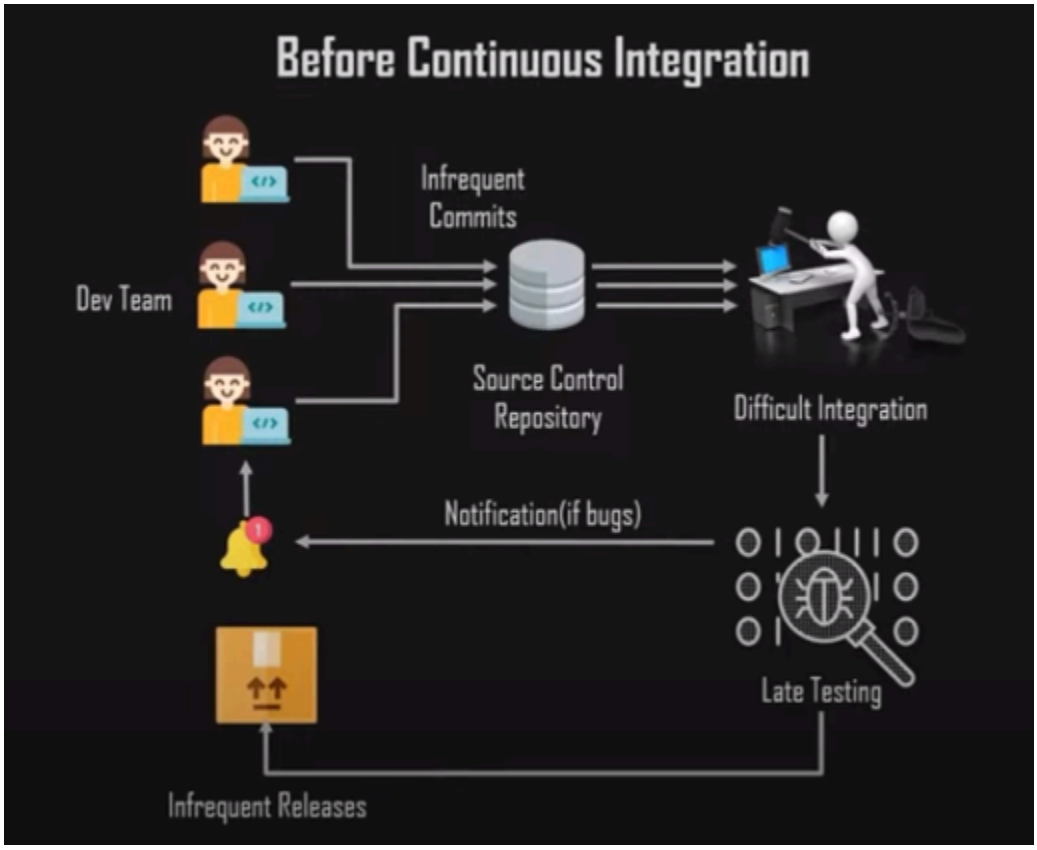
Jenkins

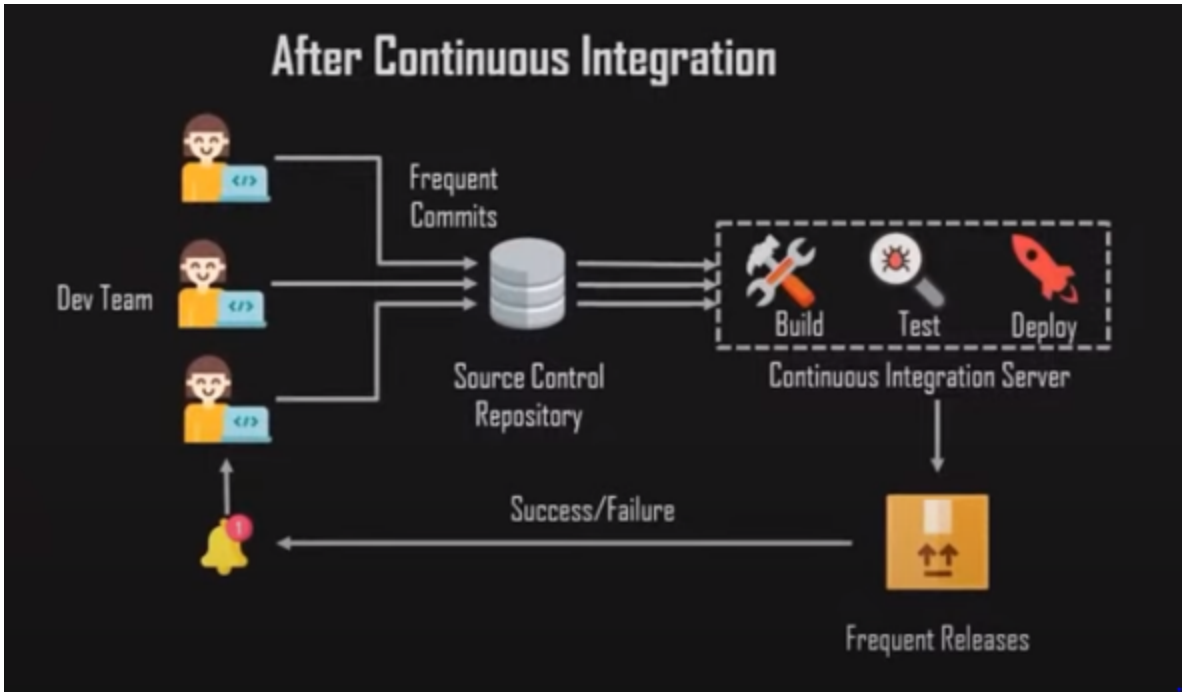
Index

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

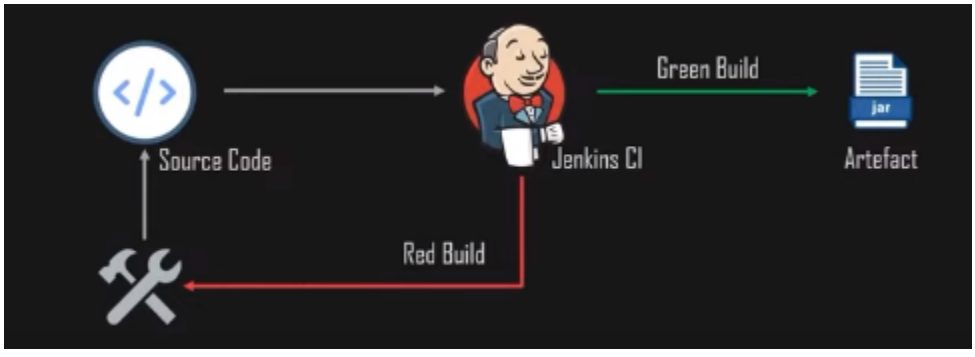




Continuous integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently.

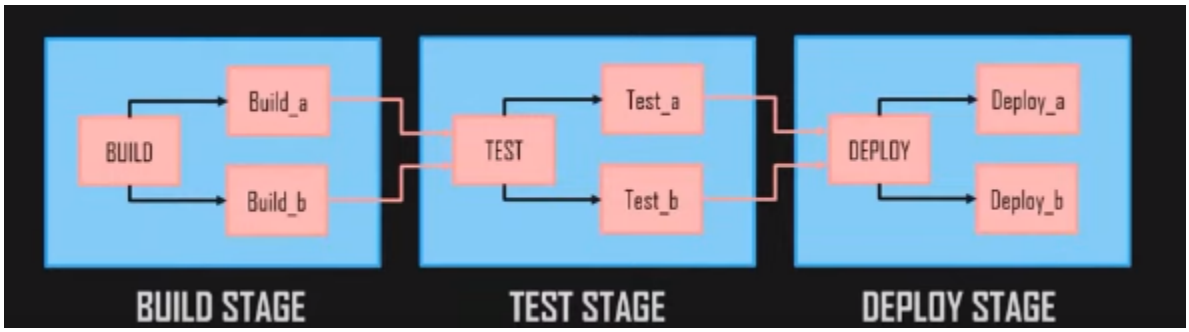
1.2. What is jenkins?

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

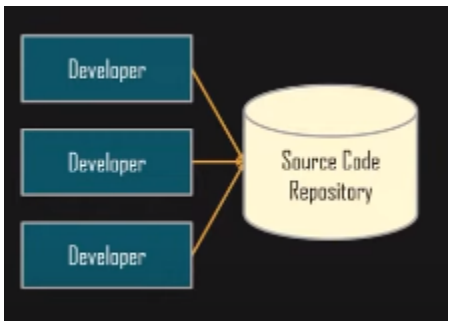
1.4. What is a jenkinsfile?

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

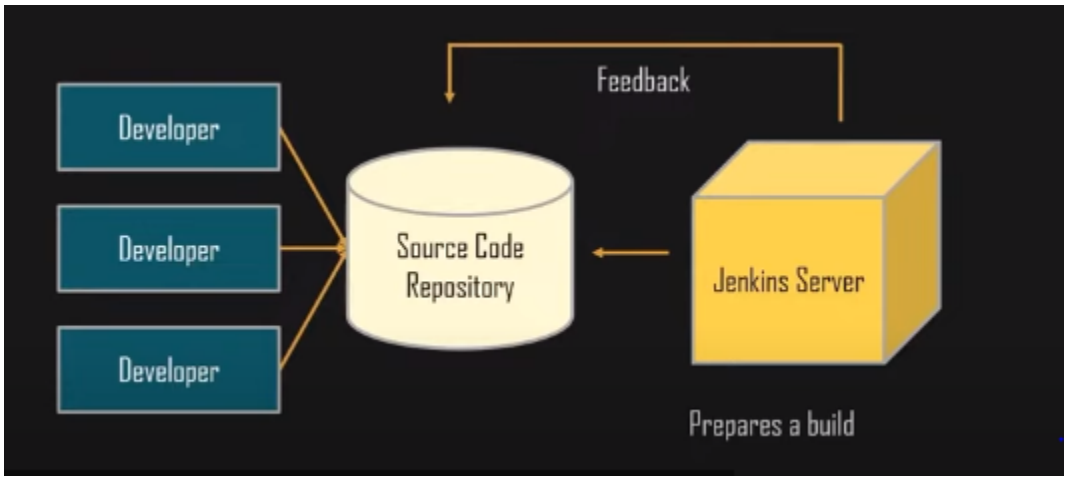
1.5. Jenkins workflow

- Developers commit changes to the source code:

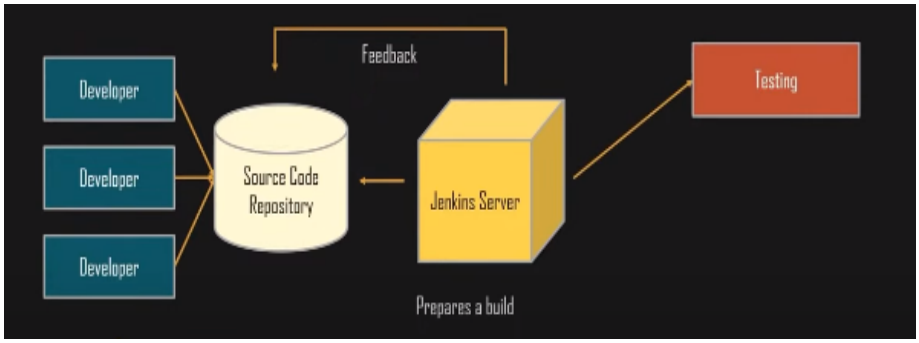
  Jenkins workflow always starts with the version control system or Source code repository.



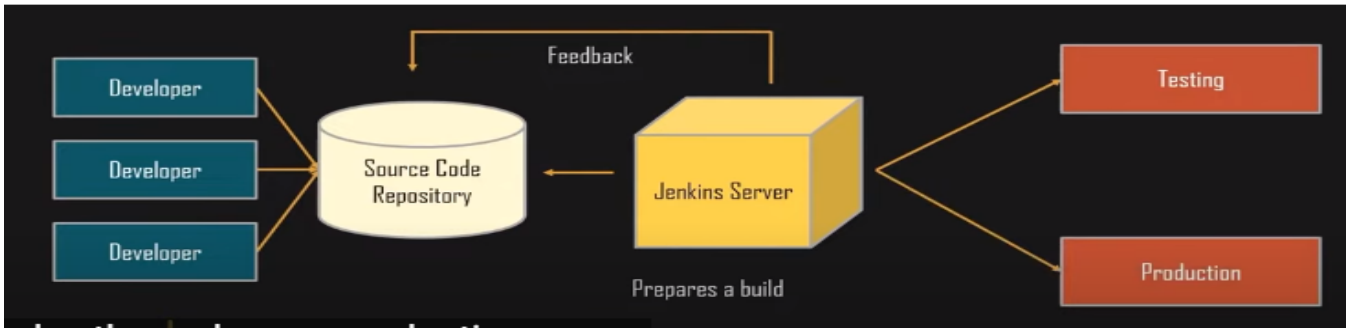
- CI server pulls that code, triggers a build



- 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/ | sudo tee \
  /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

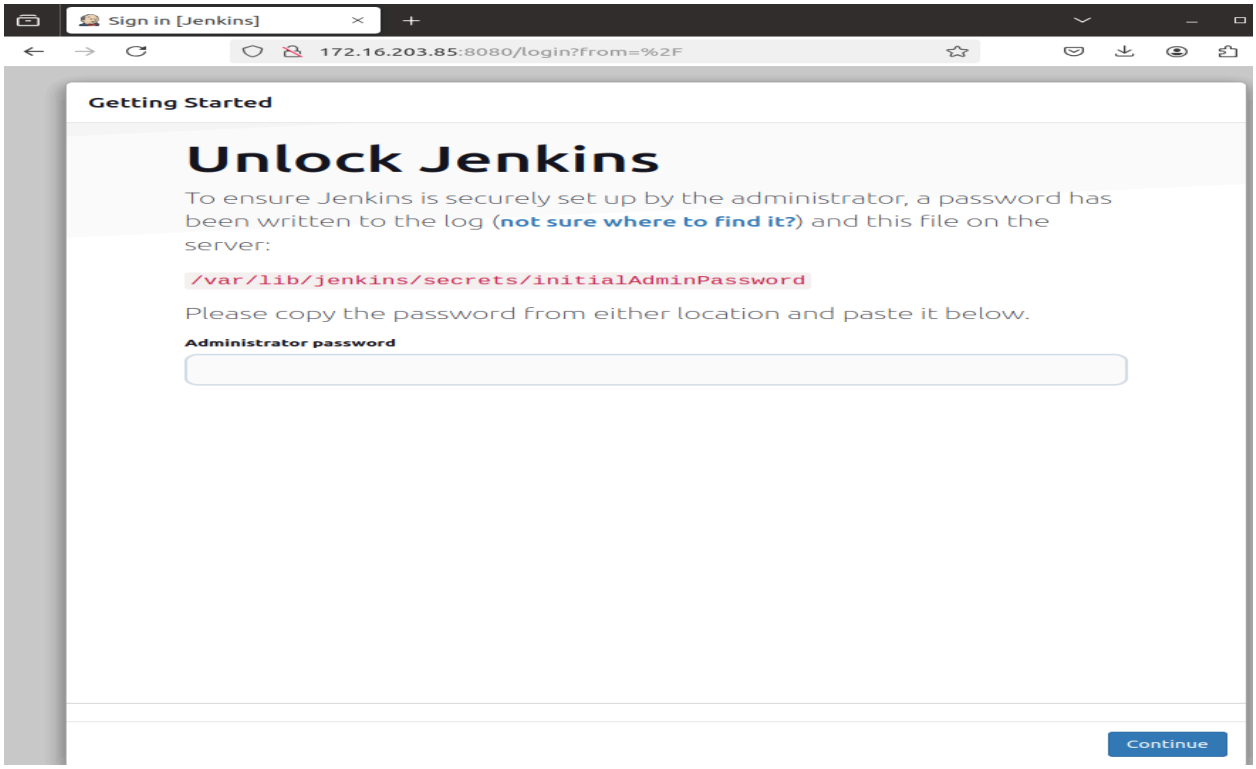
https://172.16.203.85:8080

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

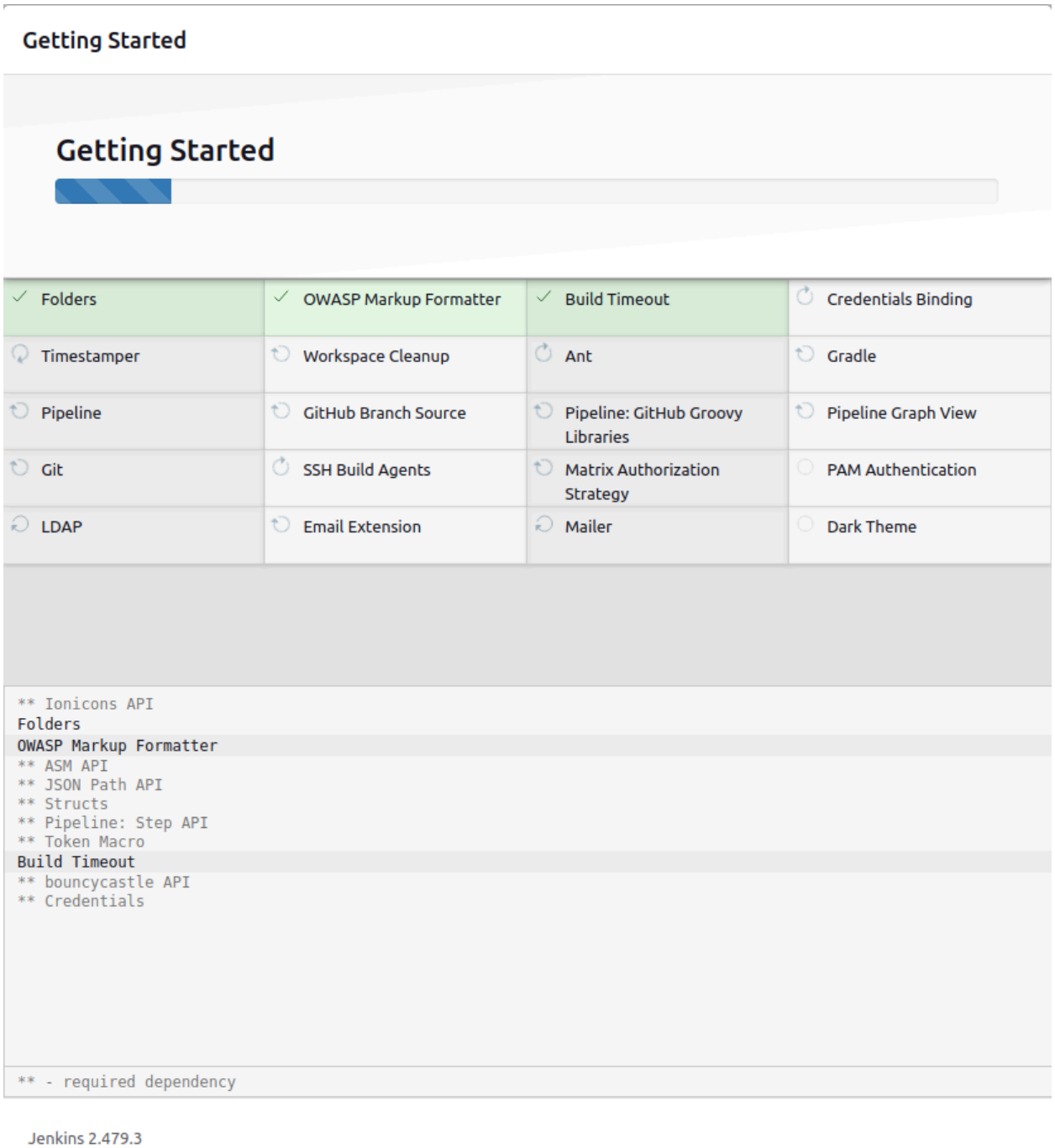
```
vlab@HYVLAB7:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2025-01-24 16:26:05 IST; 5s ago
   Main PID: 173232 (java)
      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 --webr>

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/initial>
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>
```

