

1) Configure 2 slave machines in jenkins master.

Here I have one master Server and Two Slave Nodes.

The screenshot shows the AWS CloudWatch Metrics interface. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Instances, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, and Dedicated Hosts. The main area has a title 'Instances (3) Info' with a search bar and a 'Last updated less than a minute ago' indicator. Below this is a table with columns: Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. Three instances are listed: Jenkins_Master (t2.large, running, 2/2 checks passed), Slave_01 (t2.micro, running, 2/2 checks passed), and Slave_02 (t2.micro, running, 2/2 checks passed). At the bottom, there's a modal window titled 'Select an instance' with a close button.

- Connect two slave servers and install java only because these is prerequisite.
- No need to install the Jenkins in the slave servers.
- I am connect slave_01 server and install java-17 version.

```
[root@Slave_01 ~]# yum -y install java-17*
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Resolving Dependencies
--> Running transaction check
--> Package java-17-amazon-corretto.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Processing Dependency: libX11 for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libX1 for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXinerama for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXt for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXrender for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXrandr for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXtst for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: giflib for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: Package java-17-amazon-corretto-devel.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Package java-17-amazon-corretto-headless.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Processing Dependency: jpackage-utils for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: fontconfig for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: dejavu-sans-fonts for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: dejavu-serif-fonts for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: dejavu-sans-mono-fonts for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: alsalib for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: log4j-cve-2021-40228-cve-mitigations for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Package java-17-amazon-corretto-javadoc.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Package java-17-amazon-corretto-jmودs.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Running transaction check
--> Package alsalib.x86_64 0:1.1.4.1-2.amzn2 will be installed
--> Package dejavu-sans-fonts.noarch 0:2.33-6.amzn2 will be installed
--> Processing Dependency: dejavu-fonts-common = 2.33-6.amzn2 for package: dejavu-sans-fonts-2.33-6.amzn2.noarch
--> Package dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2 will be installed
--> Package dejavu-serif-fonts.noarch 0:2.33-6.amzn2 will be installed
--> Package fontconfig.x86_64 0:2.13.0-4.3.amzn2 will be installed
--> Processing Dependency: fontpackages-filesystem for package: fontconfig-2.13.0-4.3.amzn2.x86_64
--> Package giflib.x86_64 0:4.1.6-9.amzn2.0.2 will be installed
--> Processing Dependency: libICE.so.6()(64bit) for package: giflib-4.1.6-9.amzn2.0.2.x86_64
--> Processing Dependency: libSM.so.6()(64bit) for package: giflib-4.1.6-9.amzn2.0.2.x86_64
--> Package javapackages-tools.noarch 0:3.4.1-11.amzn2 will be installed
--> Processing Dependency: python-javapackages = 3.4.1-11.amzn2 for package: javapackages-tools-3.4.1-11.amzn2.noarch
--> Processing Dependency: libxml2 for package: javapackages-tools-3.4.1-11.amzn2.noarch
--> Package libX11.x86_64 0:1.6.7-3.amzn2.0.5 will be installed
--> Processing Dependency: libX11-common >= 1.6.7-3.amzn2.0.5 for package: libX11-1.6.7-3.amzn2.0.5.x86_64
```

- I am connect slave_02 server and install java-17 version.

```
[root@Slave_02 ~]#
[root@Slave_02 ~]# yum -y install java-17*
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Resolving Dependencies
--> Running transaction check
--> Package java-17-amazon-corretto.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Processing Dependency: libX11 for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libX1 for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXinerama for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXt for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXrender for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXrandr for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: libXtst for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: giflib for package: 1:java-17-amazon-corretto-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: Package java-17-amazon-corretto-devel.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Package java-17-amazon-corretto-headless.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Processing Dependency: jpackage-utils for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: fontconfig for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: dejavu-sans-fonts for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: dejavu-serif-fonts for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: dejavu-sans-mono-fonts for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: alsalib for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Processing Dependency: log4j-cve-2021-40228-cve-mitigations for package: 1:java-17-amazon-corretto-headless-17.0.12+7-1.amzn2.1.x86_64
--> Package java-17-amazon-corretto-javadoc.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Package java-17-amazon-corretto-jmودs.x86_64 1:17.0.12+7-1.amzn2.1 will be installed
--> Running transaction check
--> Package alsalib.x86_64 0:1.1.4.1-2.amzn2 will be installed
--> Package dejavu-sans-fonts.noarch 0:2.33-6.amzn2 will be installed
--> Processing Dependency: dejavu-fonts-common = 2.33-6.amzn2 for package: dejavu-sans-fonts-2.33-6.amzn2.noarch
--> Package dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2 will be installed
--> Package dejavu-serif-fonts.noarch 0:2.33-6.amzn2 will be installed
--> Package fontconfig.x86_64 0:2.13.0-4.3.amzn2 will be installed
--> Processing Dependency: fontpackages-filesystem for package: fontconfig-2.13.0-4.3.amzn2.x86_64
--> Package giflib.x86_64 0:4.1.6-9.amzn2.0.2 will be installed
--> Processing Dependency: libICE.so.6()(64bit) for package: giflib-4.1.6-9.amzn2.0.2.x86_64
--> Processing Dependency: libSM.so.6()(64bit) for package: giflib-4.1.6-9.amzn2.0.2.x86_64
```

Steps to be done on slave

- Create ssh-keygen
- cat id_rsa.pub > authorized_keys

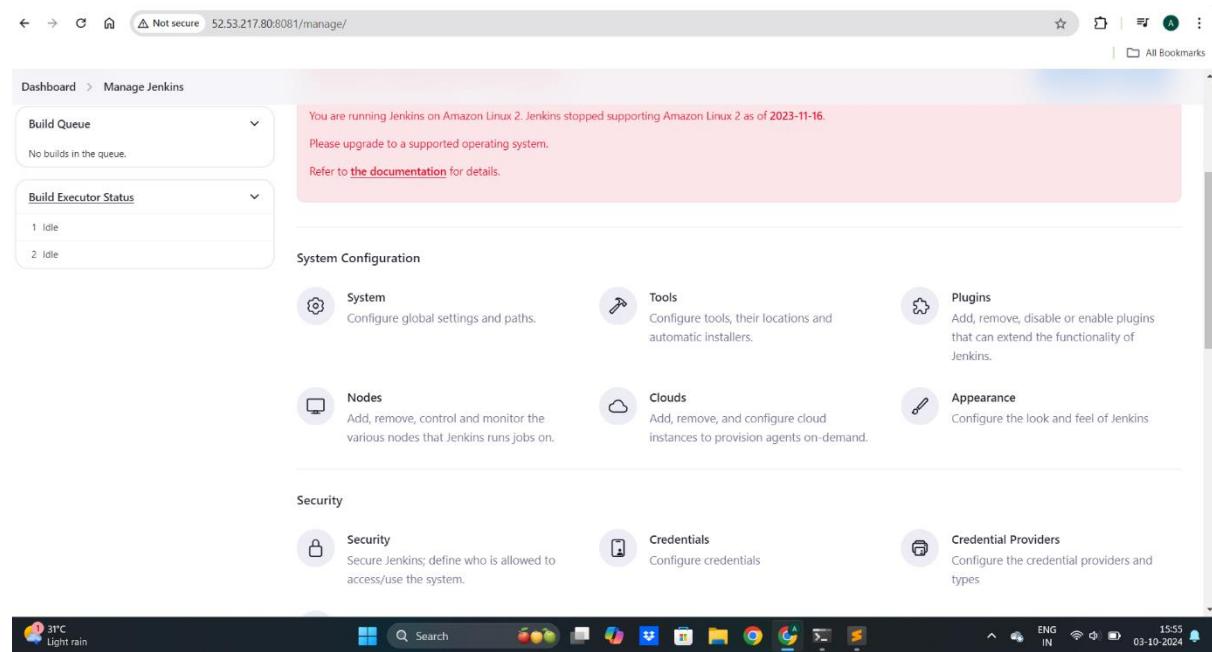
chmod 700 authorized_keys

Next Steps to be done on master machine:

