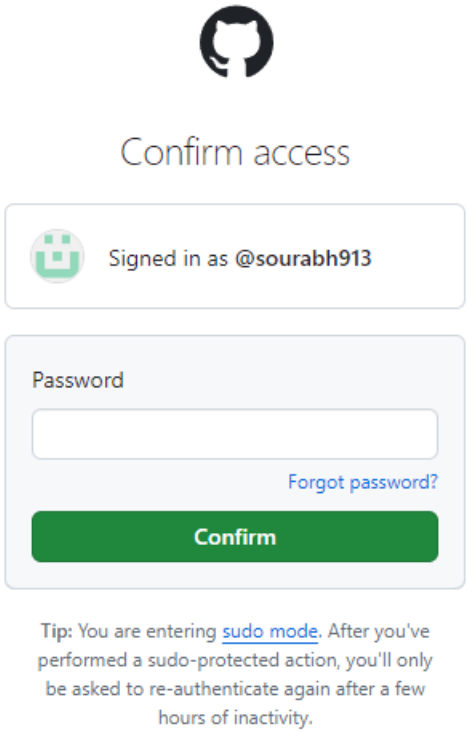



Select Webhooks > Add webhook




Need to put in password for your GitHub account.

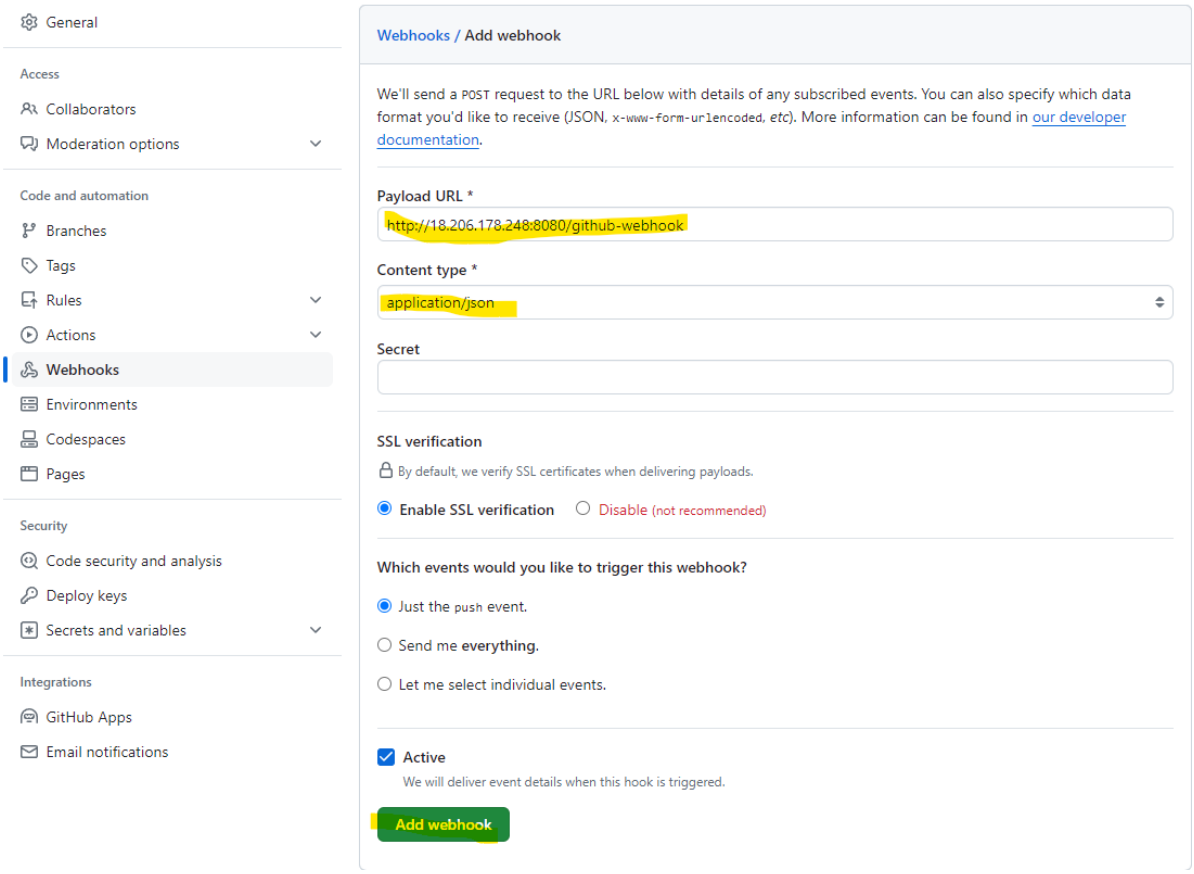
---

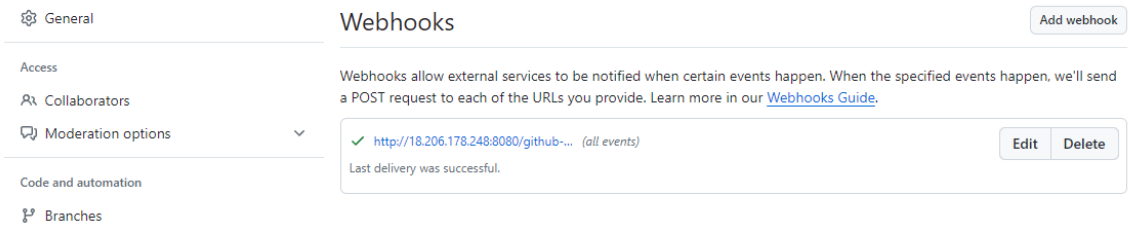



Then under the Payload URL, give the Jenkins URL /github webhook

And under Content type > Select application/json.