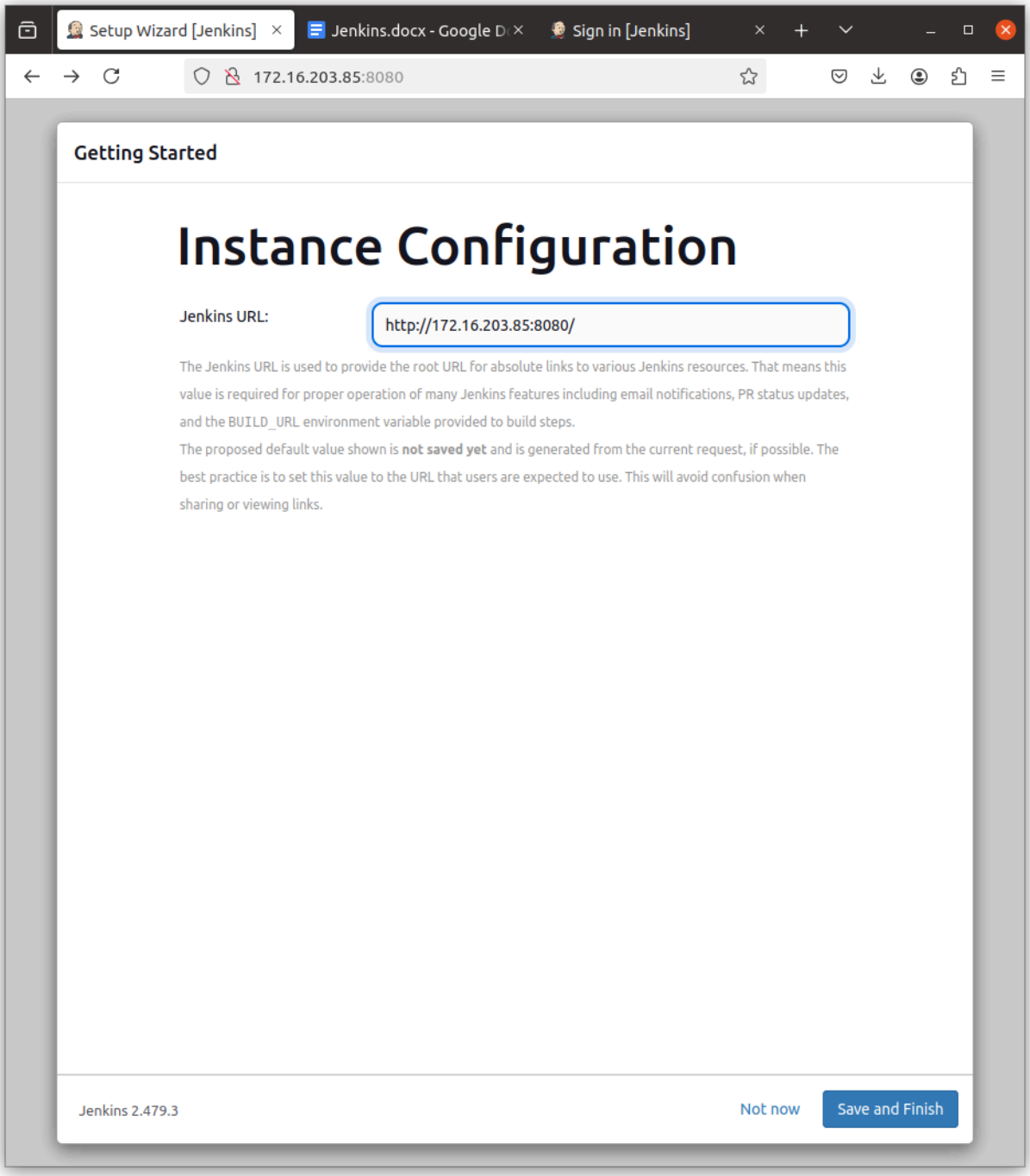


```
vlab@HYVLAB7:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
2d7092649c194c8fbfb5bd4d5140166a
```

2.2.2. Set user and password

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





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
```

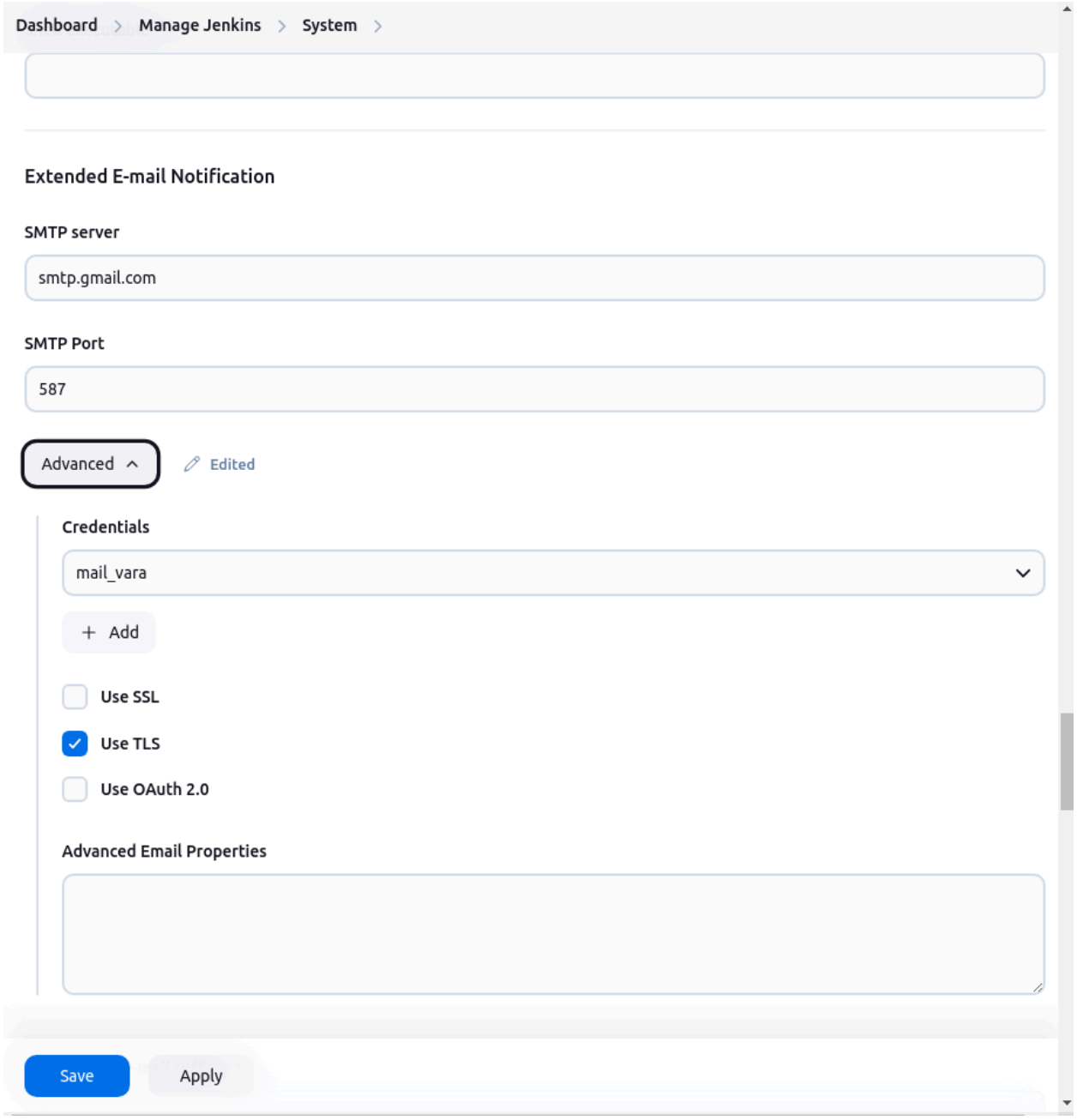
```
vlab@HYVLAB7:/var/lib/jenkins/users$ cd vara_16419438915906767594/
vlab@HYVLAB7:/var/lib/jenkins/users/vara_16419438915906767594$ ls
config.xml
vlab@HYVLAB7:/var/lib/jenkins/users/vara_16419438915906767594$ cat config.xml
<?xml version='1.1' encoding='UTF-8'?>
<user>
  <version>10</version>
  <id>vara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