- Login to master machine
- switch to root user.
- mkdir -p /var/lib/jenkins/.ssh
- cd /var/lib/jenkins/.ssh
- ssh-keyscan -H SLAVE-NODE-IP-OR-HOSTNAME >>/var/lib/jenkins/.ssh/known_hosts
- # ssh-keyscan -H 172.31.38.42 >>/var/lib/jenkins/.ssh/known_hosts
- chown jenkins:jenkins known_hosts
- #we need to change the owner as we ran ssh-keyscan command using “root” user.
- # default user of Jenkins will be “jenkins”
- chmod 700 known_hosts

these all configurations Done Then go to the jenkins GUI

Go to the manage Jenkins , and click on Nodes.



Here click on the New Node.

The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main area is titled 'Nodes' and contains a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. One node is listed: 'Built-In Node' (Linux (amd64)), Data obtained 24 min, In sync, 27.10 GiB free disk space, 0 B free swap space, 27.10 GiB free temp space, 0ms response time, and 24 min clock difference. A 'New Node' button is located at the top right of the table area.

Give Node name and select permanent Agent.

Click on create.

The screenshot shows the 'New node' creation form. The 'Node name' field is filled with 'Slave_01'. The 'Type' section has a radio button selected for 'Permanent Agent'. A tooltip explains: '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.' A 'Create' button is at the bottom.

- If you want Build Executors extra increase.
- Remote root directory ---- give path --/home/ec2-user
- Label as Java

The screenshot shows the 'Configure' page for the node 'Slave_01'. The 'Name' field is set to 'Slave_01'. The 'Number of executors' is set to '2'. The 'Remote root directory' is set to '/home/ec2-user'. The 'Labels' field contains 'Java'. On the left sidebar, there are links for Status, Delete Agent, Configure (which is selected), Build History, Load Statistics, Script Console, Log, System Information, and Disconnect. A 'Build Executor Status' section at the bottom indicates 'built-in node (0 of 2 executors busy)'.

Launch method – launch agents via SSH

Host --- give here you slave IP

Under the credentials click on add.

Dashboard > Manage Jenkins > Nodes >

Launch method ?
Launch agents via SSH

Host ?
3.101.38.181

Credentials ?
- none -

+ Add ▾

① The selected credentials cannot be found

Host Key Verification Strategy ?
Known hosts file Verification Strategy

Advanced ▾

Select ---- SSH username with Private key.

Scope ---- Global and Give description.

User name --- ec2-user

Dashboard > Manage Jenkins > Nodes >

Launch

Jenkins Credentials Provider: Jenkins

SSH Username with private key

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

ID ?
Slave_01

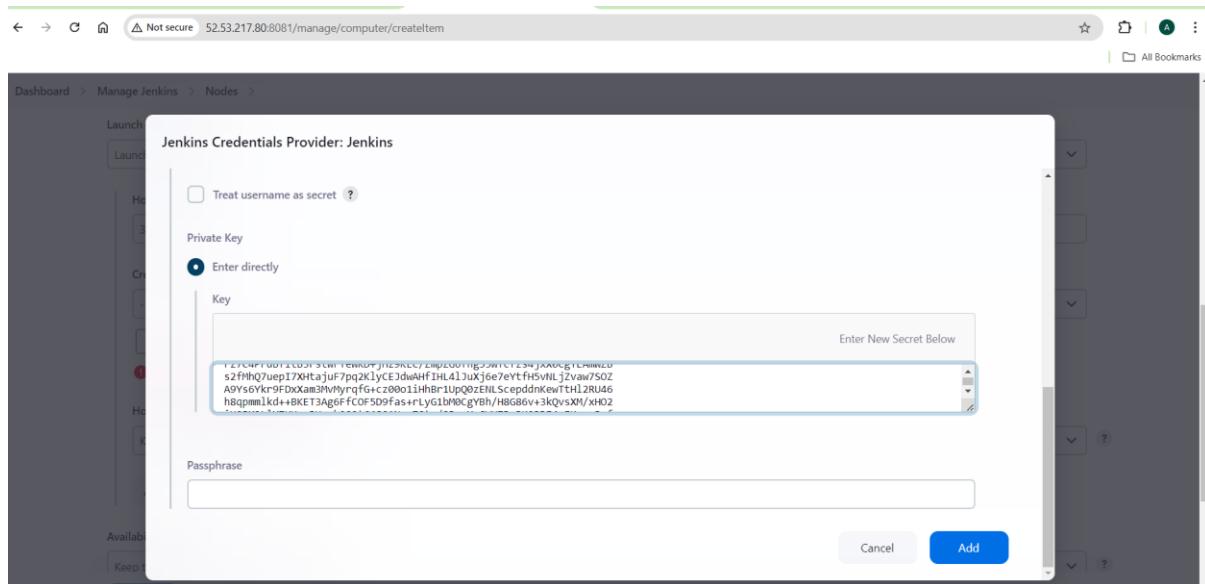
Description ?
Slave_01

Username
ec2-user

Treat username as secret ?

Here click on enter directly and paste the pem key.

And click on add.

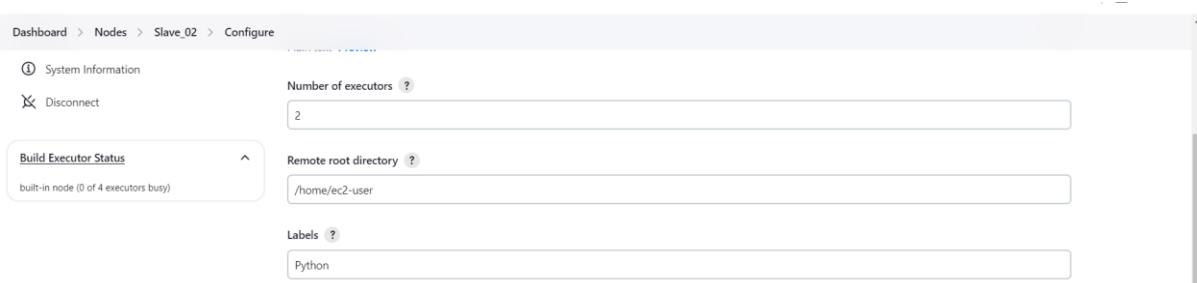


After that select the select credential and save.

