

Jenkins

Index

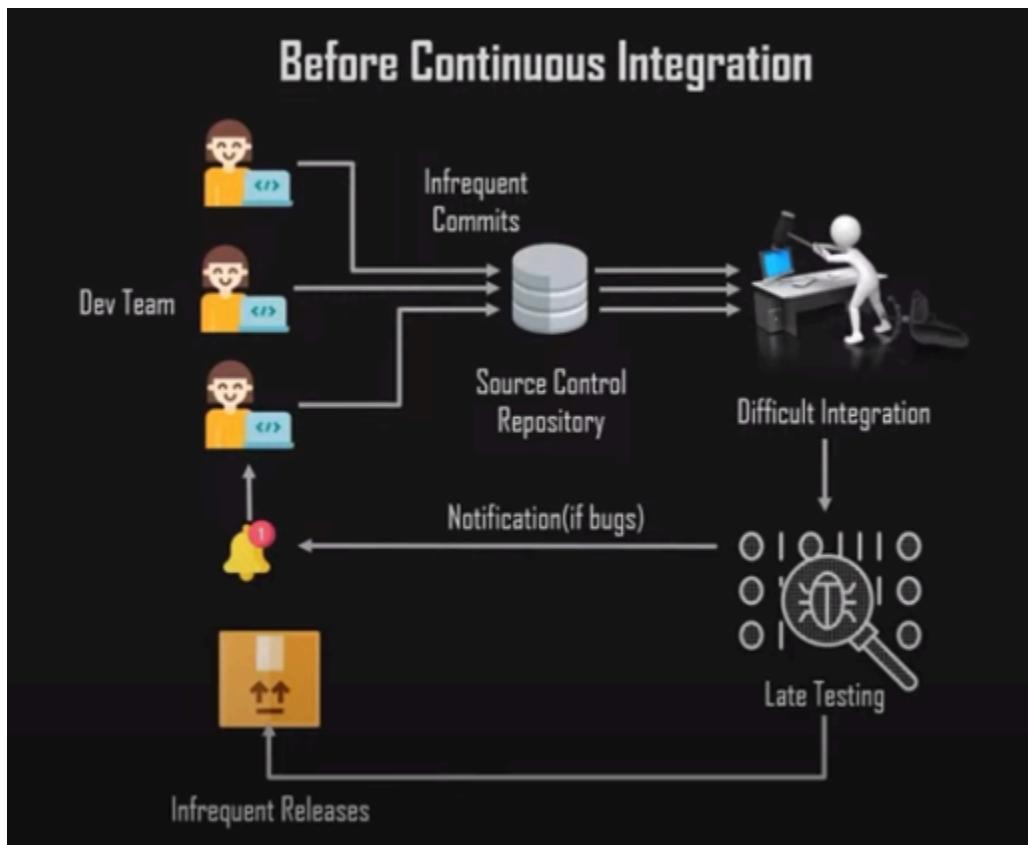
1. Introduction.....	4
1.1. Why Continuous Integration?.....	4
1.2. What is jenkins?.....	5
1.3. What is a pipeline?.....	6
1.4. What is a jenkinsfile?.....	6
1.5. Jenkins workflow.....	6
2. Installation.....	9
2.1. Installation of Java.....	9
2.2. Installation of Jenkins.....	9
2.2.1. Access the Jenkins Web Interface.....	9
2.2.2. Set user and password.....	11
2.3. Email Notifications.....	13
2.3.1. Extended Email Notification (Email Extension Plugin).....	13
2.3.2. E-mail Notification (Mailer Plugin).....	15
2.3.3. Setup.....	16
2.3.4. Configure the SMTP.....	18
2.3.5. Configure Email Notifications for a Job.....	22
2.3.5.1. Attach Build log.....	25
2.4. Test Results Analyzer Plugin.....	29
3. Jenkins.....	30
3.1. Reset the admin Password.....	31
3.2. Available plugins.....	33
3.3. Create Jobs.....	35
3.3.1. Using Freestyle Project.....	35
3.3.2. Using Pipeline.....	39
3.3.2.1. Status.....	42
3.3.2.2. Console Output.....	42
3.3.2.3. No workspace Initially.....	44
3.3.2.4. Workspace creation when files are needed.....	44
3.3.2.5. Pipeline console.....	44
3.3.2.6. Replay.....	45
3.3.2.7. Pipeline Steps.....	46
3.3.2.8. Generate Pipeline Script.....	47
3.3.3. Using Maven Project.....	52
3.3.3.1. Maven Integration Plugin.....	52
3.4. Nodes.....	53
3.4.1. Why ?.....	54
3.4.2. Configure a Node.....	55
3.4.2.1. Build Queue.....	55

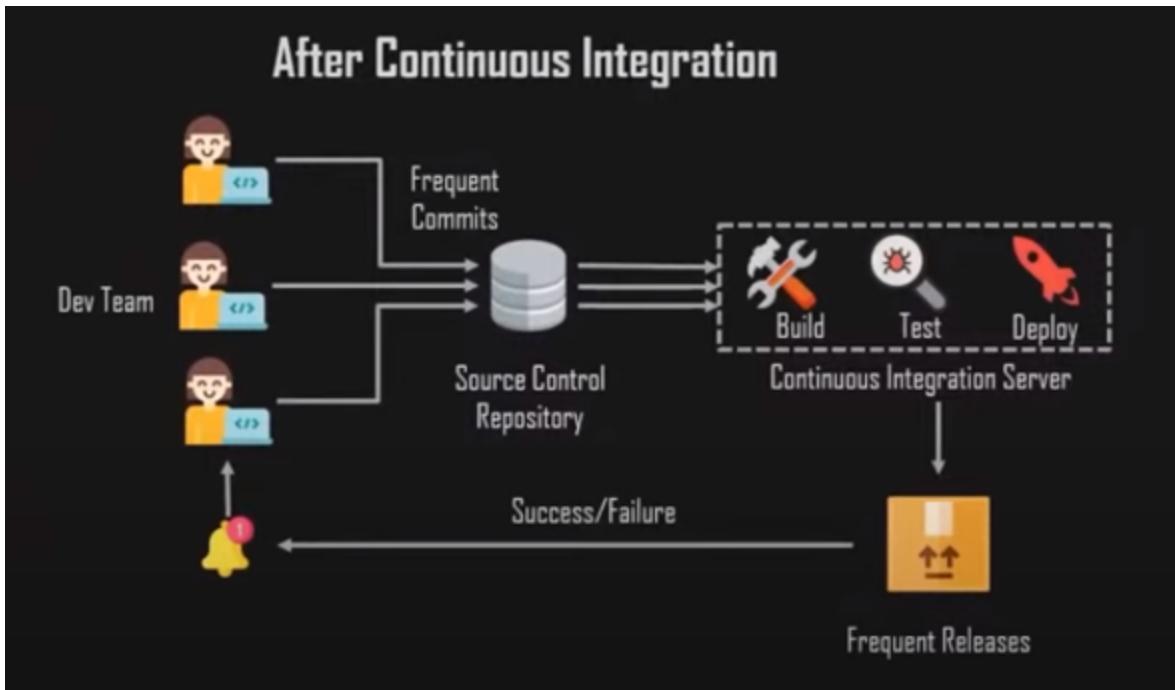
3.4.2.2. Build Executor Status.....	55
3.4.2.3. Nodes Section.....	56
3.4.2.4. Create Agent1 and Configuration.....	57
3.4.2.4.1. Requirements.....	57
3.4.2.4.2. Set up.....	59
3.4.2.4.3. Log.....	64
3.4.2.4.4. System Information.....	65
3.4.2.5. Run a job on a specific node in jenkins.....	65
3.4.2.5.1. Check the builds.....	68
4. Automate Job.....	70
4.1. Use organization folders.....	70
4.1.1. Git Polling.....	70
4.1.1.1. Using HTTP.....	70
4.1.1.2. Using SSH.....	76
4.1.2. Git Webhook.....	79
4.1.2.1. Required Plugin (Generic Webhook Trigger Plugin).....	79
4.1.2.2. ngrok.....	79
4.1.3. Webhook Trigger for Multibranch Pipeline.....	86
4.1.3.1. Plugin (Multibranch Scan Webhook Trigger).....	87
4.1.3.2. Each branch of repository can have its own Jenkinsfile.....	87
4.1.3.2.1. Scan Repository Log.....	93

1. Introduction

1.1. Why Continuous Integration?

- For example in a development team there are multiple developers now all the developers work on different components for a project for a code a developer a may be writing code let's say python 1.0 and developer b will be write 1.2 , 1.3 and so, no one of the biggest challenges that we face in an industry is how to make sure that all the developers codes get collaborated and that we having properly release getting deployed that is biggest challenge in the industry.
- Another challenge difficult in debug

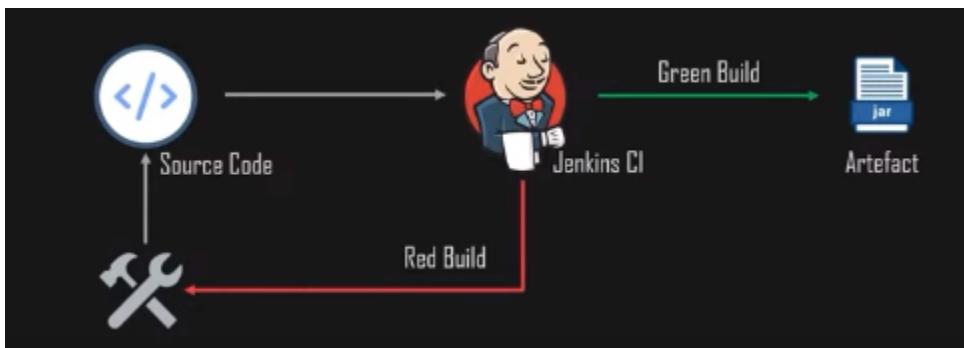




Continuous integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently.

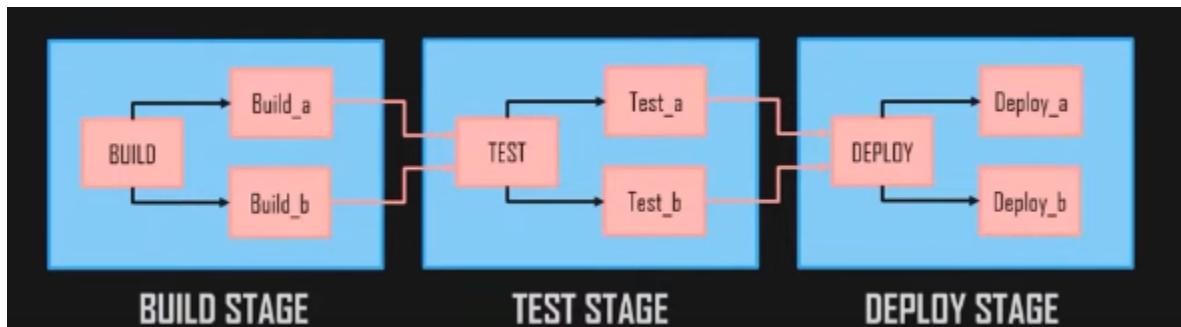
1.2. What is jenkins?

Jenkins is an open source automation tool written in Java with plugins built for continuous integration purposes. Plugins allow integration of various Devops stages.



1.3. What is a pipeline?

Jenkins pipeline is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins. A pipeline has an extensible automation server for creating complex delivery pipelines as code.



They are two approach

1. Declarative → Groovy Syntax
2. Scripted → strict and Traditional

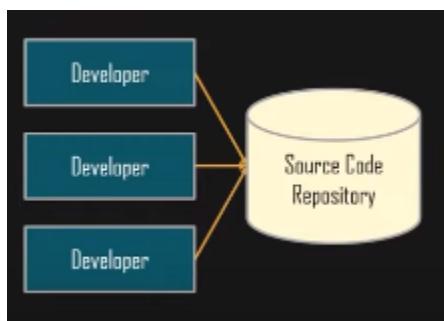
1.4. What is a jenkinsfile?

A jenkinsfile is a text file that contains where we define our entire structure and syntax of our jenkins.

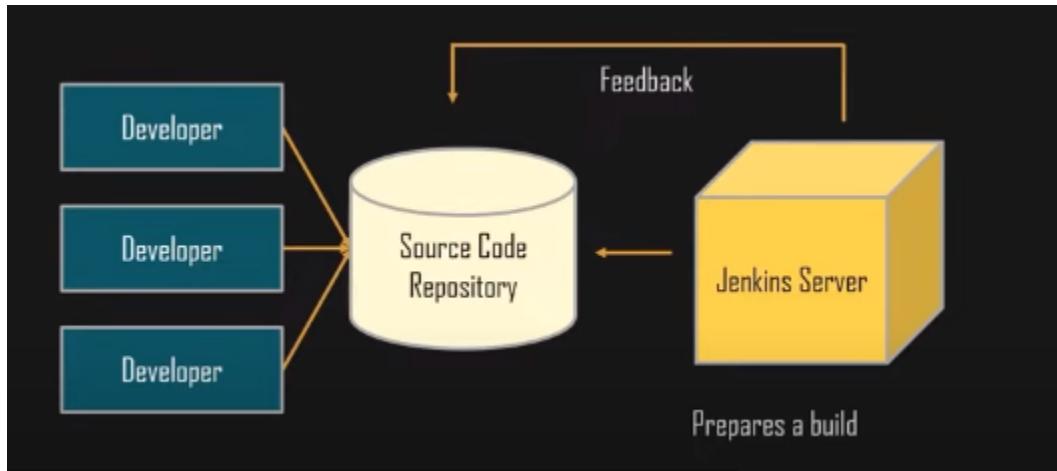
1.5. Jenkins workflow

- Developers commit changes to the source code:

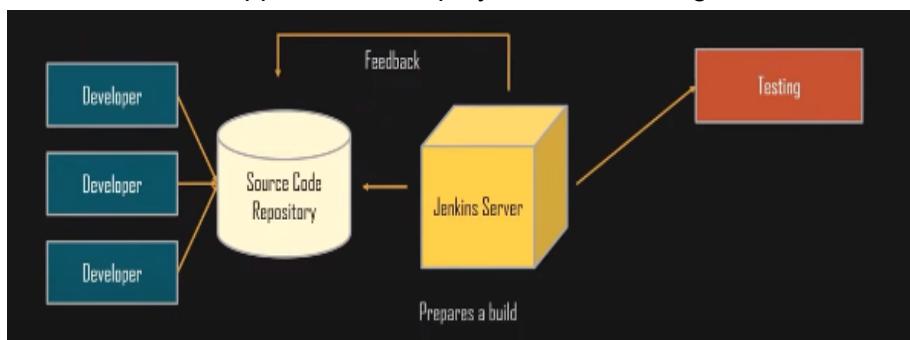
Jenkins workflow always starts with the version control system or Source code repository.



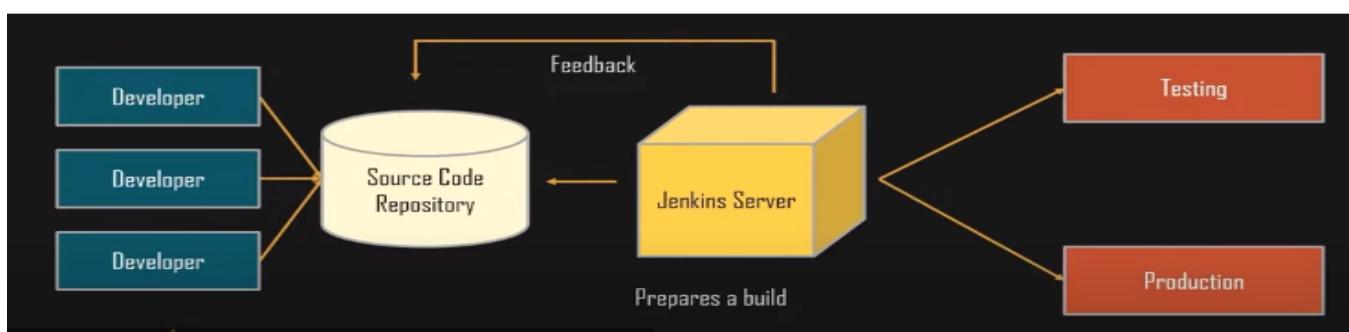
- CI server pulls that code, triggers a build



- The build application is deployed on the testing server



- Then the application is deployed on the production server



Create a pipeline on jenkins to execute a job which does the following tasks:

- Pull code from git
- Define environment variables
- Run unit tests
- Compile and package code
- Build code
- Combine this code into an article in a .jar format

2. Installation

2.1. Installation of Java

Jenkins requires Java to run. Install OpenJDK

```
sudo apt update
```

```
sudo apt install fontconfig openjdk-17-jre
```

```
vlab@HYVLAB7:/var/lib/jenkins$ sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                  Priority   Status
-----*
*  0          /usr/lib/jvm/java-17-openjdk-amd64/bin/java  1711      auto mode
    1          /usr/lib/jvm/java-11-openjdk-amd64/bin/java  1111      manual mode
    2          /usr/lib/jvm/java-17-openjdk-amd64/bin/java  1711      manual mode

Press <enter> to keep the current choice[*], or type selection number:
vlab@HYVLAB7:/var/lib/jenkins$ java --version
openjdk 17.0.13 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Ubuntu-2ubuntu120.04)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Ubuntu-2ubuntu120.04, mixed mode, sharing)
```

2.2. Installation of Jenkins

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/" \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