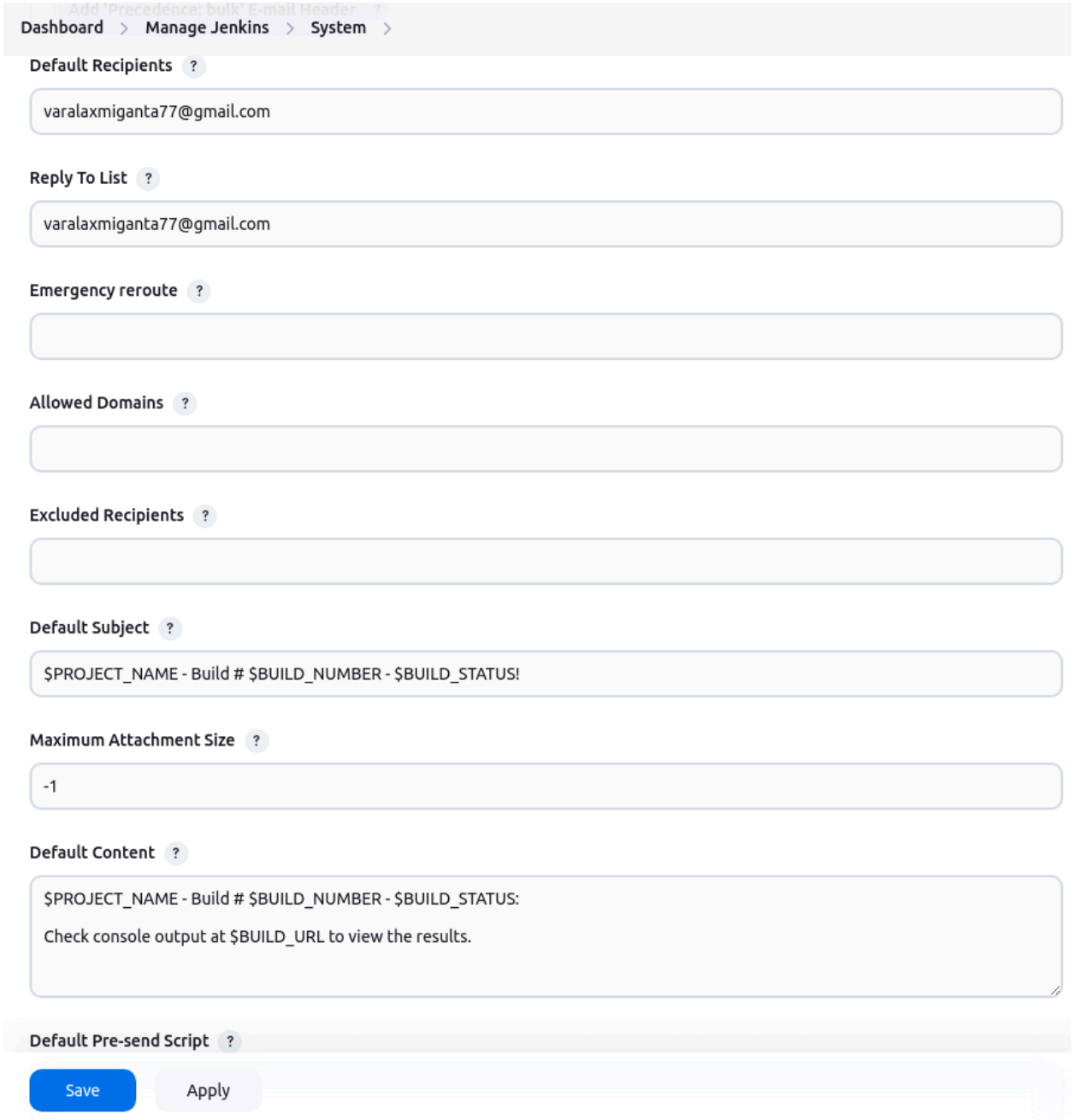
```
vlab@HYVLAB7:/var/lib/jenkins/users$ cat users.xml
<?xml version='1.1' encoding='UTF-8'?>
<hudson.model.UserIdMapper>
  <version>1</version>
  <idToDirectoryNameMap class="concurrent-hash-map">
    <entry>
      <string>vara</string>
      <string>vara_16419438915906767594</string>
    </entry>
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

2.3. Email Notifications

2.3.1. Extended Email Notification (Email Extension Plugin)

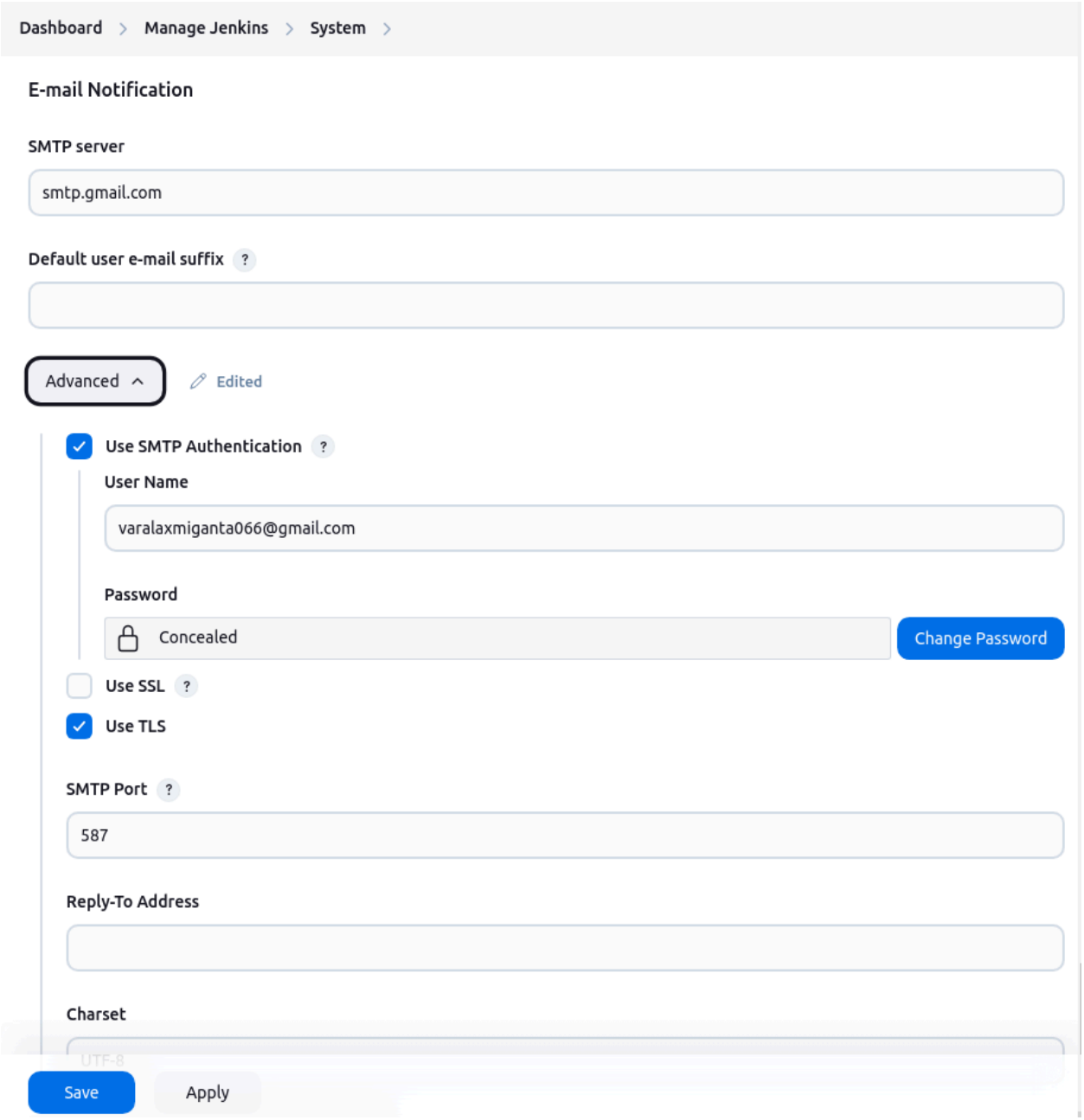
- This plugin allows administrators to create global templates for the Extended Email Publisher.
- Provides **advanced email customization**.
- Supports **multiple triggers** (Success, Failure, Unstable, Always, etc.).
- Allows **custom email templates** with Groovy scripts and HTML formatting.
- Can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


Dashboard
Manage Jenkins
System
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



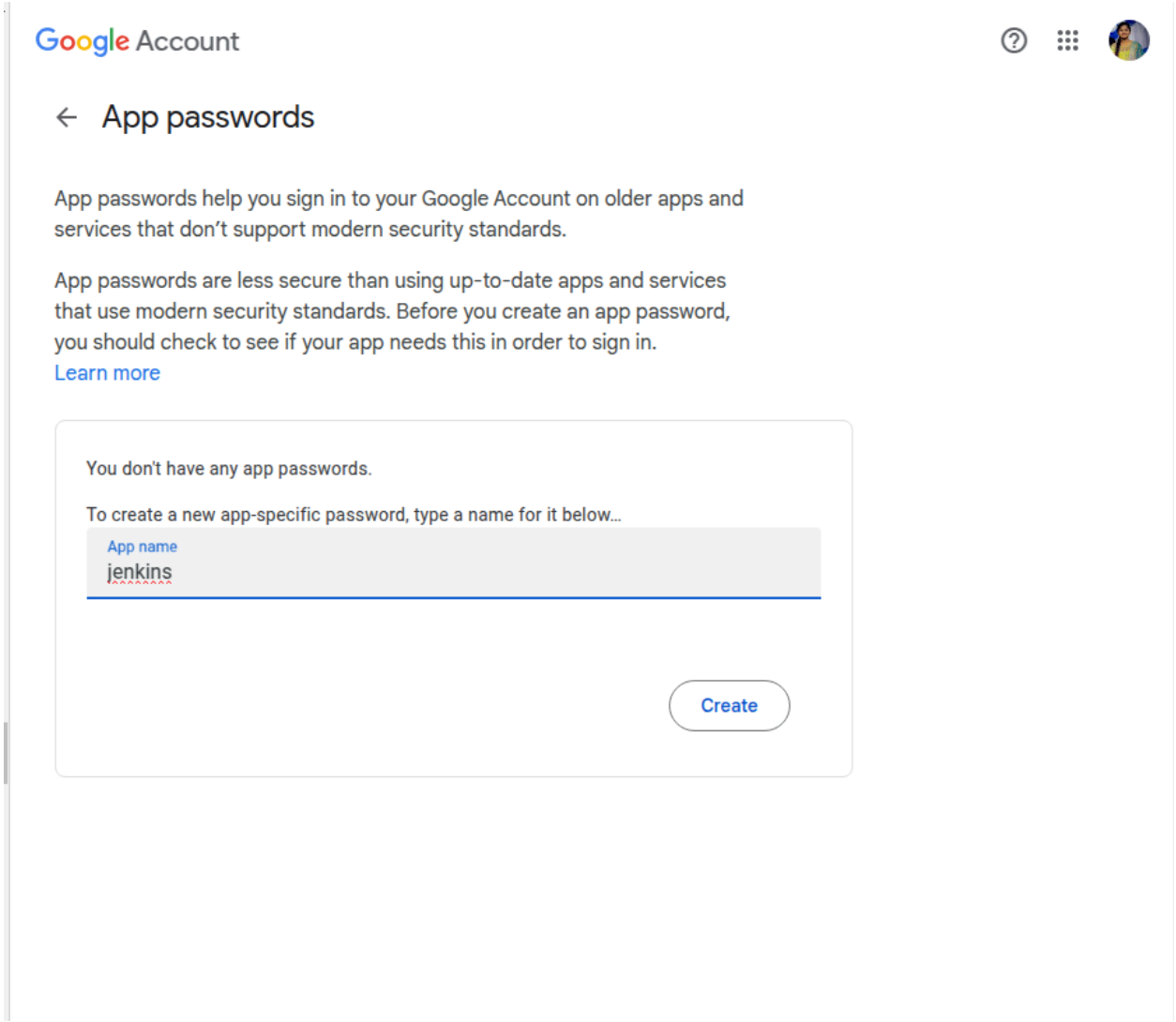
2.3.2. E-mail Notification (Mailer Plugin)

- **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.\