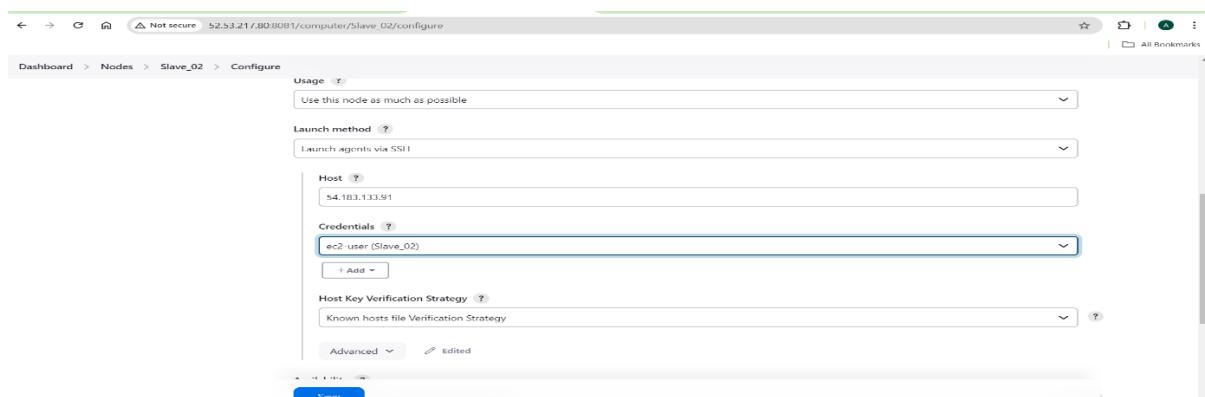
As it's create the second slave node. but need give some changes.

Give name --- Slave-02

Label is --- python

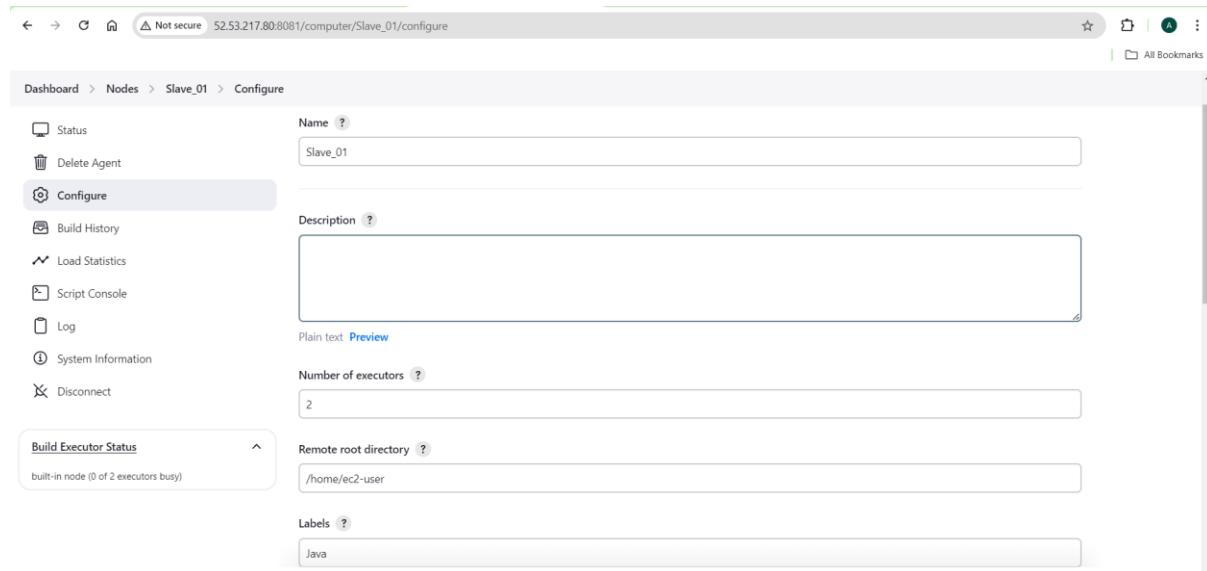


Give credential of slave Node and click on save.



Now go to jobs and attach labels.

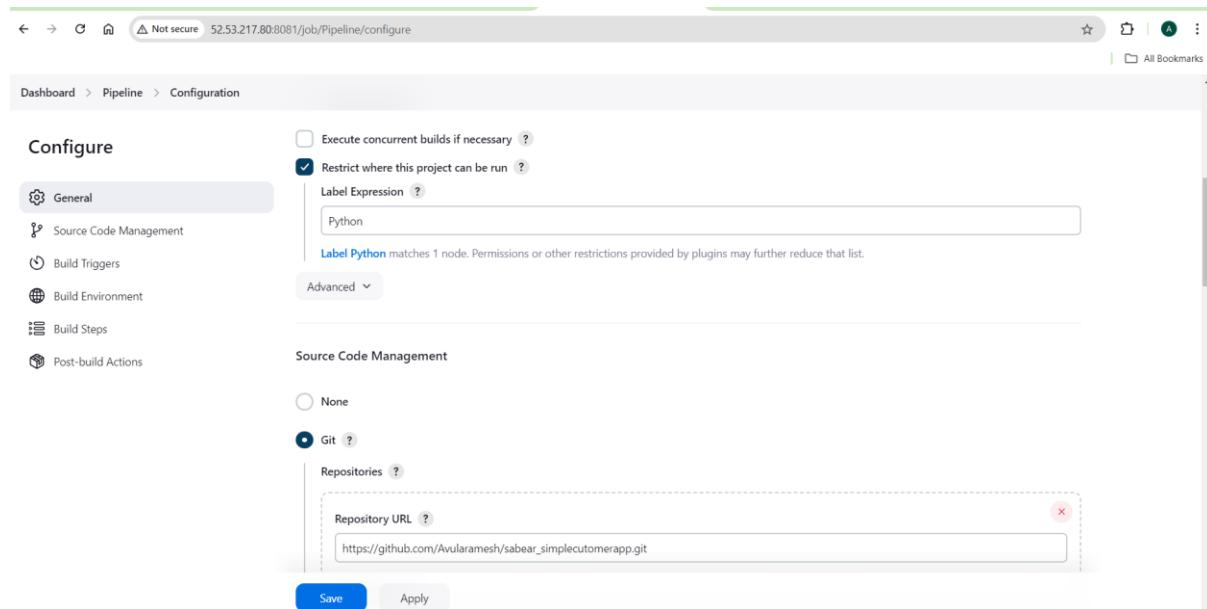
The below one is freestyle job attached the Java label.



Just click on save.

Next go to the another Job.

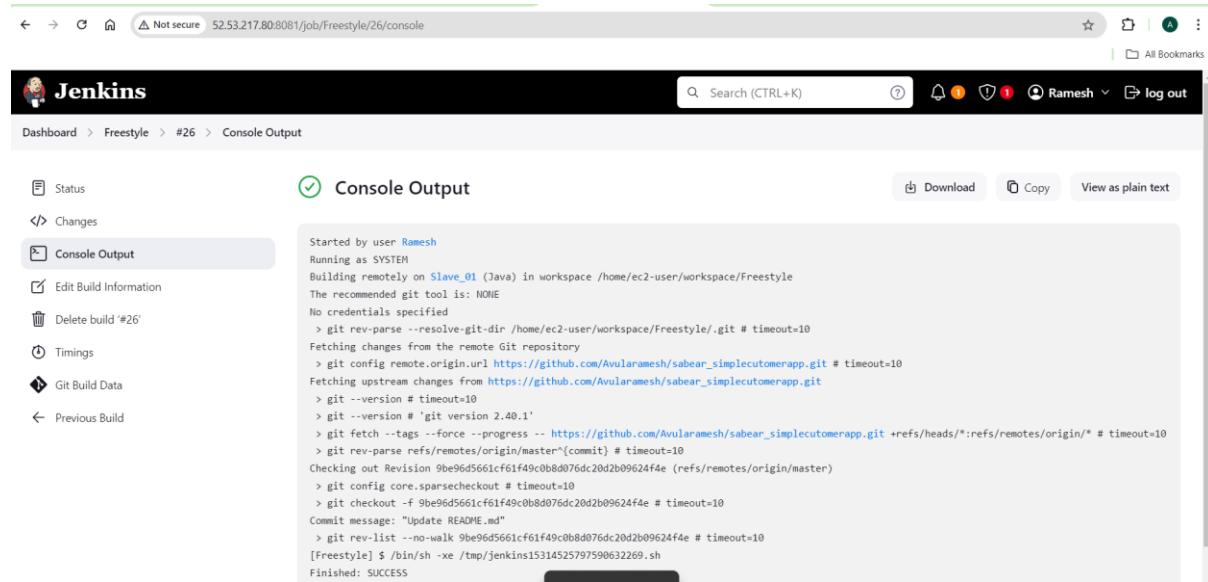
The below one is Pipeline job attached the python label.