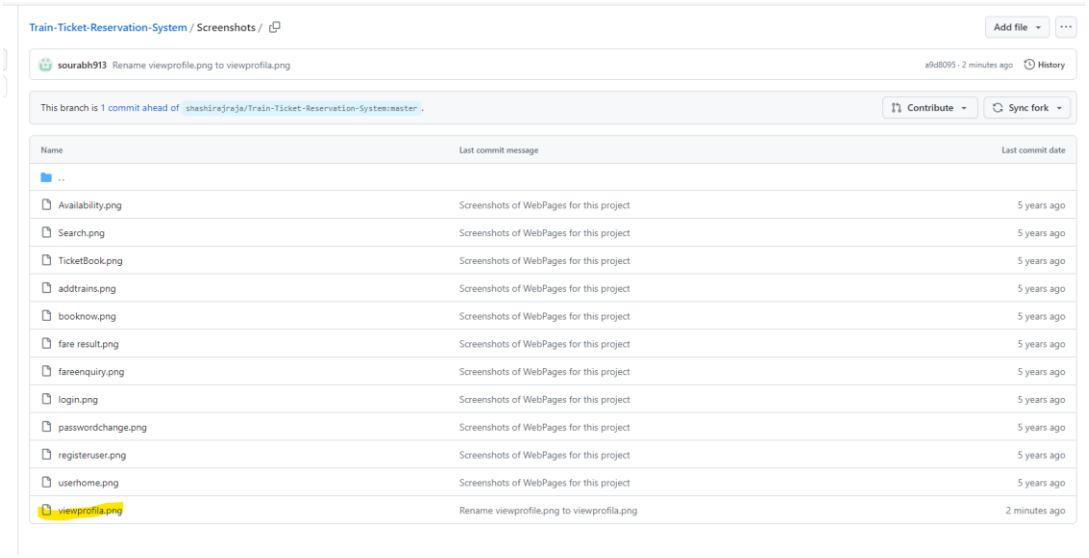
Click Add Webhook

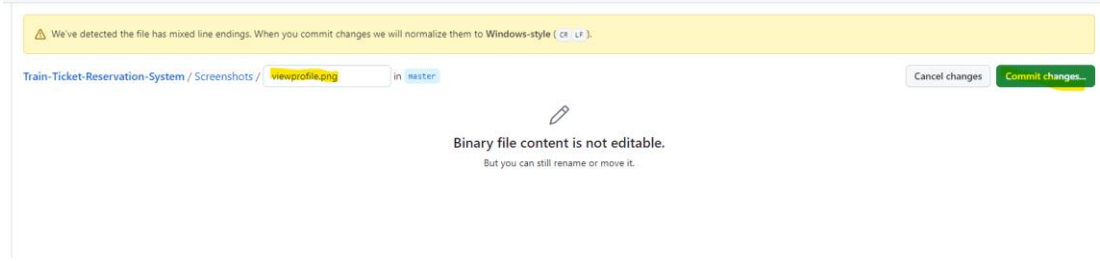


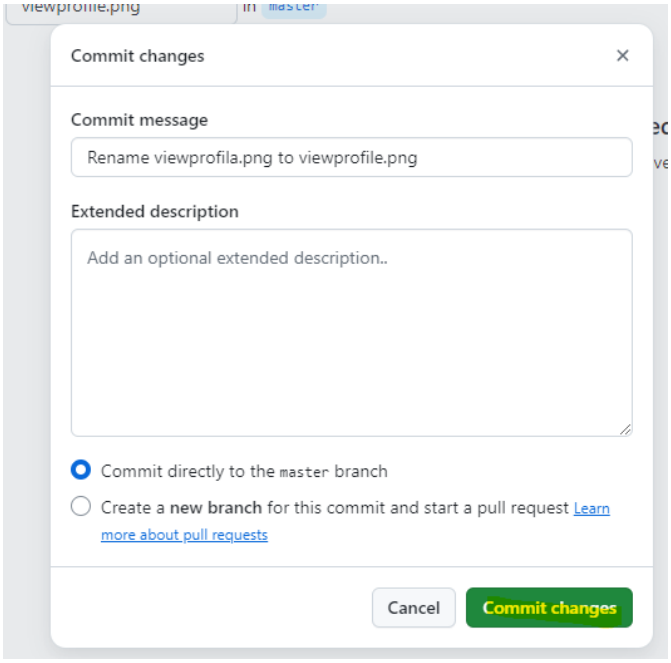




For Example, in my case I made a change under my repo > screenshots > rename viewprofila.png to viewprofile.png

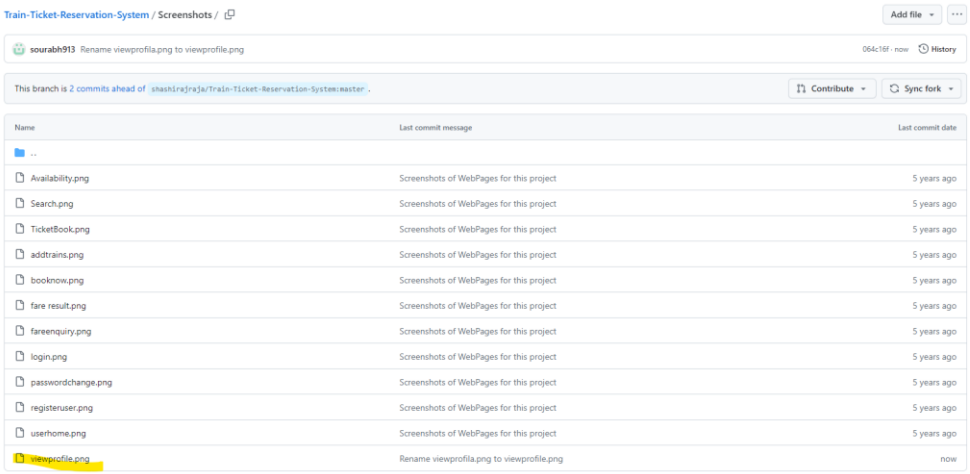
Before Change



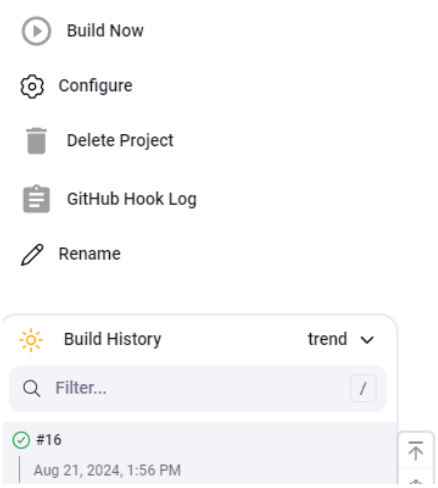




After Change

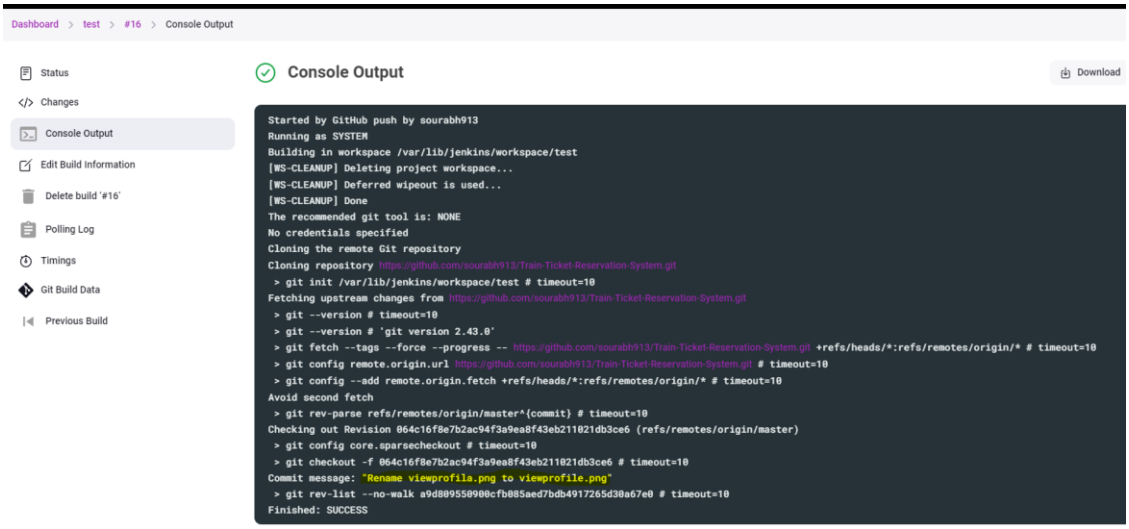



Immediately after the change, we see that build ran in Jenkins.




- 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

We can see the changes in the console output reflected successfully



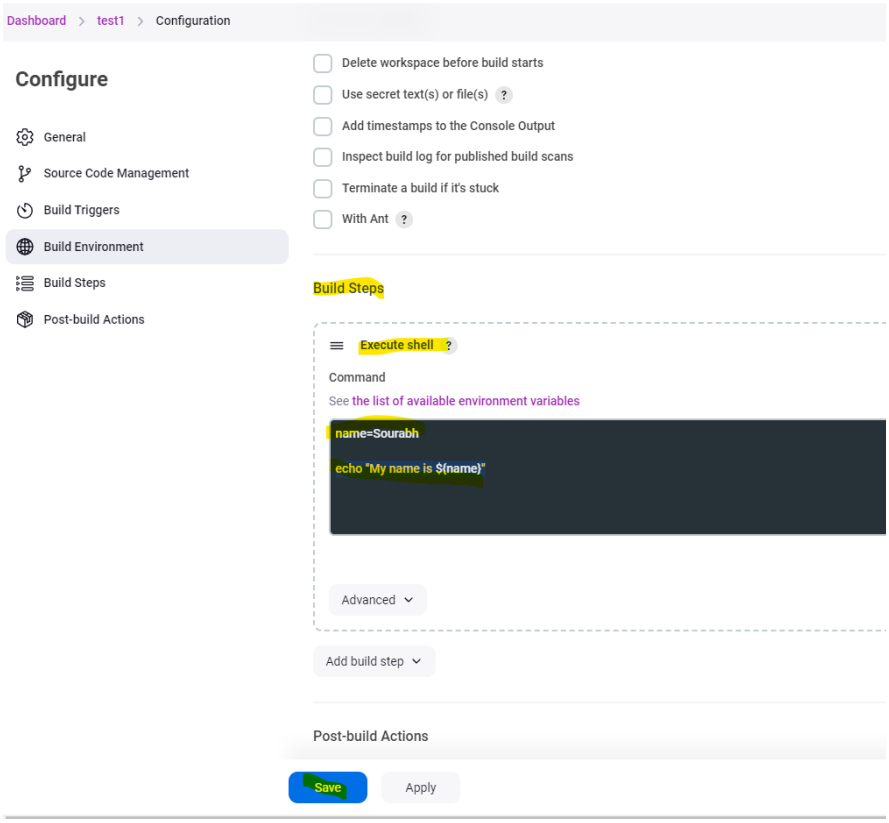
19) Setting Environment Variables in Jenkins. **[Step 16 in Ajay Sir Notes]**