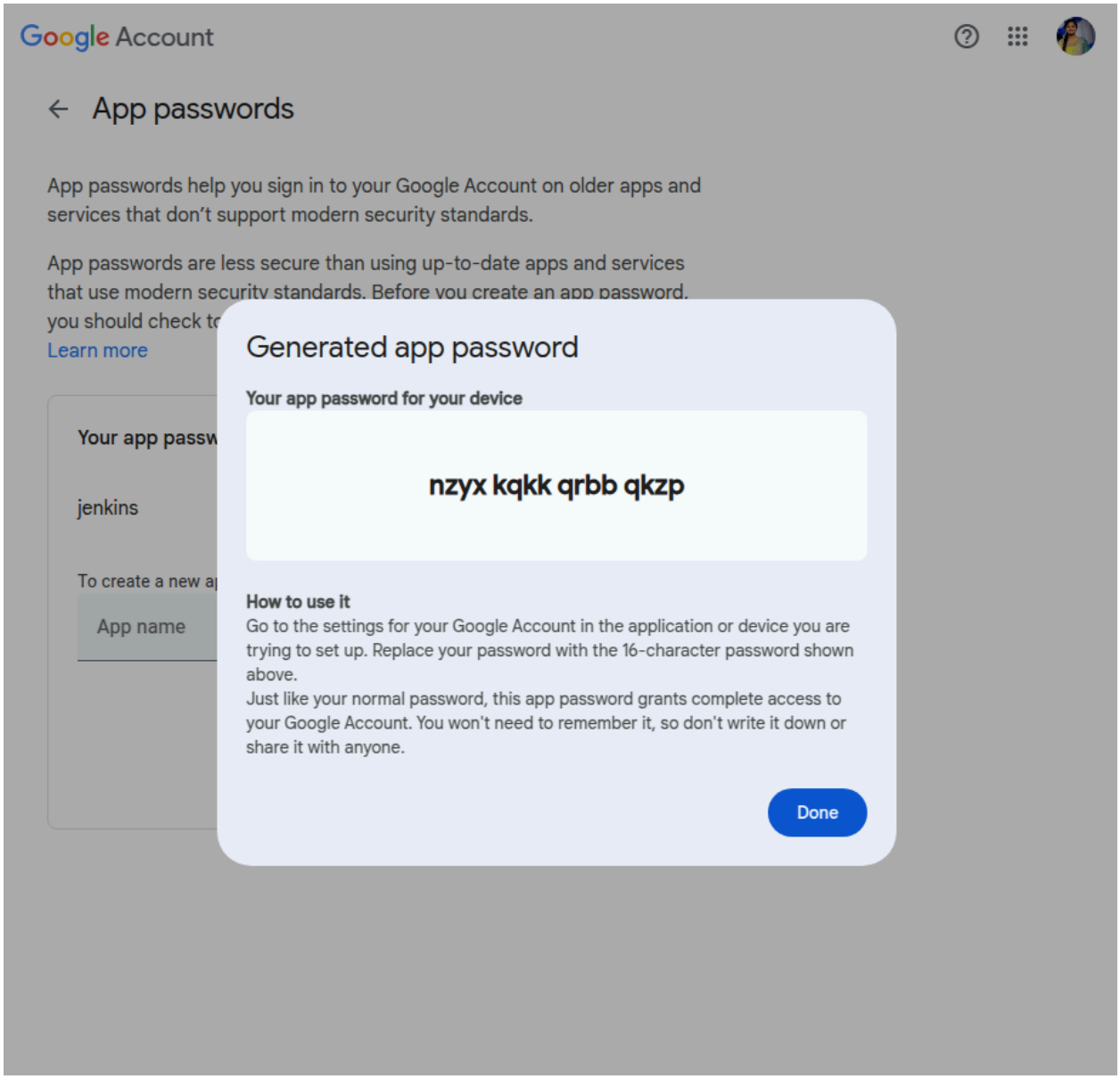


2.3.3. Setup

- 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

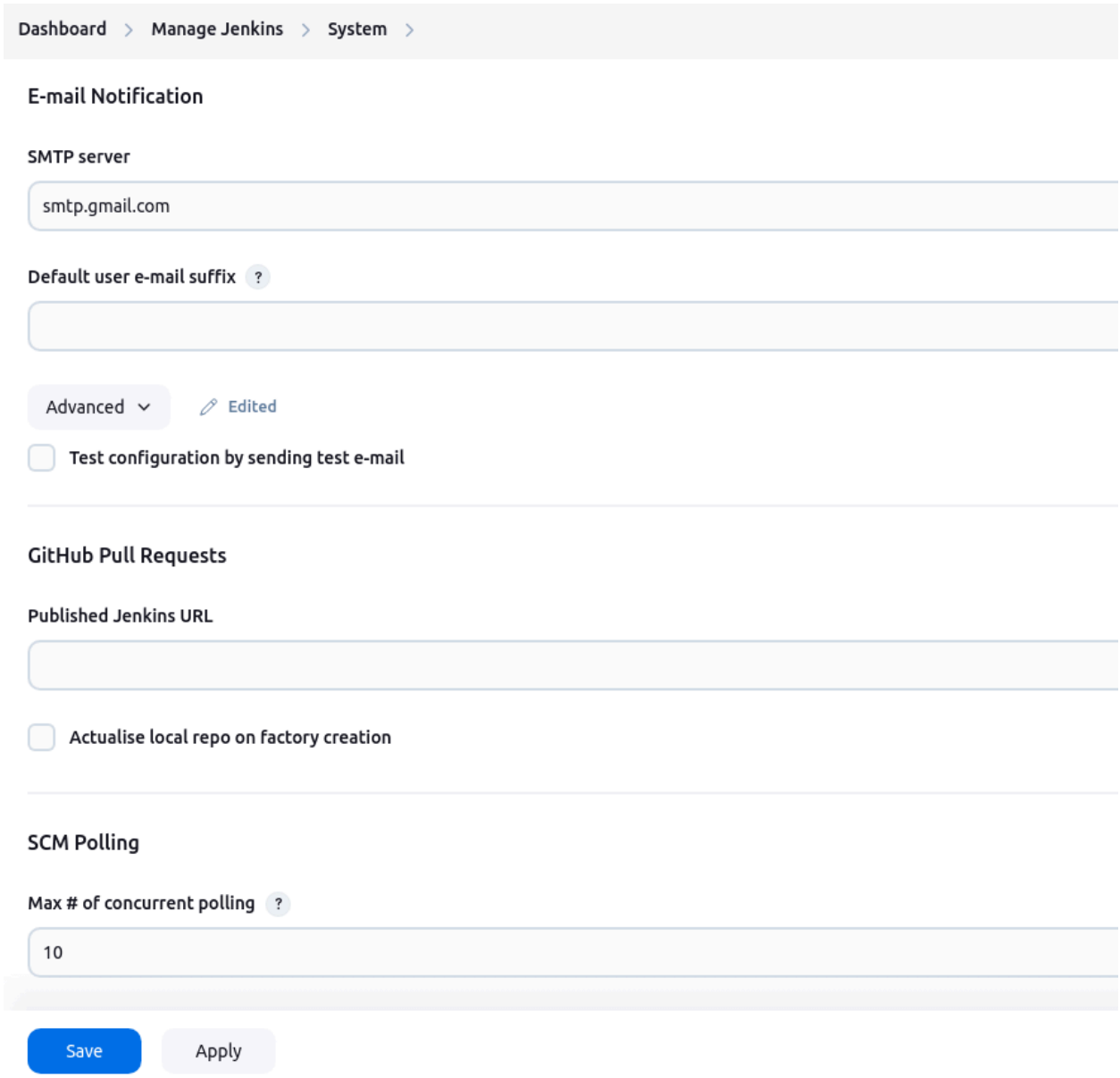


Password : nzyx kqkk qrbb qkzp

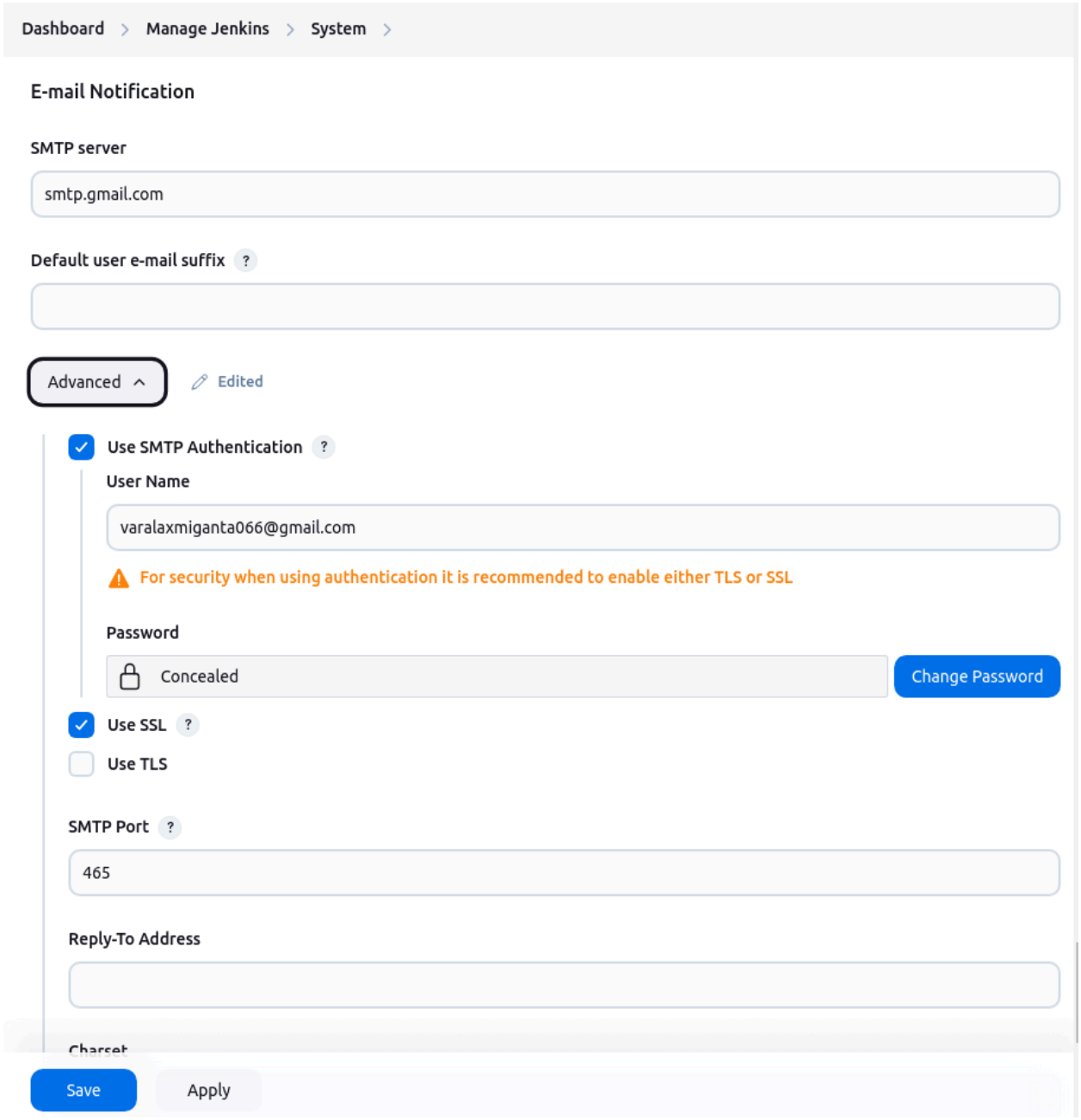


2.3.4. Configure the SMTP

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

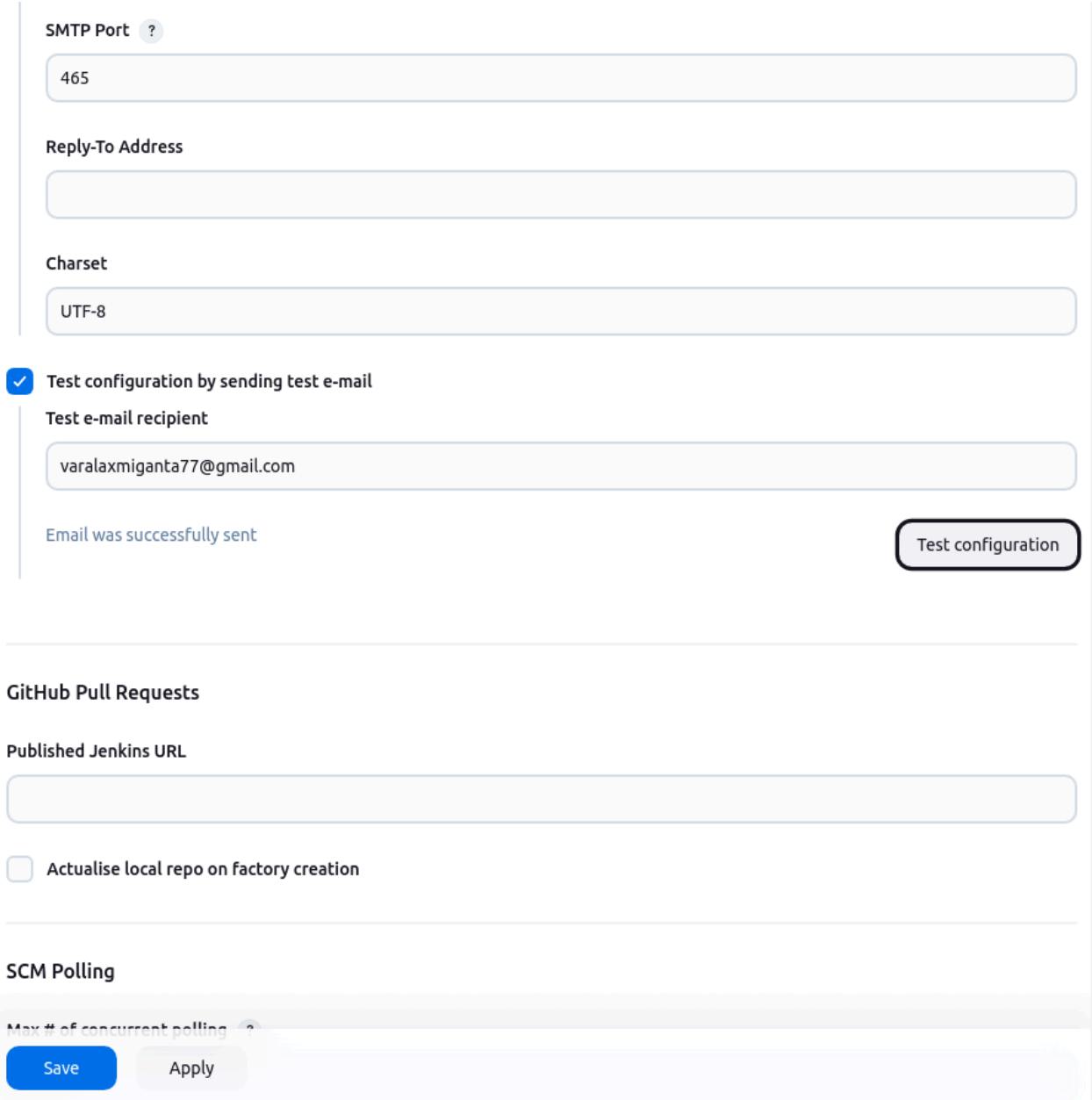


- Click on advanced
- Enter user name : email-address@gmail.com
- Enter app password
- Gmail uses port 465 for SSL or 587 for TLS.



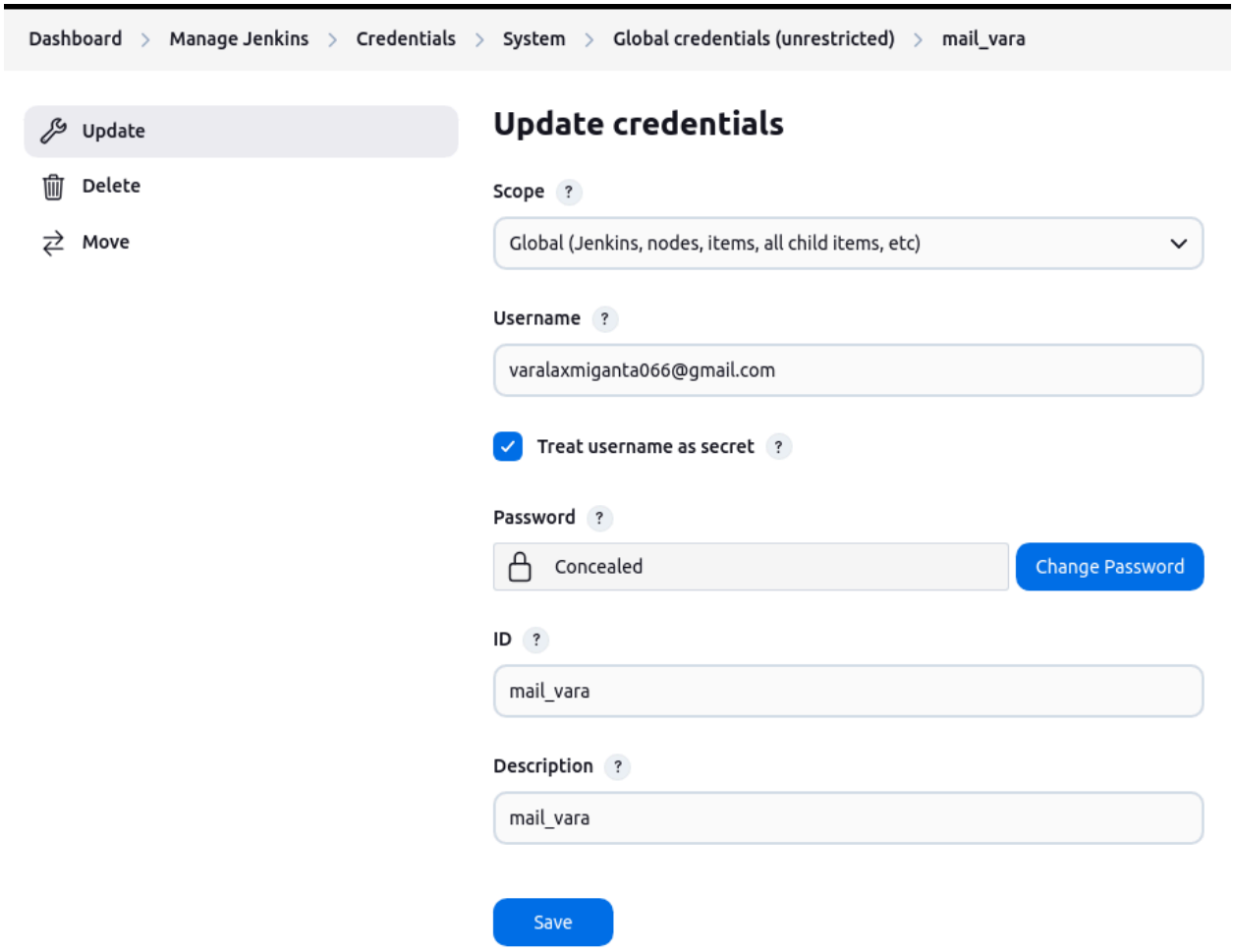
- Sender Email Address : This is the “from” address that will appear in the email notifications.
- Apply and Save.

This is the test configuration by sending test email:

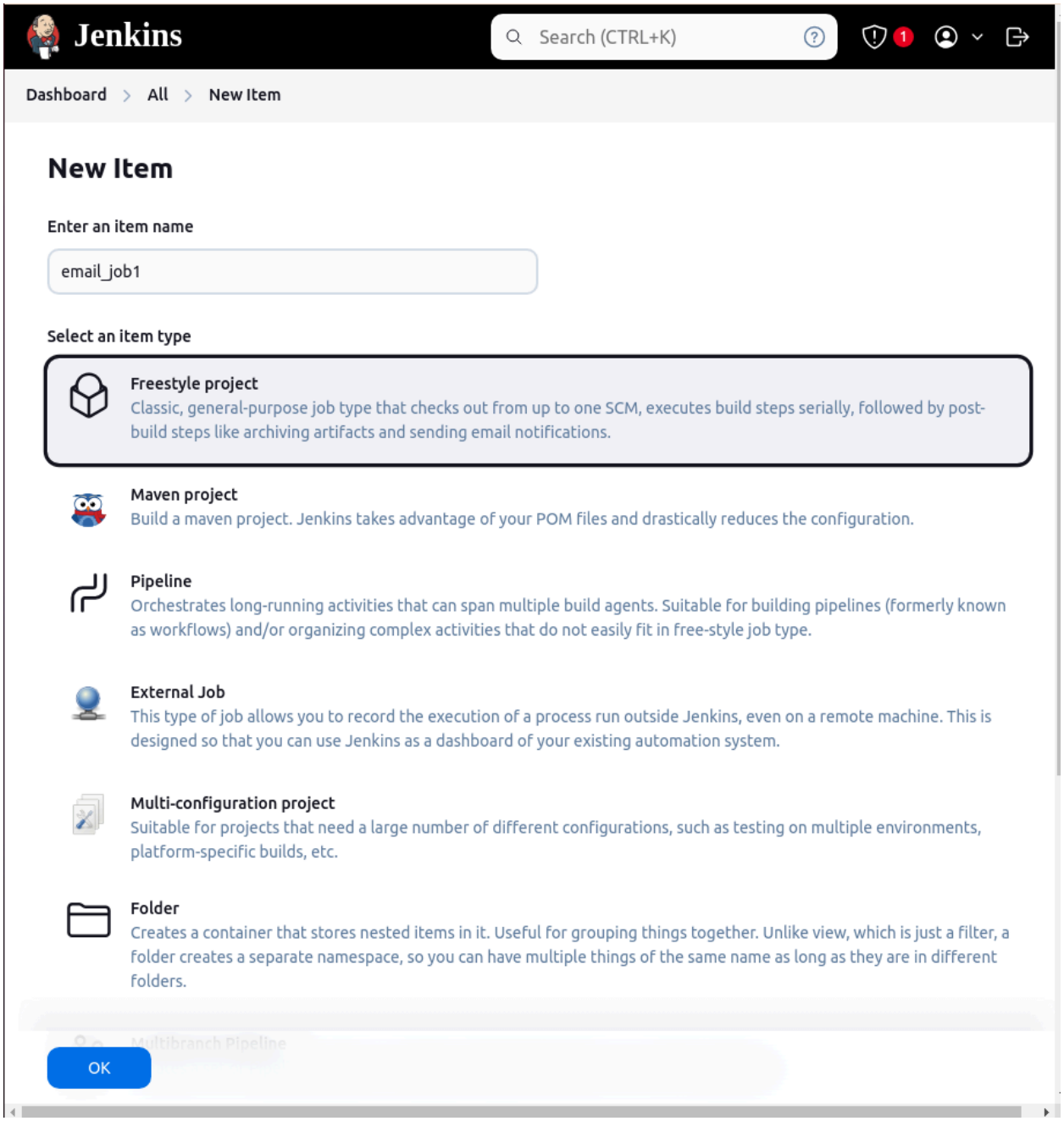


2.3.5. Configure Email Notifications for a Job

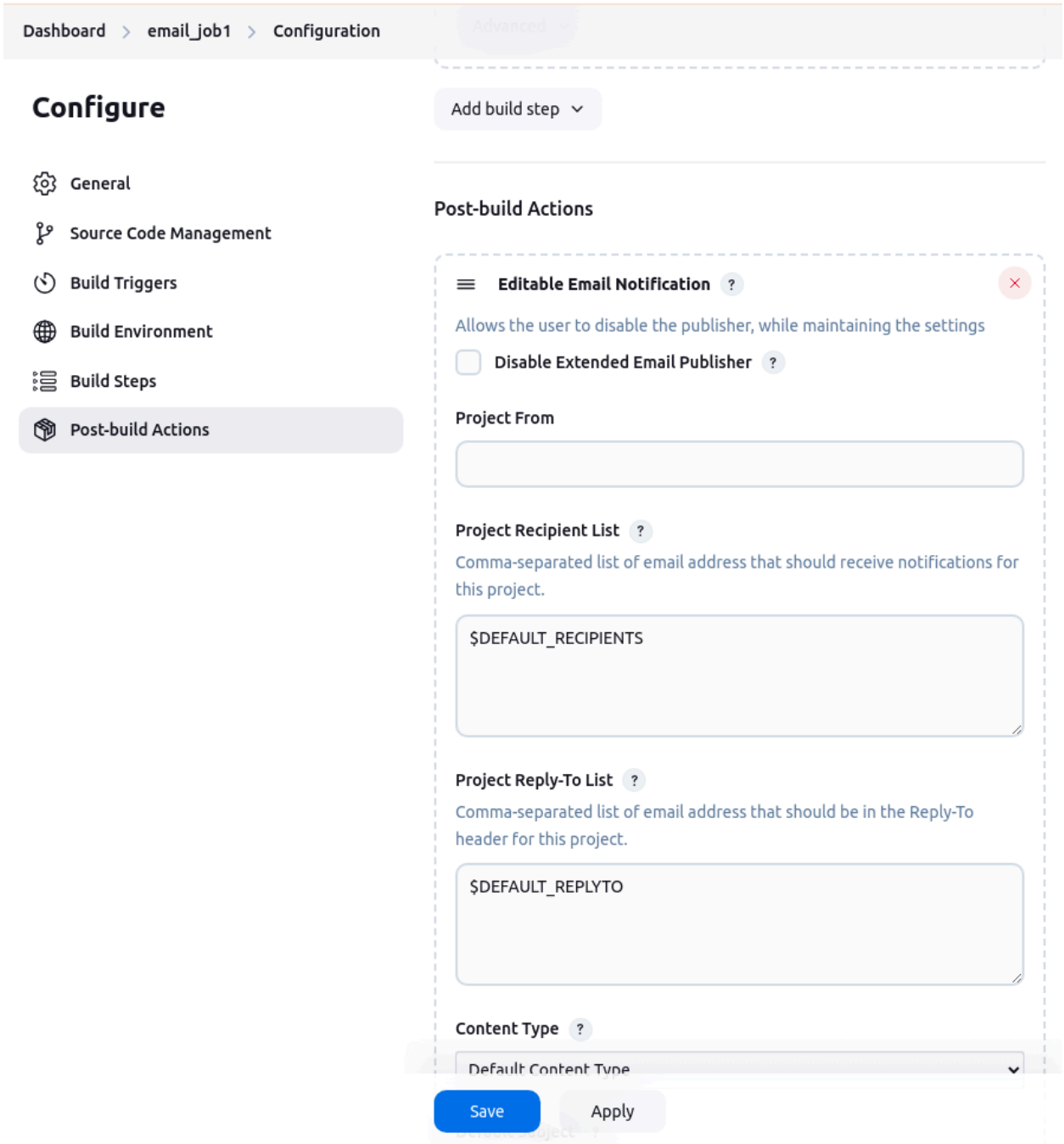
- Add the credentials to the manage jenkins



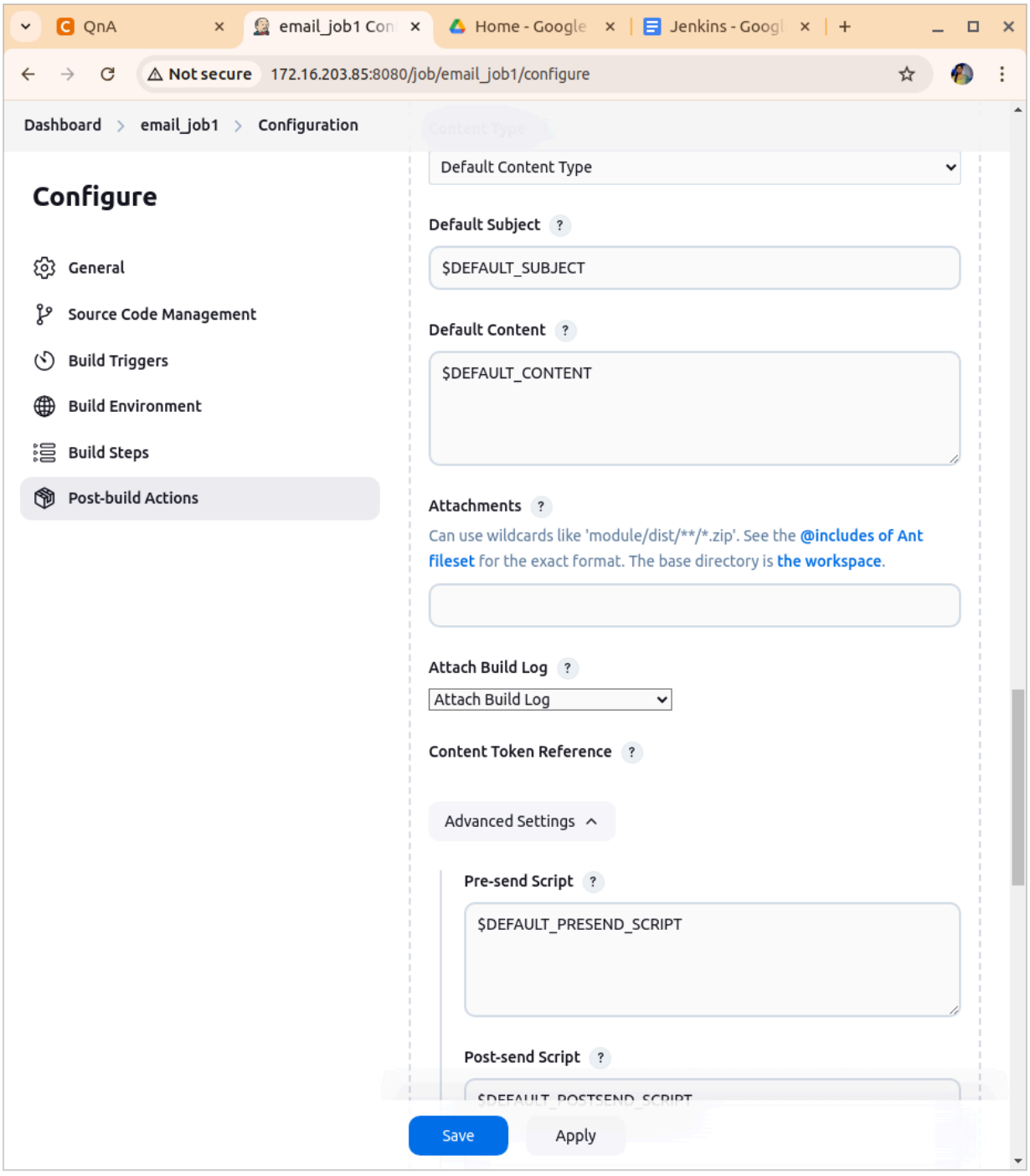
- Open your jenkins job
- Click on configure



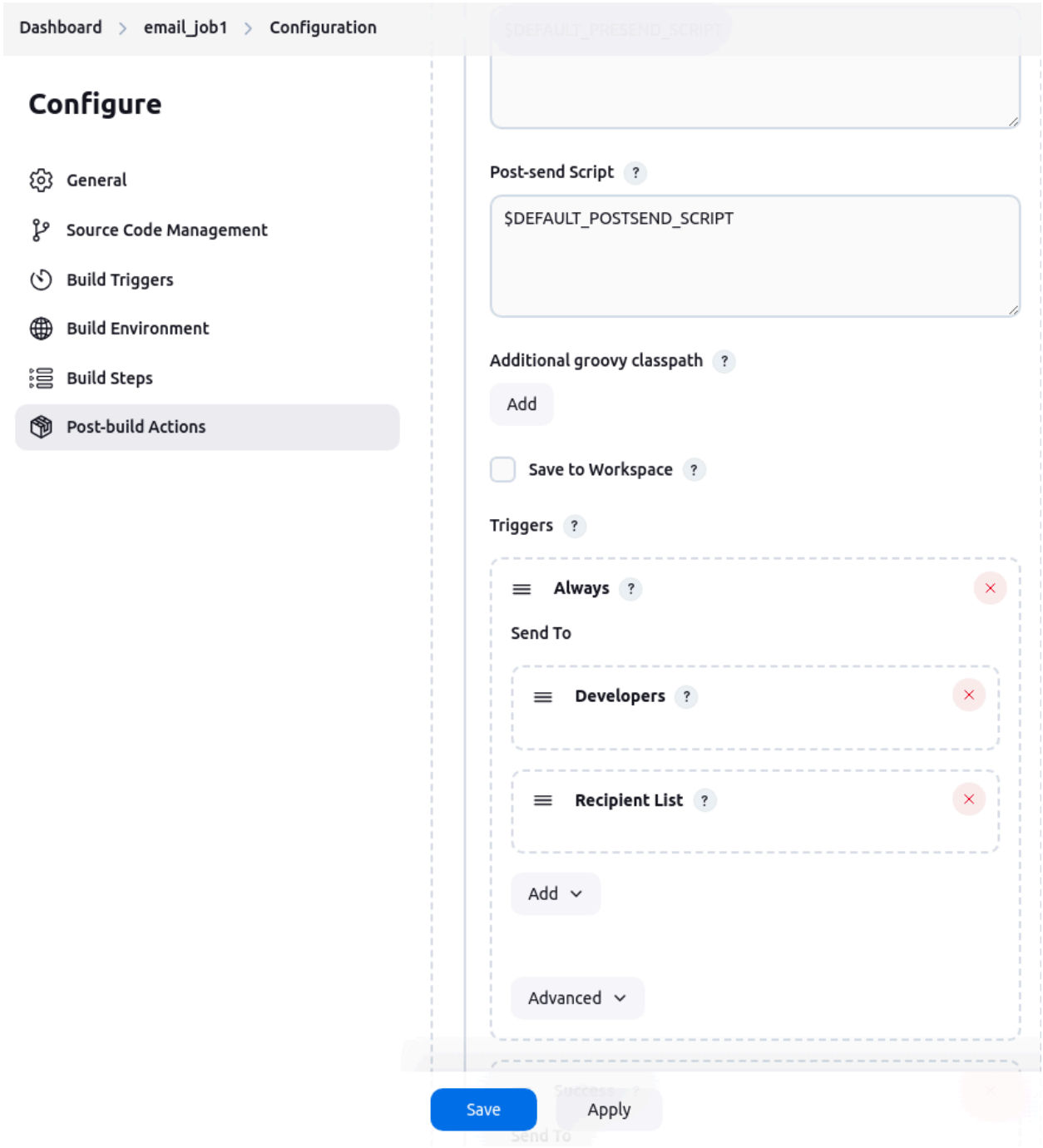
- Click **Add post-build action → Editable Email Notification**.