Make the branches are main only.

Now try to build jobs.

The below one is Freestyle Job build in slave_01



A screenshot of a web browser showing the Jenkins console output for a Freestyle job named #26. The title bar says "Console Output". The main content area shows the command-line output of the build process, which includes cloning a GitHub repository, running tests, and updating the README file. The build status is listed as "SUCCESS".

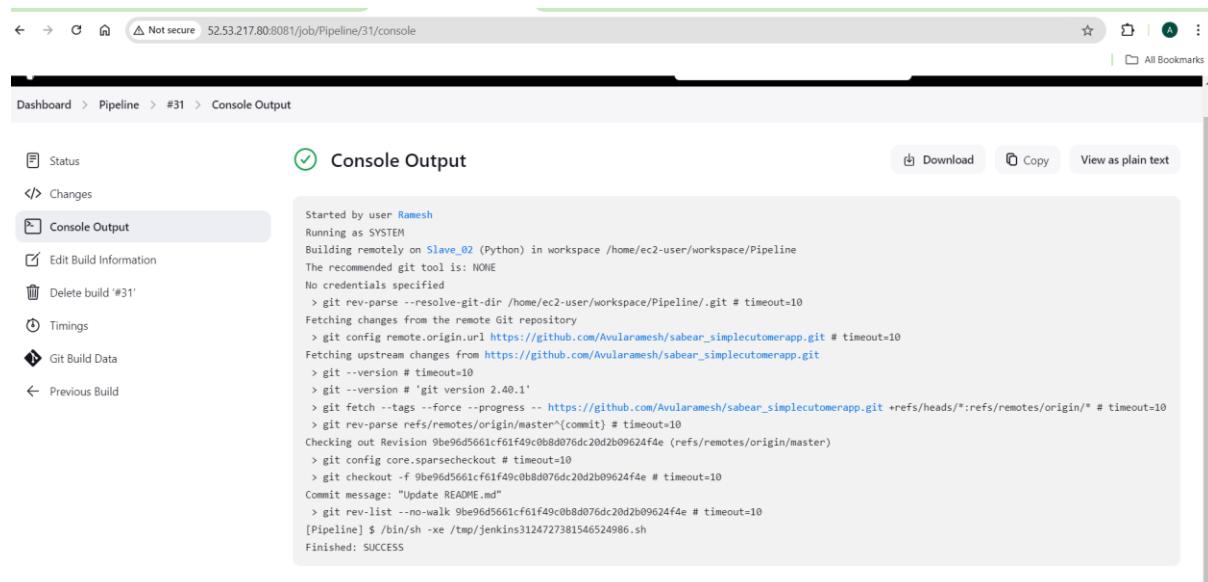
```
Started by user Ramesh
Running as SYSTEM
Building remotely on Slave_01 (Java) in workspace /home/ec2-user/workspace/Freestyle
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /home/ec2-user/workspace/Freestyle/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Avularamesh/sabear_simplecutomerapp.git # timeout=10
Fetching upstream changes from https://github.com/Avularamesh/sabear_simplecutomerapp.git
> git --version # timeout=10
> git -version # 'git' version 2.40.1'
> git fetch -tags --force --progress -- https://github.com/Avularamesh/sabear_simplecutomerapp.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 9be96d5661cf61f49c0b8d076dc20d2b09624f4e (refs/remotes/origin/master)
> git checkout -f 9be96d5661cf61f49c0b8d076dc20d2b09624f4e # timeout=10
Commit message: "Update README.md"
> git rev-list --no-walk 9be96d5661cf61f49c0b8d076dc20d2b09624f4e # timeout=10
[Freestyle] $ /bin/sh -x /tmp/jenkins1514525797590632269.sh
Finished: SUCCESS
```

Here I went slave node.

Here my code is there.

```
[ec2-user@Slave_01 Freestyle]$ pwd
/home/ec2-user/workspace/Freestyle
[ec2-user@Slave_01 Freestyle]$ ls
build build.xml Jenkinsfile pom.xml README.md Service-API.dockerfile sonar-project.properties src WebContent XMLSave.txt
[ec2-user@Slave_01 Freestyle]$ |
```

The below one is Pipeline Job build in slave_02



A screenshot of a web browser showing the Jenkins console output for a Pipeline job named #31. The title bar says "Console Output". The main content area shows the command-line output of the build process, which includes cloning a GitHub repository, running tests, and updating the README file. The build status is listed as "SUCCESS".

```
Started by user Ramesh
Running as SYSTEM
Building remotely on Slave_02 (Python) in workspace /home/ec2-user/workspace/Pipeline
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /home/ec2-user/workspace/Pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Avularamesh/sabear_simplecutomerapp.git # timeout=10
Fetching upstream changes from https://github.com/Avularamesh/sabear_simplecutomerapp.git
> git --version # timeout=10
> git -version # 'git' version 2.40.1'
> git fetch -tags --force --progress -- https://github.com/Avularamesh/sabear_simplecutomerapp.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 9be96d5661cf61f49c0b8d076dc20d2b09624f4e (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 9be96d5661cf61f49c0b8d076dc20d2b09624f4e # timeout=10
Commit message: "Update README.md"
> git rev-list --no-walk 9be96d5661cf61f49c0b8d076dc20d2b09624f4e # timeout=10
[Pipeline] $ /bin/sh -x /tmp/jenkins3124727381546524986.sh
Finished: SUCCESS
```

Here my code is there.

```
[ec2-user@Slave_02 Pipeline]$ pwd  
/home/ec2-user/workspace/Pipeline  
[ec2-user@Slave_02 Pipeline]$ ls  
build.xml Jenkinsfile pom.xml README.md Service-API.dockerfile sonar-project.properties src WebContent XMLSave.txt  
[ec2-user@Slave_02 Pipeline]$ |
```

#####Task is Done#####

2) Configure webhooks to jenkins job.

To create webhooks

The screenshot shows a GitHub repository page for 'sabear_simplecumerapp'. The 'Settings' tab is highlighted with a red box. The page displays a list of commits from 'betawins' and details about the repository's activity, releases, packages, and languages.

Here Payload URL ---- /Jenkins-url/github-webhook/

I am selecting the – send me every thing.

And click on Add webhook.

The screenshot shows the 'Webhooks' configuration page on GitHub. The 'Payload URL' field contains 'http://52.53.217.80:8081/github-webhook/'. The 'Content type' is set to 'application/x-www-form-urlencoded'. The 'Secret' field is empty. Under 'SSL verification', 'Enable SSL verification' is selected. In the 'Which events would you like to trigger this webhook?' section, 'Send me everything.' is selected. The 'Active' checkbox is checked. A green 'Add webhook' button is at the bottom.

Here webhooks is created.