2.2.1. Access the Jenkins Web Interface

By default, jenkins run on port 8080

<http://<your-server-ip>:8080>

<https://172.16.203.85:8080>

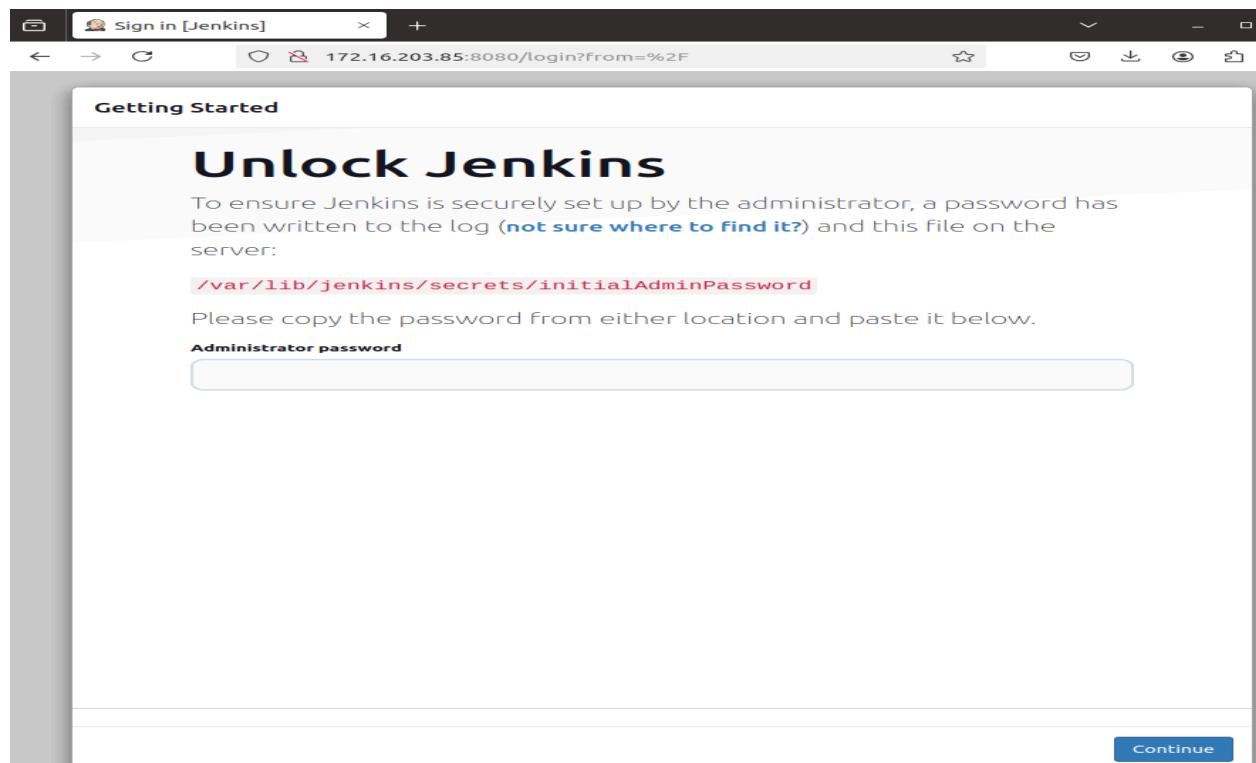
The first time you access jenkins, you'll be prompted to unlock it.
Jenkins generates a random password during installation.

```
vlab@HYVLAB7:~/jenkins$ ls /var/lib/jenkins/
config.xml          jobs           secret.key      updates
hudson.model.UpdateCenter.xml   nodeMonitors.xml secret.key.not-so-secret userContent
jenkins.telemetry.Correlator.xml plugins         Secrets        users
vlab@HYVLAB7:~/jenkins$ ls /var/lib/jenkins/secrets/
initialAdminPassword jenkins.model.Jenkins.crumbSalt master.key
```

This will print the default password to the terminal

```
vlab@HYVLAB7:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2025-01-24 16:26:05 IST; 5s ago
    Main PID: 173232 (java)
       Tasks: 61 (limit: 9364)
      Memory: 683.9M
     CGroup: /system.slice/jenkins.service
             └─173232 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/jenkins_home

Jan 24 16:25:59 HYVLAB7 jenkins[173232]: 2d7092649c194c8fbfb5bd4d5140166a
Jan 24 16:25:59 HYVLAB7 jenkins[173232]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jan 24 16:25:59 HYVLAB7 jenkins[173232]: ****
Jan 24 16:25:59 HYVLAB7 jenkins[173232]: ****
Jan 24 16:25:59 HYVLAB7 jenkins[173232]: ****
Jan 24 16:26:05 HYVLAB7 jenkins[173232]: 2025-01-24 10:56:05.489+0000 [id=39]           INFO      jen>
Jan 24 16:26:05 HYVLAB7 jenkins[173232]: 2025-01-24 10:56:05.541+0000 [id=25]           INFO      hud>
Jan 24 16:26:05 HYVLAB7 systemd[1]: Started Jenkins Continuous Integration Server.
Jan 24 16:26:07 HYVLAB7 jenkins[173232]: 2025-01-24 10:56:07.928+0000 [id=62]           INFO      h.m>
Jan 24 16:26:07 HYVLAB7 jenkins[173232]: 2025-01-24 10:56:07.929+0000 [id=62]           INFO      hud>
```



```
vlab@HYVLAB7:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
2d7092649c194c8fbfb5bd4d5140166a
```

2.2.2. Set user and password

After the above step it asks to create a user and password for admin configurations

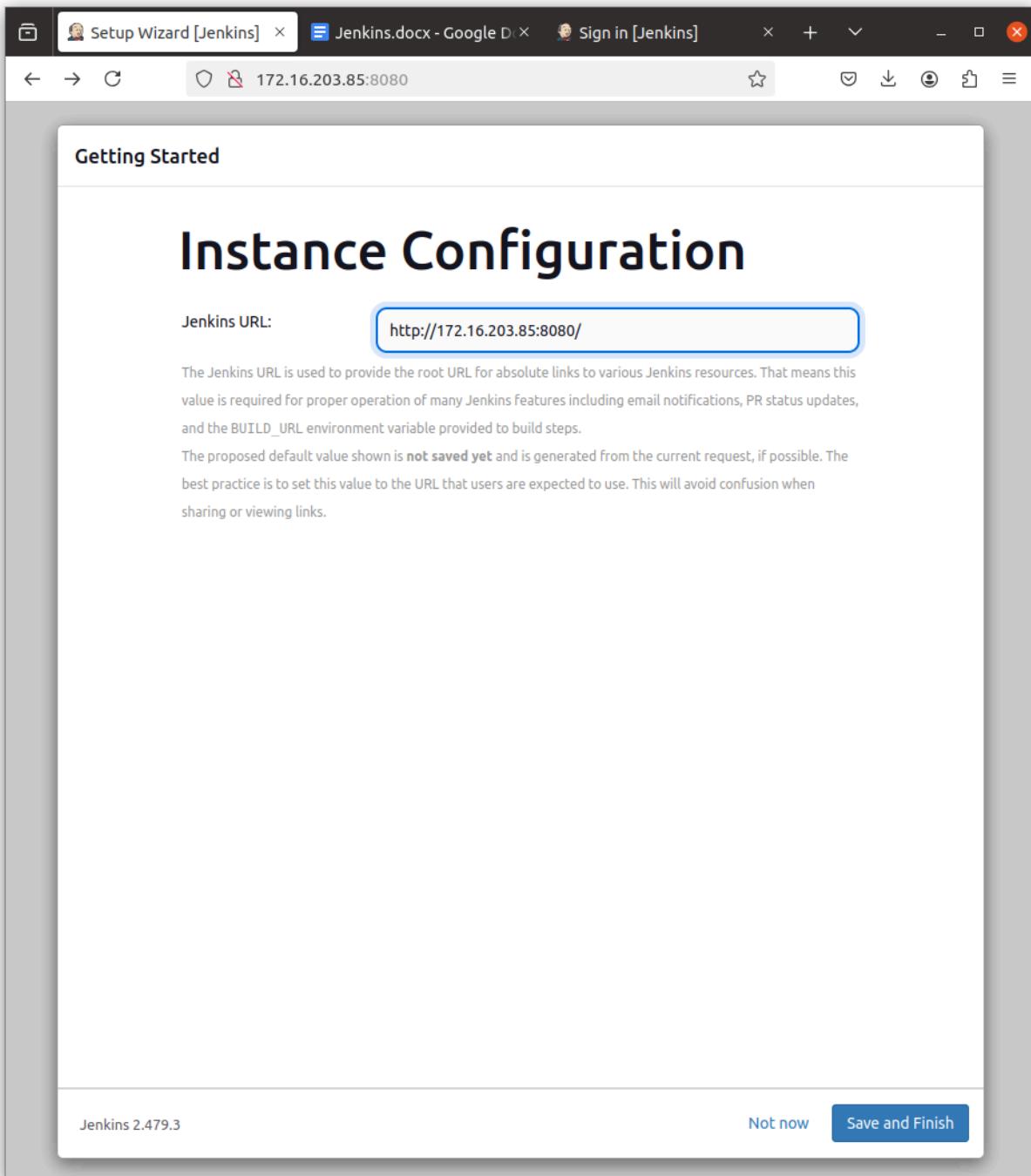
The screenshot shows the Jenkins 'Getting Started' page. At the top, there's a navigation bar with 'Getting Started'. Below it is a large section titled 'Getting Started' with a progress bar. A table lists various Jenkins features and their corresponding required dependencies:

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
⌚ Timestampper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication
⌚ LDAP	⌚ Email Extension	⌚ Mailer	⌚ Dark Theme

Below the table, there's a section titled 'Ionicons API' with a list of required dependencies:

- ** Ionicons API
- Folders
- OWASP Markup Formatter
 - ** ASM API
 - ** JSON Path API
 - ** Structs
 - ** Pipeline: Step API
 - ** Token Macro
- Build Timeout
 - ** bouncycastle API
 - ** Credentials

At the bottom of the page, there's a note: '** - required dependency'.



After reset the password successfully

Automatically removed this file /var/lib/jenkins/secrets/initialAdminPassword

```
vlab@HYVLAB7:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
2d7092649c194c8fbfb5bd4d5140166a  
vlab@HYVLAB7:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
cat: /var/lib/jenkins/secrets/initialAdminPassword: No such file or directory  
vlab@HYVLAB7:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
cat: /var/lib/jenkins/secrets/initialAdminPassword: No such file or directory
```

Check the user list

```
vlab@HYVLAB7:/var/lib/jenkins/users$ ls  
users.xml  vara_16419438915906767594
```

```
vlab@HYVLAB7:/var/lib/jenkins/users$ cd vara_16419438915906767594/  
vlab@HYVLAB7:/var/lib/jenkins/users/vara_16419438915906767594$ ls  
config.xml  
vlab@HYVLAB7:/var/lib/jenkins/users/vara_16419438915906767594$ cat config.xml  
<?xml version='1.1' encoding='UTF-8'?>  
<user>  
  <version>10</version>  
  <id>vara</id>  
  <fullName>varalaxmi</fullName>  
  <description></description>
```

```
vlab@HYVLAB7:/var/lib/jenkins/users$ cat users.xml  
<?xml version='1.1' encoding='UTF-8'?>  
<hudson.model.UserIdMapper>  
  <version>1</version>  
  <idToDirectoryNameMap class="concurrent-hash-map">  
    <entry>  
      <string>vara</string>  
      <string>vara_16419438915906767594</string>  
    </entry>  
  </idToDirectoryNameMap>  
</hudson.model.UserIdMapper>vlab@HYVLAB7:/var/lib/jenkins/users$
```

2.3. Email Notifications

2.3.1. Extended Email Notification (Email Extension Plugin)

- This plugin allows administrators to create global templates for the Extended Email Publisher.
- Provides **advanced email customization**.
- Supports **multiple triggers** (Success, Failure, Unstable, Always, etc.).
- Allows **custom email templates** with Groovy scripts and HTML formatting.
- Can send **attachments** (logs, reports, build artifacts).
- Supports **conditional email sending** based on build parameters.

Extended E-mail Notification

SMTP server

smtp.gmail.com

SMTP Port

587

Advanced ^

Edited

Credentials

mail_vara

+ Add

Use SSL

Use TLS

Use OAuth 2.0

Advanced Email Properties

Save

Apply

Add 'Precedence: bulk' E-mail Header ?

Dashboard > Manage Jenkins > System >

Default Recipients ?

varalaxmiganta77@gmail.com

Reply To List ?

varalaxmiganta77@gmail.com

Emergency reroute ?

Allowed Domains ?

Excluded Recipients ?

Default Subject ?

\$PROJECT_NAME - Build # \$BUILD_NUMBER - \$BUILD_STATUS!

Maximum Attachment Size ?

-1

Default Content ?

\$PROJECT_NAME - Build # \$BUILD_NUMBER - \$BUILD_STATUS:
Check console output at \$BUILD_URL to view the results.

Default Pre-send Script ?

Save Apply

2.3.2. E-mail Notification (Mailer Plugin)

- **Basic built-in feature** in Jenkins.
- Sends email only **on job failure**.
- Requires **SMTP settings** in **Manage Jenkins → System**.
- Cannot customize email content beyond the default format.\

E-mail Notification**SMTP server**

smtp.gmail.com

Default user e-mail suffix ?

Advanced ^

Edited

 Use SMTP Authentication ?**User Name**

varalaxmiganta066@gmail.com

Password Concealed

Change Password

 Use SSL ? **Use TLS****SMTP Port** ?

587

Reply-To Address**Charset**

UTF-8

Save**Apply****2.3.3. Setup**

- Enable 2-step verification.
- Generate an Application Specific Password:
 - Open the Google Account app on your phone, go to Google App Password
 - Once you are on the App Password page:
 - Give the app name (eg jenkins)
 - Click on create
 - Copy your app password for your device

← App passwords

App passwords help you sign in to your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

[Learn more](#)

You don't have any app passwords.

To create a new app-specific password, type a name for it below...

App name

jenkins

Create

Password : nzyx kqkk qrbb qkzp

[← App passwords](#)

App passwords help you sign in to your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to

[Learn more](#)

Generated app password

Your app password for your device

nzyx kqkk qrbb qkzp

Your app password

jenkins

To create a new app password

App name

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above.

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

[Done](#)

2.3.4. Configure the SMTP

- Goto the jenkins dashboard → manage jenkins → system
- Scroll down to the E-mail Notification section.
- Enter the SMTP server address: smtp.gmail.com

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

Advanced 

 Edited

Test configuration by sending test e-mail

GitHub Pull Requests

Published Jenkins URL

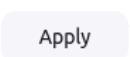
Actualise local repo on factory creation

SCM Polling

Max # of concurrent polling ?

10

 Save

 Apply

- Click on advanced
- Enter user name : email-address@gmail.com
- Enter app password
- Gmail uses port 465 for SSL or 587 for TLS.

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix [?](#)

Advanced ^

Edited

Use SMTP Authentication [?](#)

User Name

varalaxmiganta066@gmail.com

⚠ For security when using authentication it is recommended to enable either TLS or SSL

Password

 Concealed

Change Password

Use SSL [?](#)

Use TLS

SMTP Port [?](#)

465

Reply-To Address

Charset

Save

Apply

- Sender Email Address : This is the “from” address that will appear in the email notifications.
- Apply and Save.

This is the test configuration by sending test email:

SMTP Port ?

465

Reply-To Address

Charset

UTF-8

Test configuration by sending test e-mail

Test e-mail recipient

varalaxmiganta77@gmail.com

Email was successfully sent

Test configuration

GitHub Pull Requests

Published Jenkins URL

Actualise local repo on factory creation

SCM Polling

Max # of concurrent polling ?

Save

Apply

2.3.5. Configure Email Notifications for a Job

- Add the credentials to the manage jenkins

The screenshot shows the Jenkins 'Update credentials' page. The URL in the browser is `Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > mail_vara`. The page title is 'Update credentials'. On the left, there are buttons for 'Update' (with a wrench icon), 'Delete' (with a trash bin icon), and 'Move' (with a right arrow icon). A 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The main form fields include:

- Username:** varalaxmiganta066@gmail.com
- Treat username as secret:**
- Password:** Concealed (with a lock icon)
- ID:** mail_vara
- Description:** mail_vara

A blue 'Save' button is at the bottom.

- Open your jenkins job
- Click on configure

New Item

Enter an item name

Select an item type

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

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

- Click Add post-build action → Editable Email Notification.

The screenshot shows the Jenkins configuration interface for a job named 'email_job1'. The left sidebar lists various configuration sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' section is currently selected and highlighted with a grey background. On the right, under 'Post-build Actions', there is a section titled 'Editable Email Notification' which is also highlighted with a dashed border. This section contains fields for 'Project From' (a dropdown menu), 'Project Recipient List' (containing the placeholder '\$DEFAULT_RECIPIENTS'), 'Project Reply-To List' (containing the placeholder '\$DEFAULT_REPLYTO'), and 'Content Type' (a dropdown menu set to 'Default Content Type'). At the bottom of this section are 'Save' and 'Apply' buttons.

Dashboard > email_job1 > Configuration Advanced

Configure

Add build step ▾

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Post-build Actions

Editable Email Notification ? X

Allows the user to disable the publisher, while maintaining the settings

Disable Extended Email Publisher ?

Project From

Project Recipient List ?

Comma-separated list of email address that should receive notifications for this project.

\$DEFAULT_RECIPIENTS

Project Reply-To List ?

Comma-separated list of email address that should be in the Reply-To header for this project.

\$DEFAULT_REPLYTO

Content Type ?

Default Content Type

Save Apply

2.3.5.1. Attach Build log

The screenshot shows the Jenkins configuration interface for a job named 'email_job1'. The left sidebar lists various configuration sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' section is currently selected and highlighted.

The main configuration area contains several sections:

- Default Content Type:** Set to 'Default Content Type'.
- Default Subject:** Set to '\$DEFAULT_SUBJECT'.
- Default Content:** Set to '\$DEFAULT_CONTENT'.
- Attachments:** A note states: "Can use wildcards like 'module/dist/**/*.zip'. See the [@includes of Ant fileset](#) for the exact format. The base directory is the workspace." An empty attachment field is shown.
- Attach Build Log:** Set to 'Attach Build Log'.
- Content Token Reference:** An empty field.
- Advanced Settings:** A collapsed section.
- Pre-send Script:** Set to '\$DEFAULT_PRESEND_SCRIPT'.
- Post-send Script:** Set to '\$DEFAULT_POSTSEND_SCRIPT'.

At the bottom of the configuration page are two buttons: 'Save' and 'Apply'.

- Define **Triggers** (e.g., Success, Always, Unstable).
- Click save

The screenshot shows the Jenkins configuration interface for a job named 'email_job1'. The left sidebar lists several configuration sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' section is currently selected and highlighted with a grey background.

In the main configuration area, there are two main sections:

- Post-send Script**: A text input field containing the value '\$DEFAULT_POSTSEND_SCRIPT'.
- Additional groovy classpath**: A 'Add' button followed by a checkbox labeled 'Save to Workspace'.