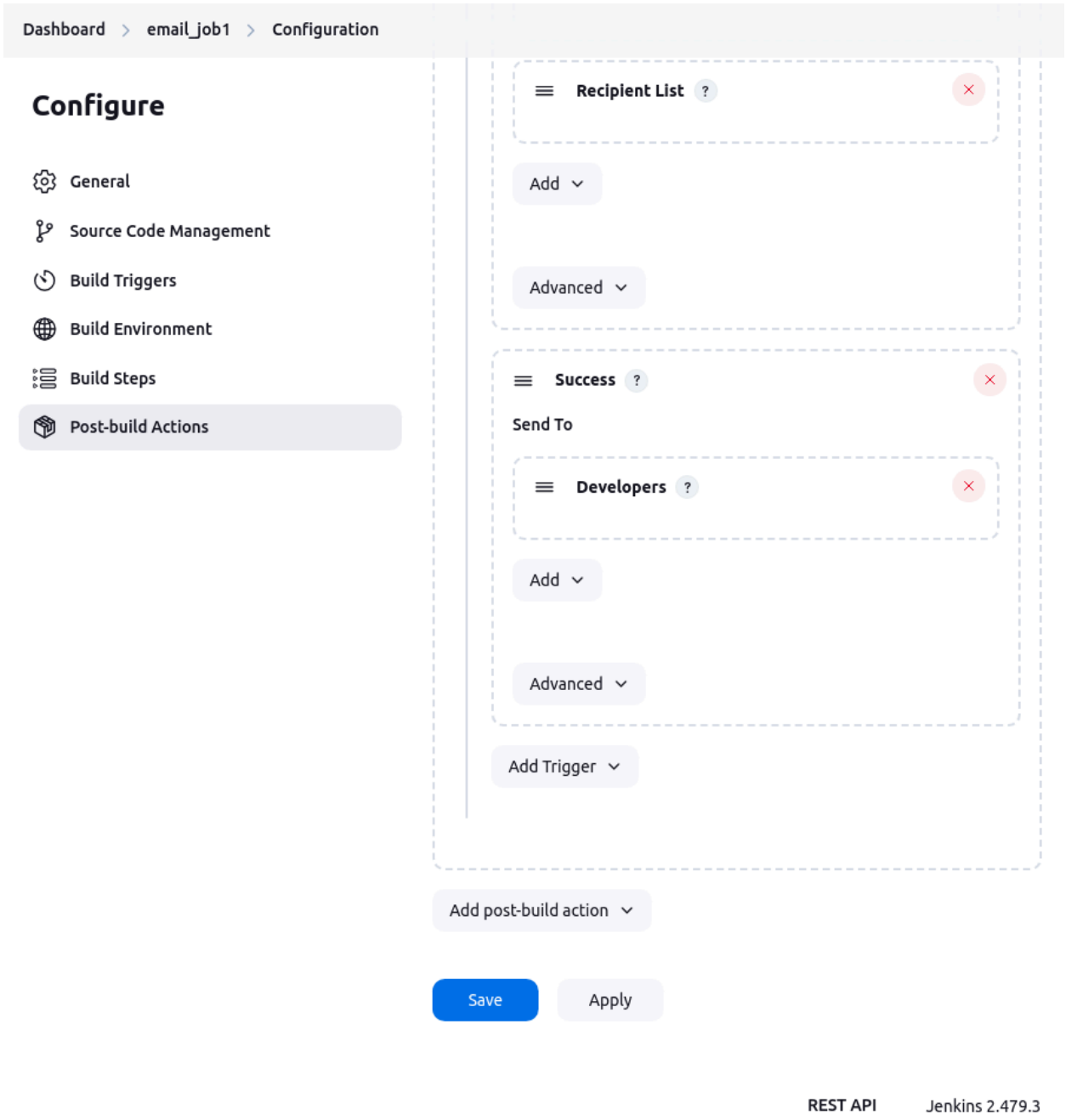


2.3.5.1. Attach Build log



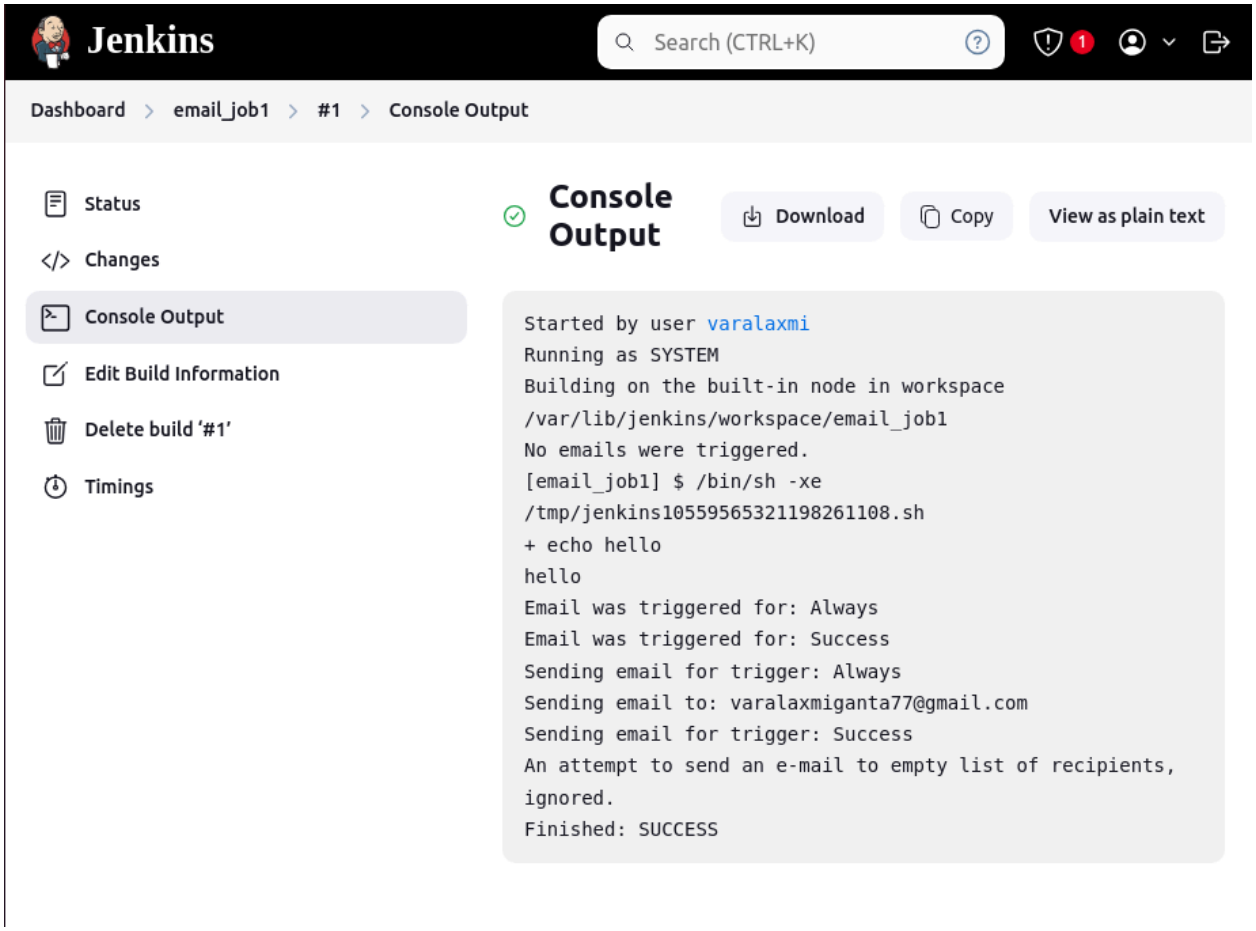
- Define **Triggers** (e.g., Success, Always, Unstable).
- Click save



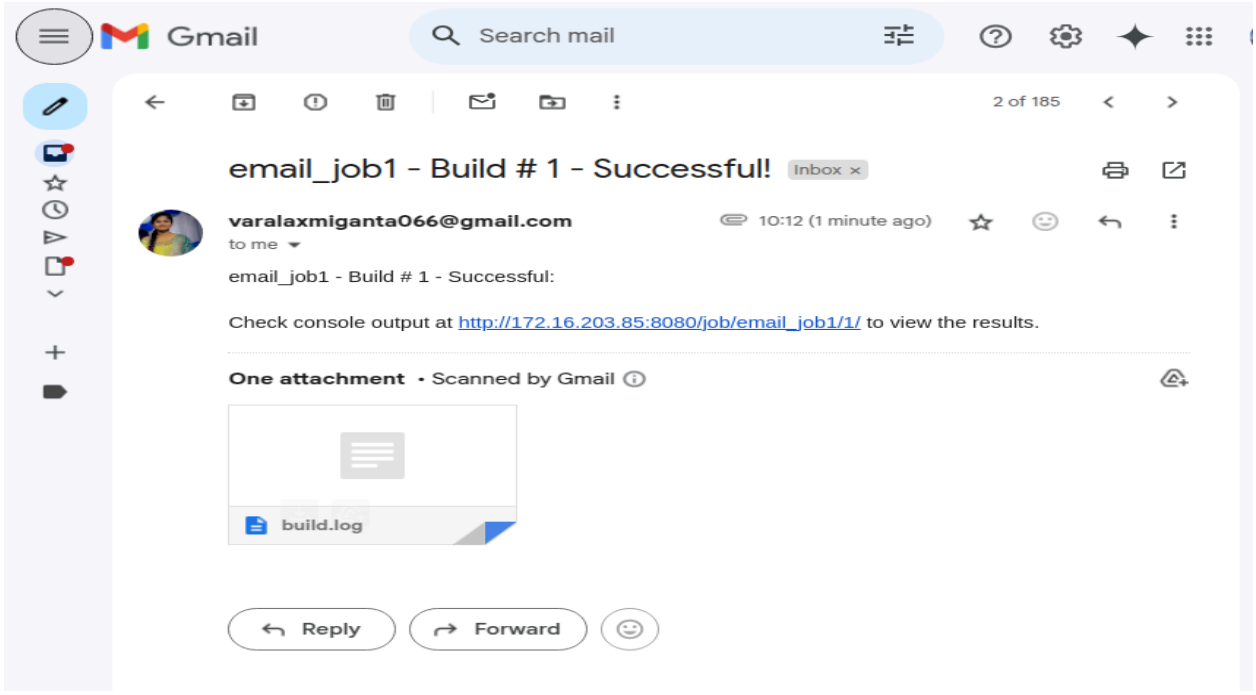


- Test Email Notifications
  - Run a build
  - Email will be sent on job completion

- Console Output

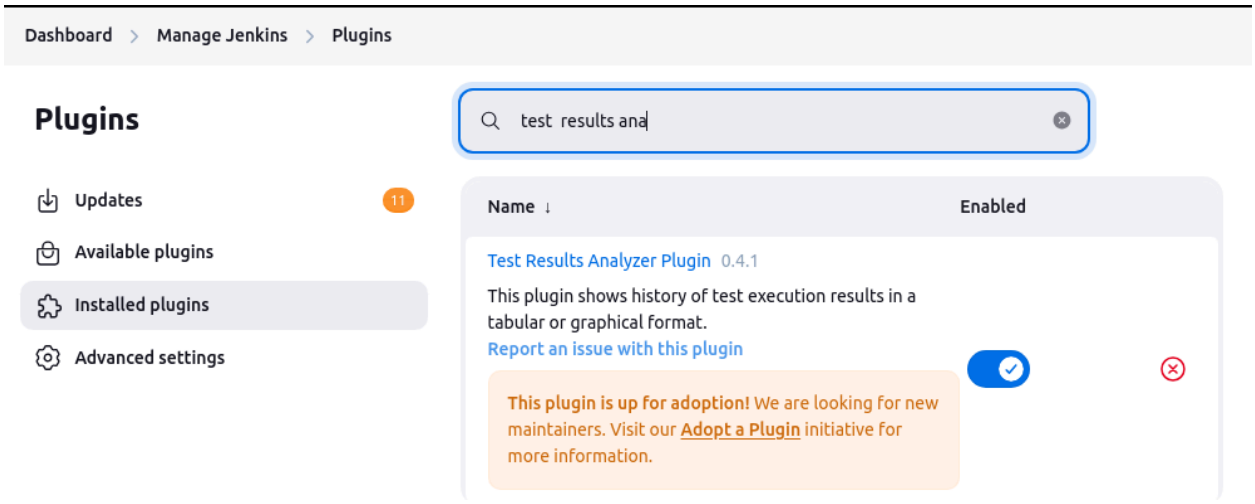


- Email Notification



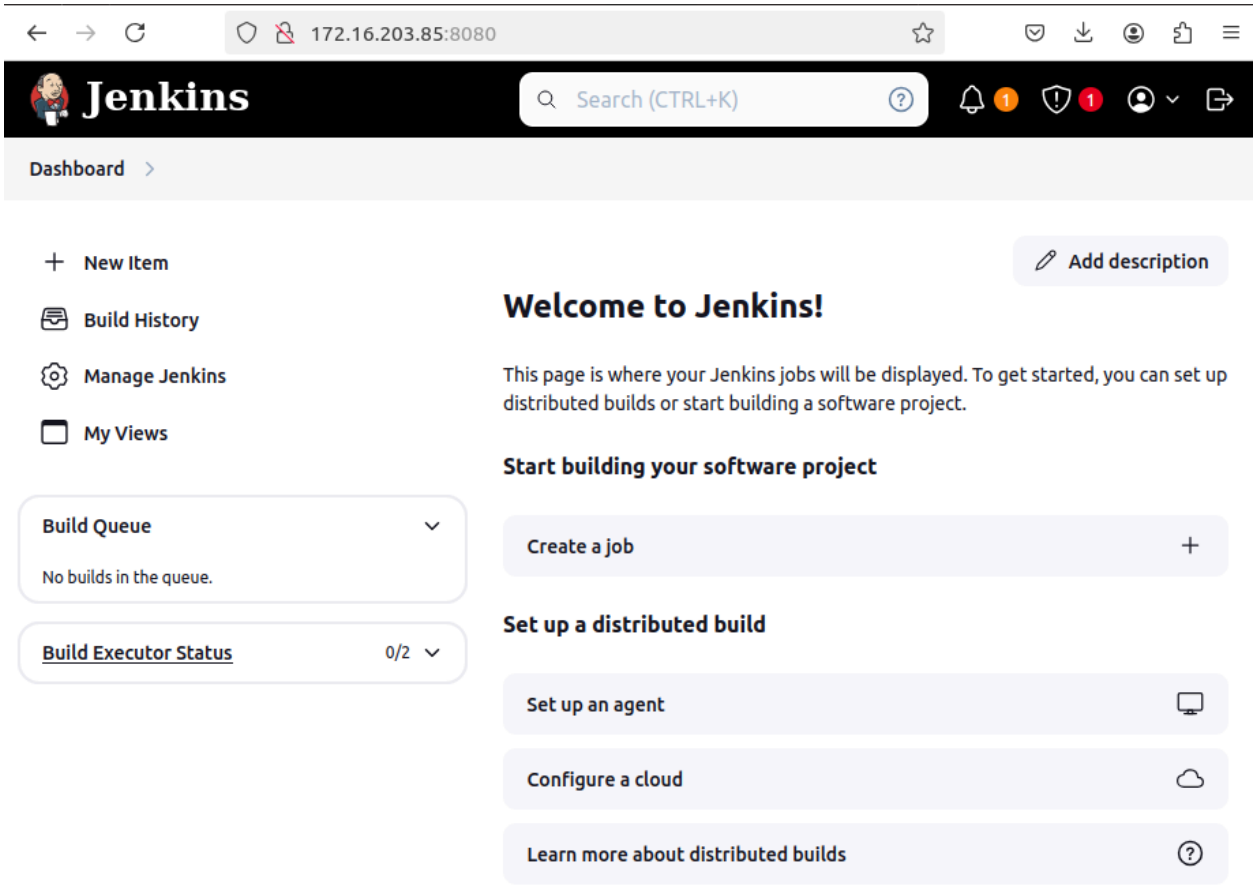
2.4. Test Results Analyzer Plugin

- 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**

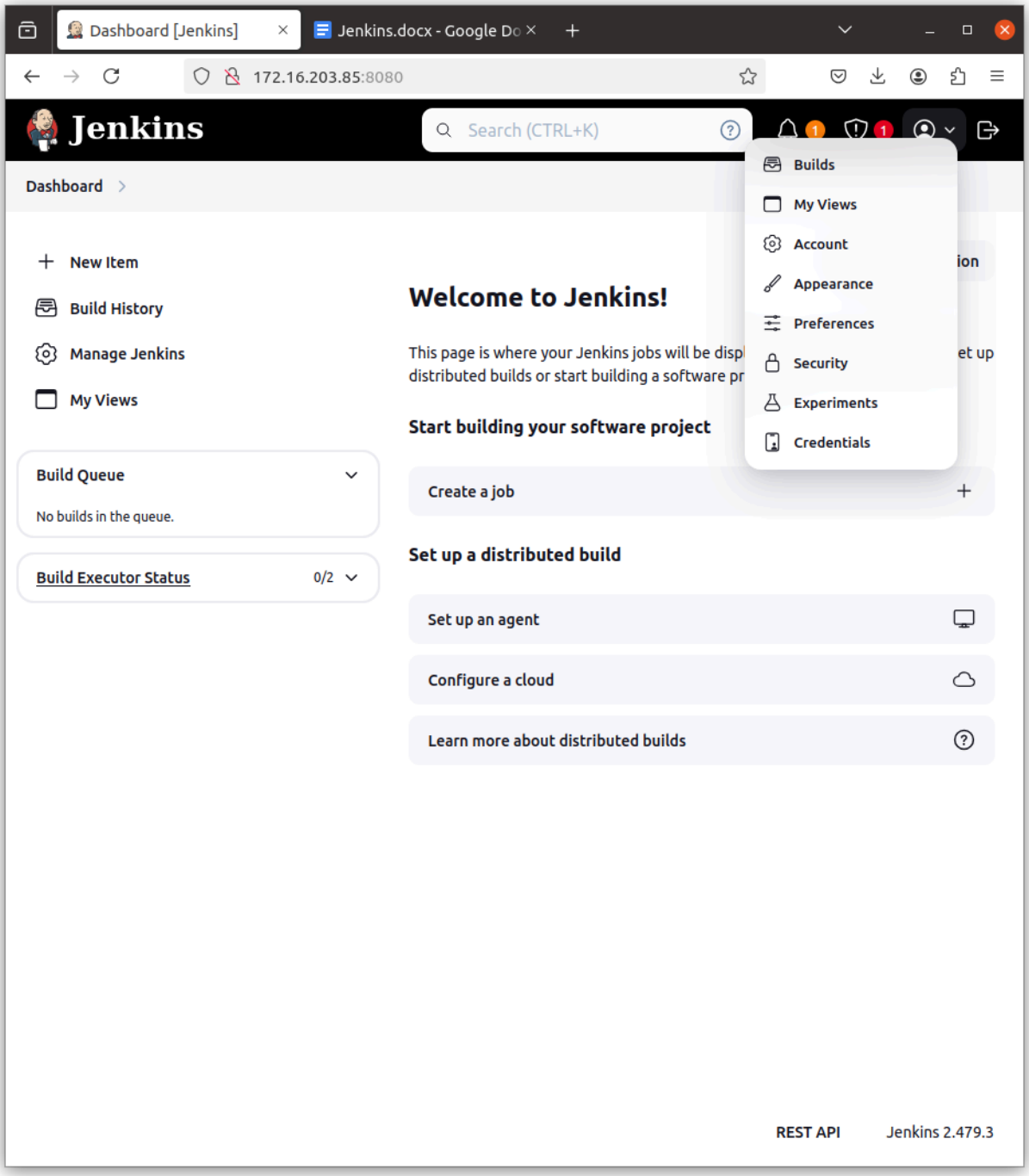


3. Jenkins

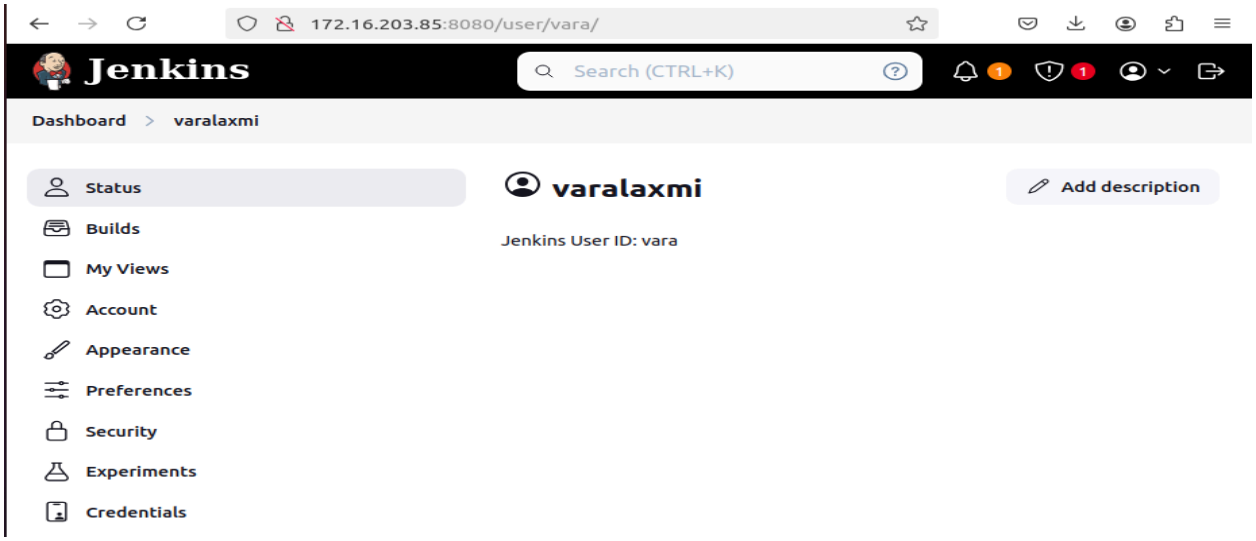
Welcome to jenkins dashboard



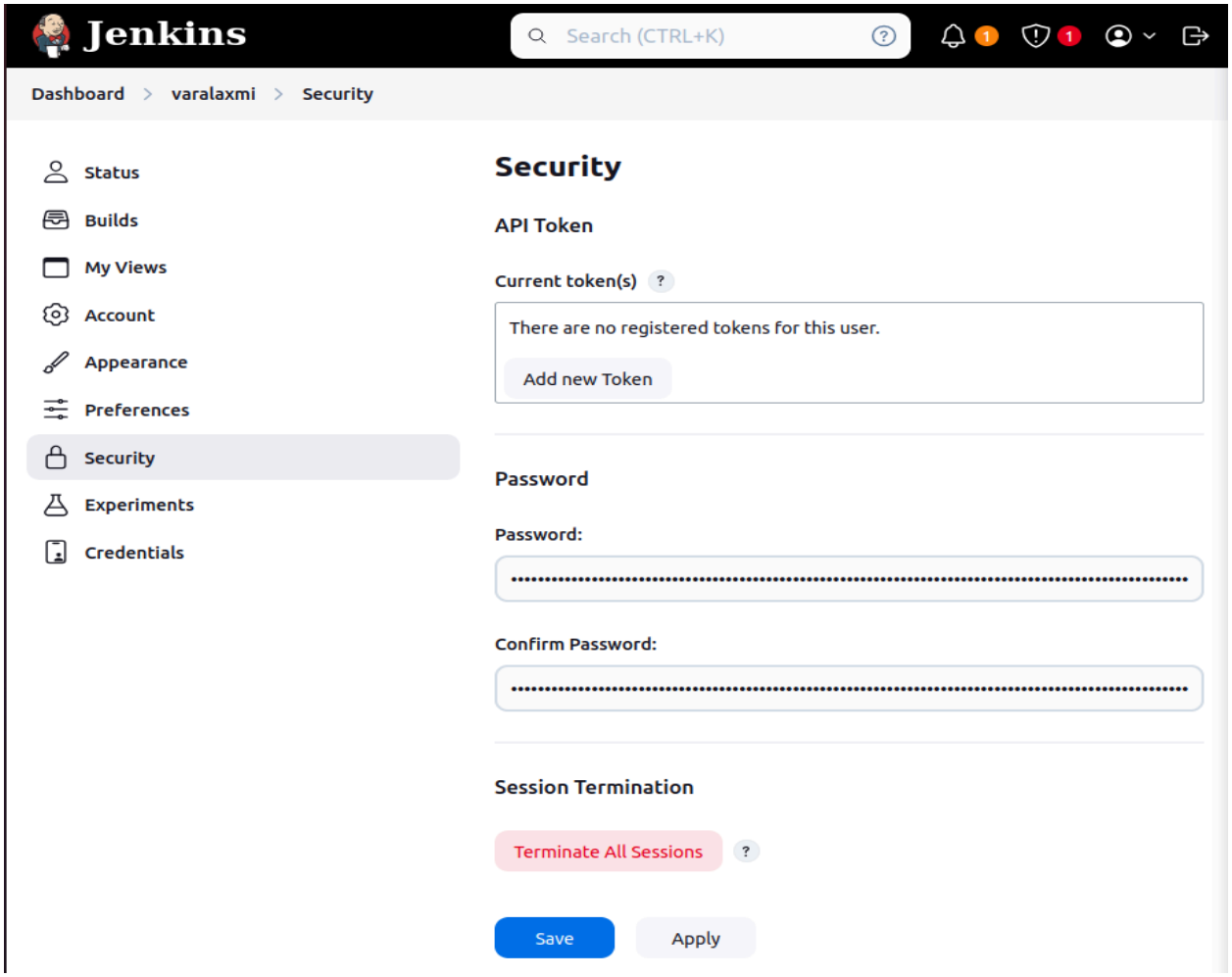
3.1. Reset the admin Password



This is the admin page



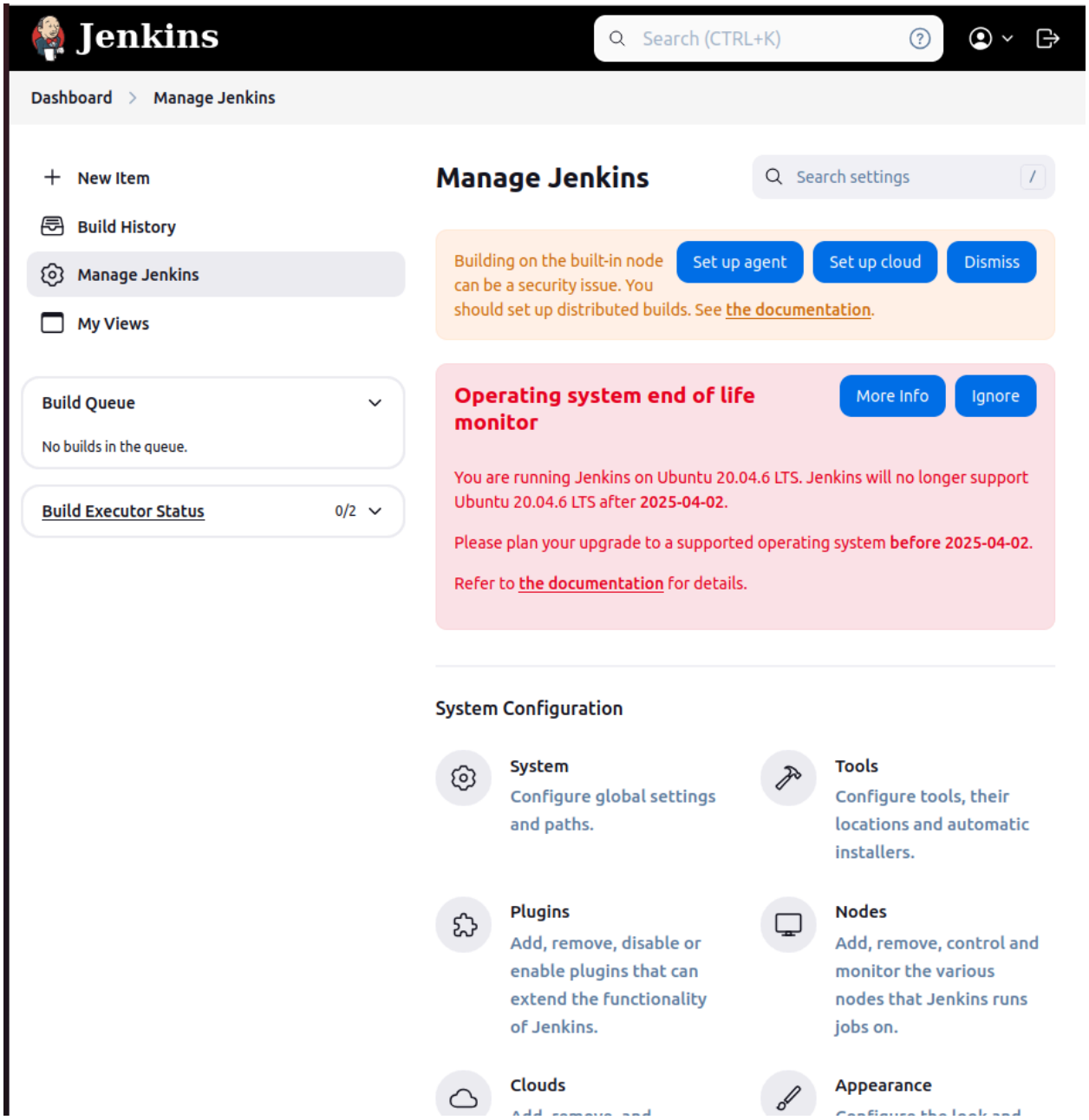
Goto the Security the change password and save



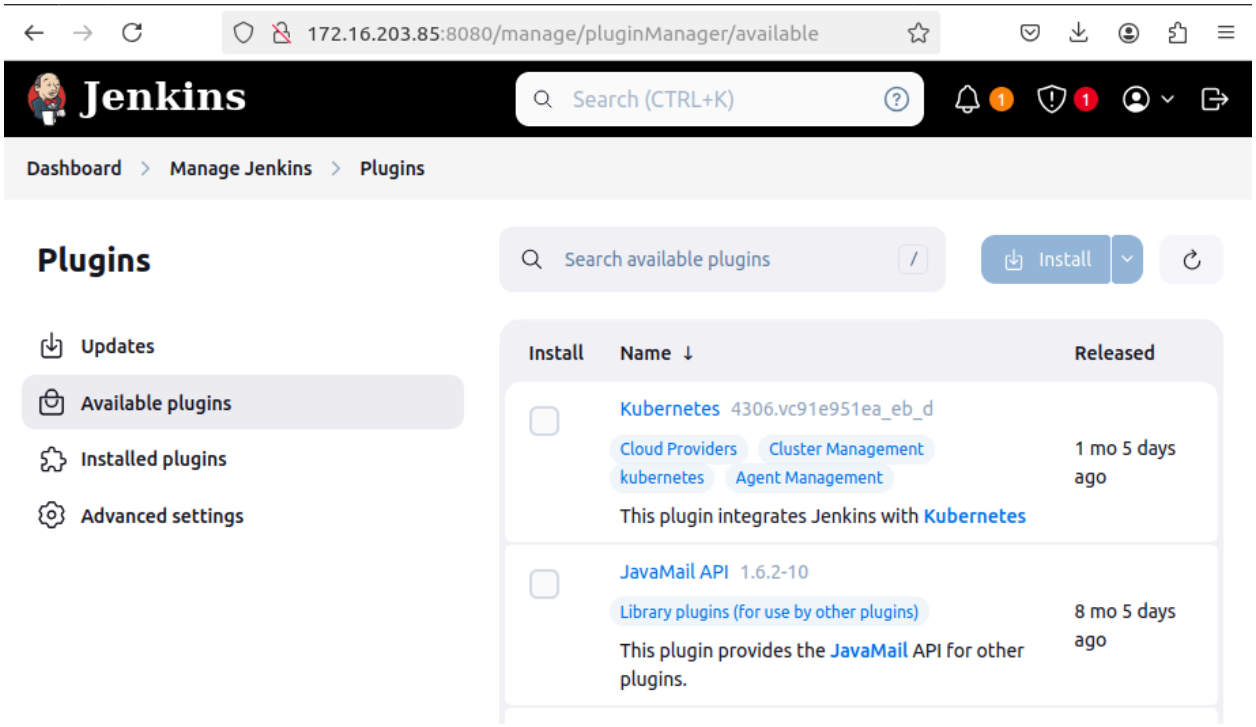
3.2. Available plugins

To get the available plugins in jenkins

Manage Jenkins ⇒ Plugins



Plugins ⇒ Available Plugins

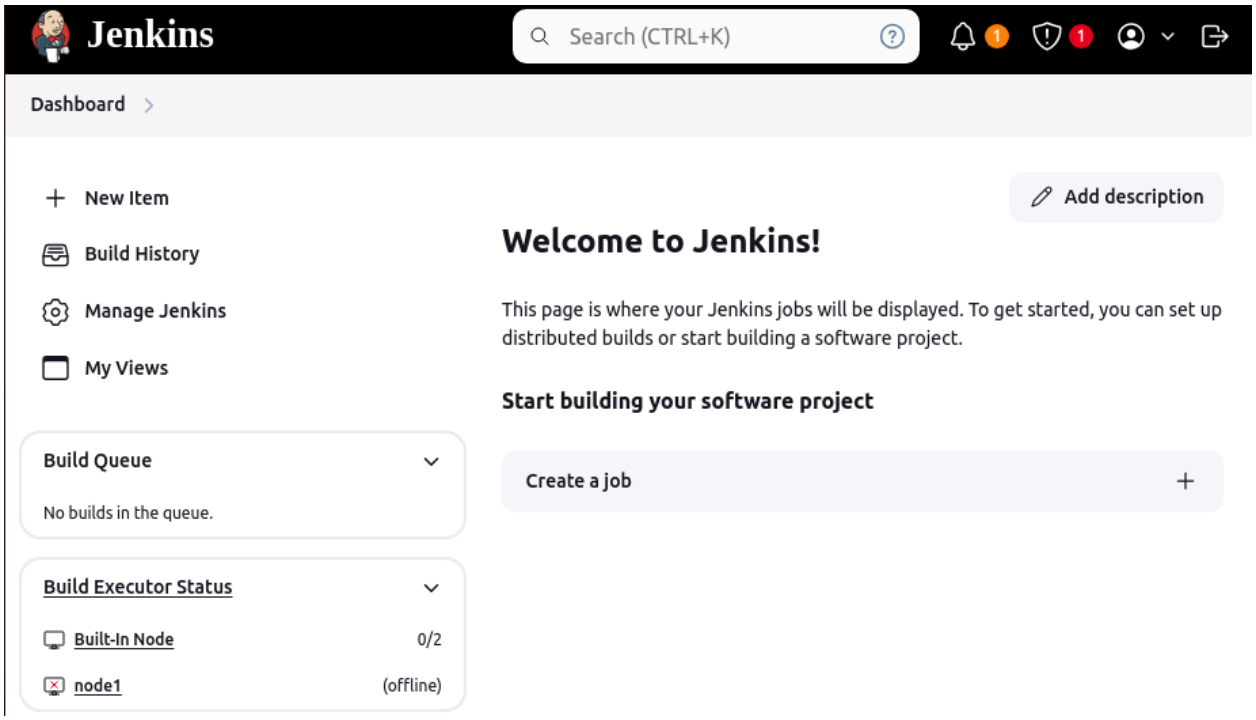


Through cmd line