The screenshot shows the GitHub settings page for a repository named 'sabear_simplecustomerapp'. The 'Webhooks' tab is selected. A message at the top says, 'Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at <https://docs.github.com/webhooks/ping-event>'. Below this, there is a list of webhooks with one entry: 'http://52.53.217.80:8081/github-webhook/ (all events)'. An 'Edit' button and a 'Delete' button are next to this entry. On the left sidebar, the 'Webhooks' option is highlighted under the 'Code and automation' section.

Now go to the Jenkins Job.

Click on these check box and save.

The screenshot shows the Jenkins configuration page for a 'Freestyle' job. Under the 'Build Triggers' section, the 'GitHub hook trigger for GITScm polling' checkbox is checked and highlighted with a red border. Other options like 'Trigger builds remotely' and 'Build after other projects are built' are also present but not selected. At the bottom of the configuration page, there are 'Save' and 'Apply' buttons, with 'Save' being highlighted with a red border.

Actually in these readme.md file 7 points is there I am adding the 8 point.

The screenshot shows the GitHub interface for editing a README.md file. The file contains a list of steps for executing a pipeline. The 8th point, 'i am adding the 8th point.', has been added at the end of the list. The code block is as follows:

```
## How to execute this pipeline
1) Sonarscanner
2) Sonarqube server
3) jenkins server
4) nexus server
5) change the credetials of nexus server as per our requirement
6) create a new repo in nexus with version policy as "snapshot"
7) modify the pipeline as per our sonarscanner name and sonarqube server name
8) i am adding the 8th point.

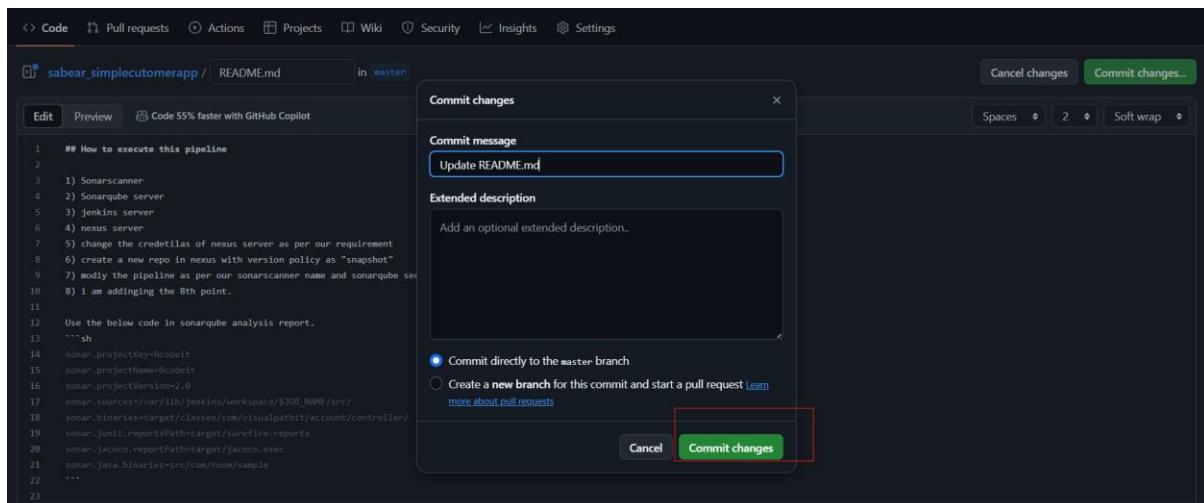
Use the below code in sonarqube analysis report.
```sh
sonar.projectKey=Nodeit
sonar.projectName=Nodeit
sonar.projectVersion=2.0
sonar.sources=/var/lib/jenkins/workspace/$JOB_NAME/src/
sonar.binaries=target/classes/com/visualpathit/account/controller/
sonar.junit.reportsPath=target/surefire-reports
sonar.jacoco.reportPath=target/jacoco.exec
sonar.java.binaries=src/com/room/sample
```

```

Here I am adding the new point.

The screenshot shows the GitHub interface for editing a README.md file. The 8th point has been added. A red box highlights the 8th point in the list. The code block is identical to the previous screenshot.

Click on commit changes.



Here automatically build job.

Dashboard > Freestyle >

Freestyle

Status

Changes

Workspace

Build Now

Configure

Delete Project

Github Hook Log

Rename

Build History

trend

Filter...

#28

Oct 4, 2024, 12:21 PM

- Last build (#28), 2 min 9 sec ago
- Last stable build (#28), 2 min 9 sec ago
- Last successful build (#28), 2 min 9 sec ago
- Last completed build (#28), 2 min 9 sec ago

After that I am deleted the webhook.

Webhooks Guide'. A 'Add webhook' button is located in the top right corner of the main content area."/>

https://github.com/Avularamesh/sabear_simplecustomerapp/settings/hooks

Avularamesh / sabear_simplecustomerapp

Type to search

Code Pull requests Actions Projects Wiki Security Insights Settings

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security

Deploy keys

Secrets and variables

Add webhook

#####Task is Done#####

3) Configure poll scm and build periodical options in jenkins job.

- **Poll SCM:** This option makes Jenkins regularly check the source code repository for any changes (like new commits). If Jenkins finds changes, it will trigger a build.
- It's useful when you only want to build the project when the code is updated.
- **Build Periodically:** This option tells Jenkins to run the job at regular intervals, no matter if there are code changes or not. It doesn't check for commits;
- it just runs the job on a schedule (like daily or weekly builds).

See the below how many builds happen and what time.

The screenshot shows the Jenkins dashboard with three active projects:

| Name | Last Success | Last Failure | Last Duration |
|-----------|----------------|----------------|---------------|
| Freestyle | 5.9 sec #30 | N/A | 0.48 sec |
| Pipeline | 23 hr #31 | 1 day 21 hr #2 | 0.53 sec |
| pvt-Repo | 1 hr 47 min #3 | 1 day 5 hr #1 | 0.83 sec |

Now we Go to the Job and select **Poll SCM**

* * * * * --- This expression will execute the job every minute of every day.

The configuration page for the Freestyle job shows the 'Build Triggers' section. The 'Poll SCM' checkbox is checked and highlighted with a red box. A tooltip below it says: "Do you really mean "every minute" when you say "*****"? Perhaps you meant "H * * * *" to poll once per hour". Other trigger options like 'Trigger builds remotely' and 'Build after other projects are built' are not checked.

After that I am click on save.

Here we haven't seen the any build's because in the repo no actions that's why.

The screenshot shows the Jenkins dashboard at 52.53.156.92:8081. The dashboard includes a sidebar with links like 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. A 'Build Queue (1)' section shows one item. The main area displays a table of builds:

| S | W | Name | Last Success | Last Failure | Last Duration |
|-------|-------|-----------|----------------|----------------|---------------|
| Green | Sun | Freestyle | 5.9 sec #30 | N/A | 0.48 sec |
| Green | Sun | Pipeline | 23 hr #31 | 1 day 21 hr #2 | 0.53 sec |
| Green | Cloud | pvt-Repo | 1 hr 47 min #3 | 1 day 5 hr #1 | 0.83 sec |

Below the table is a 'Build Executor Status' section showing 1 idle executor. The bottom right corner shows 'REST API Jenkins 2.462.2'.

So now I went to the github repo and change the code then automatically build.