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

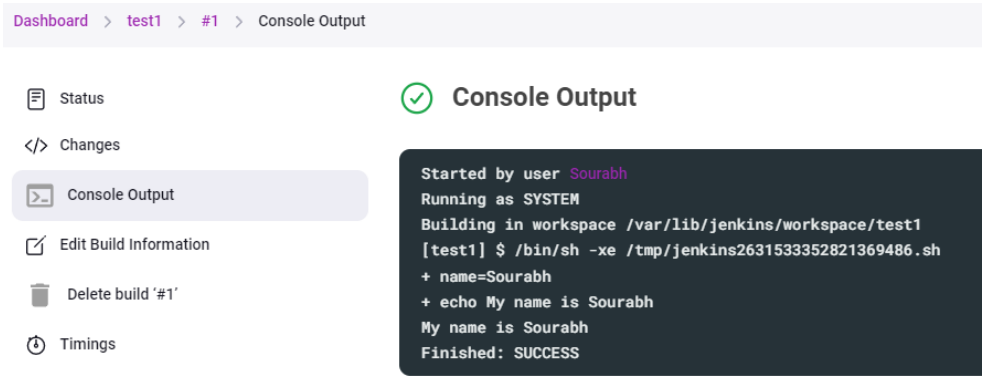
For Example -

name=Sourabh

echo "My name is ${name}"

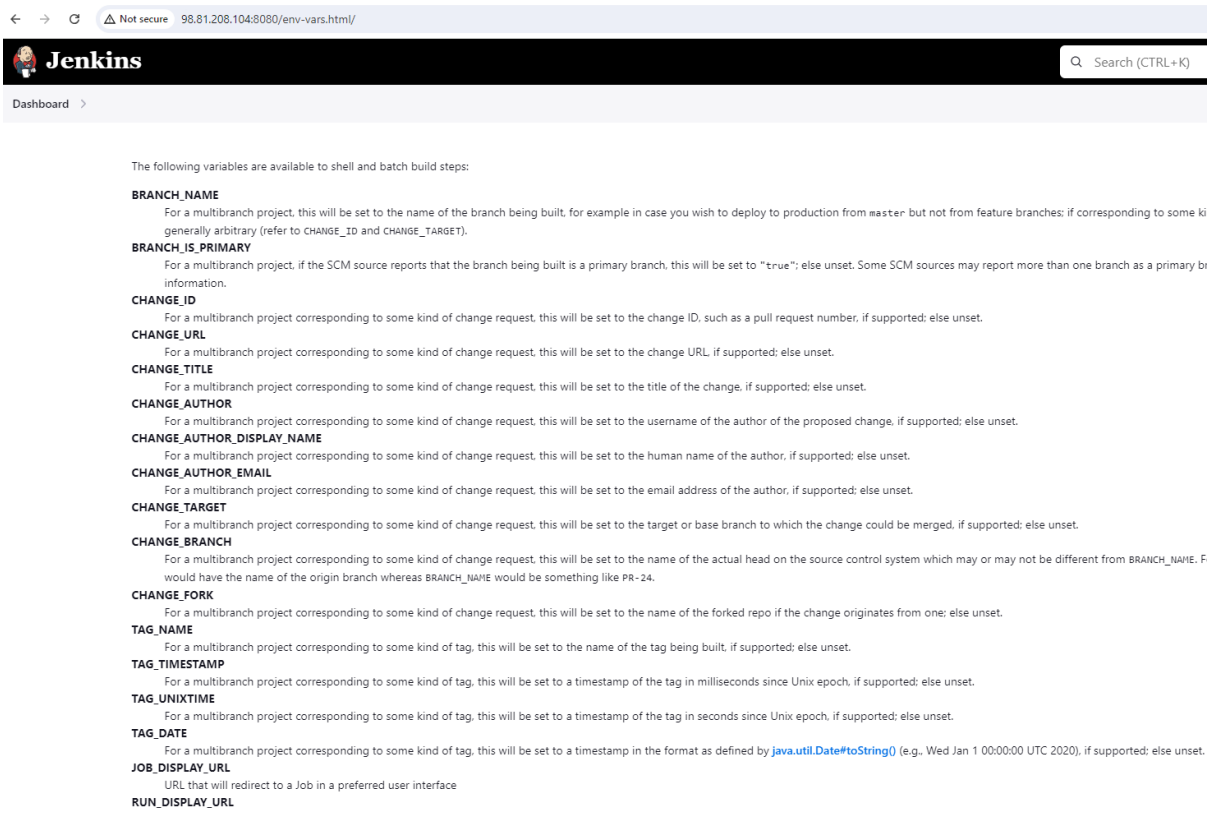



Output of the above build



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

Jenkins URL/env-vars.html

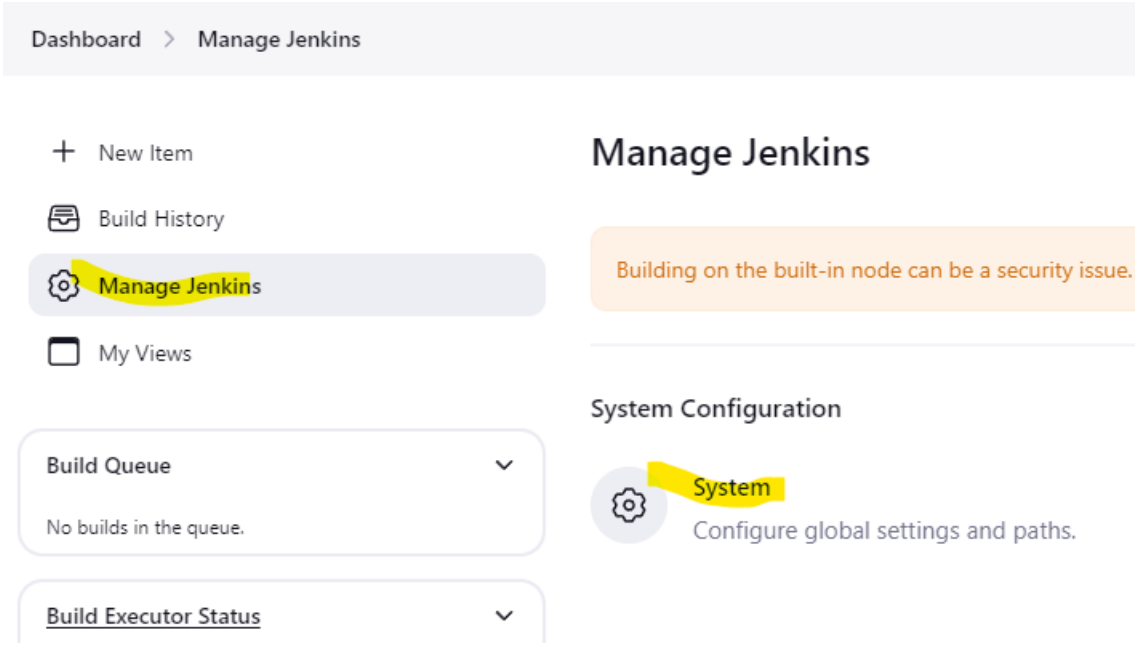



20) Setting Global Environment variables in Jenkins **[Step 17 in Ajay Sir Notes]**