Below these, the **Triggers** section is expanded, showing the following configurations:

- Always**: A trigger type with a red 'X' icon to its right.
- Send To**: A section containing three recipient definitions:
 - Developers**: A trigger type with a red 'X' icon to its right.
 - Recipient List**: A trigger type with a red 'X' icon to its right.
- Add**: A dropdown menu for adding new triggers.
- Advanced**: A dropdown menu for advanced trigger options.

At the bottom of the configuration area, there are three buttons: **Save**, **Success**, and **Apply**. The **Save** button is highlighted with a blue background.

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Recipient List ?



Add ▾

Advanced ▾

Success ?



Send To

Developers ?



Add ▾

Advanced ▾

Add Trigger ▾

Add post-build action ▾

Save

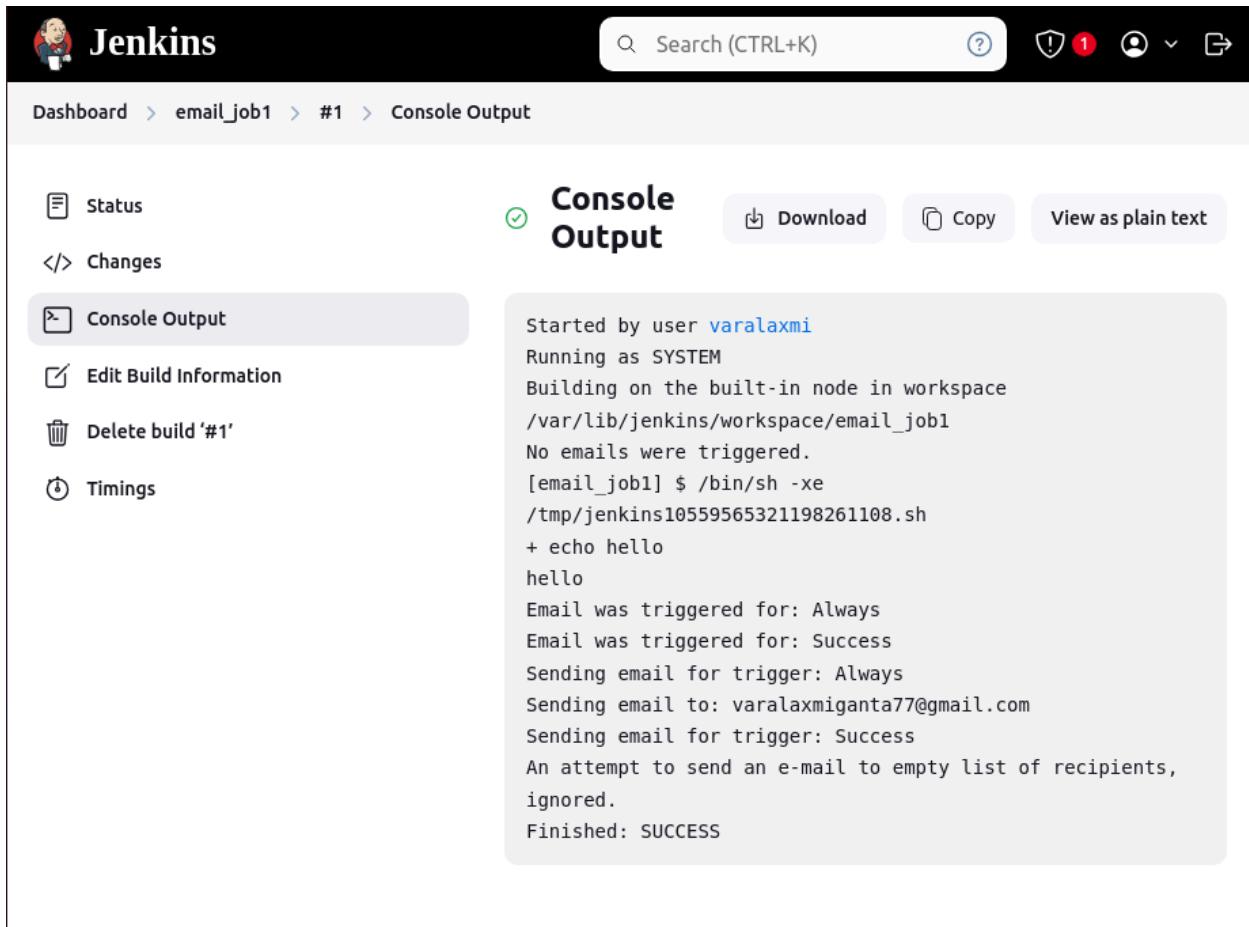
Apply

[REST API](#)

Jenkins 2.479.3

- Test Email Notifications
 - Run a build
 - Email will be sent on job completion

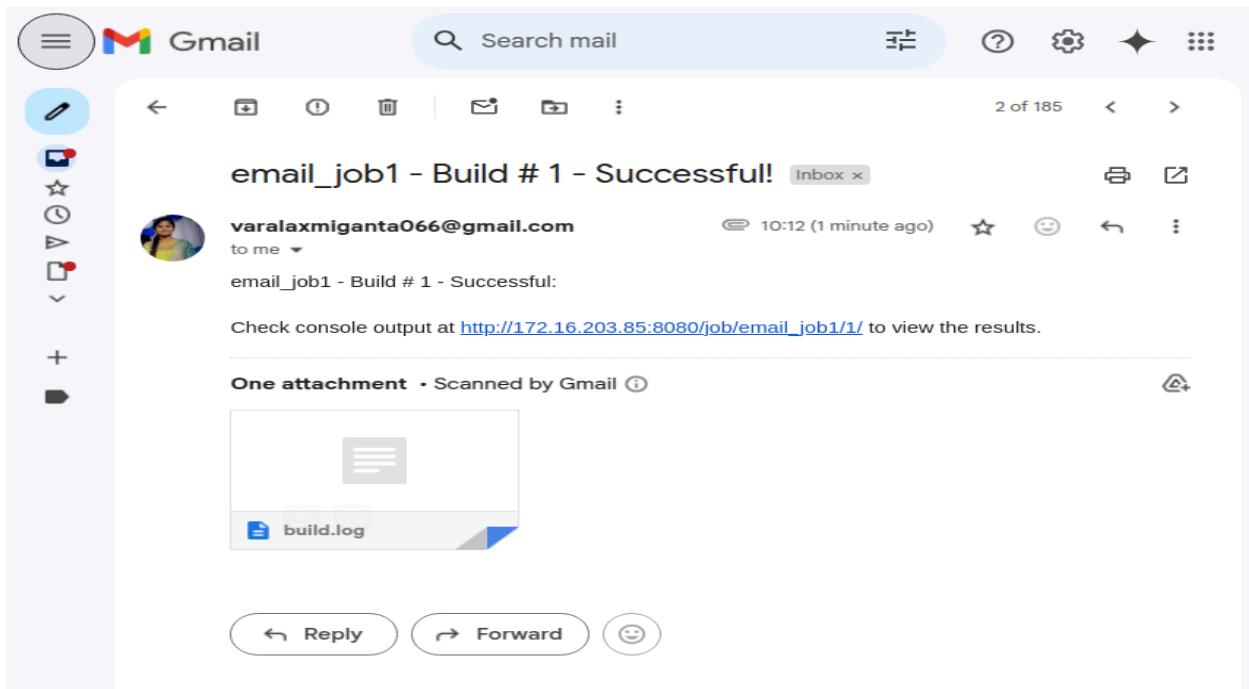
- Console Output



The screenshot shows the Jenkins interface for a build named 'email_job1' under job #1. The 'Console Output' tab is selected. The output log is displayed, showing the build was triggered by user 'varalaxmi' and ran as SYSTEM on a built-in node. It shows the command '/bin/sh -xe /tmp/jenkins10559565321198261108.sh' being executed, which includes an 'echo hello' command. The log indicates that no emails were triggered, but an attempt to send an email to an empty list was ignored. The build finished successfully.

```
Started by user varalaxmi
Running as SYSTEM
Building on the built-in node in workspace
/var/lib/jenkins/workspace/email_job1
No emails were triggered.
[email_job1] $ /bin/sh -xe
/tmp/jenkins10559565321198261108.sh
+ echo hello
hello
Email was triggered for: Always
Email was triggered for: Success
Sending email for trigger: Always
Sending email to: varalaxmiganta77@gmail.com
Sending email for trigger: Success
An attempt to send an e-mail to empty list of recipients,
ignored.
Finished: SUCCESS
```

- Email Notification



The screenshot shows an email in the Gmail inbox titled 'email_job1 - Build # 1 - Successful!' sent to 'varalaxmiganta066@gmail.com' at 10:12 (1 minute ago). The email body contains a message about a successful build and a link to view the results. An attachment named 'build.log' is attached to the email. The Gmail interface includes standard controls like reply, forward, and delete.

email_job1 - Build # 1 - Successful!

varalaxmiganta066@gmail.com
to me

10:12 (1 minute ago)

email_job1 - Build # 1 - Successful:

Check console output at http://172.16.203.85:8080/job/email_job1/1 to view the results.

One attachment • Scanned by Gmail

build.log

2.4. Test Results Analyzer Plugin

- The **Test Results Analyzer** plugin provides a graphical view of test results over multiple builds, making it easier to track failures and trends.
- Go to **Jenkins Dashboard** → **Manage Jenkins** → **Manage Plugins**.
- In the **Available** tab, search for **Test Results Analyzer**.
- Click **Install**

The screenshot shows the Jenkins Manage Plugins interface. The left sidebar has tabs for 'Updates' (11), 'Available plugins', 'Installed plugins' (selected), and 'Advanced settings'. A search bar at the top contains the text 'test results anal'. The main area lists the 'Test Results Analyzer Plugin 0.4.1' with the status 'Enabled'. It includes a description: 'This plugin shows history of test execution results in a tabular or graphical format.', a 'Report an issue with this plugin' link, and a note: 'This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.' with a blue checked button and a red close button.

3. Jenkins

Welcome to jenkins dashboard

The screenshot shows the Jenkins dashboard at the URL 172.16.203.85:8080. The top navigation bar includes a back arrow, forward arrow, refresh button, a shield icon, and the Jenkins logo. To the right of the logo is a search bar with the placeholder "Search (CTRL+K)". Below the search bar are several status icons: a bell with 1 notification, a shield with 1 alert, a user icon with 1 notification, and a gear icon. A "Dashboard" link is also present.

Welcome to Jenkins!

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

Start building your software project

Build Queue
No builds in the queue.

Create a job +

Build Executor Status 0/2

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds ?

3.1. Reset the admin Password

The screenshot shows the Jenkins dashboard at 172.16.203.85:8080. The user menu is open, displaying options such as Builds, My Views, Account, Appearance, Preferences, Security, Experiments, and Credentials. The 'Account' option is highlighted.

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. You can start building your software project by creating a new job or managing existing ones.

Start building your software project

Create a job

Set up a distributed build

- Set up an agent
- Configure a cloud
- Learn more about distributed builds

Build Queue: No builds in the queue.

Build Executor Status: 0/2

REST API Jenkins 2.479.3

This is the admin page

The screenshot shows the Jenkins user profile interface for a user named 'varalaxmi'. The top navigation bar includes links for 'Dashboard', 'Builds', 'My Views', 'Account', 'Appearance', 'Preferences', 'Security', 'Experiments', and 'Credentials'. The main content area displays the user's profile picture, name 'varalaxmi', and Jenkins User ID 'vara'. A search bar at the top right contains the placeholder 'Search (CTRL+K)'. On the far right, there are icons for notifications, security, and other user management options.

Goto the Security the change password and save

The screenshot shows the Jenkins Security settings page. The left sidebar lists the same navigation items as the previous screenshot. The main content area is titled 'Security' and contains sections for 'API Token' and 'Password'. The 'API Token' section shows a message stating 'There are no registered tokens for this user.' with a button labeled 'Add new Token'. The 'Password' section has fields for 'Password:' and 'Confirm Password:', both represented by redacted text. At the bottom, there is a 'Session Termination' section with a red button labeled 'Terminate All Sessions'. Below the main content are 'Save' and 'Apply' buttons.

3.2. Available plugins

To get the available plugins in jenkins

Manage Jenkins ⇒ Plugins

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with options like 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted), and 'My Views'. Below the sidebar are two cards: 'Build Queue' (empty) and 'Build Executor Status' (0/2). The main area is titled 'Manage Jenkins' and contains a message about security risks with built-in nodes. Below this is a large red-highlighted section titled 'Operating system end of life monitor' with a warning about Jenkins support ending on April 2, 2025. At the bottom, there's a 'System Configuration' section with links to 'System', 'Tools', 'Plugins', 'Nodes', 'Clouds', and 'Appearance'.

Manage Jenkins

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

Operating system end of life monitor

You are running Jenkins on Ubuntu 20.04.6 LTS. Jenkins will no longer support Ubuntu 20.04.6 LTS after **2025-04-02**. Please plan your upgrade to a supported operating system before **2025-04-02**. Refer to [the documentation](#) for details.

System Configuration

- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**: Add, remove and
- Appearance**: Configure the look and

Plugins ⇒ Available Plugins

The screenshot shows the Jenkins plugin manager interface at the URL 172.16.203.85:8080/manage/pluginManager/available. The page title is "Jenkins". The navigation bar includes links for "Dashboard", "Manage Jenkins", and "Plugins". The main content area is titled "Plugins" and features a sidebar with options: "Updates", "Available plugins" (which is selected and highlighted in grey), "Installed plugins", and "Advanced settings". A search bar at the top right allows searching for available plugins. Below the search bar is an "Install" button. The main table lists two available plugins:

Install	Name ↓	Released
<input type="checkbox"/>	Kubernetes 4306_vc91e951ea_eb_d Cloud Providers Cluster Management kubernetes Agent Management	1 mo 5 days ago
<input type="checkbox"/>	JavaMail API 1.6.2-10 Library plugins (for use by other plugins)	8 mo 5 days ago

This plugin integrates Jenkins with Kubernetes.

This plugin provides the JavaMail API for other plugins.

Through cmd line

```
vlab@HYVLAB7:/var/lib/jenkins$ ls plugins/ | grep "git"
git
git-client
git-client.jpi
github
github-api
github-api.jpi
github-branch-source
github-branch-source.jpi
github.jpi
git.jpi
pipeline-github-lib
pipeline-github-lib.jpi
```

3.3. Create Jobs

- From the Jenkins Dashboard, click on "New Item" in the left-hand menu.

Welcome to Jenkins!

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

Start building your software project

Create a job +

- Enter a name for your pipeline job (e.g., job1).
- Select "Pipeline" from the list of options.
- Click "OK".

3.3.1. Using Freestyle Project

Note : Install python plugin

- Go to Jenkins Dashboard → Manage Jenkins → Manage Plugins → Available tab.
- Search for "Python Plugin" and install it.

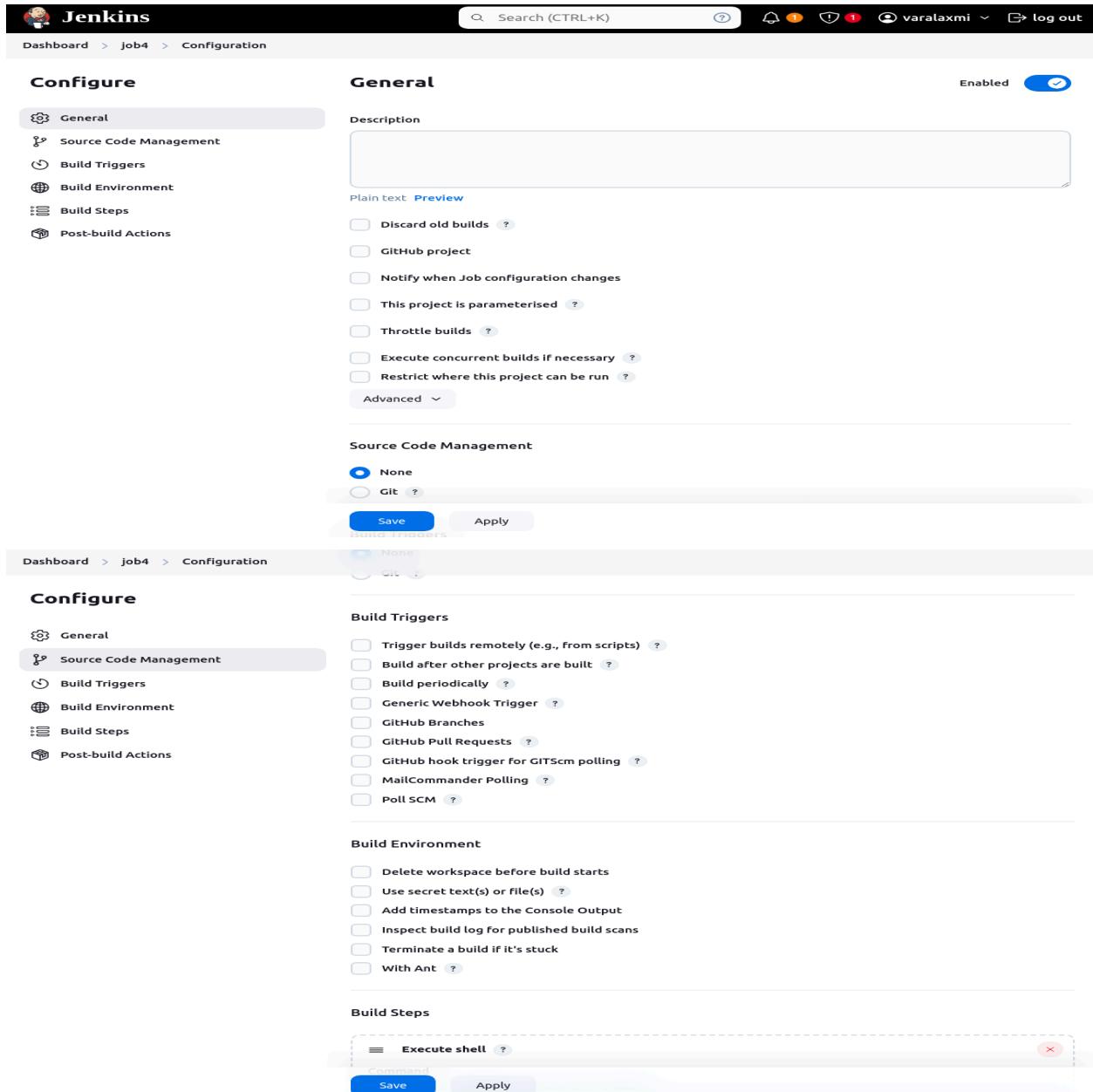
Plugins

pyth

Name	Enabled
Pyenv Pipeline Plugin 2.1.2 Allows a Durable Task (sh or bat) to be executed within a Python virtualenv Report an issue with this plugin	<input checked="" type="checkbox"/> Uninstall
Python Plugin 1.3 Adds the ability to execute python scripts as build steps. Report an issue with this plugin	<input checked="" type="checkbox"/> Uninstall

This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. [Learn more](#).