The screenshot shows the GitHub code editor for the repository `sabear.simplecutomerapp` in the `master` branch. The file is named `README.md` and contains the following content:

```
## How to execute this pipeline
1) Sonarscanner
2) Sonarqube server
3) jenkins server
4) nexus server
5) change the credetillas of nexus server as per our requirement
6) create a new repo in nexus with version policy as "snapshot"
7) modify the pipeline as per our sonarscanner name and sonarqube server name
8) Use the below code in sonarqube analysis report.
```
sonar.projectKey=lcideit
sonar.projectName=lcideit
sonar.projectVersion=2.0
sonar.sources=/var/lib/jenkins/workspace/$JOB_NAME/src/
sonar.binaries=target/classes/com/visualpathit/account/controller/
sonar.junit.reportsPath=target/surefire-reports
sonar.jacoco.reportPath=target/jacoco.exec
sonar.java.binaries=src/com/room/sample
```

Use Control + Shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.
```

At the bottom, it says 'Attach files by dragging & dropping, selecting or pasting them.' The bottom right corner shows 'ENG IN 18:30 04-10-2024'.

As we discuss the job build.

The screenshot shows the Jenkins dashboard with three listed jobs:

| S | W | Name | Last Success | Last Failure | Last Duration |
|---|----|-----------|----------------|----------------|---------------|
| ✓ | ☀️ | Freestyle | 37 sec #31 | N/A | 0.49 sec |
| ✓ | ☀️ | Pipeline | 1 day 0 hr #31 | 1 day 22 hr #2 | 0.53 sec |
| ✓ | ☁️ | pvt-Repo | 2 hr 29 min #3 | 1 day 6 hr #1 | 0.83 sec |

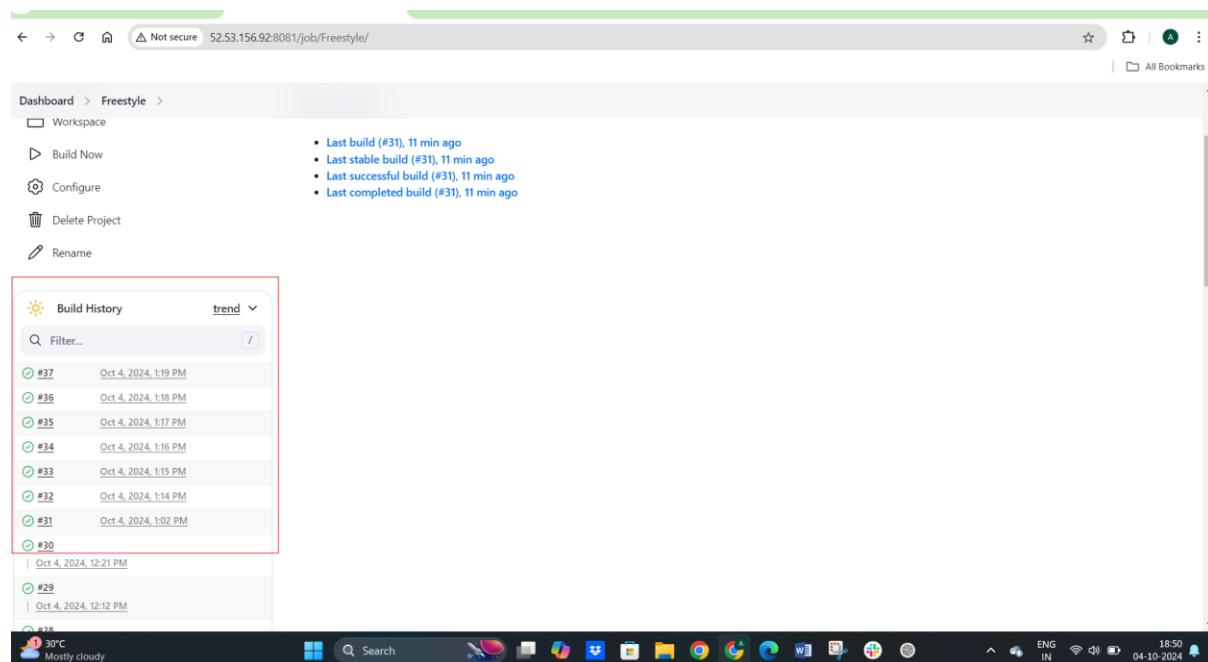
Build Queue (1) is shown below the main table. Build Executor Status shows 1 idle and 2 idle. The REST API version is 2.462.2. The system tray shows the date as 04-10-2024.

Now went to Job and change – Build Periodically

The screenshot shows the configuration page for the Freestyle job. Under the Build Triggers tab, the 'Build periodically' checkbox is checked. The schedule dropdown is set to 'H * * * *'. A warning message at the bottom states: 'Do you really mean "every minute" when you say "H * * * *"? Perhaps you meant "H * * * *" to poll once per hour'. Other trigger options like 'Trigger builds remotely' and 'Build after other projects are built' are also present.

Now here you see the After selecting these build Periodically.

It will build the job every minute.



The screenshot shows a Jenkins Freestyle project's build history. A red box highlights the list of builds from Oct 4, 2024, at 1:19 PM to 1:30 PM, illustrating periodic builds every minute. The build numbers range from #30 to #37. The Jenkins interface includes a dashboard, workspace, build now, configure, delete project, and rename options. The status bar at the bottom shows the date as 04-10-2024, time as 18:50, and weather as 30°C mostly cloudy.

#####Task is Done#####

4) Take backup of jenkins server by using bash script.

I went to the Jenkins server and there I am creating the on file --- Script_for_Jenkins_Backup

I went to these file and Add the script.

```
[root@Jenkins_Master ~]# vi Script_for_Jenkins_Backup
[root@Jenkins_Master ~]# p" [New] 26L, 674B written
```

```
#!/bin/bash
# Jenkins home directory (where Jenkins stores all jobs and configs)
JENKINS_HOME="/var/lib/jenkins"

# Directory where you want to store the backup
BACKUP_DIR="/backup/jenkins"

# Add the current date to the backup file name
DATE=$(date +%F)
BACKUP_FILE="jenkins_backup_$DATE.tar.gz"

# Create the backup directory if it doesn't exist
mkdir -p $BACKUP_DIR

# Create a compressed backup of the Jenkins home directory
echo "Backing up Jenkins..."
tar -czvf $BACKUP_DIR/$BACKUP_FILE $JENKINS_HOME

# Check if the backup was successful
if [ $? -eq 0 ]; then
    echo "Backup successful! Backup saved as $BACKUP_DIR/$BACKUP_FILE"
else
    echo "Backup failed!"
fi
```

After change the permissions of the file and Run the file.

```
[root@Jenkins_Master ~]# ls
Script_for_Jenkins_Backup
[root@Jenkins_Master ~]# chmod 777 Script_for_Jenkins_Backup
[root@Jenkins_Master ~]# ls
Script_for_Jenkins_Backup
[root@Jenkins_Master ~]# ./Script_for_Jenkins_Backup
Backing up Jenkins...
tar: Removing leading '/' from member names
/var/lib/jenkins/
/var/lib/jenkins/%C/
/var/lib/jenkins/%C/jenkins/
/var/lib/jenkins/%C/jenkins/war/
/var/lib/jenkins/%C/jenkins/war/META-INF/
/var/lib/jenkins/%C/jenkins/war/META-INF/MANIFEST.MF
/var/lib/jenkins/%C/jenkins/war/META-INF/JENKINS.SF
/var/lib/jenkins/%C/jenkins/war/META-INF/JENKINS.RSA
/var/lib/jenkins/%C/jenkins/war/META-INF/maven/
/var/lib/jenkins/%C/jenkins/war/META-INF/maven/org.jenkins-ci.main/
/var/lib/jenkins/%C/jenkins/war/META-INF/maven/org.jenkins-ci.main/jenkins-war/
/var/lib/jenkins/%C/jenkins/war/META-INF/maven/org.jenkins-ci.main/jenkins-war/pom.xml
/var/lib/jenkins/%C/jenkins/war/META-INF/maven/org.jenkins-ci.main/jenkins-war/pom.properties
/var/lib/jenkins/%C/jenkins/war/WEB-INF/
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/META-INF/
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/META-INF/licenses.html
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/META-INF/licenses.xml
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/dependencies.txt
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/LICENCE
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/add.svg
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/analytic.svg
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/arrow-left.svg
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/arrow-right.svg
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/arrow-up.svg
/var/lib/jenkins/%C/jenkins/war/WEB-INF/classes/images/symbols/browsers.svg
```