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

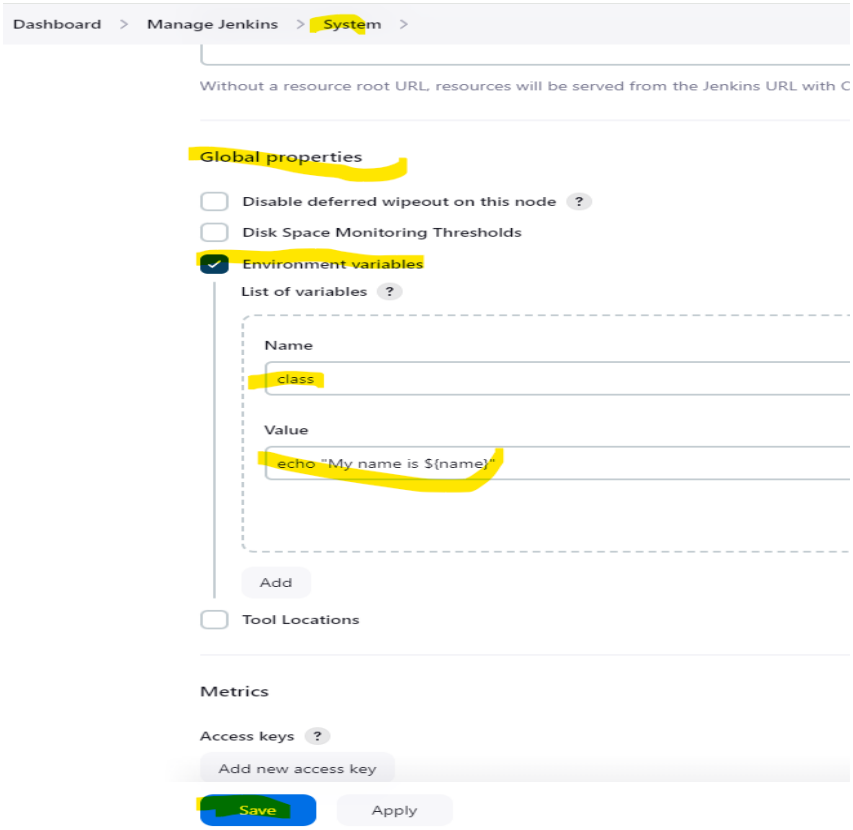
Goto Manage Jenkins > System




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

Name > class

Value > echo "My name is ${name}



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

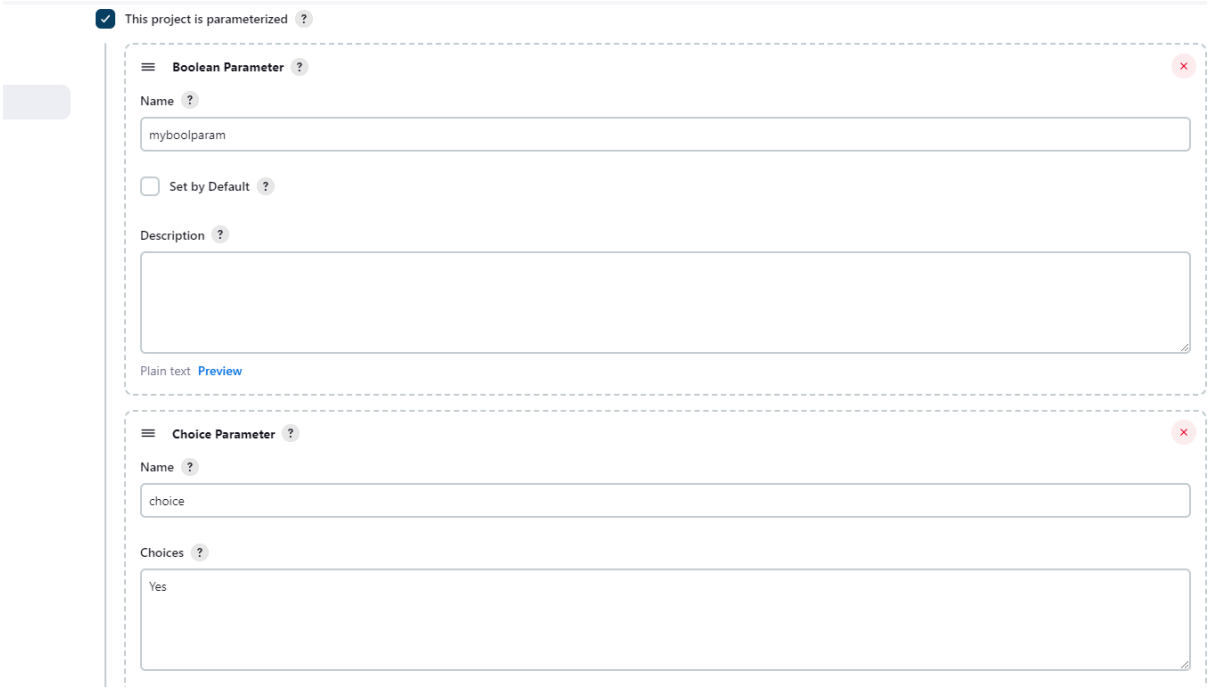
Credentials Parameter

File Parameter

Password Parameter

String Parameter

Run Parameter



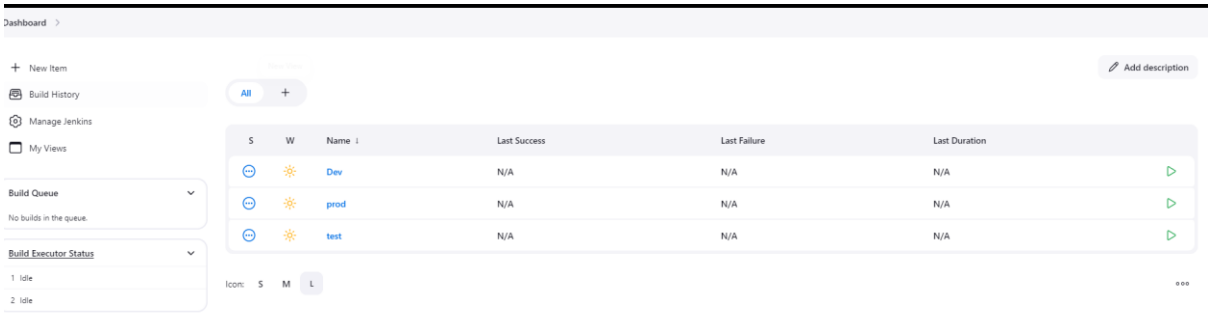

22) How to create a pipeline in Jenkins [**Step 19 in Ajay Sir Notes]**

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

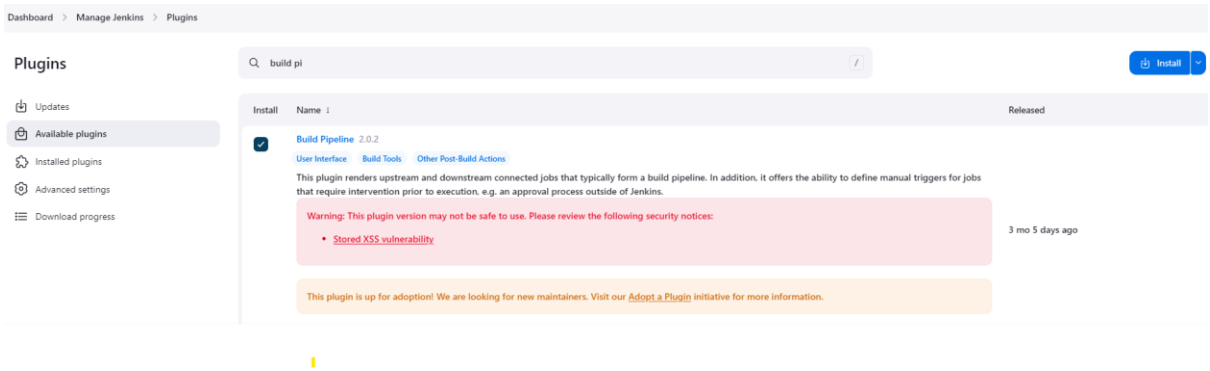
Steps to create a pipeline

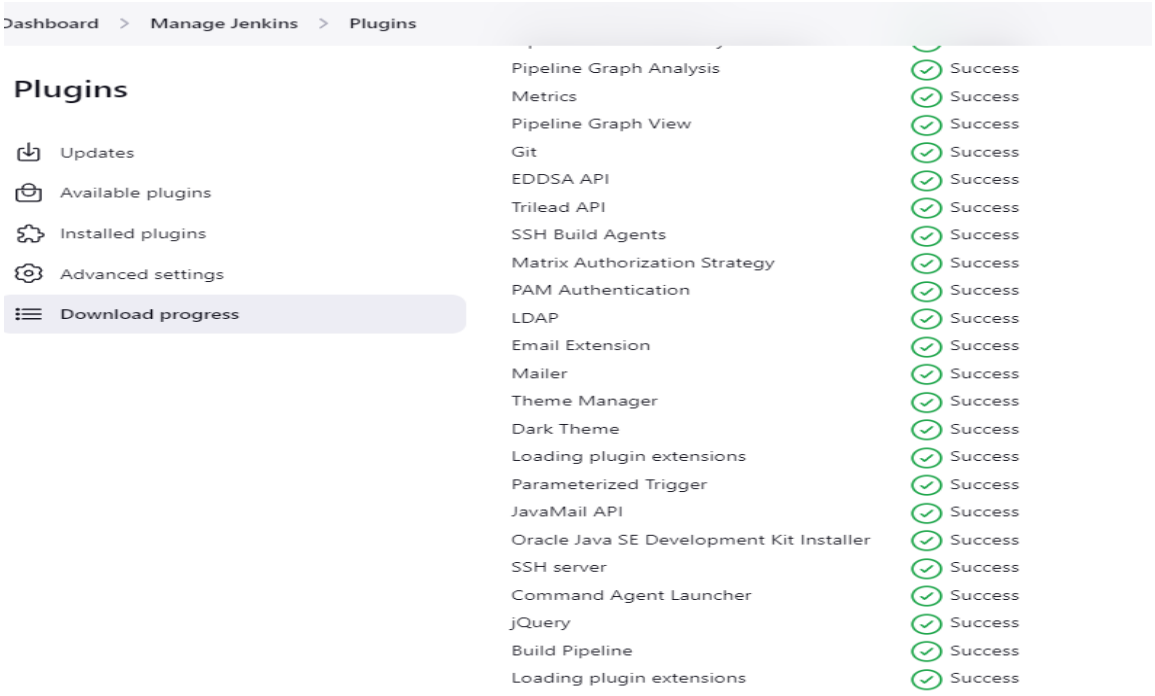
We need to create two jobs at least and create a dependency on each other (Upstream/Downstream)