Template



The screenshot shows the Jenkins configuration interface for a job named "job4". The top navigation bar includes links for "Dashboard", "Job4", and "Configuration", along with user information for "varalaxmi". The main area is titled "Configure" and contains several tabs: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The "General" tab is currently selected.

General Tab:

- Description:** A large text input field with "Plain text" and "Preview" options. It contains the following text:

```
Discard old builds
GitHub project
Notify when Job configuration changes
This project is parameterised
Throttle builds
Execute concurrent builds if necessary
Restrict where this project can be run
```
- Enabled:** A toggle switch is set to "Enabled".
- Source Code Management:** A radio button group showing "None" (selected) and "Git".
- Buttons:** "Save" and "Apply".

Source Code Management Tab:

- Source Code Management:** Shows "None" selected.

Build Triggers Tab:

- Build Triggers:** A list of triggers including "Trigger builds remotely (e.g., from scripts)", "Build after other projects are built", "Build periodically", "Generic Webhook Trigger", "GitHub Branches", "GitHub Pull Requests", "GitHub hook trigger for GITScm polling", "MailCommander Polling", and "Poll SCM".

Build Environment Tab:

- Build Environment:** A list of environment settings including "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", and "With Ant".

Build Steps Tab:

- Build Steps:** A step named "Execute shell" is listed. The configuration for this step is:

```
#!/bin/sh -xe
#(standard output)
#(standard error)
```
- Buttons:** "Save" and "Apply".

Dashboard > job4 > Configuration

Add timestamps to the Console Output

Inspect build log for published build scans
 Terminate a build if it's stuck
 With Ant ?

Configure

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

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

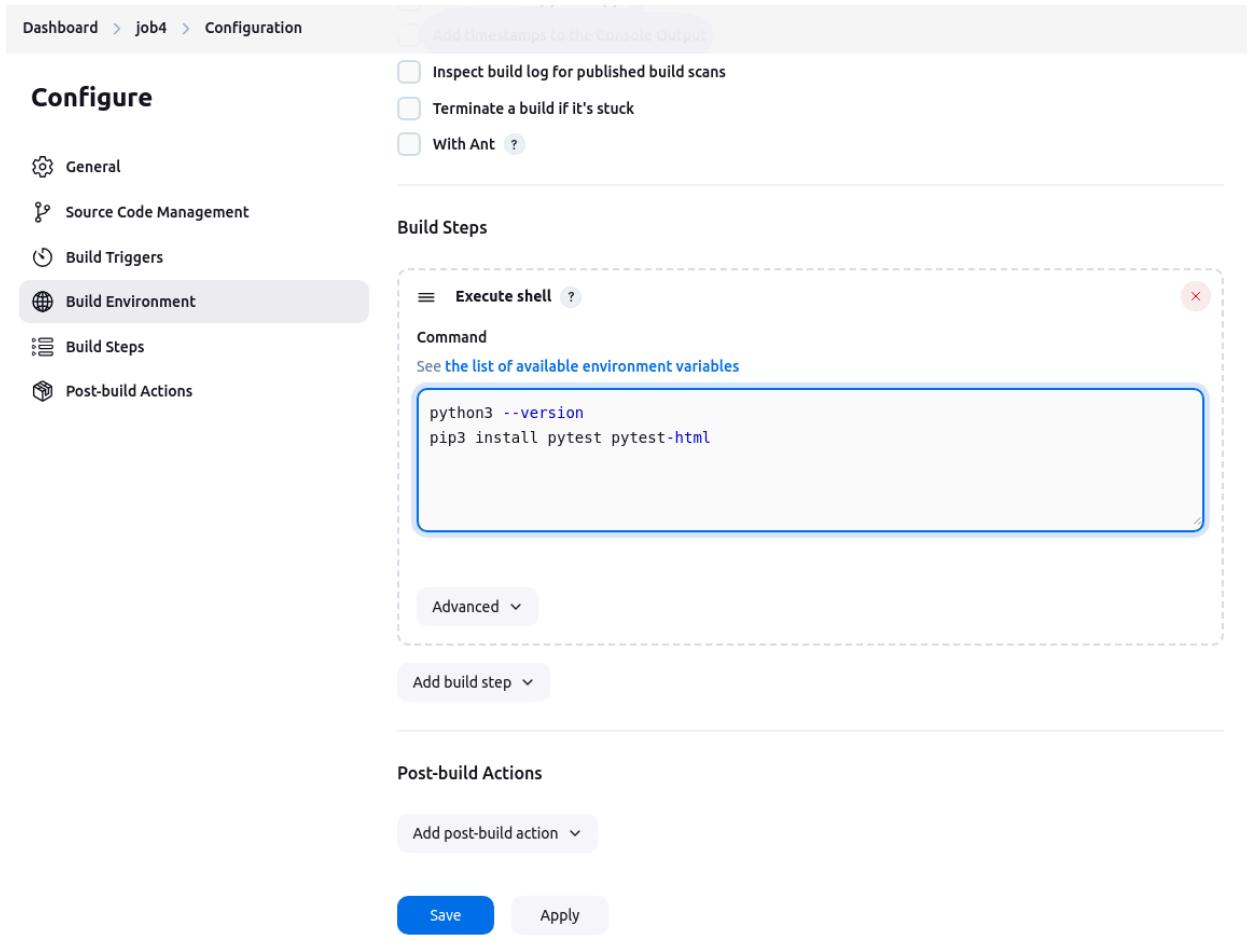
```
python3 --version
pip3 install pytest pytest-html
```

Advanced

Add build step

Post-build Actions

Add post-build action



- Click save button after click on Build Now

[REST API](#) Jenkins 2.479.3

[Status](#)[Changes](#)[Console Output](#)[Edit Build Information](#)[Delete build '#3'](#)[Timings](#)

Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user varalaxmi
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins/workspace/job4
[job4] $ /bin/sh -xe /tmp/jenkins741568884248209173.sh
+ python3 --version
Python 3.8.10
+ pip3 install pytest pytest-html
Requirement already satisfied: pytest in /var/lib/jenkins/.local/lib/python3.8/site-packages (8.3.4)
Requirement already satisfied: pytest-html in
/var/lib/jenkins/.local/lib/python3.8/site-packages (4.1.1)
Requirement already satisfied: pluggy<2,>=1.5 in
/var/lib/jenkins/.local/lib/python3.8/site-packages (from pytest) (1.5.0)
Requirement already satisfied: tomli>=1; python_version < "3.11" in
/var/lib/jenkins/.local/lib/python3.8/site-packages (from pytest) (2.2.1)
Requirement already satisfied: exceptiongroup>=1.0.0rc8; python_version < "3.11" in
/var/lib/jenkins/.local/lib/python3.8/site-packages (from pytest) (1.2.2)
Requirement already satisfied: packaging in /var/lib/jenkins/.local/lib/python3.8/site-packages (from pytest) (24.2)
Requirement already satisfied: configparser in /var/lib/jenkins/.local/lib/python3.8/site-packages (from pytest) (2.0.0)
Requirement already satisfied: jinja2>=3.0.0 in
/var/lib/jenkins/.local/lib/python3.8/site-packages (from pytest-html) (3.1.5)
Requirement already satisfied: pytest-metadata>=2.0.0 in
/var/lib/jenkins/.local/lib/python3.8/site-packages (from pytest-html) (3.1.1)
Requirement already satisfied: MarkupSafe>=2.0 in
/var/lib/jenkins/.local/lib/python3.8/site-packages (from jinja2>=3.0.0->pytest-html) (2.1.5)
Finished: SUCCESS
```

3.3.2. Using Pipeline

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there's a navigation bar with the Jenkins logo, a search bar labeled 'Search (CTRL+K)', and various status icons. Below the bar, the path 'Dashboard > New Item' is shown. The main area is titled 'New Item' and contains a text input field with 'job1' typed into it. A section titled 'Select an item type' lists several options:

- Freestyle project**: Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type. This option is highlighted with a blue border.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

- Add a description for your pipeline.
- Select the **Pipeline script** option.
- Write the pipeline script directly in the Jenkins UI.
- Simple Hello World

Configure

General

Pipeline

Advanced

Triggers

- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?
- Trigger builds remotely (e.g., from scripts) ?

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}  
12
```

Use Groovy Sandbox ?

Pipeline Syntax

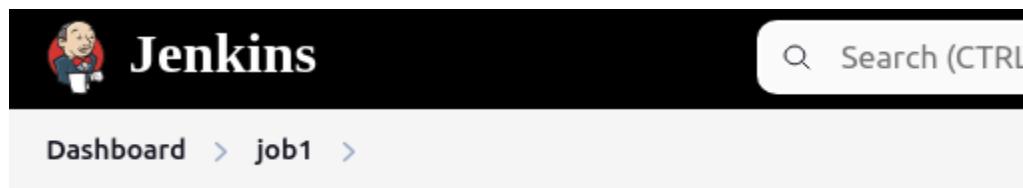
Save

Apply

Pipeline script :

Script ?

```
1 > pipeline {  
2     agent any  
3  
4 >     stages {  
5 >         stage('Hello') {  
6 >             steps {  
7 >                 script {  
8                     echo 'Hello World'  
9                 }  
10            }  
11        }  
12    }  
13 }
```



The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with the Jenkins logo and a search bar labeled "Search (CTRL+F)". Below the bar, the path "Dashboard > job1 >" is visible. The main content area is titled "job1". On the left, there's a sidebar with various options: "Status" (which is currently selected and highlighted in grey), "Changes", "Build Now", "Configure", "Delete Pipeline", "Stages", "Rename", and "Pipeline Syntax". To the right of the sidebar, the text "this is first job" is displayed. Further down, there's a section titled "Permalinks". At the bottom, there's a "Builds" summary card showing "Today" and a recent build entry "#1 15:04".

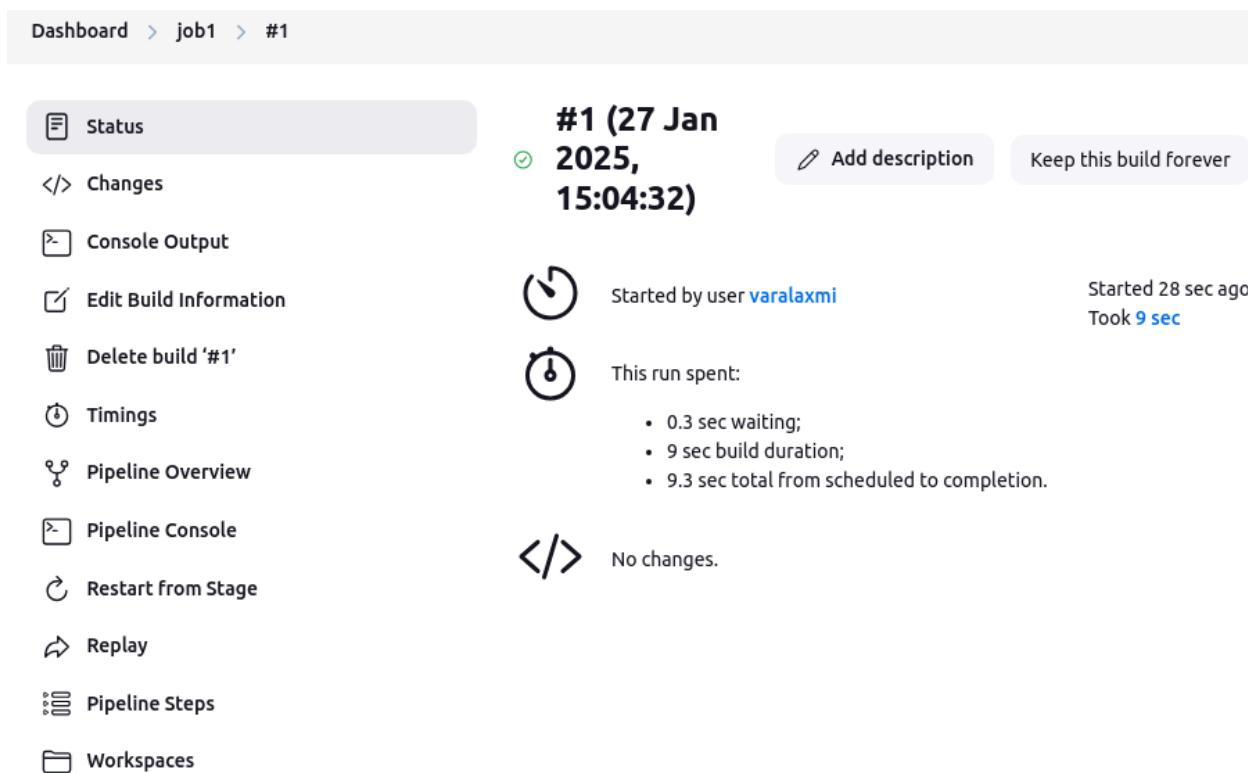
Builds

Today

✓ #1 15:04

3.3.2.1. Status

- Success(Green): The job has completed successfully without any error or failures.
- Failure(Red): The job encountered errors during execution and was unable to complete as expected.



The screenshot shows the Jenkins interface for a job named 'job1'. The top navigation bar indicates 'Dashboard > job1 > #1'. On the left, a sidebar lists various options: Status (selected), Changes, Console Output, Edit Build Information, Delete build '#1', Timings, Pipeline Overview, Pipeline Console, Restart from Stage, Replay, Pipeline Steps, and Workspaces. The main content area displays the build information for '#1 (27 Jan 2025, 15:04:32)'. It shows that the build was started by user 'varalaxmi' 28 seconds ago and took 9 seconds. Below this, it says 'This run spent:' with a list: 0.3 sec waiting, 9 sec build duration, and 9.3 sec total from scheduled to completion. A large '</>' icon indicates 'No changes.'

3.3.2.2. Console Output

- The **console output** in Jenkins refers to the log of messages and information generated during the execution of a job or pipeline.
- It provides a detailed trace of what happens during the execution, including any commands run, steps executed, and output from those steps (like errors, warnings, or success messages).

[Status](#)[Changes](#)[Console Output](#)[Edit Build Information](#)[Delete build '#1'](#)[Timings](#)[Pipeline Overview](#)[Pipeline Console](#)[Restart from Stage](#)[Replay](#)[Pipeline Steps](#)[Workspaces](#)

Console Output

[Download](#)[Copy](#)[View as plain text](#)

```

Started by user varalaxmi
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/job1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

```

vlab@HYVLAB7:/var/lib/jenkins$ cd jobs/
vlab@HYVLAB7:/var/lib/jenkins/jobs$ ls
job1
vlab@HYVLAB7:/var/lib/jenkins/jobs$ cd job1/
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1$ ls
builds config.xml nextBuildNumber
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1$ cd builds/
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1/builds$ ls
1 legacyIds permalinks
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1/builds$ cd 1
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1/builds/1$ ls
build.xml log log-index workflow-completed
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1/builds/1$ cat log
Started by user
                           varalaxmi

[Pipeline] Start of Pipeline
[Pipeline] node
                           Jenkins in /var/lib/jenkins/workspace/job1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline

```

Running on Jenkins in /var/lib/jenkins/workspace/job1

Hello World

Finished: SUCCESS

3.3.2.3. No workspace Initially

```
vlab@HYVLAB7:~$ cd /var/lib/jenkins/woekspace/
bash: cd: /var/lib/jenkins/woekspace/: No such file or directory
```

- When a pipeline is defined but does not involve any files creation or manipulation, jenkins does not create a workspace.
- This is because there are no resources or files that need to be stored or processed during the job executions.

3.3.2.4. Workspace creation when files are needed

Definition

Pipeline script

Script ?