The below we can see image Backup successful.

```
/var/lib/jenkins/workspace/pvt-Repo/.git/logs/refs/remotes/
/var/lib/jenkins/workspace/pvt-Repo/.git/logs/refs/remotes/origin/
/var/lib/jenkins/workspace/pvt-Repo/.git/logs/refs/remotes/origin/main
/var/lib/jenkins/workspace/pvt-Repo/.git/logs/HEAD
/var/lib/jenkins/workspace/pvt-Repo/.git/index
/var/lib/jenkins/workspace/pvt-Repo/Java_app_3.0-main.zip
/var/lib/jenkins/workspace/pvt-Repo@tmp/
/var/lib/jenkins/queue.xml
/var/lib/jenkins/hudson.plugins.emaiext.ExtendedEmailPublisher.xml
/var/lib/jenkins/jenkins.security.ResourceDomainConfiguration.xml
/var/lib/jenkins/jenkins.metrics.api.MetricsAccessKey.xml
/var/lib/jenkins/org.jenkinsci.plugins.workflow.flow.GlobalDefaultFlowDurabilityLevel.xml
/var/lib/jenkins/com.cloudbees.hudson.plugins.folder.config.AbstractFolderConfiguration.xml
/var/lib/jenkins/hudson.plugins.build_timeout.global.GlobalTimeoutConfiguration.xml
/var/lib/jenkins/hudson.plugins.gradle.enriched.EnrichedSummaryConfig.xml
/var/lib/jenkins/hudson.plugins.gradle.injection.InjectionConfig.xml
/var/lib/jenkins/hudson.plugins.timestamp.TimestamperConfig.xml
/var/lib/jenkins/io.jenkins.plugins.junit.storage.JunitTestResultStorageConfiguration.xml
/var/lib/jenkins/org.jenkins.model.ArtifactManagerConfiguration.xml
/var/lib/jenkins/org.jenkinsci.plugins.displayurlapi.DefaultDisplayURLProviderGlobalConfiguration.xml
/var/lib/jenkins/github-plugin-configuration.xml
/var/lib/jenkins/org.jenkinsci.plugins.github_branch_source.GitHubConfiguration.xml
/var/lib/jenkins/org.jenkinsci.plugins.workflow.libs.GlobalLibraries.xml
/var/lib/jenkins/org.jenkinsci.plugins.workflow.libs.GlobalUntrustedLibraries.xml
/var/lib/jenkins/hudson.plugins.build_timeout.operations.BuildStepOperation.xml
Backup successful! Backup saved as /backup/jenkins/jenkins_backup_2024-10-04.tar.gz
```

Go to the dir and check file there or not.

```
[root@Jenkins_Master ~]# find / -name "jenkins_backup_2024-10-04.tar.gz"
/backup/jenkins/jenkins_backup_2024-10-04.tar.gz
[root@Jenkins_Master ~]# cd /backup/jenkins/jenkins_backup_2024-10-04.tar.gz
-bash: cd: /backup/jenkins/jenkins_backup_2024-10-04.tar.gz: Not a directory
[root@Jenkins_Master ~]# cd /backup/jenkins
[root@Jenkins_Master jenkins]# ls
jenkins_backup_2024-10-04.tar.gz
[root@Jenkins_Master jenkins]# pwd
/backup/jenkins
```

The Backup file is Available.

#####Task is Done#####

5) Take backup of jenkins using thin backup plugin.

- To Take backup of jenkins using thin backup plugin first we need to install the thin Plugin.
- Go to manage Jenkins, Plugins, Available Plugins.
- Search for the thinBackup.

Dashboard > Manage Jenkins > Plugins

Plugins

- Updates
- Available plugins **3**
- Installed plugins
- Advanced settings

thin

Install

| | | | |
|-----------------------------------|----------------------|-----------------------------------------------------------|--------------------------------------------------------------------|
| AWS Credentials | 231.v08a_59f17d742 | aws | 6 mo 10 days ago |
| Build Name and Description Setter | 2.4.3 | Build Wrappers | 2 mo 25 days ago |
| ThinBackup | 2.1.1 | Miscellaneous | 3 mo 8 days ago |
| Pipeline Maven Integration | 1457.vf7a_de13b_c0d4 | pipeline Maven | 2 days 8 hr ago |
| Icon Shim | 3.0.0 | User Interface Library plugins (for use by other plugins) | Deprecated: No longer does anything, you can uninstall this plugin |

Thin backup is installed.

Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

ThinBackup Success

Loading plugin extensions Success

Download progress

→ [Go back to the top page](#)
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

Now Go to the manage Jenkins and scroll down you see the thinbackup.

Not secure 52.53.156.92:8081/manage/

Dashboard > **Manage Jenkins**

Troubleshooting

Manage Old Data
Scrub configuration files to remove remnants from old plugins and earlier versions.

Tools and Actions

ThinBackup Backup your global and job specific configuration.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

Jenkins CLI
Access/manage Jenkins from your shell, or from your script.

Script Console
Executes arbitrary script for administration/troubleshooting/diagnostics.

Prepare for Shutdown
Stops executing new builds, so that the system can be eventually shut down safely.

And click on ThinkBackup.

6) Setup a new jenkins server and dump the backup taken in task4.

Here I am in Jenkins master Server our Backup file is Available.

```
[root@Jenkins_Master ~]# cd /backup/jenkins
[root@Jenkins_Master jenkins]# ls
jenkins_backup_2024-10-04.tar.gz
[root@Jenkins_Master jenkins]# |
```

Extract these file—using tar -xvf <tarfile name>

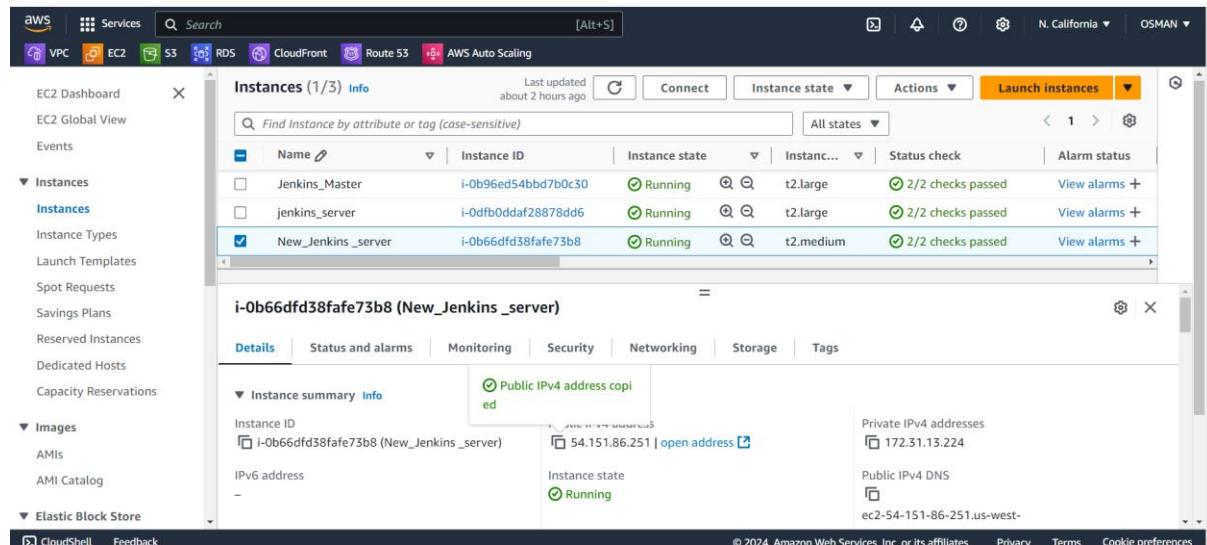
After that Jenkins file move to /home/ec2-user/ location