Example - Dev, Test, Prod

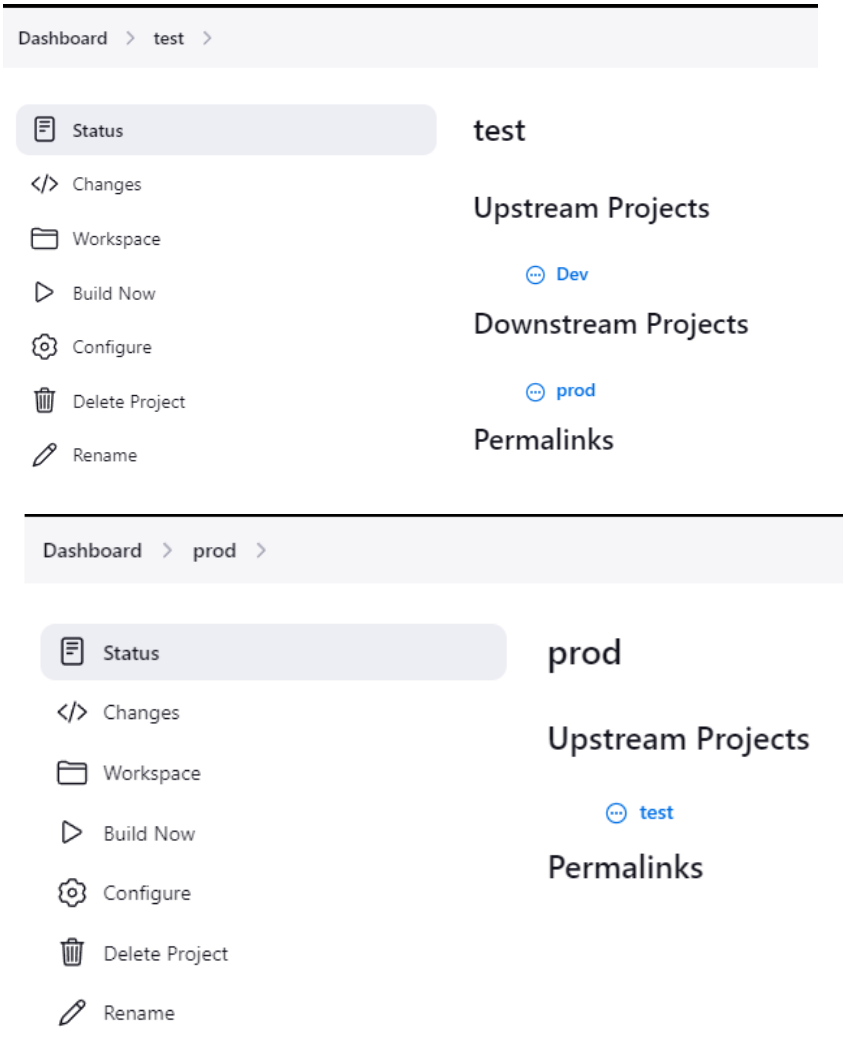



Now Install a plugin build pipeline

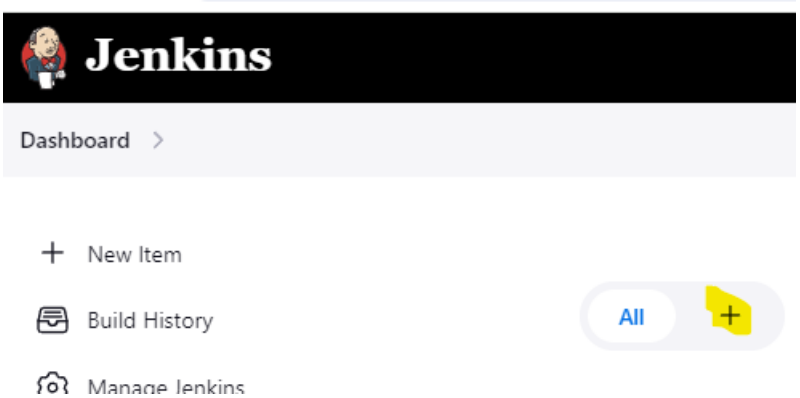


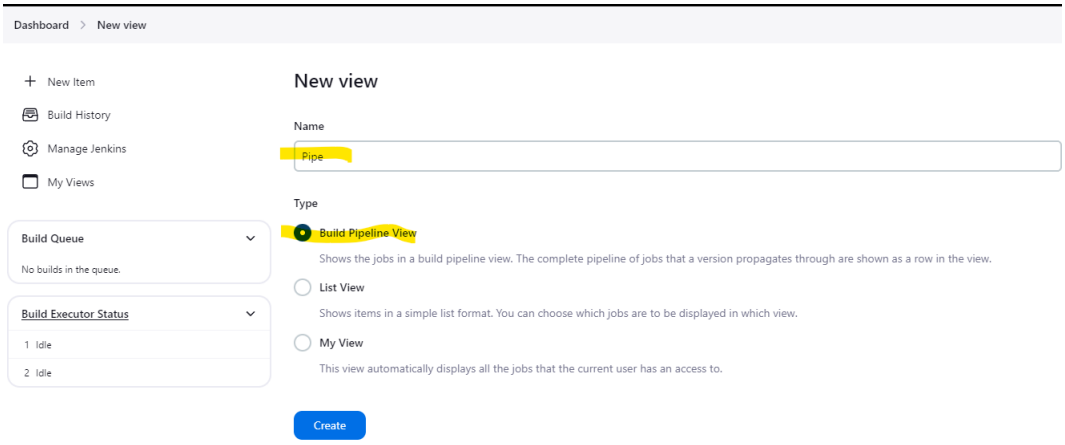
set project dependency on test to dev and prod to test