```
1  pipeline {
2      agent any
3
4      stages {
5          stage('Hello') {
6              steps {
7                  script {
8                      echo 'Hello World'
9                      writeFile file: 'example.txt', text: 'This is a test file'
10                 }
11             }
12         }
13     }
14 }
```

- If your pipeline includes steps that require creating or accessing files(eg.., creating a file, cloning a repository, reading a file) jenkins creates a workspace directory automatically.
- The workspace serves as the job's working directory where jenkins performs file-related operations.

3.3.2.5. Pipeline console

Dashboard > job1 > #1 > Pipeline Console

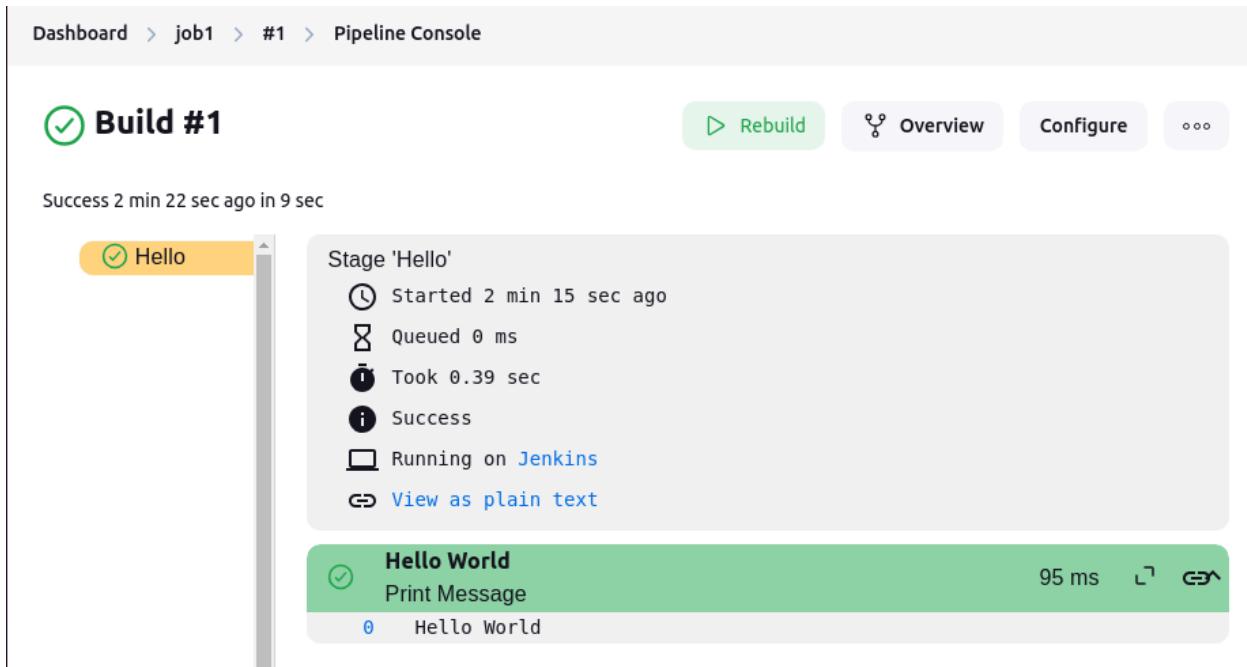
Build #1

Success 2 min 22 sec ago in 9 sec

Hello

Stage 'Hello'
Started 2 min 15 sec ago
Queued 0 ms
Took 0.39 sec
Success
Running on Jenkins
[View as plain text](#)

Hello World
Print Message 95 ms
Hello World



3.3.2.6. Replay

- In Jenkins, **Replay** is a feature that allows you to re-run a previously executed pipeline with modified code, typically from the Jenkinsfile.

Dashboard > job1 > #1 > Replay

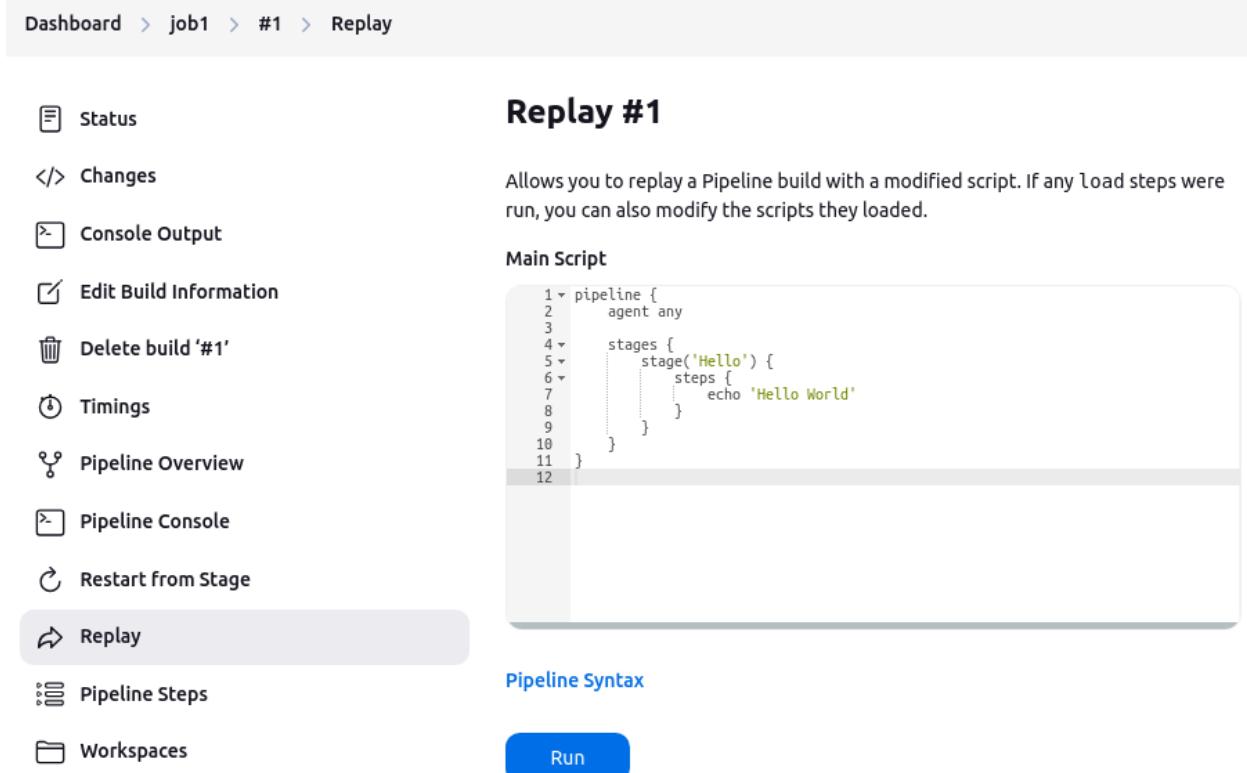
Replay #1

Status Changes Console Output Edit Build Information Delete build '#1' Timings Pipeline Overview Pipeline Console Restart from Stage Replay Pipeline Steps Workspaces

Main Script

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}  
12
```

Pipeline Syntax Run



3.3.2.7. Pipeline Steps

Dashboard > job1 > #1 > Pipeline Steps

Step	Arguments	Status
Start of Pipeline - (6.4 sec in block)		✓
node - (2.1 sec in block)	<input type="button" value="X"/>	✓
node block - (1 sec in block)		✓
stage - (0.62 sec in block)	Hello	<input type="button" value="X"/> ✓
stage block (Hello) - (0.39 sec in block)		✓
echo - (95 ms in self)	Hello World	<input type="button" value="X"/> ✓

Status
 Changes
 Console Output
 Edit Build Information
 Delete build '#1'
 Timings
 Pipeline Overview
 Pipeline Console
 Restart from Stage
 Replay
 Pipeline Steps
 Workspaces

Dashboard > job1 > #1 > Workspaces

Workspaces for job1 #1

- /var/lib/jenkins/workspace/job1 on built-in

Status
 Changes
 Console Output
 Edit Build Information
 Delete build '#1'
 Timings
 Pipeline Overview
 Pipeline Console
 Restart from Stage
 Replay
 Pipeline Steps
 Workspaces

3.3.2.8. Generate Pipeline Script

- **agent { label 'agent1_66' }**
This tells Jenkins to run the entire pipeline on any node that has the label agent1_66.
- **stages { ... }**
The pipeline is broken into several stages (Checkout, Build, Test, Deploy). Each stage can contain one or more steps.
- **steps { ... }**
Each stage's steps include commands (like echo or shell commands via sh) that will be executed on the selected node.
- **post { ... }**
The post section lets you specify actions that occur after the pipeline completes, whether it was successful or not. Here, for example, it cleans up the workspace and prints a success or failure message.
- **Open a Job with Pipeline Configuration:**
 - If you already have a Pipeline job, go to its configuration page.
 - If not, create a new Pipeline job by selecting "New Item" and choosing "Pipeline".
- **Access the Snippet Generator:**
 - Scroll down to the "**Pipeline**" section of the job configuration.
 - Click on the "**Pipeline Syntax**" link (often located at the bottom of the configuration page).

Dashboard > pip_1 > Pipeline Syntax

Snippet Generator

- Declarative Directive Generator
- Declarative Online Documentation
- Steps Reference
- Global Variables Reference
- Online Documentation
- Examples Reference
- IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step

git: Git

git

Repository URL

https://github.com/VaralaxmiGanta/j_repo1.git

Branch

main

Credentials

varalaxmiganta/******** (git credentials)

+ Add

Include in polling?

Include in changelog?

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with links like Snippet Generator, Declarative Directive Generator, etc. The main area has a heading 'Overview' with a detailed description of what the generator does. Below that is a section for 'Steps' with a 'Sample Step' dropdown set to 'git: Git'. A large panel on the right contains fields for 'git': 'Repository URL' (https://github.com/VaralaxmiGanta/j_repo1.git), 'Branch' (main), 'Credentials' (varalaxmiganta/********), and two checkboxes for 'Include in polling?' and 'Include in changelog?' both of which are checked.

- **Generate a Snippet:**

- In the Pipeline Syntax page, choose the type of step you want to generate (for example, "sh" for shell commands, "git" for source control, etc.).
- Enter the Repo URL, branch and give the credentials.
- Click "**Generate Pipeline Script**".
- The generated Groovy snippet will appear. You can copy this snippet and paste it into your Jenkinsfile or Pipeline script.

Repository URL ?

Branch ?

Credentials ?

[+ Add](#) [Include in polling? ?](#) [Include in changelog? ?](#)[Generate Pipeline Script](#)

```
git branch: 'main', credentialsId: '1', url:  
'https://github.com/VaralaxmiGanta/j_repo1.git'
```



Global Variables

There are many features of the Pipeline that are not steps. These are often exposed via global variables, which are not supported by the snippet generator. See the [Global Variables Reference](#) for details.

- After past it into your pipeline script and save it

Configure

Pipeline

 General

 Pipeline

 Advanced

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 git branch: 'main', credentialsId: '1', url: 'https://gitlab.com/...'  
8                 echo 'Hello World'  
9             }  
10        }  
11    }  
12}  
13
```

try sample Pipeline...

Use Groovy Sandbox ?

Pipeline Syntax

Advanced

Advanced ▾

Save

Apply

[REST API](#)

Jenkins 2.479.3

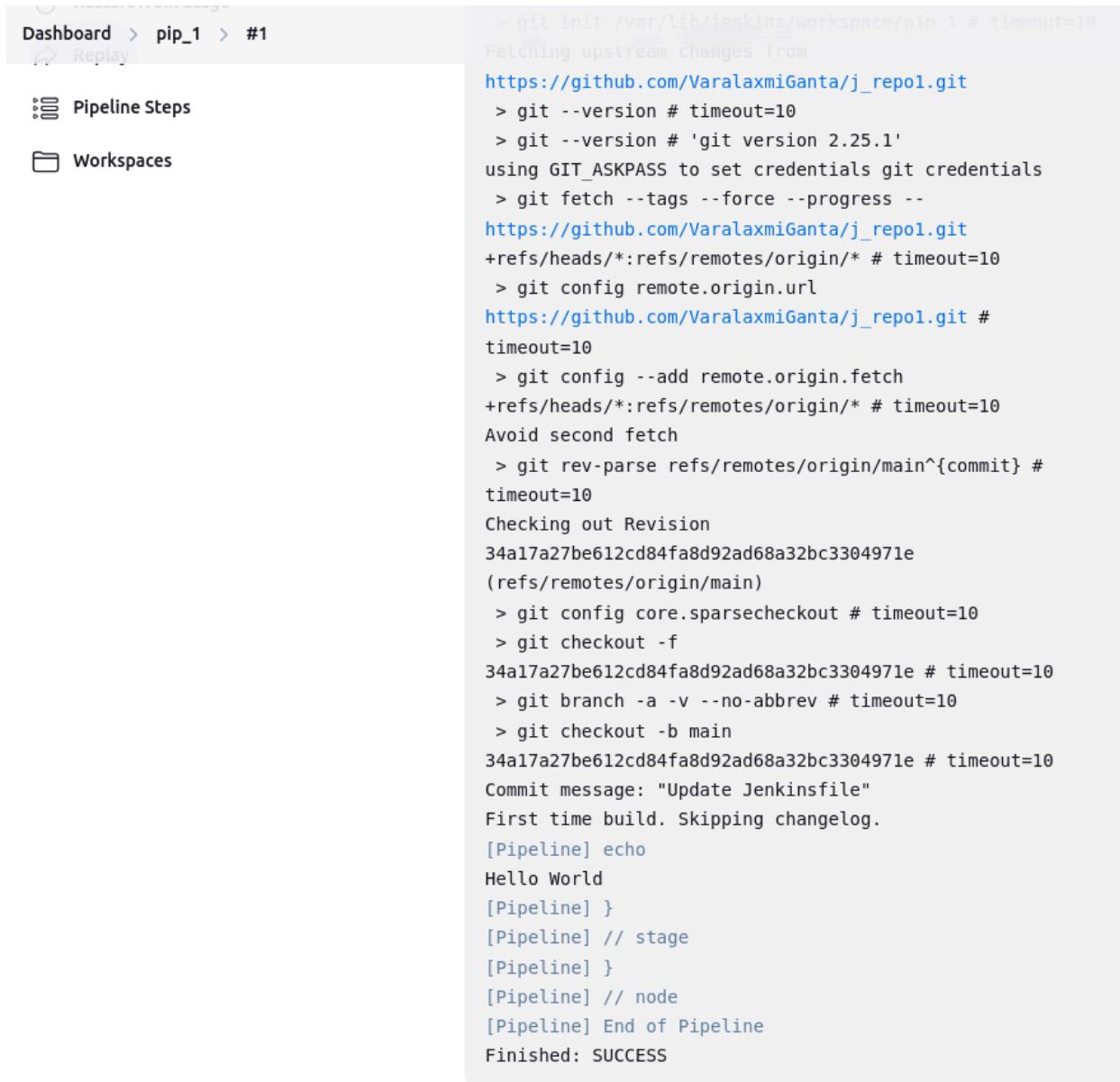
- Click on build now and observe the console output.

[Status](#)[Changes](#)[Console Output](#)[Edit Build Information](#)[Delete build '#1'](#)[Timings](#)[Git Build Data](#)[Pipeline Overview](#)[Pipeline Console](#)[Restart from Stage](#)[Replay](#)[Pipeline Steps](#)[Workspaces](#)

Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user varalaxmi
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/pip_1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] git
The recommended git tool is: NONE
using credential 1
Cloning the remote Git repository
Cloning repository
https://github.com/VaralaxmiGanta/j_repol.git
> git init /var/lib/jenkins/workspace/pip_1 # timeout=10
Fetching upstream changes from
https://github.com/VaralaxmiGanta/j_repol.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_ASKPASS to set credentials git credentials
> git fetch --tags --force --progress --
https://github.com/VaralaxmiGanta/j_repol.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url
https://github.com/VaralaxmiGanta/j_repol.git #
timeout=10
> git config --add remote.origin.fetch
+refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} #
timeout=10
Checking out Revision
34a17a27be612cd84fa8d92ad68a32bc3304971e
```



The screenshot shows a Jenkins pipeline step named 'pip_1' with build number '#1'. The left sidebar has 'Pipeline Steps' and 'Workspaces' options. The main area displays a command-line log of the pipeline execution:

```

> git init /var/lib/jenkins/workspace/pip_1 # timeout=10
Fetching upstream changes from
https://github.com/VaralaxmiGanta/j_repol.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_ASKPASS to set credentials git credentials
> git fetch --tags --force --progress --
https://github.com/VaralaxmiGanta/j_repol.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url
https://github.com/VaralaxmiGanta/j_repol.git #
timeout=10
> git config --add remote.origin.fetch
+refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} #
timeout=10
Checking out Revision
34a17a27be612cd84fa8d92ad68a32bc3304971e
(refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f
34a17a27be612cd84fa8d92ad68a32bc3304971e # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git checkout -b main
34a17a27be612cd84fa8d92ad68a32bc3304971e # timeout=10
Commit message: "Update Jenkinsfile"
First time build. Skipping changelog.
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

3.3.3. Using Maven Project

- Maven is a build automation and project management tool primarily used for Java projects. It simplifies the process of building, testing, and packaging projects, managing dependencies, and creating reports.
- A Maven project is a software project managed using **Apache Maven**. A Maven project uses a **pom.xml** file (Project Object Model) to define the configuration and structure of the project. This file contains information such as dependencies, build settings, plugins, and other configurations.

3.3.3.1. Maven Integration Plugin

- Install Maven Integration Plugins

Plugins

Updates

11

Available plugins

Installed plugins

Advanced settings

maven

Name ↓

Enabled

Maven Integration plugin 3.24

This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.

[Report an issue with this plugin](#)



Jenkins

Dashboard > All > New Item

New Item

Enter an item name
maven_job1

Select an item type

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

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, Folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different Folders.

[OK](#)

3.4. Nodes

- In Jenkins, a **Node** is a machine (physical or virtual) where Jenkins runs jobs (builds, tests, deployments, etc.).
- Nodes can be categorized into two types:
 - **Master (Controller) Node**
 - The main Jenkins instance.
 - Manages the job queue, schedules builds, and distributes work to agent nodes.
 - Can also run jobs, but it's recommended to offload heavy workloads to agents.

- Goto jenkins dashboard → Nodes

The screenshot shows the Jenkins interface for managing nodes. At the top, a navigation bar indicates the path: Dashboard > Nodes > Built-In Node. Below this, a sidebar on the left lists several options: Status (selected), Configure, Build History, Load Statistics, and Script Console. The main content area is titled "Built-In Node". It contains a brief description: "This is the Jenkins controller's built-in node. Builds running on this node will execute on the same system and as the same user as the Jenkins controller. This is appropriate e.g. for special jobs performing backups, but in general you should run builds on agents. [Learn more about distributed builds.](#)". There is also a link to "Monitoring Data". A section titled "Build Executor Status" shows "(0 of 2 executors busy)". Another section titled "Projects tied to Built-In Node" shows "None".

- **Agent (slave) Node**
 - A secondary machine that connects to the master.
 - Executes jobs as assigned by the master.

3.4.1. Why ?

- **Scalability** – Run builds on multiple machines to handle large projects efficiently.
- **Load Distribution** – Offload CPU/memory-intensive tasks from the master to agents.
- **Platform-Specific Builds** – Run jobs on different OS platforms (Linux, Windows, macOS).
- **Parallel Execution** – Execute multiple builds at the same time.

3.4.2. Configure a Node

The screenshot shows the Jenkins 'Manage Jenkins' page with the 'Nodes' section selected. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'System', 'Configure global settings and paths.', and 'Tools' (Configure tools, their locations and automatic installers). The 'Nodes' section contains four items: 'Plugins' (13), 'Nodes', 'Clouds', and 'Appearance'. Each item has a brief description and an icon.

- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins

- Goto Manage Jenkins → Nodes

The screenshot shows the Jenkins 'Nodes' management interface. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Nodes', '+ New Node', 'Configure Monitors', and a refresh icon. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0 of 2 executors busy). The main area is titled 'Nodes' and displays a table of configured nodes. A 'New Node' button is available at the top right of the table.

S	Name	Architecture	Clock Difference	Free Disk Space	Fr Sw Spa
	Built-In Node	Linux (amd64)	In sync	27.04 GiB	22.69
		last checked	20 min	20 min	20 min

Icon: S M L Legend

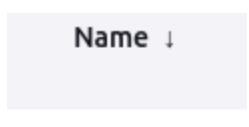
3.4.2.1. Build Queue

- “No builds in the queue” → indicates that currently, no jobs are waiting to be executed.
- If jobs are in the queue, Jenkins schedules them based on available executors.

3.4.2.2. Build Executor Status

- “0 of 2 executors busy”
- Indicates that jenkins has 2 executors (workers that execute jobs).
- currently , none are running a job
- If a job starts, the status changes.

3.4.2.3. Nodes Section

Column	Description	Purpose	
Name	The node's name (eg built-in node for the master/ controller node)	Identifies the node	 Built-In Node
Architecture	The system architecture (eg linux (amd64) for a 64-bit linux machine)	Displays the system architecture	<pre>vlab@HYVLAB7:~\$ uname -x86_64</pre>
Clock difference	Shows if there is any time difference between the jenkins master and the agent and sync status is always yes	Shows if there is any time drift between the Jenkins controller and agents.	<pre>vlab@HYVLAB7:~\$ timedatectl status Local time: Thu 2025-02-06 Universal time: Thu 2025-02-06 RTC time: Thu 2025-02-06 Time zone: Asia/Kolkata (I) System clock synchronized: yes NTP service: active RTC in local TZ: no</pre>
Free disk space	Available disk space on the node	Indicates available disk space on the node. Low space may cause build failures.	<pre>vlab@HYVLAB7:~\$ df -h / Filesystem Size Used Avail Use% /dev/sda1 46G 17G 27G 38%</pre>
Free swap space	Amount of swap memory available	Displays available swap memory, which helps prevent crashes when RAM is full.	<pre>vlab@HYVLAB7:~\$ free -h total used free shared buff/cache Mem: 7.7Gi 3.4Gi 138Mi 215Mi Swap: 2.3Gi 945Mi 22Gi</pre>
Free temp space	Space available in the temporary directory	Shows available space in /tmp, which is used for temporary build files and logs.	<pre>vlab@HYVLAB7:~\$ df -h /tmp Filesystem Size Used Avail Use% /dev/sda1 46G 17G 27G 38%</pre>
Response time	Network response time from the master to the node To check response time from the jenkins master to a node (eg an agent with IP 192.168.1.100)	Measures the network response time from the controller to the agent. High response time may indicate network issues.	<pre>vlab@HYVLAB7:~\$ ping -c 4 192.168.1.100 PING 192.168.1.100 (192.168.1.100) 56(84) bytes of 192.168.1.100 ping statistics ... 4 packets transmitted, 0 received, 100% packet loss</pre>

3.4.2.4. Create Agent1 and Configuration

3.4.2.4.1. Requirements

- Add the credentials to allow jenkins to connect via SSH to the agent machine (172.16.203.40).
 - Under credentials, make sure added the correct SSH username and private key for the agent machine

Username and ip address