```
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.

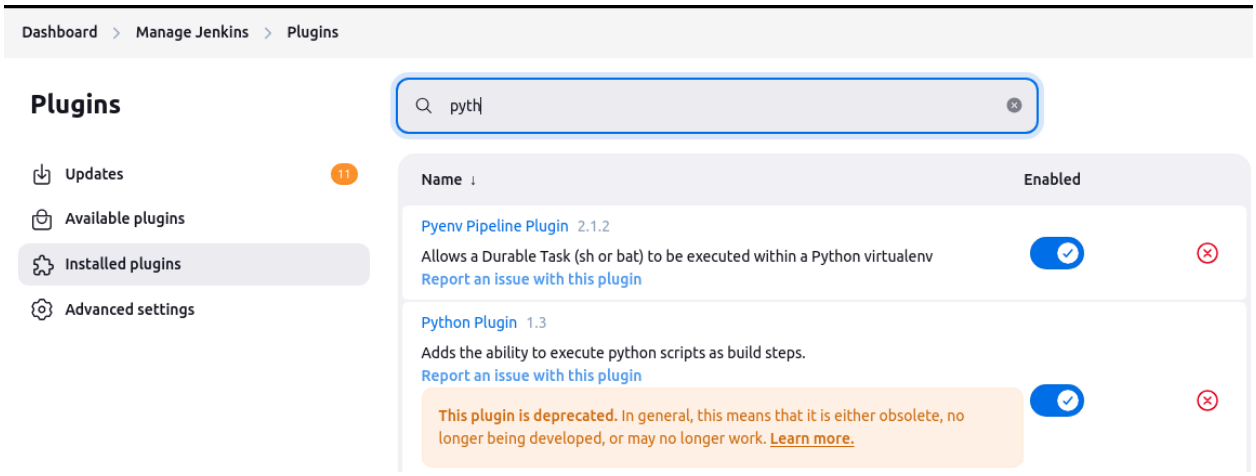


- Enter a name for your pipeline job (e.g., job1).
- Select "Pipeline" from the list of options.
- Click "OK".

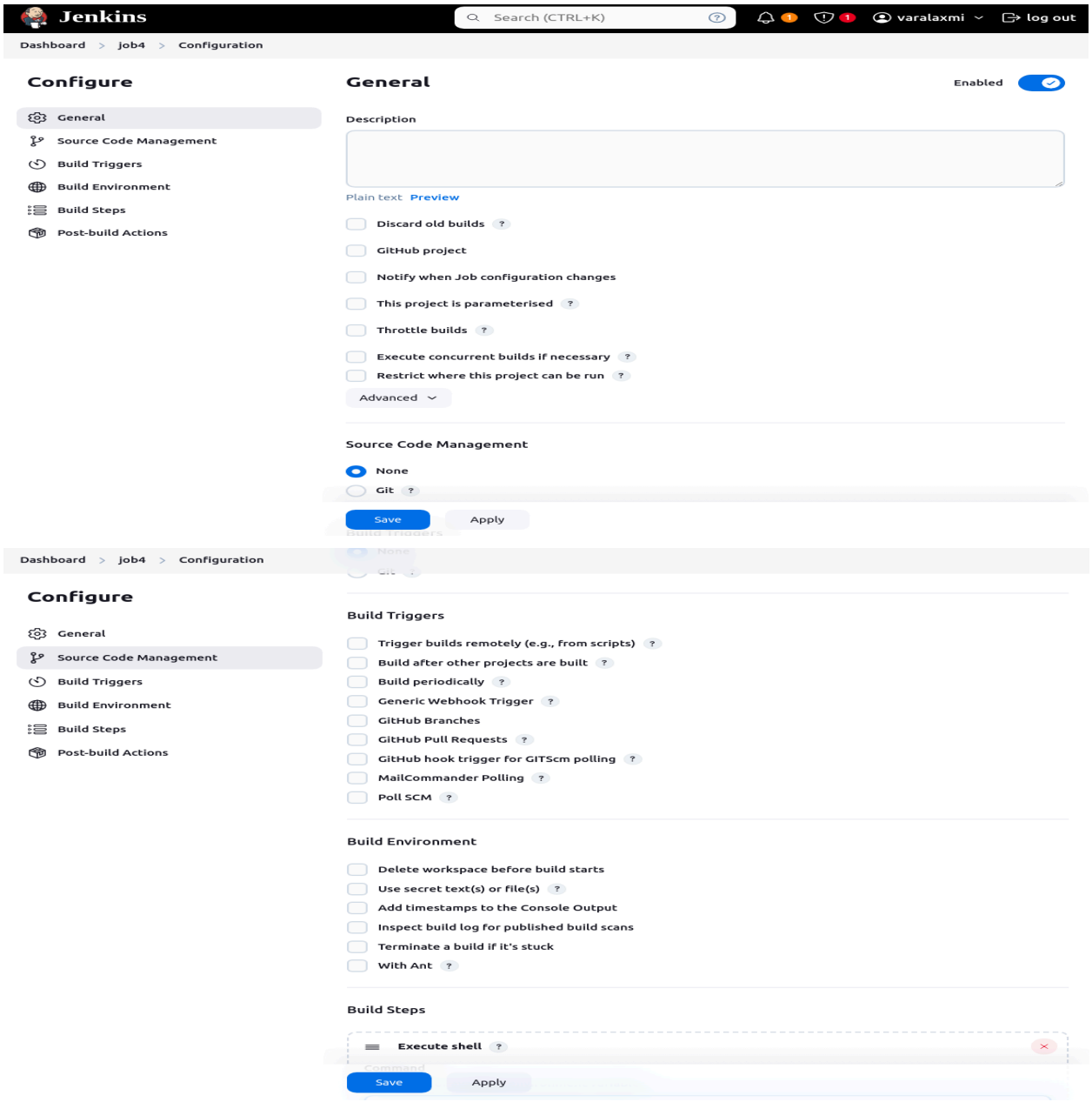
3.3.1. Using Freestyle Project

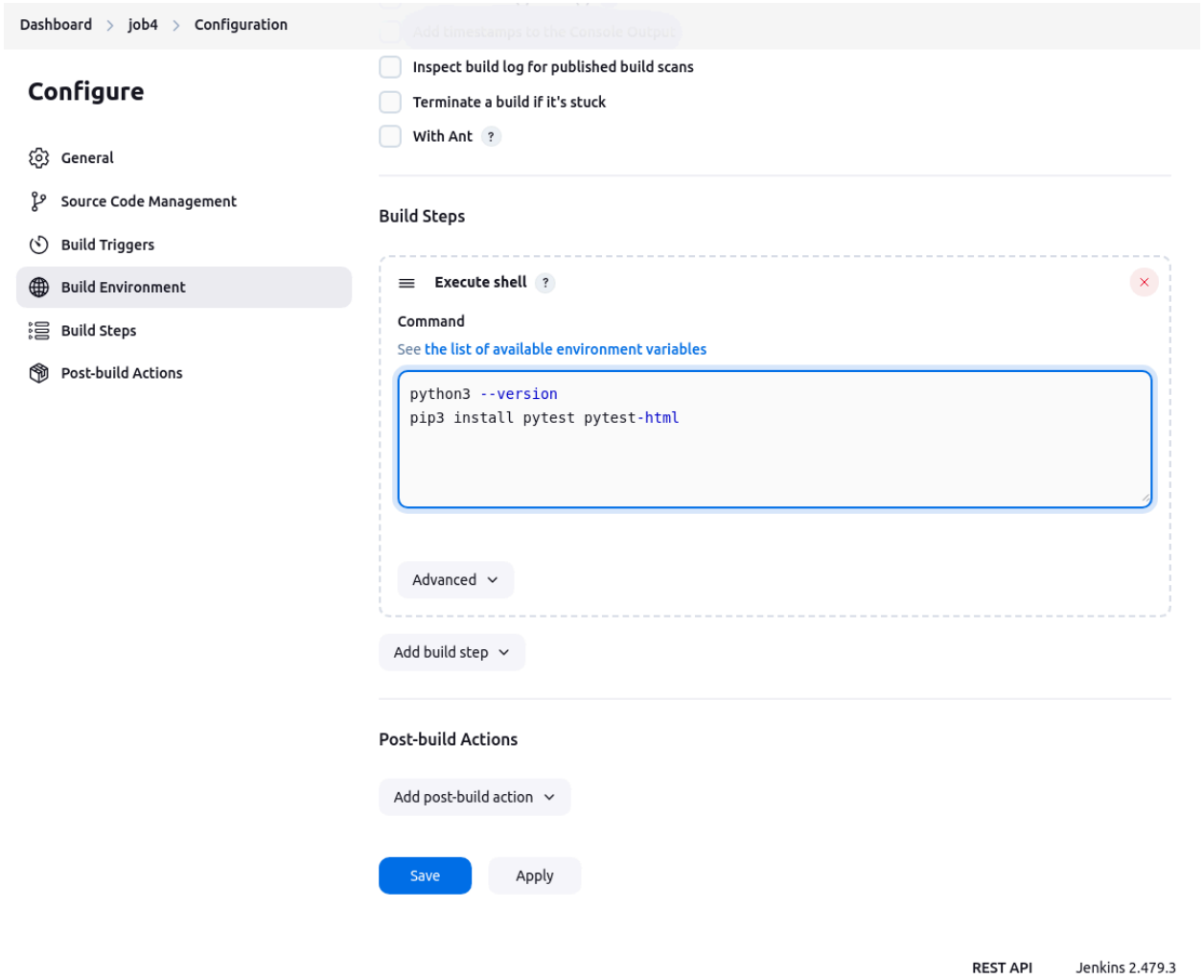
Note : Install python plugin

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



Template


Jenkins
Search (CTRL+K)
varalaxmi
log out
Dashboard > job4 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
General
Enabled
Description
Plain text Preview
Discard old builds
GitHub project
Notify when Job configuration changes
This project is parameterised
Throttle builds
Execute concurrent builds if necessary
Restrict where this project can be run
Advanced
Source Code Management
None
Git
Save
Apply
Dashboard > job4 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Generic Webhook Trigger
GitHub Branches
GitHub Pull Requests
GitHub hook trigger for GITScm polling
MailCommander Polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Add timestamps to the Console Output
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
Build Steps
Execute shell
Save
Apply



- Click save button after click on Build Now

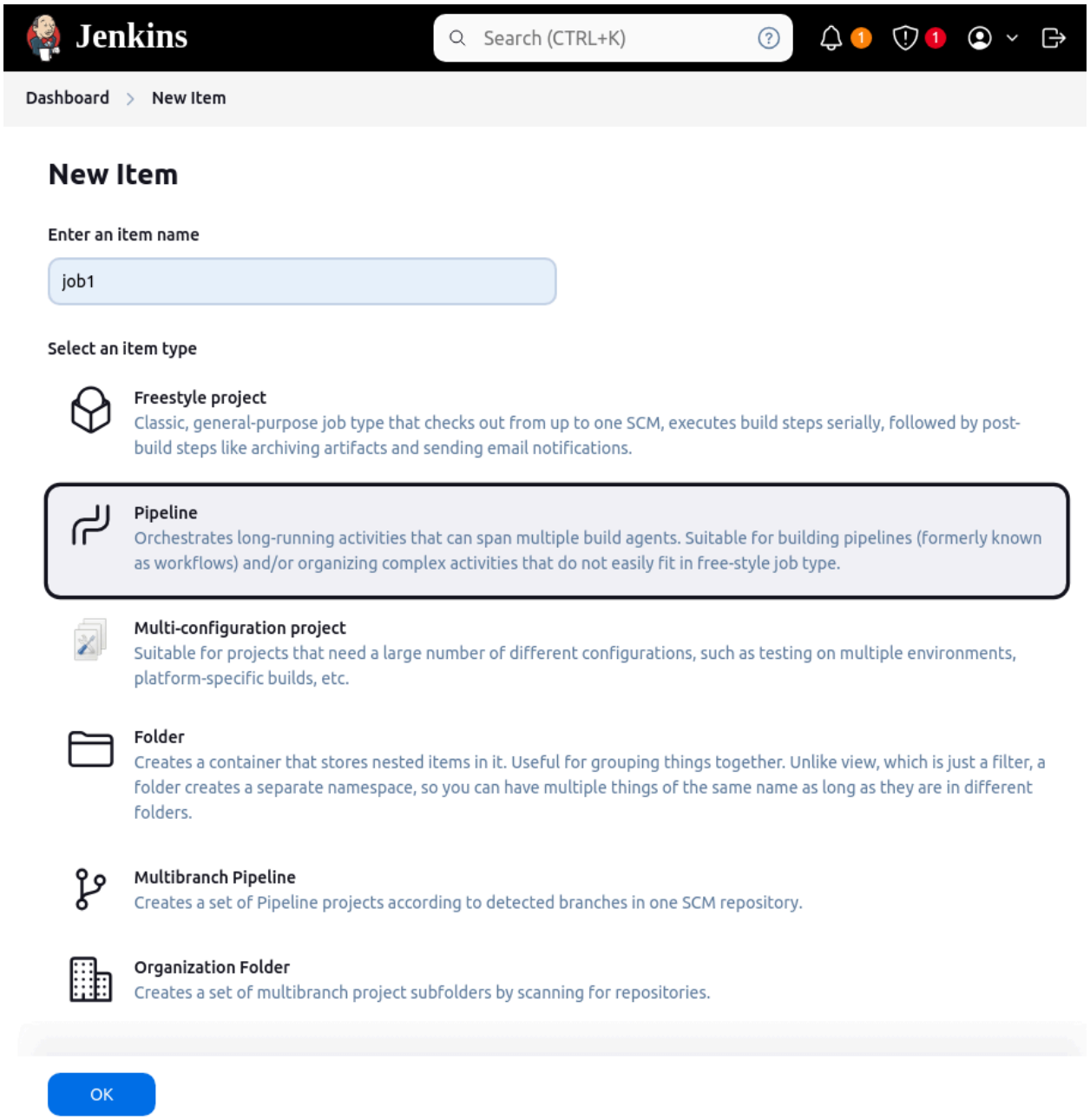
Status
Changes
Console Output
Edit Build Information
Delete build '#3'
Timings