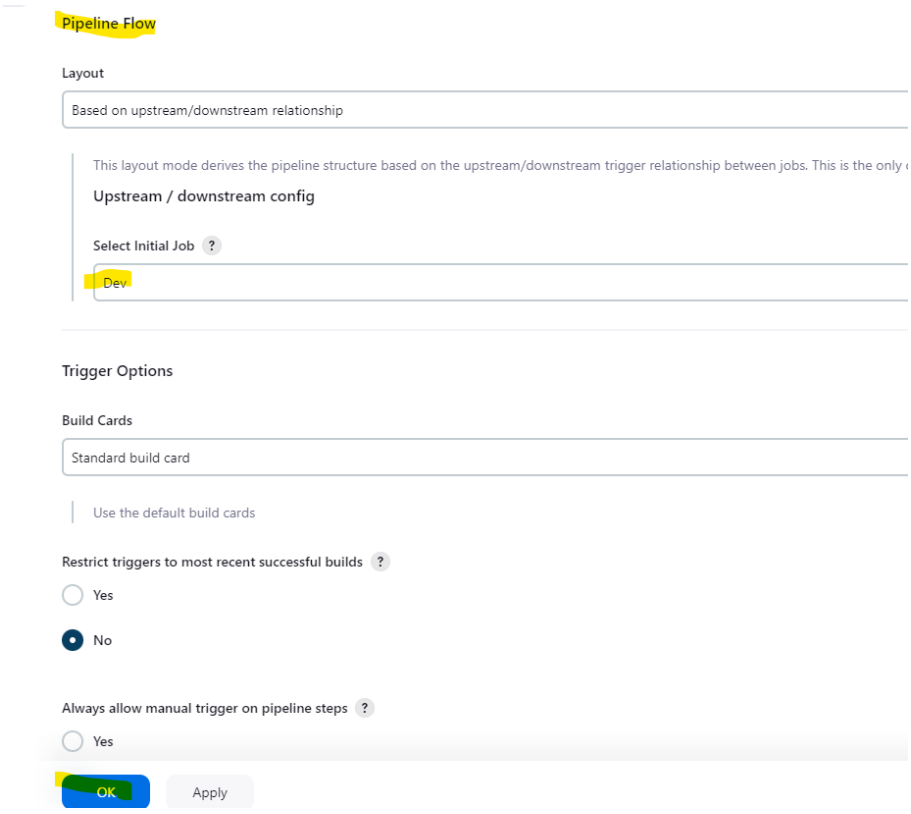

Now click on + sign on dashboard



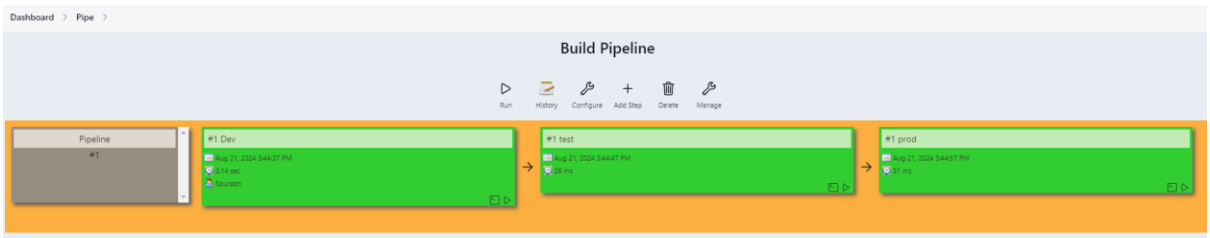
Give Pipeline view a name and select Build Pipeline View




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




Now will run the Build Pipeline and see the output below

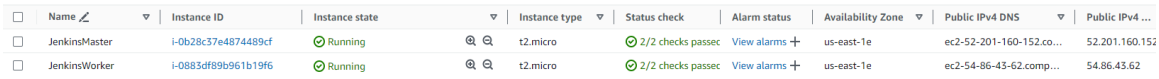


23) Running job on Jenkins Worker/Slave node [**Step 21 in Ajay Sir Notes]**

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 | View alarms | us-east-1e | ec2-52-201-160-152.co... | 52.201.160.152 |
| JenkinsWorker | i-0883df89b961b19f6 | Running | t2.micro | 2/2 checks passed | View alarms | us-east-1e | ec2-54-86-43-62.comp... | 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:VJPDTHSsr2O5HXxI78w/50JzSwxzCNySumk8u8XJGR0 root@JenkinsMaster
The key's randomart image is:
+--[ED25519 256]--+
|         =*o+    |
|         .=B..   |
|        . .ooE.  |
|       . .. .+.. |
|        S oo..=  |
|         *oo=ooo.|
|        . oO+.+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 AAAAC3NzaC1lZDI1NTE5AAAAIGQUP9GF6tuLBLGofjUG4u10ObGXMkYaTl5Ux+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# 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)

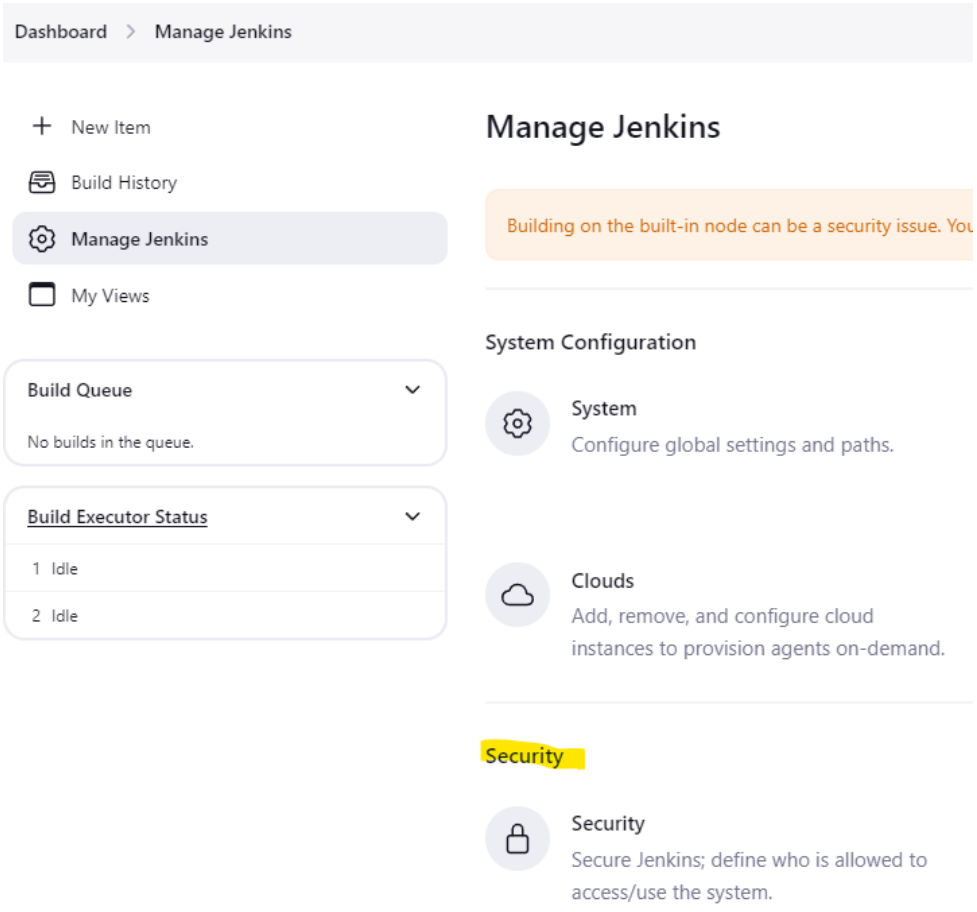
```
root@JenkinsWorker:~/.ssh# cat >> authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGQUP9GF6tuLBLGofjUG4u10ObGXMkYaTl5Ux+mav7s1 root@JenkinsMaster
^C
root@JenkinsWorker:~/.ssh#
root@JenkinsWorker:~/.ssh#
root@JenkinsWorker:~/.ssh# cat authorized_keys
no-port-forwarding,no-agent-forwarding,no-X11-forwarding,command="echo 'Please login as the user \"ubuntu\" rather than the user \"root\".';echo;sleep 10;ex
it 142" ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC/wR6CHuFqnuxDjo2/aoaAvoARTzOMJEU7iMkHRCPO1IESNFVE4hXPIQETnwa2C1sonQYruh+sd3uRHVkA8SEZ4zpqrxv2NafGw5k6R6/G2hll
U3E8YjP6xwAu0GN9Z0sNy9wdmUQqFEL8ANAcl+K66oIUg4of6Sbwr/vcVmuvjai+6dAzLXar5hd2A1dJI2zEyLgApJvu63akps8rTUsDc7NovXPecsEnISLvIQHd/5xweVErVR5fmFKjvPuod6n9LAp6JCcz
bULtR/1KwcujO9FwXpa4hw2yB8vmR3svQMG+UKnkuhGB07cl14p1c4KyNr9haQ/8mRMQvrLb9ngf 22July
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGQUP9GF6tuLBLGofjUG4u10ObGXMkYaTl5Ux+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.

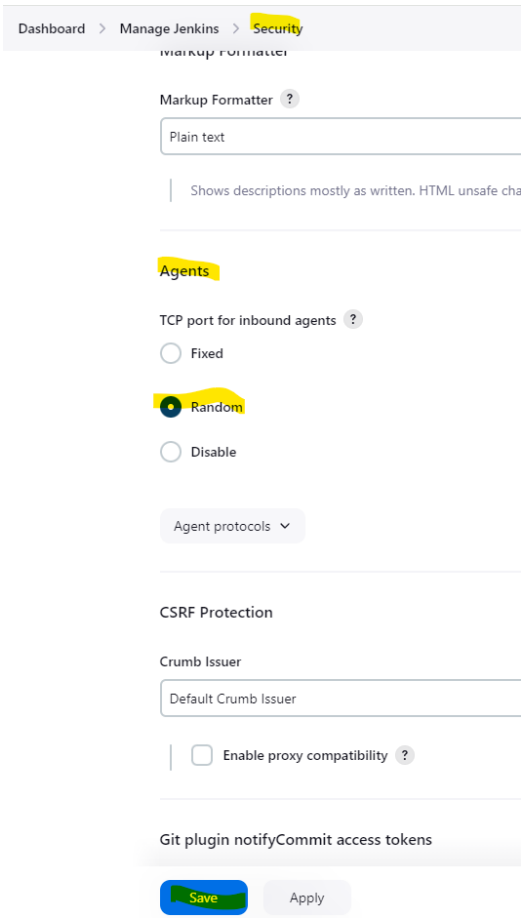
```
root@JenkinsMaster:/home/ubuntu# ssh 54.86.43.62
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * 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.