```
vlab@HYVLAB8:~$ whoami  
vlab
```

```
vlab@HYVLAB8:~$ hostname -I  
172.16.203.40 172.18.0.1 172.17.0.1
```

Private key :

```
vlab@HYVLAB8:~/.ssh$ ls  
authorized_keys  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

New credentials

Kind

SSH Username with private key

Scope

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

ID

ip_40

Description

this is agent(jyothsna) machine

Username

vlab

Treat username as secret

Private Key

Enter directly

Key

```
-----BEGIN OPENSSH PRIVATE KEY-----
```

Enter New Secret Below

Create

- **Remote Root Directory :**

- Path : /home/vara/jenkins_agent1/
- This is the working directory on the agent machine where jenkins stores: job workspace, build artifacts, and logs
- Give the correct ownership
- **\$ sudo chown -R vlab:vlab /home/vara**

```
vlab@HYVLAB8:~$ sudo chown vlab:vlab /home/vara
vlab@HYVLAB8:~$ ls -ld /home/vara
drwxr-xr-x 4 vlab vlab 4096 Feb  6 12:16 /home/vara
```

```
vlab@HYVLAB8:~$ sudo useradd -m -d /var/lib/jenkins -s /bin/bash jenkins
vlab@HYVLAB8:~$ getent passwd jenkins
jenkins:x:1001:1001::/var/lib/jenkins:/bin/bash
vlab@HYVLAB8:~$ sudo chown -R jenkins:jenkins /home/vara/jenkins-agent1/
vlab@HYVLAB8:~$ ls -ld /home/vara/jenkins-agent1/
drwxr-xr-x 2 jenkins jenkins 4096 Feb  6 11:41 /home/vara/jenkins-agent1/
```

```
vlab@HYVLAB7:/var/lib/jenkins$ cat .ssh/known_hosts
172.16.203.40 ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCaTwR/qJnLFE0jk9AsGVTZTNYiouX7cP8sqpNS25cylsEHfI9
+PIRb/MU2ADUuFHEnBexSd88AXPi4TFFJTPjOHY3ES/7r0L7uDwX4d2lExm8f/3N7zcpzqeUbfq2ZR2u1lQsWnFxhsX/v1SR3c7hG
VDloOW1Bwr0HDCs5dMRjUS2gqWpjH4/IH0ld4HAM5d2VG0YMLn+NJ6vEvYf2xqDexaEZ9SsDG2jsCKhVgeuezq9NbENXdolc6wrdK
Mvb189cfgSdPH1uoX6LztA29DIDJoXaI4okuYTrVq/hhZCmLsr44yLotWBx8xTENSE3TAx8sksd92mRLSxaofWrDFIWAfsVLLqBI
1Ly/6gvTjY5W7GNo00Y2iytq9/jsAv7vQ8aGzm3UUUsMBel2j1LIZkYIRLgNVnxPD/1JQCX83W3VEmjTwK9mrVI/YrW+k6X+VX7cS
RscpcWiSqeUoD0Wdg1g5e0iiapp8ycNk3Aqfj3xerdckZh4YtpnEDXywLtc=
172.16.203.40 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAIBmlzdHAyNTYAAABBBJzyz7FZxdtvVMM
+xgs7bFsKZ0gDW08Rb/9MANONMJC6Swj0V023kqvJ0Zzi/DLgf8Zkem9TE5R7pJzIoeIEWuI=
172.16.203.40 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAIbRCp10wJs+c7/QfGQpSESX3djXEfoRKBZz9BKq2kEe
```

```
vlab@HYVLAB7:/var/lib/jenkins$ sudo ssh-keygen -R 172.16.203.40 -f .ssh/known_hosts
# Host 172.16.203.40 found: line 1
# Host 172.16.203.40 found: line 2
# Host 172.16.203.40 found: line 3
.ssh/known_hosts updated.
Original contents retained as .ssh/known_hosts.old
```

3.4.2.4.2. Set up

- Click on new node and enter the node name

Dashboard > Nodes > New node

New node

Node name

agent1_66

Type



Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

- **Name:** agent1_40 → This is the identifier for the node.
- **Description:** "This is agent1" → A human-readable description to help distinguish it.
- **Number of Executors:** 4 → This agent can run **4 parallel builds** at a time.
- **Labels:** these labels help assign specific jobs to this agent.
 - A Job can use agent {label 'docker'} to ensure it runs on an agent with Docker installed.
- **Usage: Use this node as much as possible**
 - This means Jenkins can schedule **any available job** on this node.
 - If you want to restrict it to certain jobs, change this to "**only build job with matching labels**".
- **Launch method:**
 - **Launch agents via SSH**
 - This means the Jenkins controller will connect to the agent over SSH.

Status

Delete Agent

Configure (selected)

Build History

Load Statistics

Script Console

Log

System Information

Disconnect

Build Executor Status (0 of 4 executors busy)

Name ?
agent1_40

Description ?
This is agent1
[Plain text](#) [Preview](#)

Number of executors ?
4

Remote root directory ?
/home/vara/jenkins_agent1/

Labels ?
linux docker maven agent1_40

Usage ?
Use this node as much as possible

Launch method ?
Launch agents via SSH

Save

- Host : this should be the IP address or hostname of the agent machine.
- Jenkins requires **SSH credentials** to connect the agent machine.
- Check the Known Hosts file:
 - The **known hosts** file on the Jenkins master must include the agent machine's SSH fingerprint.
 - On the Jenkins master, check if the agent machine's fingerprint is listed in `~/.ssh/known_hosts`. If not, run this command from the Jenkins master to add the fingerprint: `$ ssh-keyscan 172.16.203.40 >> ~/.ssh/known_hosts`

Dashboard > Nodes > agent1_40 > Configure

Launch agents via SSH

Host ?
172.16.203.40

Credentials ?
this is agent(jyothsna) machine

+ Add

Host Key Verification Strategy ?
Non verifying Verification Strategy

Advanced ▾ Edited

Availability ?
Keep this agent online as much as possible

Node Properties

Disable deferred wipeout on this node ?
 Disk Space Monitoring Thresholds
 Environment variables
 Notify when Node online status changes ?
 Tool Locations

Save

REST API Jenkins 2.479.3

- **Host Key Verification Strategy :**
 - **Non verifying verification strategy:**
 - Disables host key verification and is less secure because it does not validate the server's identity.
 - **Known Hosts File Verification Strategy:**
 - Uses a known_hosts file (typically located in the Jenkins master's or agent's .ssh directory) to verify the SSH host key.

```
$ ssh-keyscan 172.16.203.40 | sudo tee -a .ssh/known_hosts
```

```
vlab@HYVLAB7:/var/lib/jenkins$ ssh-keyscan 172.16.203.40 | sudo tee -a .ssh/known_hosts
# 172.16.203.40:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.11
172.16.203.40 ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQCaTwR/qJnLFE0jk9AsGVTZTNyiouX7cP8sqpNS25cYlsEHFI9
+PIRb/MU2ADUuFHEnBexSd88AXPi4TFFJTPj0HY3ES/7r0L7uDwX4d2lExm8f/3N7zcpzqeUbfq2ZR2u1lQsWnFxhsX/v1SR3c7hG
VDloOW1Bwr0HDCs5dMRjUS2gqWpjH4/IH0ld4HAM5d2VGoYMLn+NJ6vEvYf2xqDexaEz9SsDG2jsCKhVgeuezq9NbENXdolc6wrdK
Mvb189cfgSdPH1uoX6LztA29DIDJoXaI4okuYTrVq/hhZCmLsr44yLotWBx8xTENSE3Tax8sksd92miRLSxaofWrDFIWFsVLLqBI
1Ly/6gvTjY5W7GN0o0Y2iytq9/jsAv7vQ8aGzm3UUsMBe1j1LIZkYIRLgVNvxPD/1JQCX83W3VEmjTwK9mrVI/YrW+w+k6X+vX7cS
RscpcWiSqueuUoD0Wdg1g5e0iapp8ycNk3Aqfj3xerdckZh4YtpnEDXywLtc=
172.16.203.40 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoyTItbmlzdHAyNTYAAAAIBmlzdHAyNTYAAABBBJzyz7FZxdtvVMM
+xgs7bFsKZ0gDW08Rb/9MANONMJC6Swj0V023kqvj0ZZi/DLgf8Zkem9TE5R7pJzIoeIEwUI=
172.16.203.40 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBmRCp10wJs+c7/QfGQpSESX3djXEfoRKBZz9BKq2kEe
```

- When the agent connects, Jenkins compares the host key provided by the agent with the entries in the known_hosts file. If there's a match, the connection proceeds; if not, it fails.
- Requires maintenance of the known_hosts file, and if the remote host's key changes (for example, due to a server rebuild), you must update the file accordingly.

Check if the system is online or offline at status:

The screenshot shows the Jenkins Node configuration page for 'agent1_40'. The top navigation bar shows 'Dashboard > Nodes > agent1_40'. The main content area has several tabs: 'Status' (selected), 'Delete Agent', 'Configure', 'Build History', 'Load Statistics', 'Script Console', 'Log', 'System Information', and 'Disconnect'. The 'Status' tab displays the message 'This is agent1'. Below this are sections for 'Monitoring Data' (with a dropdown menu) and 'Labels' (with tags: maven, linux, docker). A section titled 'Projects tied to agent1_40' shows 'None'. At the bottom, a 'Build Executor Status' box indicates '(0 of 4 executors busy)'.

3.4.2.4.3. Log

Dashboard > Nodes > agent3 > Log

Status
Delete Agent
Configure
Build History
Load Statistics
Script Console
Log
System Information
Disconnect

Build Executor Status
(0 of 4 executors busy)

```
SSHLauncher{host='172.16.203.40', port=22,
credentialsId='ip_40', jvmOptions='', javaPath='',
prefixStartSlaveCmd='', suffixStartSlaveCmd='',
launchTimeoutSeconds=60, maxNumRetries=10,
retryWaitTime=15,
sshHostKeyVerificationStrategy=hudson.plugins.sshslaves.Verifiers.NonVerifyingKeyVerificationStrategy,
tcpNoDelay=true, trackCredentials=true}
[02/07/25 09:29:16] [SSH] Opening SSH connection to 172.16.203.40:22.
[02/07/25 09:29:17] [SSH] WARNING: SSH Host Keys are not being verified. Man-in-the-middle attacks may be possible against this connection.
[02/07/25 09:29:17] [SSH] Authentication successful.
[02/07/25 09:29:19] [SSH] The remote user's environment is:
BASH=/usr/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:extquote
:force_fignore:globasciiranges:hostcomplete:interactive_comments:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=([0]="0")
BASH_ARGV=()
BASH_CMDS=()
BASH_EXECUTION_STRING=set
BASH_lineno=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="5" [1]="0" [2]="17" [3]="1"
[4]="release" [5]="x86_64-pc-linux-gnu")
BASH_VERSION='5.0.17(1)-release'
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
DIRSTACK=()
EUID=1000
GROUPS=()
HOME=/home/vlab
HOSTNAME=HVVIARR
```

SSH_CONNECTION= 172.16.203.85 41094 172.16.203.40 22
TERM=dumb
UID=1000
USER=vlab
XDG_RUNTIME_DIR=/run/user/1000
XDG_SESSION_CLASS=user
XDG_SESSION_ID=206
XDG_SESSION_TYPE=tty
_=']'

[02/07/25 09:29:19] [SSH] Starting sftp client.
[02/07/25 09:29:19] [SSH] Copying latest remoting.jar...
Source agent hash is 28E3A3D224EDC0F7379AD95EB0A5D4B2.
Installed agent hash is 28E3A3D224EDC0F7379AD95EB0A5D4B2
Verified agent jar. No update is necessary.
Expanded the channel window size to 4MB
[02/07/25 09:29:19] [SSH] Starting agent process: cd "/home/veena/jenkins" && java -jar remoting.jar -workDir /home/veena/jenkins -jar-cache /home/veena/jenkins/remoting/jarCache
Feb 07, 2025 9:29:20 AM
org.jenkinsci.remoting.engine.WorkDirManager
initializeWorkDir
INFO: Using /home/veena/jenkins/remoting as a remoting work directory
Feb 07, 2025 9:29:20 AM
org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/veena/jenkins/remoting
<===[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 3261.v9c670a_4748a_9
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
Agent successfully connected and online

3.4.2.4.4. System Information

The screenshot shows the Jenkins Node details page for 'agent1_40'. The top navigation bar shows 'Dashboard > Nodes > agent1_40'. On the left, a sidebar lists various management options: Status (highlighted), Delete Agent, Configure, Build History, Load Statistics, Script Console, Log, System Information (highlighted), and Disconnect. Below this is a 'Build Executor Status' section indicating '(0 of 4 executors busy)'. The main content area has tabs for 'Agent agent1_40' (selected), 'Edit description', and 'Mark this node temporarily offline'. A status message says 'This is agent1'. A 'Monitoring Data' section contains a table with system metrics:

Architecture	Linux (amd64)
Clock Difference	In sync
Free Disk Space	82.95 GiB
Free Swap Space	843.13 MiB
Free Temp Space	23.98 GiB
Response Time	82ms

The 'Labels' section shows 'maven', 'linux', and 'docker' assigned to this node. The 'Projects tied to agent1_40' section shows 'None'.

3.4.2.5. Run a job on a specific node in jenkins

- From the Jenkins dashboard, create a new Freestyle project or open an existing one.
- In the job configuration, check the option "**Restrict where this project can be run**".
- In the field that appears, enter the **label** that you assigned to your node
- Save the configuration.
- Click "**Build Now**" to start the job.

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

Description

[Plain text](#) [Preview](#) Discard old builds  GitHub project Notify when Job configuration changes This project is parameterised  Throttle builds  Execute concurrent builds if necessary  Restrict where this project can be run Label Expression 

agent3

Label **agent3** matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Advanced ▾

 Save Apply

- Jenkins will dispatch the job to an available node that matches the specified label.
- **Console output:**

The screenshot shows a web browser window with three tabs open:

- QnA
- Jenkins - Google Docs
- job1 #7 Console [Jenkins]

The active tab is "job1 #7 Console [Jenkins]". The URL in the address bar is `172.16.203.85:8080/job/job1/7/console`. The page title is "Jenkins".

The main content area displays the Jenkins Console Output for build #7 of the "job1" job. The output is as follows:

```
Started by user varalaxmi
Running as SYSTEM
Building remotely on agent3 (linux docker) in workspace
/home/veena/jenkins/workspace/job1
No emails were triggered.
[job1] $ /bin/sh -xe /tmp/jenkins3989172492026491720.sh
+ echo hello
hello
No emails were triggered.
Finished: SUCCESS
```

On the left sidebar, the following options are listed:

- Status
- </> Changes
- Console Output** (highlighted)
- Edit Build Information
- Delete build '#7'
- Timings
- ← Previous Build

At the bottom right of the page, there are links for "REST API" and "Jenkins 2.479.3".