Here to transfer the Jenkins folder to new sever using the below command.

```
ec2-user@Jenkins_Master ~]$ sudo su -
Last login: Tue Oct  8 16:22:37 UTC 2024 on pts/3
[root@Jenkins_Master ~]# cd /backup/jenkins/var/lib
[root@Jenkins_Master lib]# ls
jenkins
[root@Jenkins_Master lib]# cp jenkins /home/ec2-user/
[root@Jenkins_Master lib]# cp -r jenkins /home/ec2-user/
[root@Jenkins_Master lib]# cd
[root@Jenkins_Master ~]# exit
Logout
ec2-user@Jenkins_Master ~]$ ls
jenkins  Ncalifornia.pem  Thin_Backup_file
ec2-user@Jenkins_Master ~]$ sudo scp -i /home/ec2-user/Ncalifornia.pem -r jenkins ec2-user@54.151.86.251:/home/ec2-user/
MANIFEST.MF
JENKINS_SF
JENKINS_RSA
bam.xml
bam.properties
licenses.html
licenses.xml
dependencies.txt
LICENCE
bdd.svg
analytics.svg
arrow-left.svg
arrow-right.svg
arrow-up.svg
browsers.svg
brush-outline.svg
build-history.svg
build-steps.svg
```

| Time | Speed | Total |
|------|-------|-----------|
| 100% | 127KB | 23.8MB/s |
| 100% | 126KB | 29.6MB/s |
| 100% | 12KB | 7.9MB/s |
| 100% | 5049 | 2.4MB/s |
| 100% | 67 | 23.4KB/s |
| 100% | 28KB | 9.4KB/s |
| 100% | 337KB | 1.0KB/s |
| 100% | 8335 | 2.5MB/s |
| 100% | 1099 | 460.4KB/s |
| 100% | 237 | 97.7KB/s |
| 100% | 793 | 248.4KB/s |
| 100% | 256 | 89.5KB/s |
| 100% | 259 | 88.8KB/s |
| 100% | 254 | 69.5KB/s |
| 100% | 407 | 104.6KB/s |
| 100% | 498 | 148.2KB/s |
| 100% | 502 | 138.6KB/s |
| 100% | 4227 | 1.1MB/s |

Go to the new server check there Jenkins is available.



```

PS C:\Users\ramee\downloads> ssh -i "Ncalifornia.pem" ec2-user@ec2-54-151-86-251.us-west-1.compute.amazonaws.com
  _\_ #####
  ~\_\####\ Amazon Linux 2023
  ~~ \###|
  ~~ \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
  ~~ V~'__->
  ~~~ /_
  ~~_. /_
  _/ /_
  _m/ /_
Last login: Tue Oct  8 17:48:06 2024 from 49.206.53.132
[ec2-user@ip-172-31-13-224 ~]$ ls
jenkins

```

Here you need to install the java.

CMD: sudo yum -y install java-17*

Next install Jenkins.

Go the cd /var/lib --- there you need to delete the Jenkins folder

Next remove the Jenkins folder ----- sudo rm -rf Jenkins

CMD : sudo cp -r Jenkins /var/lib ---- move to the Jenkins folder to /var/lib Location.

After that check the files.

```

PS C:\Users\ramee\downloads> ssh -i "Ncalifornia.pem" ec2-user@ec2-54-151-86-251.us-west-1.compute.amazonaws.com
  _\_ #####
  ~\_\####\ Amazon Linux 2023
  ~~ \###|
  ~~ \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
  ~~ V~'__->
  ~~~ /_
  ~~_. /_
  _/ /_
  _m/ /_
Last login: Tue Oct  8 18:04:21 2024 from 49.206.53.132
[ec2-user@ip-172-31-13-224 ~]$ cd /var/lib
[ec2-user@ip-172-31-13-224 lib]$ ls
alternatives chrony dnf gssproxy initramfs kpatch misc os-prober private rpm-state sss update-motd yum
amazon   cloud   games   hibinit-agent Jenkins logrotate nfs portables rpm    selinux   systemd xfsdump
[ec2-user@ip-172-31-13-224 lib]$ sudo rm -rf Jenkins
[ec2-user@ip-172-31-13-224 lib]$ ls
alternatives chrony dnf gssproxy initramfs logrotate nfs    portables rpm    selinux   systemd xfsdump
amazon   cloud   games   hibinit-agent kpatch   misc   os-prober private rpm-state sss   update-motd yum
[ec2-user@ip-172-31-13-224 lib]$ cd
[ec2-user@ip-172-31-13-224 ~]$ ls
jenkins
[ec2-user@ip-172-31-13-224 ~]$ cp jenkins /var/lib
cp: -r not specified; omitting directory 'jenkins'
[ec2-user@ip-172-31-13-224 ~]$ sudo cp jenkins /var/lib
cp: -r not specified; omitting directory 'jenkins'
[ec2-user@ip-172-31-13-224 ~]$ cd /var/lib/jenkins
[ec2-user@ip-172-31-13-224 jenkins]$ ls
{
com.cloudbees.hudson.plugins.folder.config.AbstractFolderConfiguration.xml
config.xml
credentials.xml
fingerprints
github-plugin-configuration.xml
hudson.model.UpdateCenter.xml
hudson.plugins.build_timeout.global.GlobalTimeOutConfiguration.xml
hudson.plugins.build_timeout.operations.BuildStepOperation.xml
}

```

Go to the browser and browse.

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with links for 'Dashboard', 'Manage Jenkins', 'Manage Plugins', 'Manage Nodes', 'Manage Queue', 'Manage View', 'Manage Pipeline', 'Manage Freestyle', 'Manage System', and 'Logout'. Below the navigation bar is a search bar with placeholder text 'Search (CTRL+K)'. To the right of the search bar are icons for 'All Bookmarks' and 'All Bookmarks'.

The main content area has a dark header with the Jenkins logo and the word 'Jenkins'. On the left, there's a sidebar with links for 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. A 'Build Queue' section indicates 'No builds in the queue.' A 'Build Executor Status' section shows '1 Idle' and '2 Idle' executors.

The central part of the dashboard displays a table of build projects:

| S | W | Name ↓ | Last Success | Last Failure | Last Duration |
|---|-------|-----------|------------------|----------------|---------------|
| ✗ | cloud | Freestyle | 7 hr 5 min #3519 | 14 sec #3525 | 0.36 sec |
| ✓ | sun | Pipeline | 5 days 2 hr #31 | 6 days 0 hr #2 | 0.53 sec |
| ✓ | cloud | pvt-Repo | 4 days 4 hr #3 | 5 days 7 hr #1 | 0.83 sec |

At the bottom, there are buttons for 'Icon: S M L' and a 'More' button.

Here my data is Available Task is Done