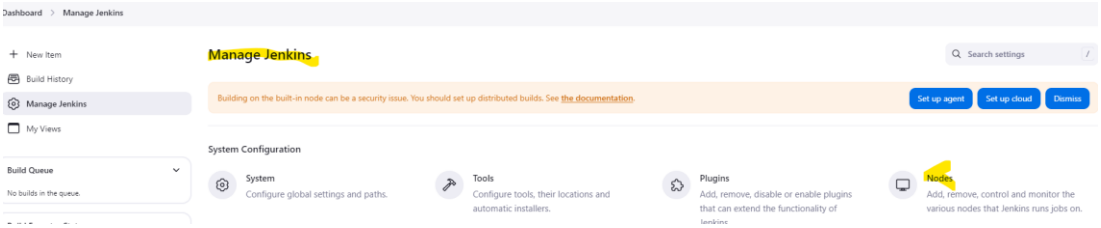
  From Jenkins dashboard, go to Manage Jenkins > Security



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

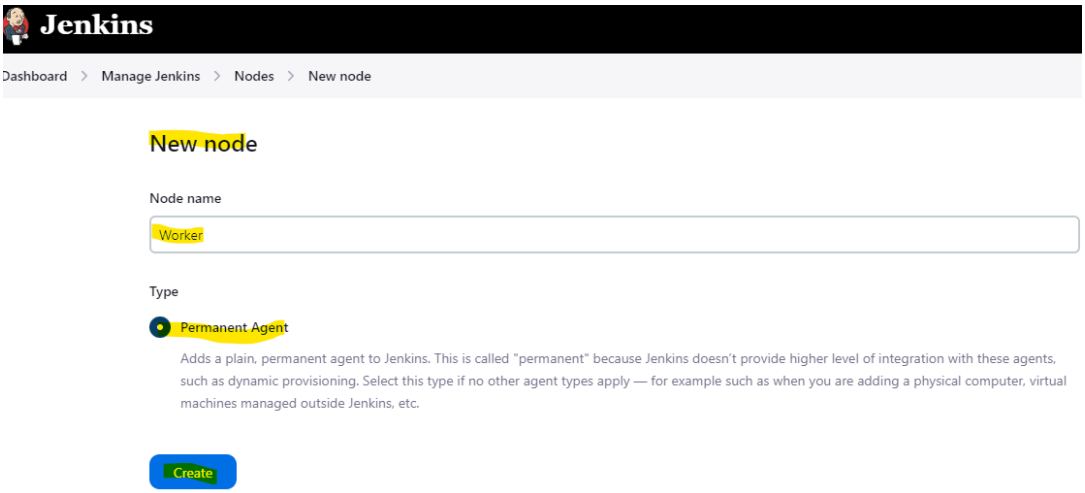



- Now go to Manage Jenkins > Nodes.

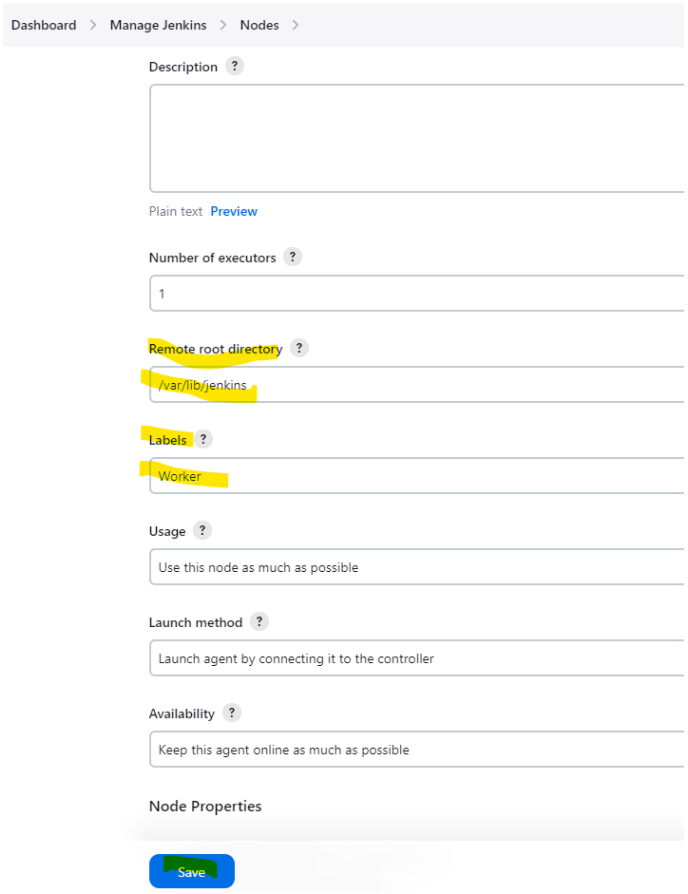



- Click on + New Node.

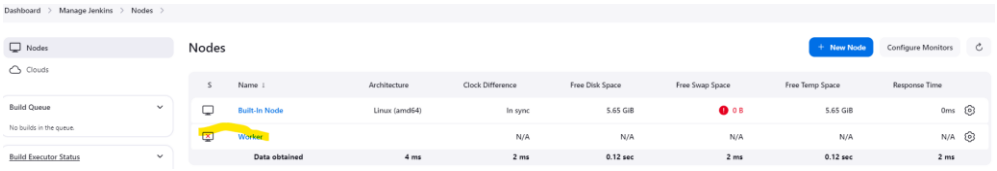
- Give a Name to the node e.g. Worker and select permanent agent and click create.




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



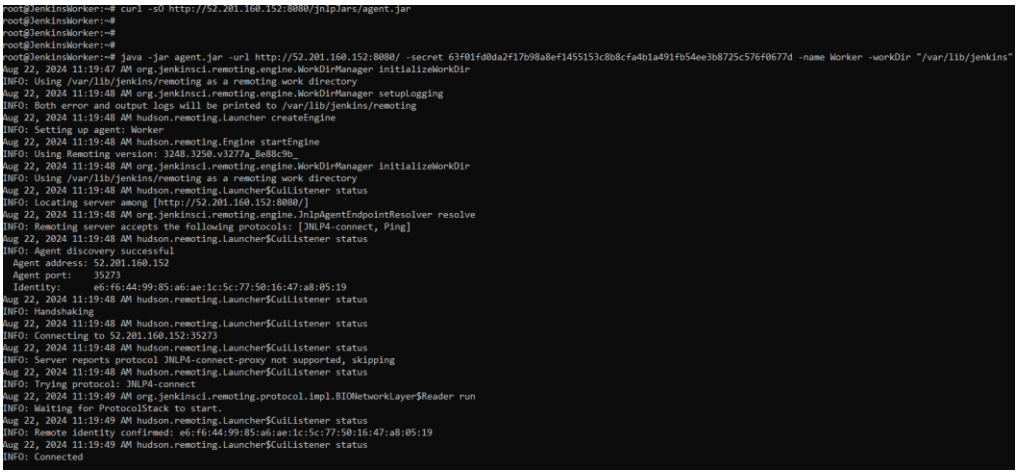

- Now you will see the Node showing added in Nodes dashboard.



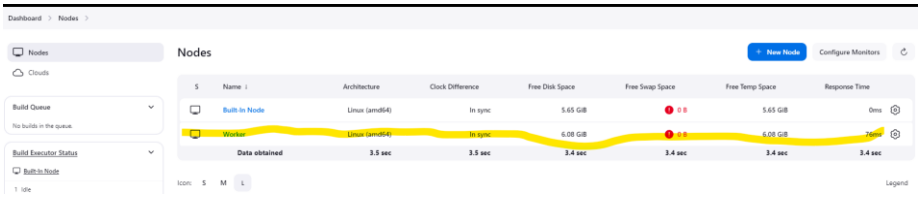

- Now we will click on Worker node.