Check Build Executor status:

The screenshot shows the Jenkins dashboard interface. At the top left, there is a breadcrumb navigation: "Dashboard >". Below this is a horizontal menu bar with several items: "+ New Item", "Build History" (with a folder icon), "Project Relationship" (with a circular icon), "Check File Fingerprint" (with a fingerprint icon), "Manage Jenkins" (with a gear icon), and "My Views" (with a square icon). The main content area contains two expandable sections. The first section, titled "Build Queue" with a downward arrow icon, displays the message "No builds in the queue.". The second section, titled "Build Executor Status" with an upward arrow icon, displays the message "built-in node + 2 agents (-4 of 6 executors busy)".

Dashboard >

- + New Item
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views

Build Queue ▼

No builds in the queue.

Build Executor Status ^

built-in node + 2 agents (-4 of 6 executors busy)

3.4.2.5.1. Check the builds

- Agents do not maintain important data. All job configuration, build logs and artifacts are stored on the controller, so it would be possible to use a temporary directory as the agent root directory.

Number of executors ?

4

Remote root directory ?

/home/veena/jenkins/

An agent needs to have a directory dedicated to Jenkins. Specify the path to this directory on the agent. It is best to use an absolute path, such as /var/jenkins or c:\jenkins . This should be a path local to the agent machine. There is no need for this path to be visible from the controller.

Agents do not maintain important data; all job configurations, build logs and artifacts are stored on the controller, so it would be possible to use a temporary directory as the agent root directory.

However, by giving an agent a directory that is not deleted after a machine reboot, for example, the agent can cache data such as tool installations, or build workspaces. This prevents unnecessary downloading of tools, or checking out source code again when builds start to run on this agent again after a reboot.

If you use a relative path, such as ./jenkins-agent , the path will be relative to the working directory provided by the *Launch method*.

- For launchers where Jenkins controls starting the agent process, such as SSH, the current working directory will typically be consistent, e.g. the user's home directory.
- For launchers where Jenkins has no control over starting the agent process, such as inbound agents launched from the command line, the current working directory may change between launches of the agent and use of a relative path may prove problematic.

The principal issue encountered when using relative paths with inbound launchers is the proliferation of stale workspaces and tool installation on the agent machine. This can cause disk space issues.

Labels ?

Save

This is master machine:

```
vlab@HYVLAB7:/var/lib/jenkins/jobs$ cd job1
job1/ job11/
vlab@HYVLAB7:/var/lib/jenkins/jobs$ cd job1/
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1$ ls
builds config.xml nextBuildNumber
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1$ cd builds/
vlab@HYVLAB7:/var/lib/jenkins/jobs/job1/builds$ ls
1 10 11 2 3 4 5 6 7 8 9 legacyIds permalinks
```

This is Agent machine:

```
vlab@HYVLAB8:~$ cd /home/veena/jenkins/
vlab@HYVLAB8:/home/veena/jenkins$ ls
remoting remoting.jar workspace
vlab@HYVLAB8:/home/veena/jenkins$ cd workspace/
vlab@HYVLAB8:/home/veena/jenkins/workspace$ ls
job1 job11 workspaces.txt
vlab@HYVLAB8:/home/veena/jenkins/workspace$ cd job1
```

4. Automate Job

- Jenkins has the ability to automatically create, update, and delete jobs based on the repositories.
- There are multiple alternatives for automatic job management
 - Use organization folders
 - Use multibranch pipelines
 - Use pipeline

4.1. Use organization folders

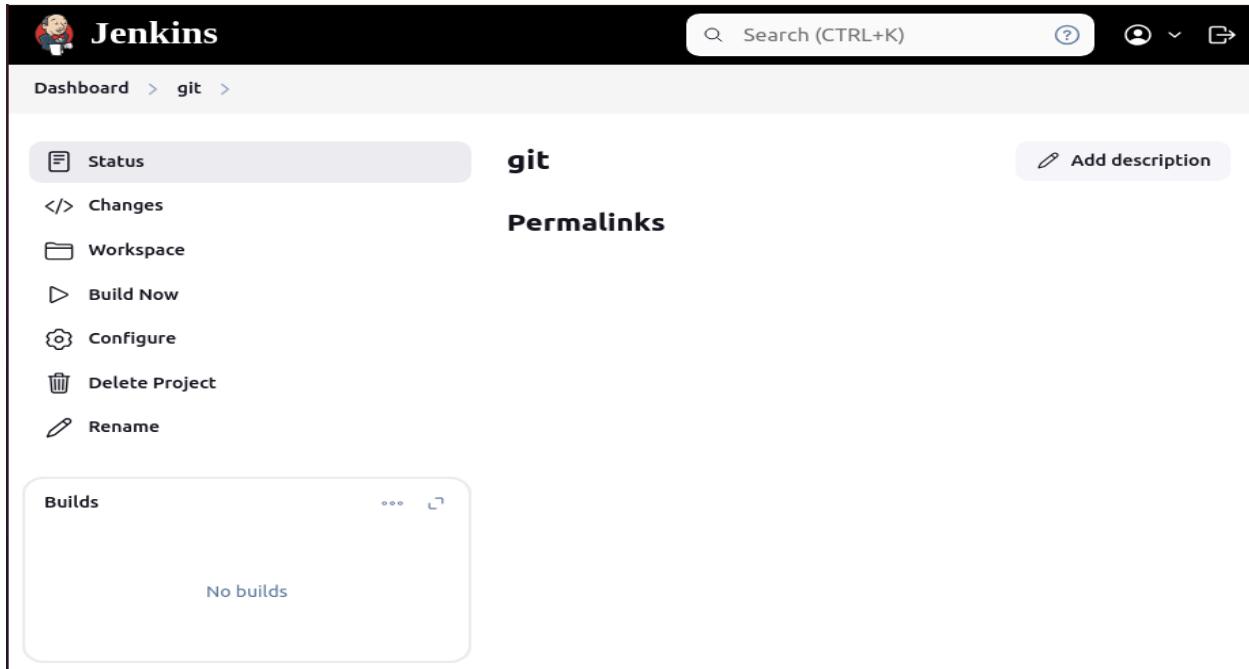
- Management of jobs based on the repositories and branches within a source code management (SCM) organization.
- Automatically create and delete jobs as repositories or branches are added or removed from the SCM system.

4.1.1. Git Polling

- Jenkins periodically checks the Git repository for changes using the configured schedule.

4.1.1.1. Using HTTP

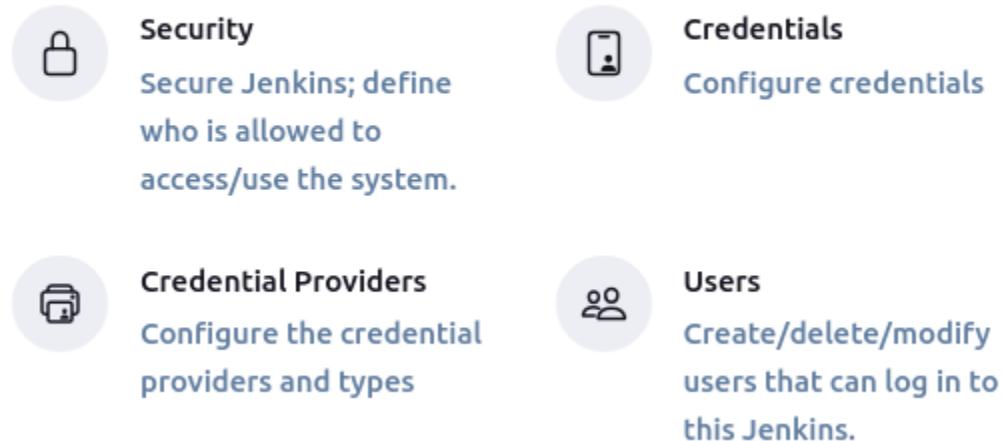
- Click new item then enter name and select freestyle project after click ok



- Add the Token to Jenkins Credentials:

- Go to Jenkins dashboard.
- Click on **Manage Jenkins** > In Security select **Credentials -> add credentials**

Security



- Set the **Kind** to **Username with password**.

The screenshot shows the Jenkins 'Update credentials' page for a GitHub credential. The URL in the browser is: [Dashboard > Manage Jenkins > Credentials > System > Global credentials \(unrestricted\) > varalaxmiganta/***** \(git cred\)](#)

The page has the following fields:

- Update**, **Delete**, **Move** buttons.
- Scope**: Global (Jenkins, nodes, items, all child items, etc).
- Username**: varalaxmiganta
- Treat username as secret**:
- Password**: Concealed, **Change Password** button.
- ID**: 1
- Description**: git credentials
- Save** button.

- In **Username**, put your **GitHub username**.

- In **Password**, paste the **Personal Access Token (PAT)** you generated.

The screenshot shows the GitHub Developer Settings page under Personal access tokens (classic). It lists two tokens:

- jenkins1** — admin:enterprise, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages
Last used within the last week
Expires on Sat, Mar 1 2025.
- jenkins** — admin:public_key, delete:packages, delete_repo, repo, user, workflow, write:packages
Last used within the last week
Expires on Thu, Feb 27 2025.

A note at the bottom explains that Personal access tokens (classic) function like ordinary OAuth access tokens and can be used instead of a password for Git over HTTPS, or to authenticate to the API over Basic Authentication.

- **Update Git Repository URL in Jenkins Job:**

The screenshot shows a GitHub repository page for **j_repo1**. The repository details include:

- Code** tab selected.
- Owner**: VaralaxmiGanta
- Private**
- Clone** section:
 - Local
 - Codespaces
 - HTTPS** (selected)
 - SSH**
 - GitHub CLI**
 - Clone using the web URL: https://github.com/VaralaxmiGanta/j_repo1
 - Download ZIP**
- About** section:
 - No description, website, or topics provided.
 - Readme**
 - Activity**
 - 0 stars
 - 1 watching
 - 0 forks
- Releases** section: No releases published. Create a new release.
- Packages** section: No packages published. Publish your first package.
- Languages** section: Shell 100.0%

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

 None Git [?](#)

Repositories [?](#)

Repository URL [?](#)

Credentials [?](#)

[+ Add](#)[Advanced ▾](#)[Add Repository](#)

Branches to build [?](#)

Branch Specifier (blank for 'any') [?](#)

[Add Branch](#)

Repository browser [?](#)

Additional Behaviours

[Save](#)[Apply](#)

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 ?**Schedule** ?

⚠ Do you really mean "every minute" when you say "***"? Perhaps you meant "H *****" to poll once per hour**

Would last have run at Thursday, 30 January, 2025 at 4:22:35 pm India Standard Time; would next run at Thursday, 30 January, 2025 at 4:22:35 pm India Standard Time.

 Ignore post-commit hooks ?**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 ?**Save****Apply****In the Schedule :**Goto the https://crontab.guru/#* * * *

Crontab Guru is a simple, web-based tool that helps you understand and generate cron expressions. A cron expression is used in the cron utility to schedule jobs on Unix-like systems

← → ⌂ crontab.guru/#* _ _ _ _

☆ ⚙ 🌐 :

Cronitor

Cron Job Monitoring

crontab guru

The quick and simple editor for cron schedule expressions by [Cronitor](#)

“At every minute.”

[next](#) at 2025-01-30 14:19:00 [random](#)

* * * * *

minute	hour	day (month)	month	day (week)
*	*	*	*	*
* any value				
' value list				
, separator				
- range of values				
/ step values				
@yearly (non-standard)				
@annually (non-standard)				
@monthly (non-standard)				
@weekly (non-standard)				
@daily (non-standard)				
@hourly (non-standard)				
@reboot (non-standard)				

How to monitor your cron jobs:

The screenshot shows the Jenkins configuration interface for a job named 'git3'. The left sidebar lists configuration sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps (which is selected and highlighted in grey), and Post-build Actions. The main area is titled 'Build Steps' and contains a single step named 'Run with timeout'. Under 'Time-out strategy', 'Absolute' is selected with a value of '1'. Under 'Build Step', 'Execute shell' is chosen, and the command 'echo "success"' is entered in the text area. At the bottom, there are 'Save' and 'Apply' buttons.

when you change the code in a Git repository, a new commit is created. If you have configured Jenkins to trigger a build upon detecting changes in the Git repository, Jenkins will automatically start a build when it detects this new commit.

4.1.1.2. Using SSH

- **Generate SSH Key Pair:**
 - \$ ssh-keygen -t rsa -b 4096 -C "your_email@gmail.com"

- **Key Pair Created:** After completing the above steps, your SSH key pair will be created:
 - The **private key** will be saved in the file: `~/.ssh/id_rsa`.
 - The **public key** will be saved in the file: `~/.ssh/id_rsa.pub`.

```
$ ls .ssh/
config id_rsa id_rsa.pub known_hosts known_hosts.old veena
veena.pub
```
- Copy the private key and it to the global credentials in jenkins.
- **Test the SSH key:**
 - `$ ssh -T git@github.com`

The screenshot shows the Jenkins 'Update credentials' page for a GitHub SSH Key. The navigation path is: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > VaralaxmiGanta (GitHub SSH Key). The page title is 'Update credentials'. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' is 'GitHub_SSH_Key', and the 'Description' is 'GitHub SSH Key'. The 'Username' is 'VaralaxmiGanta'. There is an unchecked checkbox for 'Treat username as secret'. Under 'Private Key', the 'Enter directly' option is selected, and the key value is shown as 'Concealed for Confidentiality' with a 'Replace' button. A 'Passphrase' field is present at the bottom, and a 'Save' button is at the very bottom right.

Note : user name must be exact

- **Configure Git Repository in Jenkins:**
 - Update your Git repository URL in Jenkins to use the SSH URL

The screenshot shows the Jenkins 'Configuration' page for a Git repository. The top navigation bar includes 'Dashboard', 'git', 'Configuration', and 'Source Code Management'. The left sidebar lists 'General', 'Source Code Management' (which is selected), 'Build Triggers', 'Build Environment', 'Build Steps', and 'Post-build Actions'. The main configuration area is titled 'Configure'.

Source Code Management: The 'Git' option is selected. The 'Repository URL' is set to `git@github.com:VaralaxmiGanta/j_repo1.git`. The 'Credentials' dropdown is set to 'VaralaxmiGanta (GitHub SSH Key)'. A red 'X' icon is visible next to the repository URL field.

Branches to build: The 'Branch Specifier' is set to `*/main`.

Repository browser: The dropdown is set to '(Auto)'.

Additional Behaviours: Buttons for 'Save' and 'Apply' are present.

- **Git Polling Log**

Git Polling Log

```

Started on 31-Jan-2025, 12:37:00 pm
Using strategy: Default
[poll] Last Built Revision: Revision
b6f4cee62747018cacf1610ceae1987339162cbf
(refs/remotes/origin/main)
The recommended git tool is: NONE
using credential GitHub_SSH_Key
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_SSH to set credentials GitHub SSH Key
Verifying host key using known hosts file
> git ls-remote -h --
git@github.com:VaralaxmiGanta/j_repol.git # timeout=10
Found 3 remote heads on
git@github.com:VaralaxmiGanta/j_repol.git
[poll] Latest remote head revision on refs/heads/main is:
b6f4cee62747018cacf1610ceae1987339162cbf - already built
by 9
Done. Took 2.3 sec
No changes

```

4.1.2. Git Webhook

4.1.2.1. Required Plugin (Generic Webhook Trigger Plugin)

- Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.

<https://plugins.jenkins.io/generic-webhook-trigger>

4.1.2.2. ngrok

- **ngrok** is a tool that creates a secure tunnel from a public endpoint (usually an internet address) to your local machine, enabling external access to local applications or services behind a firewall or NAT (Network Address Translation).
- It is especially useful for testing, development, and sharing web services or APIs running on your local machine.
- **Login into the ngrok dashboard:**
<https://dashboard.ngrok.com/get-started/setup/linux>

- Create one account to access it.
- **Install ngrok via Apt with the following command:**

```
$ curl -sSL https://ngrok-agent.s3.amazonaws.com/ngrok.asc \
| sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null \
&& echo "deb https://ngrok-agent.s3.amazonaws.com buster main" \
| sudo tee /etc/apt/sources.list.d/ngrok.list \
&& sudo apt update \
&& sudo apt install ngrok
```

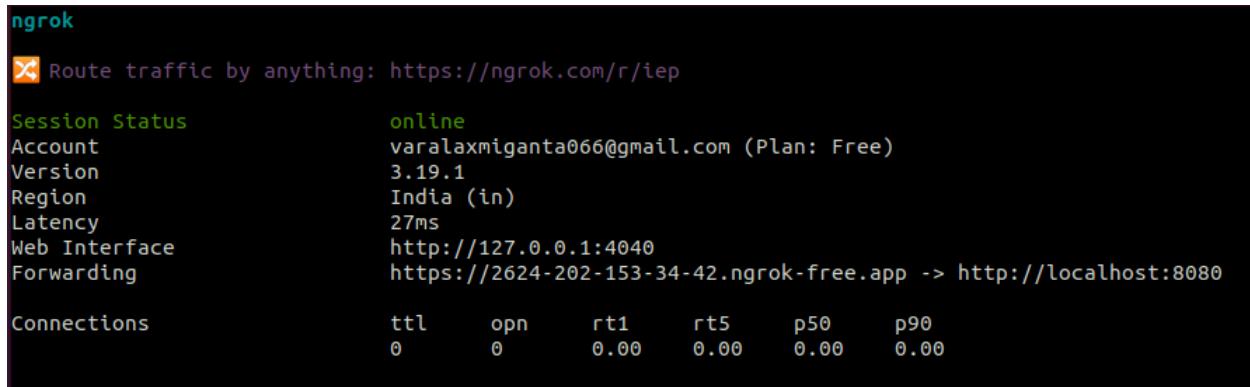
The screenshot shows the ngrok website's installation page for Linux. At the top, there's a navigation bar with a menu icon, the word "ngrok", and a "Choose another platform" button. Below the navigation is a section for "Linux" with a penguin icon and the word "Agent". Underneath, there's a heading "Installation" and a navigation bar with "Apt" (which is underlined), "Download", and "Snap". A sub-section titled "Install ngrok via Apt with the following command:" contains a code block with the same command shown above. Below this, another code block shows the command "ngrok config add-authtoken 2s0BKGpyfVZ1pB8RBZSw0P5ayog_49yKxMvmCZ31Wky9mqPpq". At the bottom of the page, there's a section titled "Deploy your app online" with tabs for "Ephemeral Domain" (which is underlined) and "Static Domain". A note says to put the app online at an ephemeral domain. Below that is a code block with "ngrok http http://localhost:8080" and a copy icon. A note at the bottom says "Once running, your endpoints will be listed on the [endpoints page](#)".