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: iniconfig 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



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

Dashboard > job1 > Configuration

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

```
pipeline {
    agent any

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

- [x] Use Groovy Sandbox

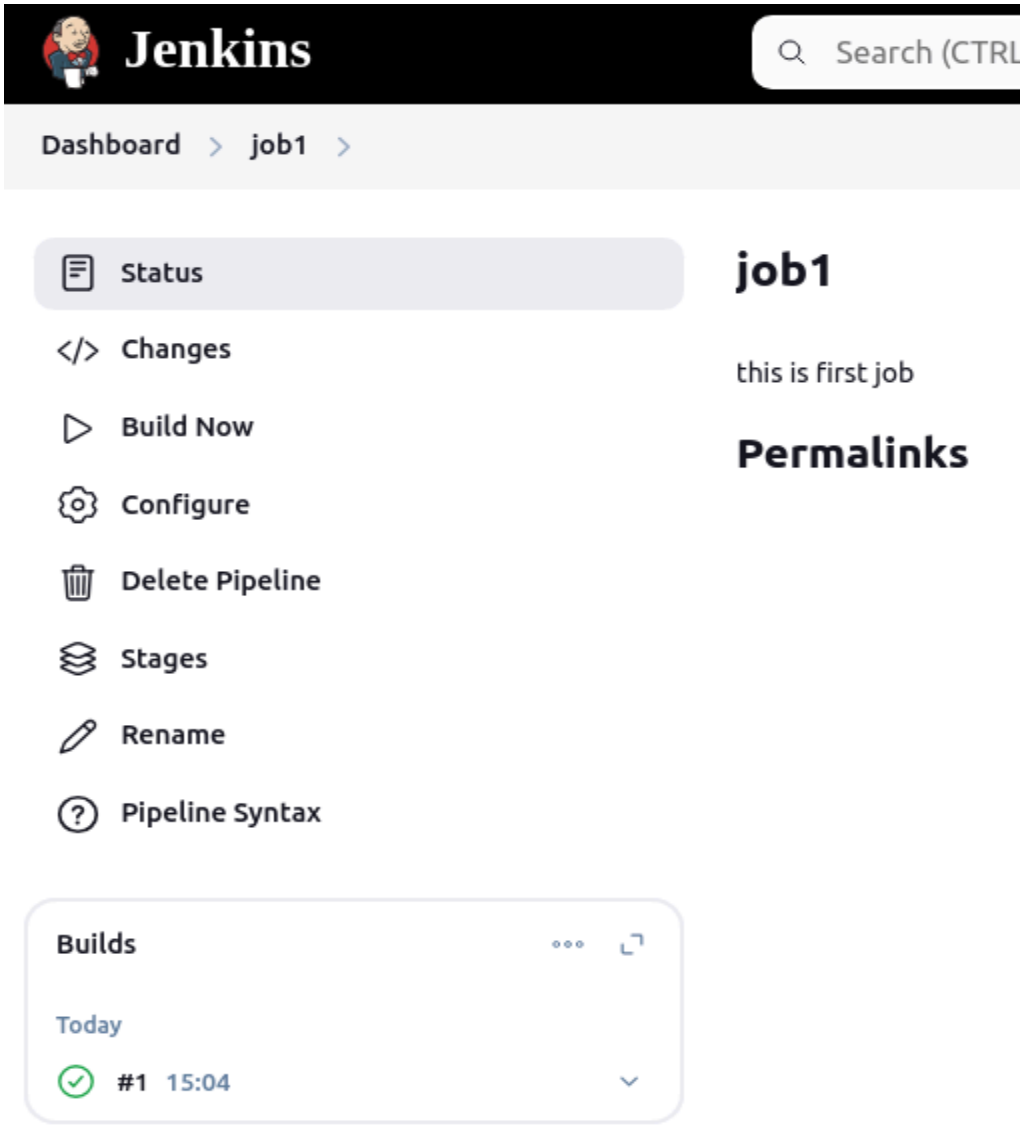
Pipeline Syntax

Save Apply

Pipeline script :

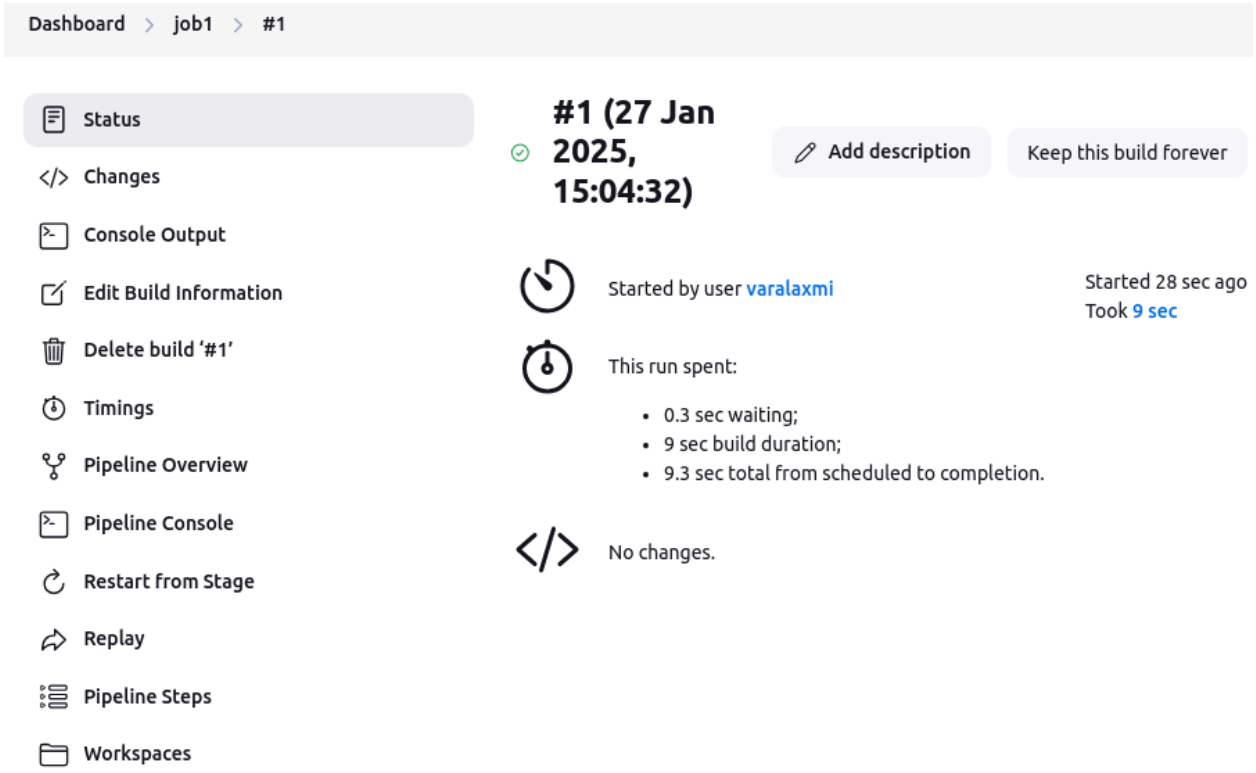
Script ?

```
pipeline {
    agent any

    stages {
        stage('Hello') {
            steps {
                script {
                    echo 'Hello World'
                }
            }
        }
    }
}
```



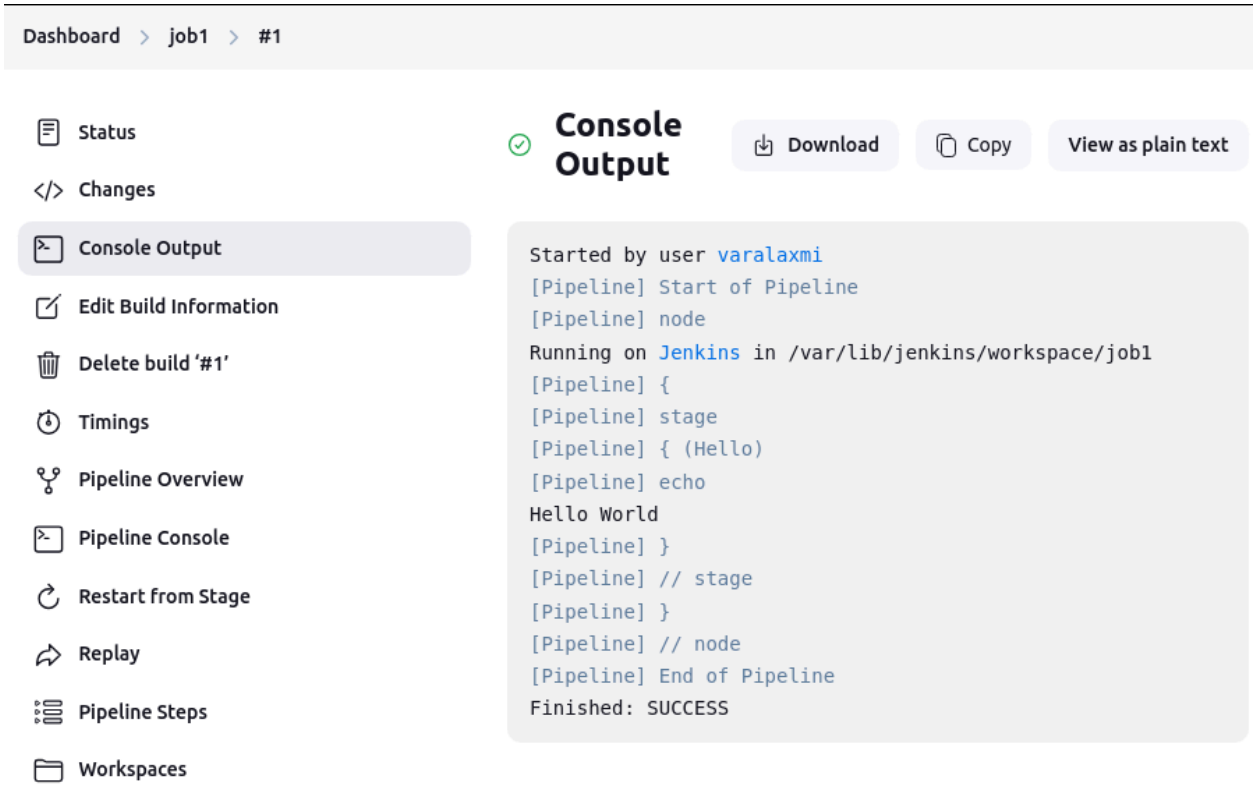
3.3.2.1. Status

- 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.



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).