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


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



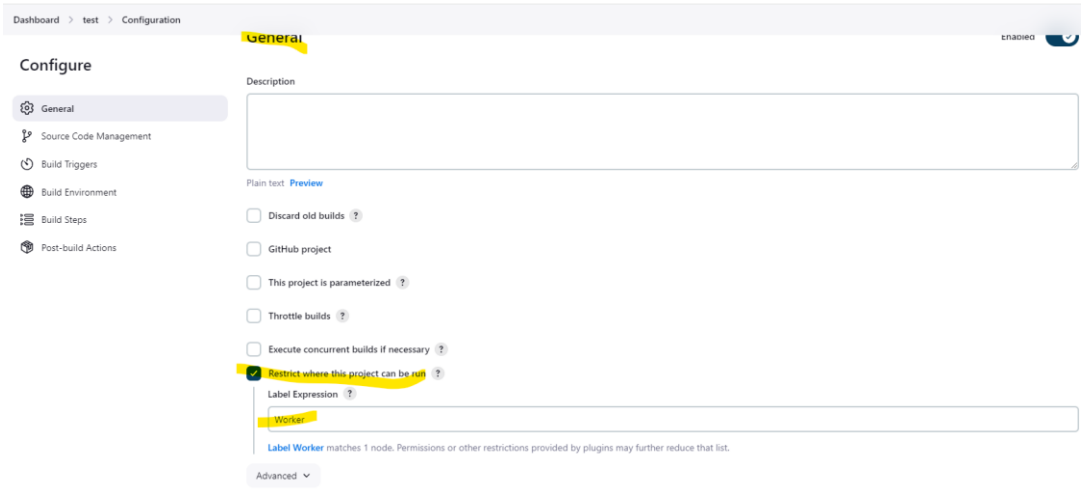

- We can see in the node dashboard that Worker in Online now as a node.



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

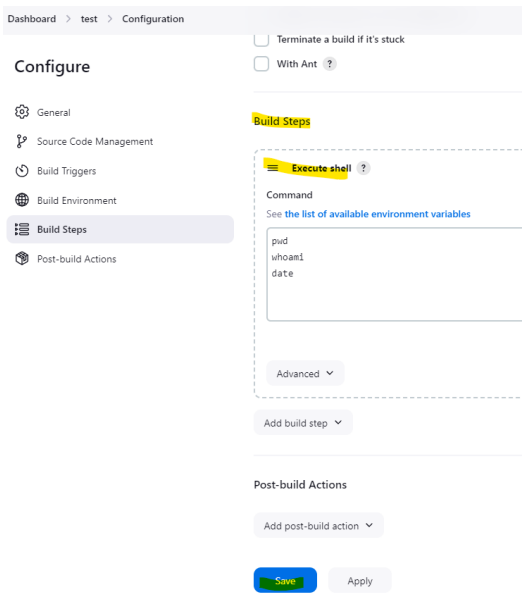
  While configuring, we will select Restriction where this project can be run under General tab.

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

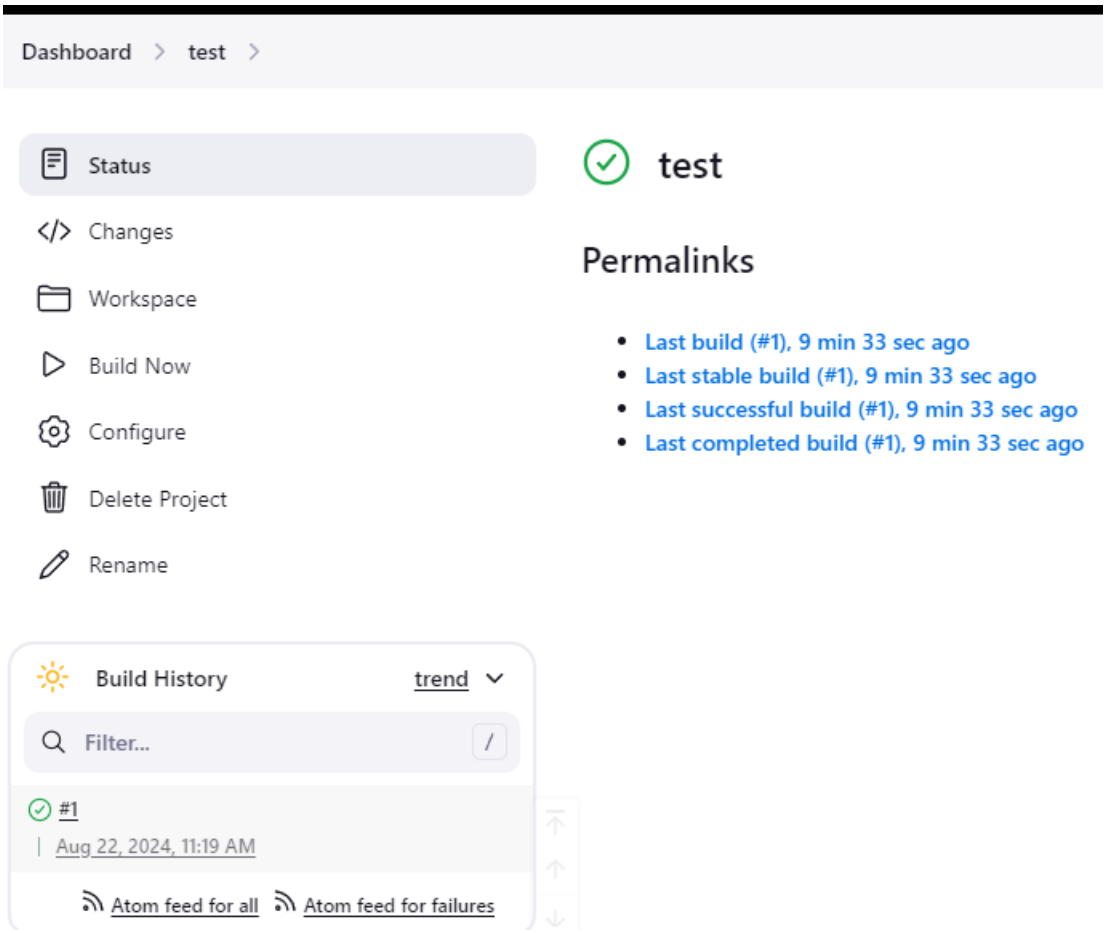


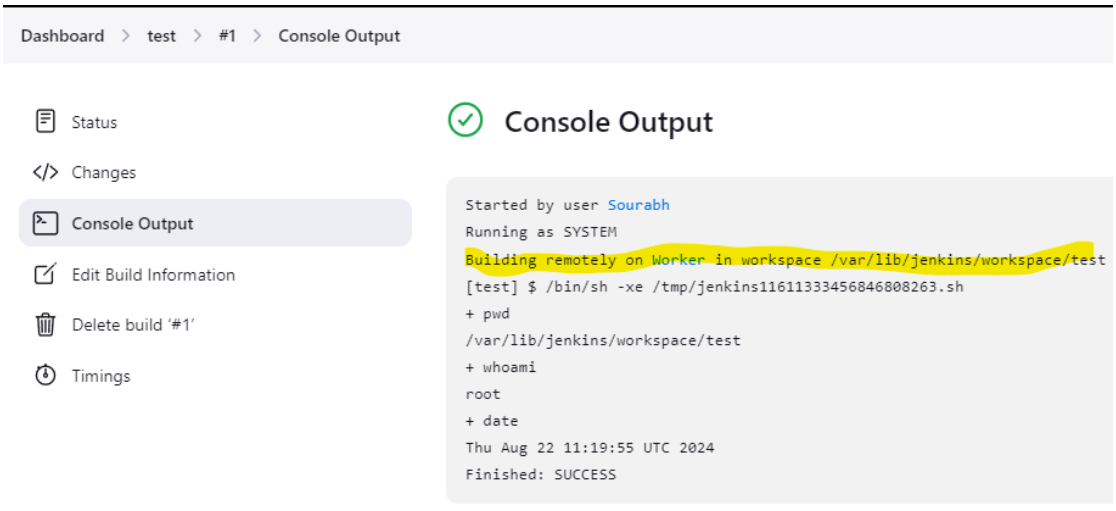

  Then we give some commands which we want to run as an example on the worker node under Build Steps > Execute shell

  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 console output confirms that the build ran on Worker Node.