- Run the following command to add your authtoken to the default ngrok.yml

```
$ sudo ngrok config add-authtoken  
2sOBKGpyfVZ1pB8RBZSw0P5ayog_49yKxMvmCZ31Wky9mqPpq
```

- **Open terminal**

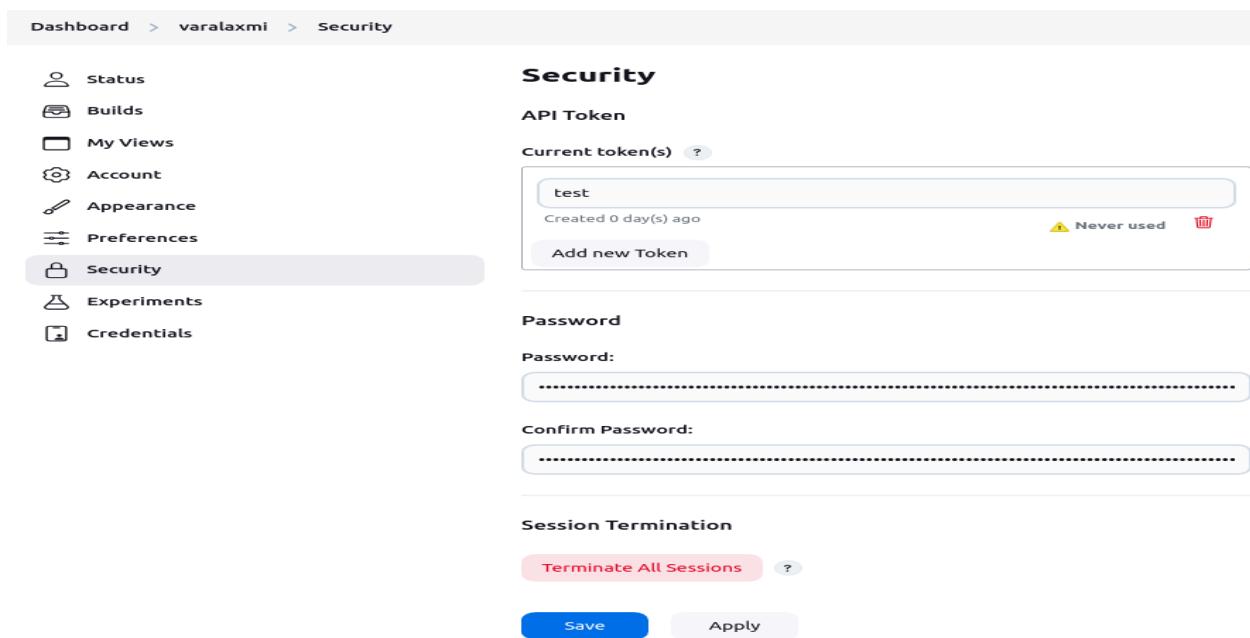
```
$ sudo ngrok http http://localhost:8080
```



```
ngrok  
X Route traffic by anything: https://ngrok.com/r/iep  
  
Session Status          online  
Account                 varalaxmiganta066@gmail.com (Plan: Free)  
Version                3.19.1  
Region                 India (in)  
Latency                27ms  
Web Interface          http://127.0.0.1:4040  
Forwarding             https://2624-202-153-34-42.ngrok-free.app -> http://localhost:8080  
  
Connections            ttl     opn      rt1      rt5      p50      p90  
                      0       0       0.00    0.00    0.00    0.00
```

- **In github add webhook for one required repo:**

- Copy the forwarding url into the payload URL.
- Select content type is **application/json**
- Secret key is get from the jenkins:



- Open jenkins dashboard → click admin icon → security
- In the API Token add new token → first enter name and click on generate token copy it and paste it on the jenkins.

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

`https://2624-202-153-34-42.ngrok-free.app/github-webhook/`

Content type *

`application/json`

Secret

`1139daaa17d47fba027b42a71ebc7e7d13`

SSL verification

By default, we verify SSL certificates when delivering payloads.

Enable SSL verification **Disable (not recommended)**

Which events would you like to trigger this webhook?

Just the push event.
 Send me everything.
 Let me select individual events.

Active
 We will deliver event details when this hook is triggered.

Add webhook

- Add the webhook it looks like this

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>

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

`✓ https://2624-202-153-34-42.ngrok-f... (all events)`

Last delivery was successful.

Edit Delete

- Goto jenkins dashboard create a job for Freestyle

The screenshot shows the Jenkins web interface for the 'webhook' project. At the top, there's a navigation bar with the Jenkins logo, a search bar containing 'Search (CTRL+K)', and user account and settings icons. Below the header, the breadcrumb navigation shows 'Dashboard > webhook >'. The main content area has a left sidebar with various project management options: 'Status' (highlighted in grey), 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', 'GitHub Hook Log', and 'Rename'. To the right of the sidebar, the project name 'webhook' is displayed in large bold letters, followed by a 'Permalinks' button and an 'Add description' button. A 'Builds' section below shows a message 'No builds'.

- Change the configure

Dashboard > webhook > Configuration

Source Code Management

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

None

Git

Repositories

Repository URL: git@github.com:VaralaxmiGanta/j_repo1.git

Credentials: VaralaxmiGanta (GitHub SSH Key)

+ Add

Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any'): */main

Add Branch

Repository browser: (Auto)

Additional Behaviours

Save

Addv

This screenshot shows the Jenkins configuration interface for a webhook. The 'Source Code Management' tab is selected. Under the 'Git' section, a repository is configured with the URL 'git@github.com:VaralaxmiGanta/j_repo1.git' and a credential named 'VaralaxmiGanta (GitHub SSH Key)'. The 'Branches to build' field contains the pattern '*/main'. The 'Repository browser' is set to '(Auto)'. At the bottom, there are 'Save' and 'Addv' buttons.

- Select build triggers is **GitHub hook trigger for GITScm polling**

Dashboard > webhook > Configuration

Additional Behaviours

Add v

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 ?
- Generic Webhook Trigger ?
- GitHub Branches
- GitHub Pull Requests ?
- GitHub hook trigger for GITScm polling ?

This screenshot shows the Jenkins configuration interface for build triggers. The 'Build Triggers' section is expanded, showing several options like 'Trigger builds remotely', 'Build after other projects are built', and 'GitHub hook trigger for GITScm polling'. The 'GitHub hook trigger for GITScm polling' option is checked. Other options like 'Trigger builds remotely' and 'Build after other projects are built' are also listed.

- Change the script in the git repo

The screenshot shows the Jenkins web interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user account information. Below the bar, the URL 'Dashboard > webhook >' is visible. The main content area has a title 'webhook' and a 'Permalinks' section. On the left, a sidebar contains links for Status, Changes, Workspace, Build Now, Configure, Delete Project, GitHub Hook Log, and Rename. A prominent 'Builds' card displays a single entry from 'Today': '#1 15:21'.

- Jenkins will trigger the changes

This screenshot shows the 'Polling Log' page for build #1. The top navigation bar includes the Jenkins logo, search, and user info, with the URL 'Dashboard > webhook > #1 > Polling Log'. The sidebar on the left includes links for Status, Changes, Console Output, Edit Build Information, Delete build '#1', Polling Log (which is selected and highlighted), View as plain text, Timings, and Git Build Data. The main content area is titled 'Polling Log' and contains a message: 'This page captures the polling log that triggered this build.' Below this, a code block shows the log output:

```

Started on 31-Jan-2025, 3:21:43 pm
Started by event from 140.82.115.250 → https://2624-202-
153-34-42.ngrok-free.app:8080/github-webhook/ on Fri Jan 31
15:21:43 IST 2025
No existing build. Scheduling a new one.
Done. Took 1 ms
Changes found

```

4.1.3. Webhook Trigger for Multibranch Pipeline

Note :

- Check the agent endpoints if it online or not

```
ngrok                                     (Ctrl+C to quit)

👋 Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status                online
Account                      varalaxmiganta066@gmail.com (Plan: Free)
Version                       3.19.1
Region                        India (in)
Latency                       22ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://433d-202-153-34-42.ngrok-free.app -> http:

Connections                   ttl     opn      rt1      rt5      p50      p90
                             23       0       0.00     0.00    30.20    30.27

HTTP Requests
-----
11:50:52.517 IST POST /github-webhook/          200 OK
11:50:50.331 IST POST /github-webhook/          200 OK
11:50:48.568 IST POST /github-webhook/          200 OK
11:50:44.794 IST POST /github-webhook/          200 OK
11:50:44.822 IST POST /github-webhook/          200 OK
```

- Give appropriate permissions to the files in your repository for Jenkins to access

```
vlab@HYVLAB7:~/vara/j_repo1$ cat Jenkinsfile
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                echo 'checking out code form main branch...'
                echo 'hello VARA'
                checkout scm
            }
        }
        stage('Build') {
            steps {
                echo 'Building project from main branch...'
                sh 'chmod +x ./build.sh'
                sh './build.sh'
            }
        }
        stage('Test') {
            steps {
                echo 'Running tests...'
                sh 'chmod +x ./run-tests.sh'
                sh './run-tests.sh'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying application...'
                sh 'chmod +x ./deploy.sh'
                sh './deploy.sh'
            }
        }
    }
}
```

```
vlab@HYVLAB7:/var/lib/jenkins/workspace$ ls
git      multi_branch_webhook_dev      pipeline_v_main@tmp
git2     multi_branch_webhook_dev@tmp  veena
git2@tmp multi_branch_webhook_main   veena@tmp
git3     multi_branch_webhook_main@tmp webhook
git3@tmp multi_branch_webhook_PR-1   webhook@tmp
git@tmp  multi_branch_webhook_PR-1@tmp workspaces.txt
job2    pipeline_v_main
```

4.1.3.1. Plugin (Multibranch Scan Webhook Trigger)

- Trigger that can receive any HTTP request and trigger a multibranch job scan when the token matched.

4.1.3.2. Each branch of repository can have its own Jenkinsfile

Note :

- List all the branches for both remote and local

```
vlab@HYVLAB7:~/vara/j_repo1$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/dev
  remotes/origin/feature-1
  remotes/origin/main
```

```
vlab@HYVLAB7:~/vara/j_repo1$ git ls-tree -r remotes/origin/dev --name-only
Jenkinsfile
README.md
vlab@HYVLAB7:~/vara/j_repo1$ git ls-tree -r remotes/origin/feature-1 --name-only
README.md
vlab@HYVLAB7:~/vara/j_repo1$ git ls-tree -r main --name-only
Jenkinsfile
README.md
build.sh
deploy.sh
run-tests.sh
```

- In the above dev and main branches are having Jenkinsfile so if getting any new commit from dev or main branches then webhook triggers but not for feature-1 because there is no jenkinsfile in that branch.
- Create a Multibranch Pipeline Job:
 - Open Jenkins, and click on **New Item**.
 - Choose **Multibranch Pipeline**, and give it a name.
 - Click **OK** to create the job.

 Jenkins

Search (CTRL+K)   

Dashboard > All > New Item

New Item

Enter an item name

Select an item type

-  **Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
-  **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
-  **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
-  **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

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

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

OK

- Configure the Multibranch Pipeline Job:
 - Under the **Branch Sources** section:
 - Select your source control management **GitHub**
 - Provide the repository URL.
 - Configure credentials if needed.



Jenkins

Search (CTRL+K)



Dashboard > multi_branch_webhook > Configuration

Configuration

General

Enabled

General

Branch Sources

Build Configuration

Scan Multibranch Pipeline Triggers

Orphaned Item Strategy

Appearance

Health metrics

Properties

Display Name [?](#)

Description

Plain text [Preview](#)

Branch Sources

Add source ^

Filter

Git

GitHub

GitHub source

Single repository & branch

Script Path [?](#)

Jenkinsfile

Scan Multibranch Pipeline Triggers

[Save](#)

[Apply](#)

Dashboard > multi_branch_webhook > Configuration

Enabled

General

Branch Sources

Build Configuration

Scan Multibranch Pipeline Triggers

Orphaned Item Strategy

Appearance

Health metrics

Properties

Display Name ?

Description

Plain text [Preview](#)

Branch Sources

GitHub Credentials ? X

varalaxmiganta/******** (git credentials)

+ Add

Repository HTTPS URL Repository HTTPS URL ?

https://github.com/VaralaxmiGanta/j_repo1.git

Credentials ok. Connected to https://github.com/VaralaxmiGanta/j_repo1. Validate

Repository Scan - Deprecated Visualization

Behaviours

Save Apply

This is from ngrok endpoint

Scan Repository Triggers

Periodically if not otherwise run ?

Scan by webhook ?

Trigger token ?

https://433d-202-153-34-42.ngrok-free.app/github-webhook/

Orphaned Item Strategy

Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed as soon as Jenkins determines their associated SCM head no longer exists. As an example, it may be useful to configure a different retention strategy to be able to examine build results of a branch after it has been removed.

Abort builds ?

Discard old items

Days to keep old items

if not empty, old items are only kept up to this number of days

Max # of old items to keep

if not empty, only up to this number of old items are kept

Appearance

Icon ?

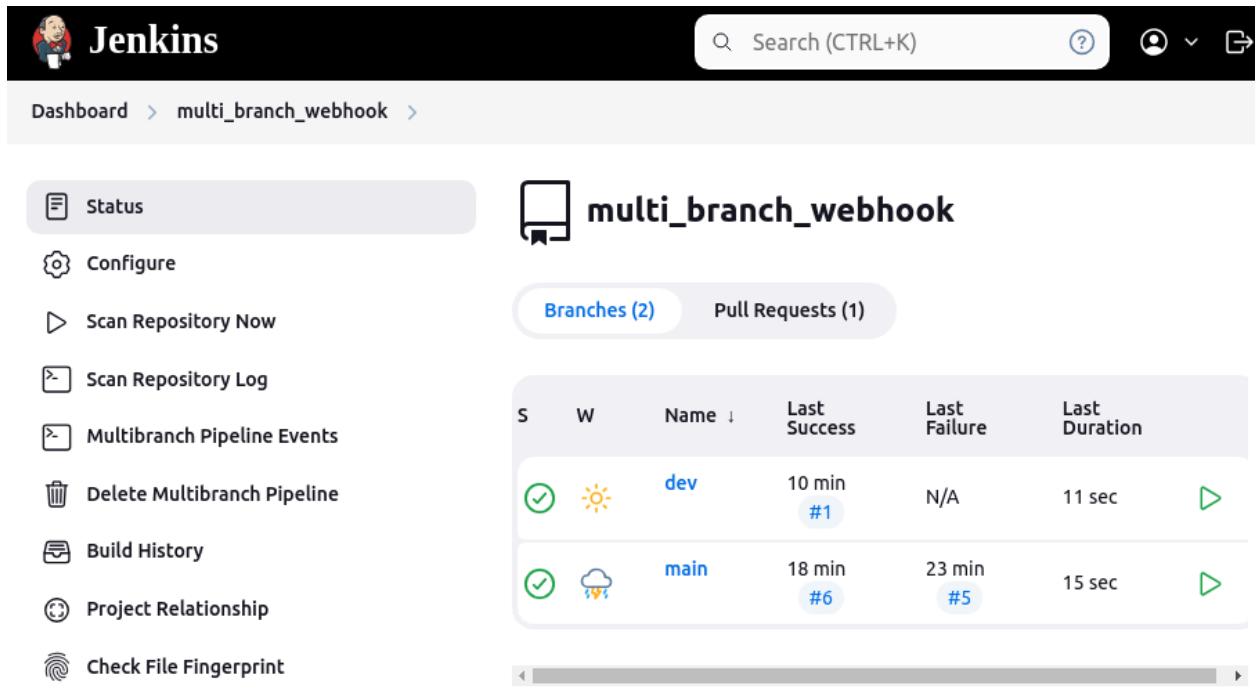
Save

Apply

Save the configurations

		multi branch _webhook	35 sec log	N/A	3.1 sec
--	--	-----------------------------	-------------------------------	-----	---------

- Jenkins should automatically detect the change via the webhook and trigger the appropriate branch's pipeline.



The screenshot shows the Jenkins multi_branch_webhook dashboard. On the left, there's a sidebar with options like Status, Configure, Scan Repository Now, Scan Repository Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, Build History, Project Relationship, and Check File Fingerprint. The main area has a title "multi_branch_webhook" with a bookmark icon. Below it, there are tabs for "Branches (2)" and "Pull Requests (1)". A table lists two branches: "dev" and "main". The "dev" branch has a green checkmark icon, a sun icon, and last ran 10 min ago (#1). The "main" branch also has a green checkmark icon, a cloud icon, and last ran 18 min ago (#6), failing 23 min ago (#5). There are arrows at the bottom of the table.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	dev	10 min #1	N/A	11 sec
✓	☁️	main	18 min #6	23 min #5	15 sec

4.1.3.2.1. Scan Repository Log

Started by user varalaxmi

[Mon Feb 03 12:26:35 IST 2025] Starting branch indexing...

12:26:36 Connecting to <https://api.github.com> using varalaxmiganta/***** (git credentials)

Examining [VaralaxmiGanta/j_repo1](#)

Checking branches...

Getting remote branches...

Checking branch [main](#)

'Jenkinsfile' found

Met criteria

No changes detected: main (still at 34a17a27be612cd84fa8d92ad68a32bc3304971e)

Checking branch [dev](#)

'Jenkinsfile' found

Met criteria

No changes detected: dev (still at b87dbbcbaef6ffbbd94f74a38c4c15d548c1b78e)

Checking branch [feature-1](#)

'Jenkinsfile' not found

Does not meet criteria

3 branches were processed

Checking pull-requests...

Getting remote pull requests...

Checking pull request #2

'Jenkinsfile' not found

Does not meet criteria

Checking pull request #1

'Jenkinsfile' found

Not mergeable, but will be built anyway

Met criteria

No changes detected: PR-1 (still at b87dbbcbaef6ffbbd94f74a38c4c15d548c1b78e)

2 pull requests were processed

Finished examining VaralaxmiGanta/j_repo1

[Mon Feb 03 12:26:40 IST 2025] Finished branch indexing. Indexing took 5 sec

Finished: SUCCESS