```
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
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
```

3.3.2.3. No workspace Initially

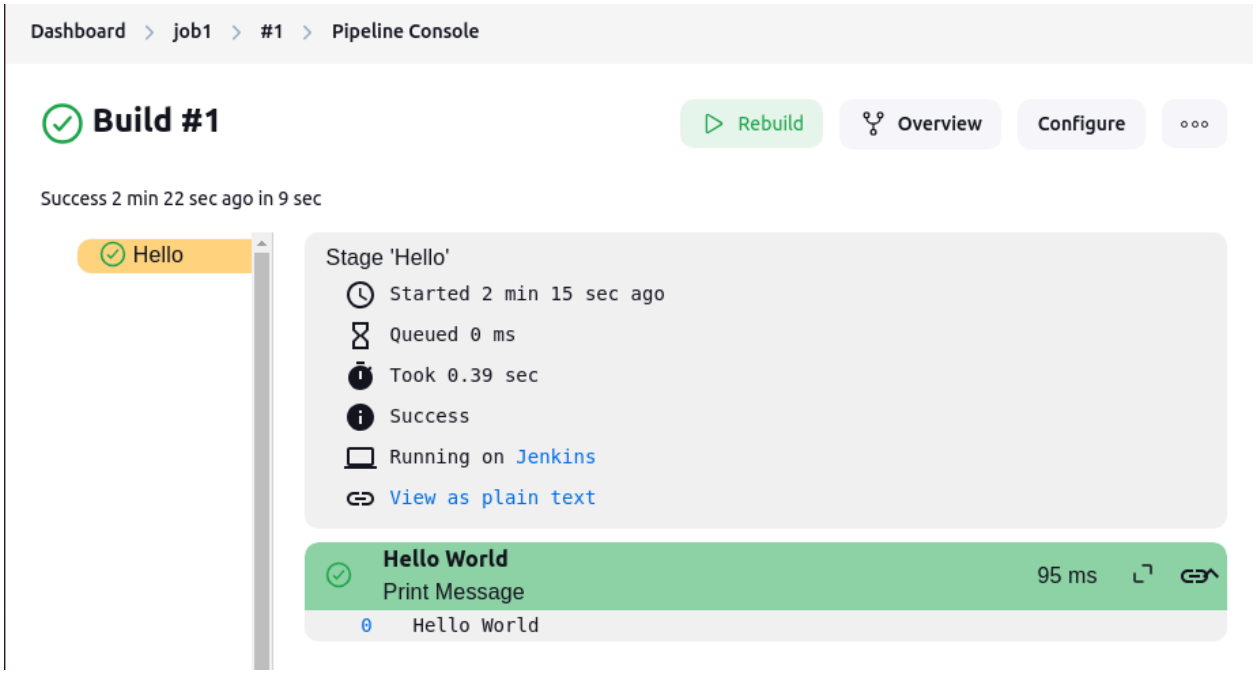
```
vlab@HYVLAB7:~$ cd /var/lib/jenkins/woekspace/
bash: cd: /var/lib/jenkins/woekspace/: No such file or directory
```

- 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

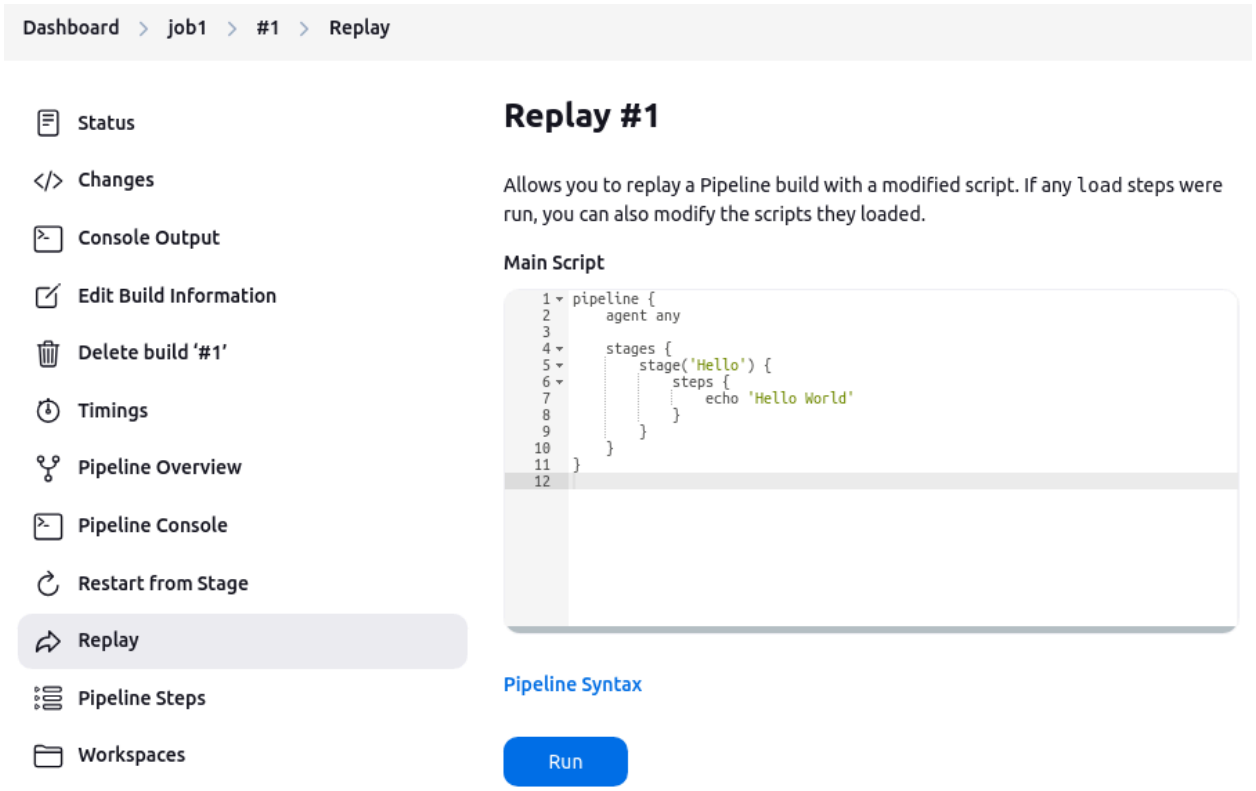
- If your pipeline includes steps that require creating or accessing files(eg.., creating a file, cloning a repository, reading a file) jenkins creates a workspace directory automatically.
- The workspace serves as the job's working directory where jenkins performs file-related operations.

3.3.2.5. Pipeline console

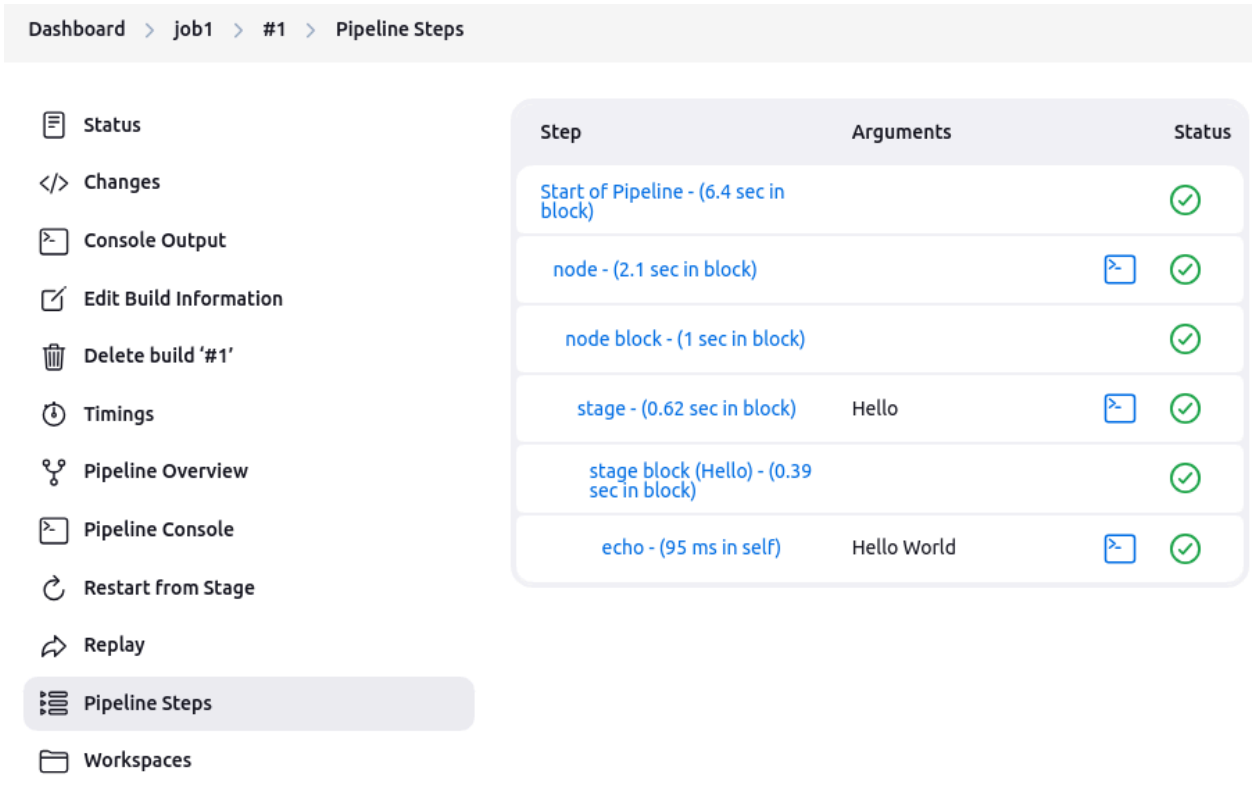


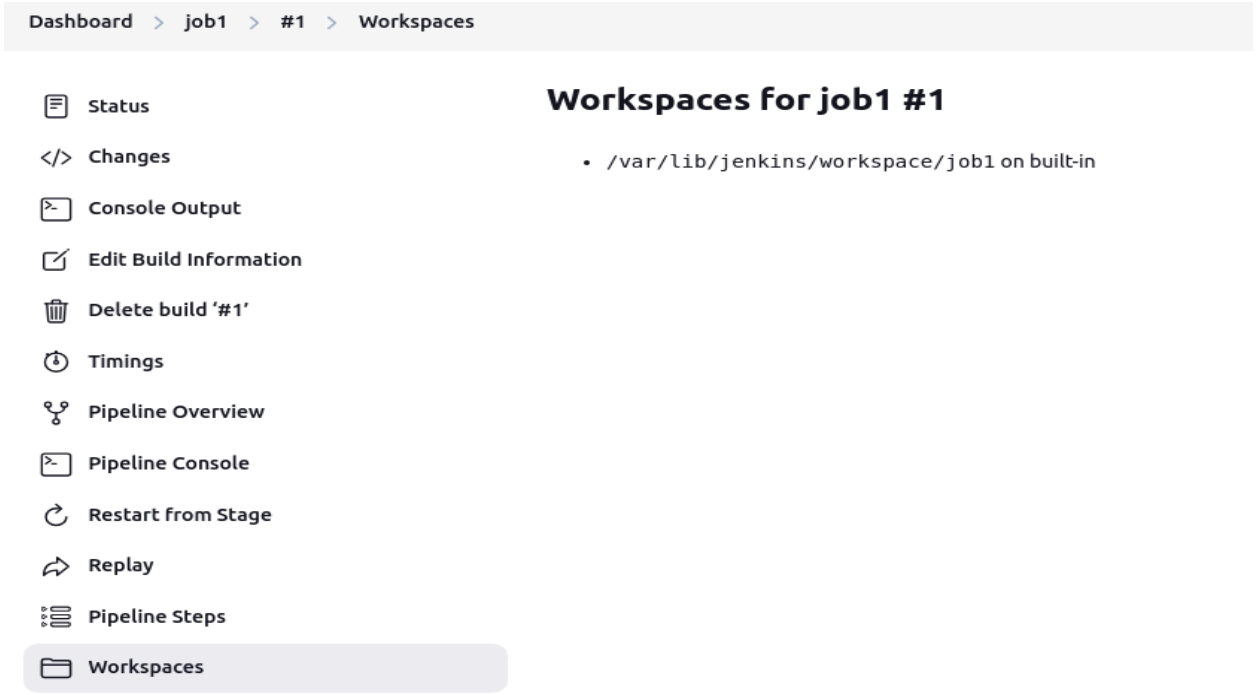
3.3.2.6. Replay

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



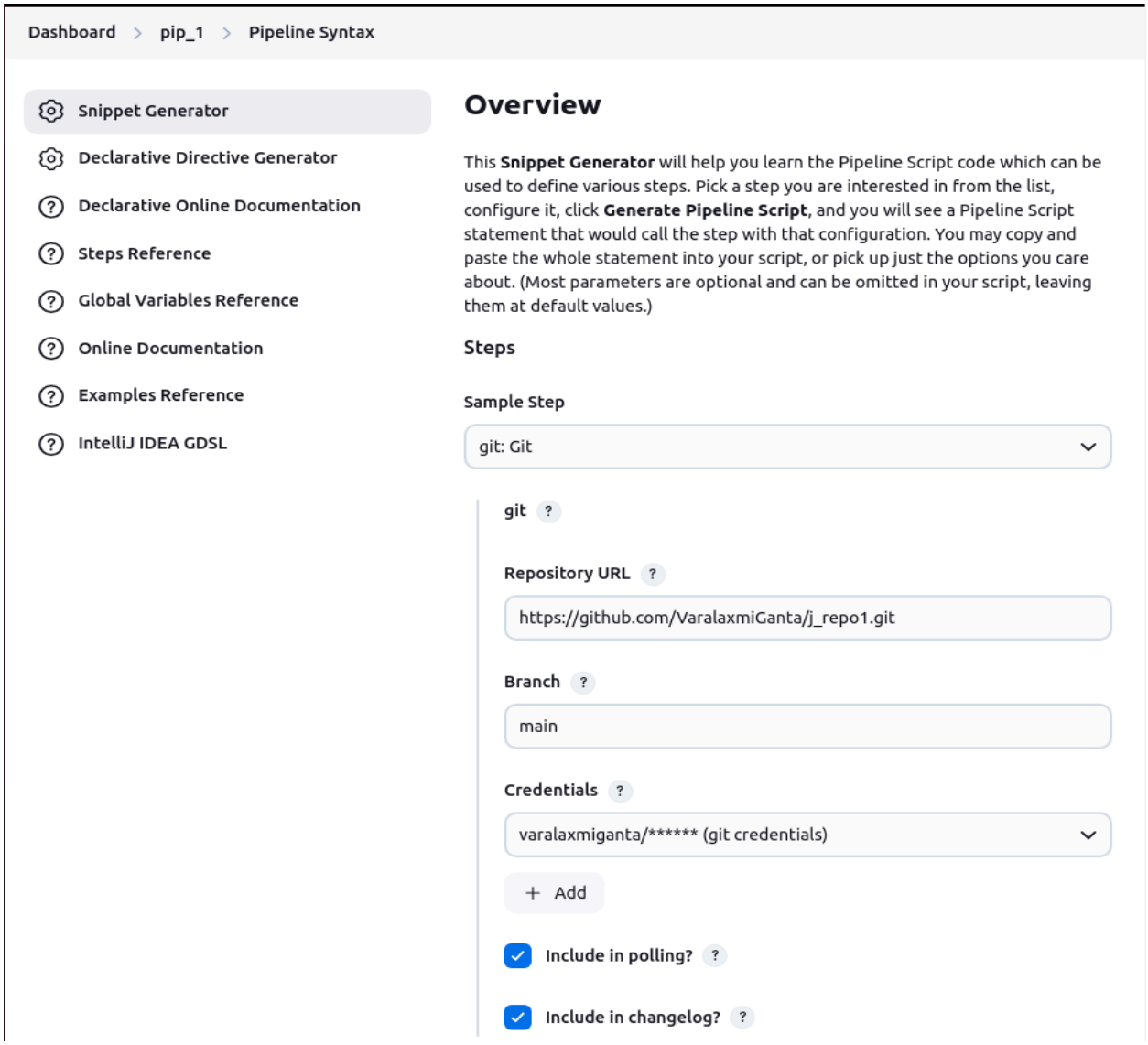
3.3.2.7. Pipeline Steps



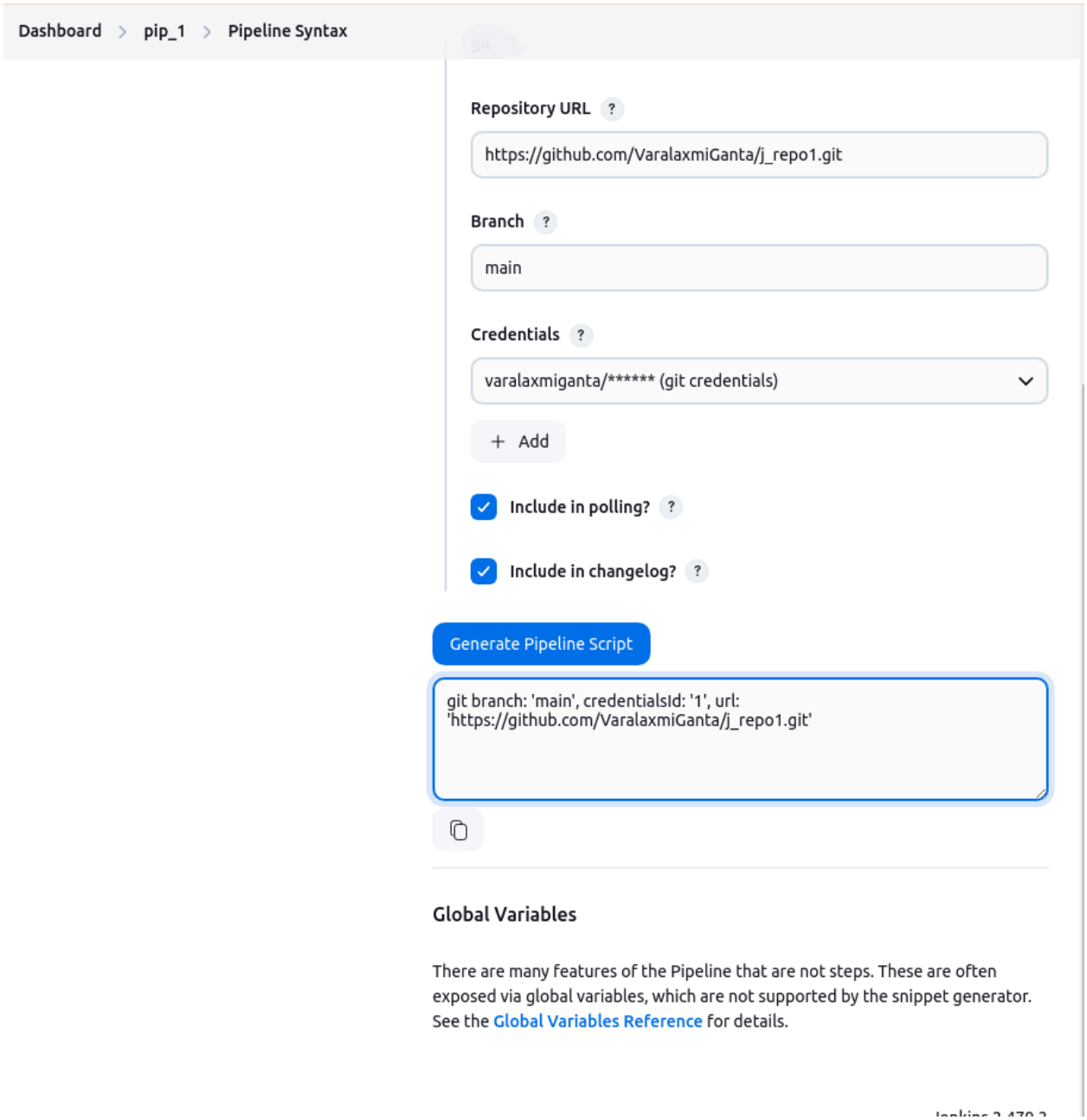


3.3.2.8. Generate Pipeline Script

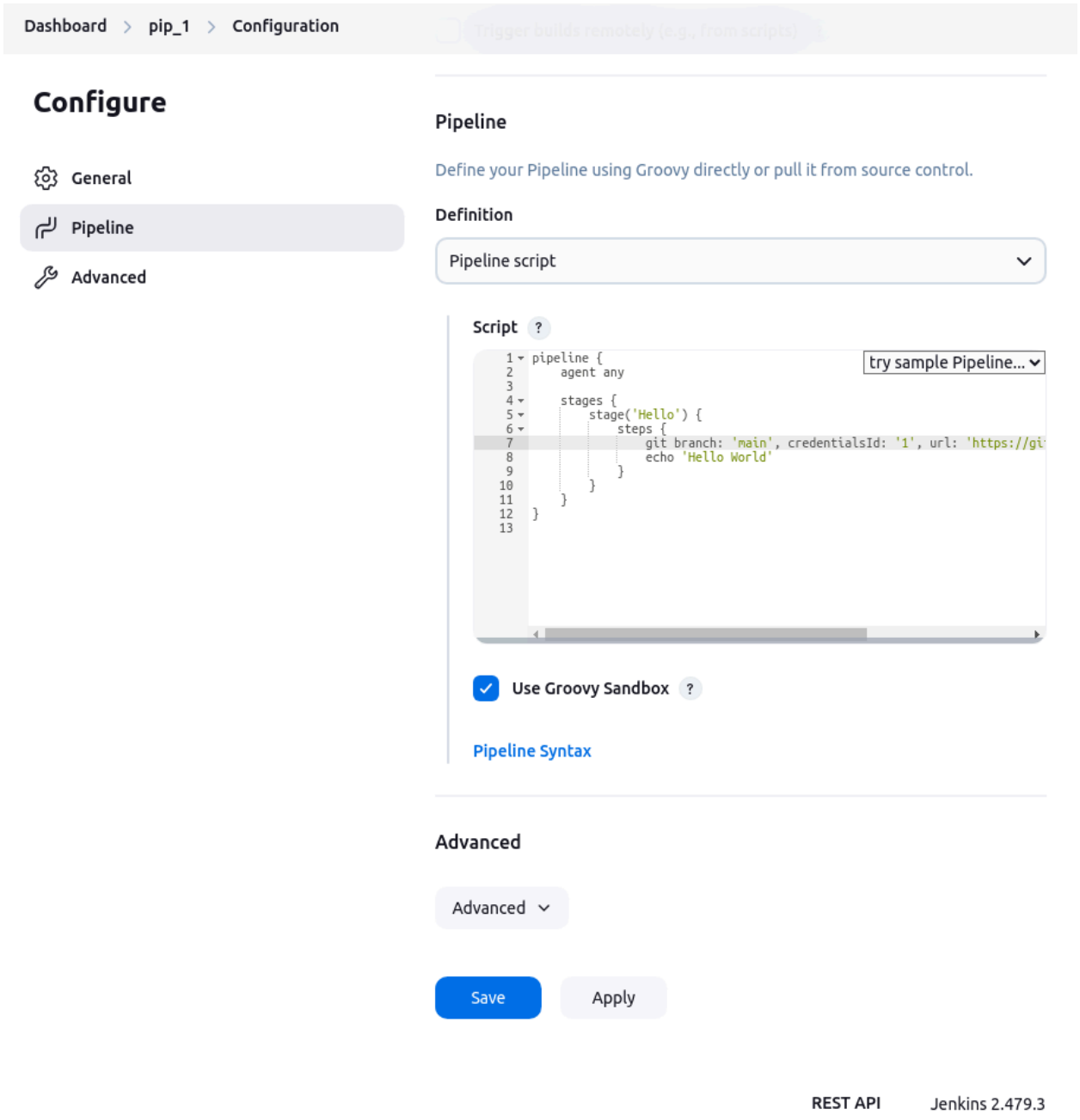
- **agent { label 'agent1_66' }**
  This tells Jenkins to run the entire pipeline on any node that has the label agent1_66.
- **stages { ... }**
  The pipeline is broken into several stages (Checkout, Build, Test, Deploy). Each stage can contain one or more steps.
- **steps { ... }**
  Each state's steps 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).



- **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.



- After past it into your pipeline script and save it



- Click on build now and observe the console output.

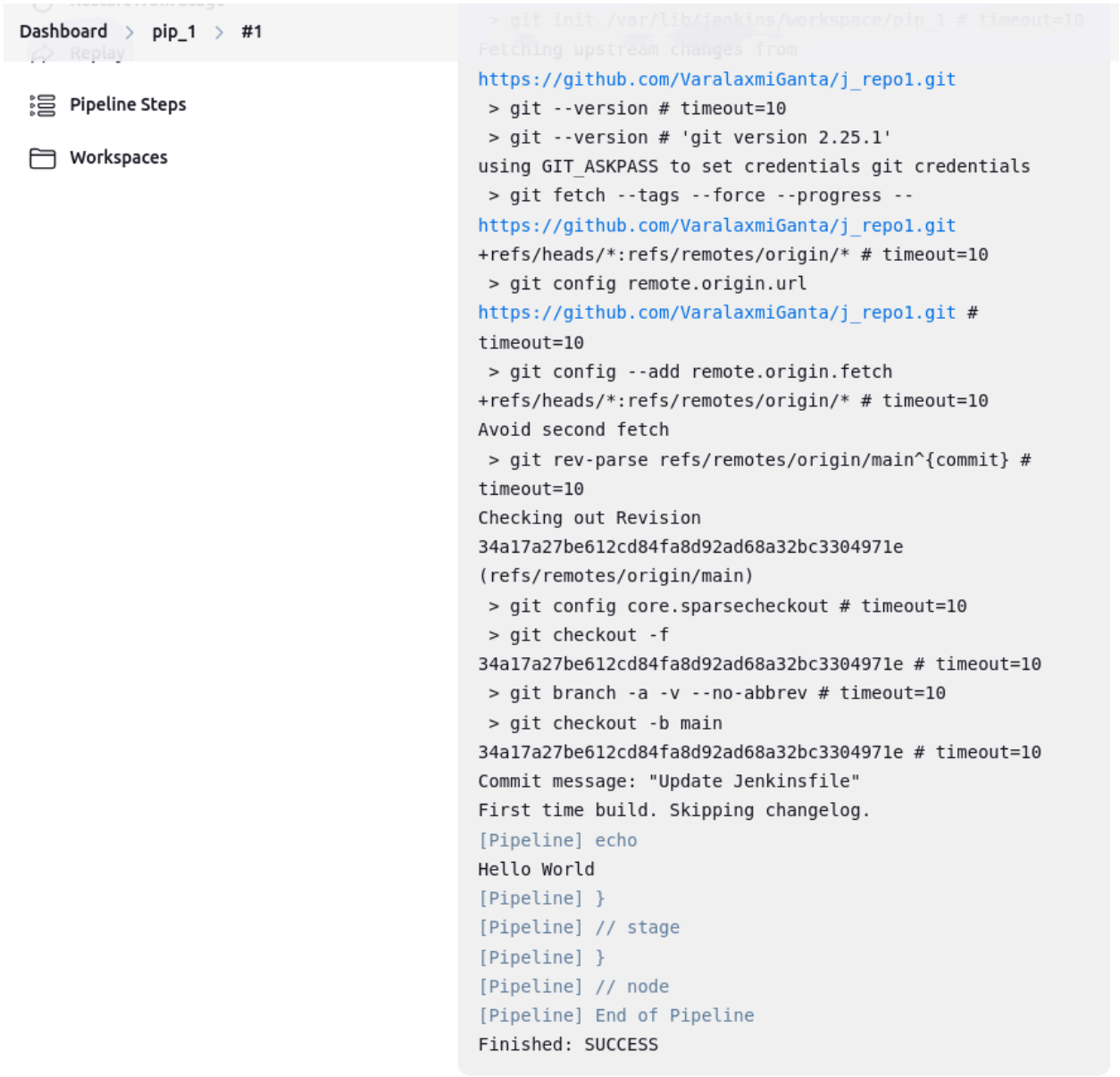
Dashboard > pip_1 > #1

- 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_repo1.git
 > git init /var/lib/jenkins/workspace/pip_1 # timeout=10
Fetching upstream changes from
https://github.com/VaralaxmiGanta/j_repo1.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_repo1.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url
https://github.com/VaralaxmiGanta/j_repo1.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
```

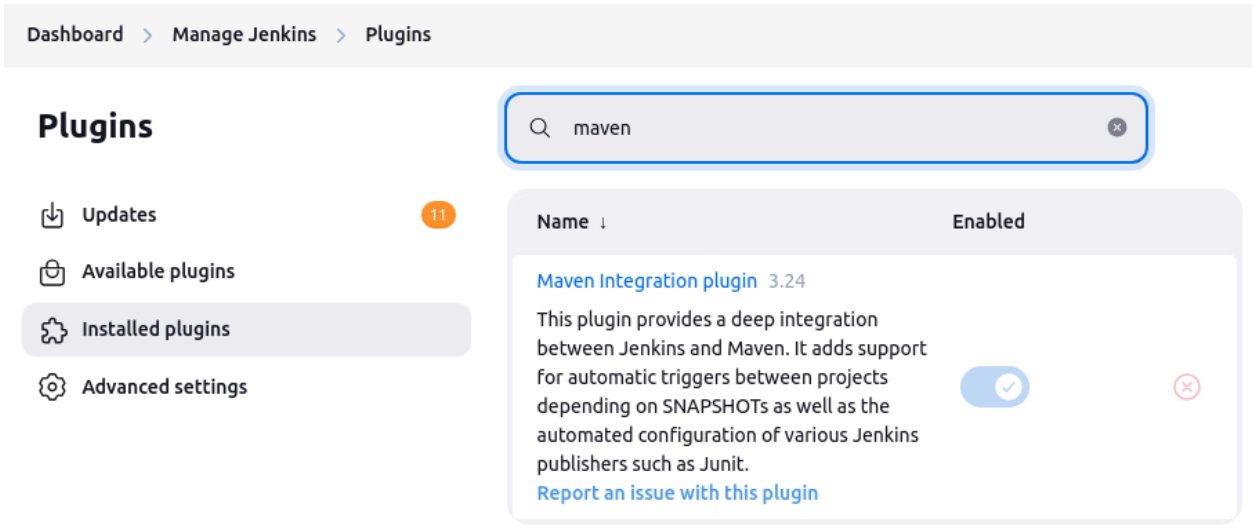


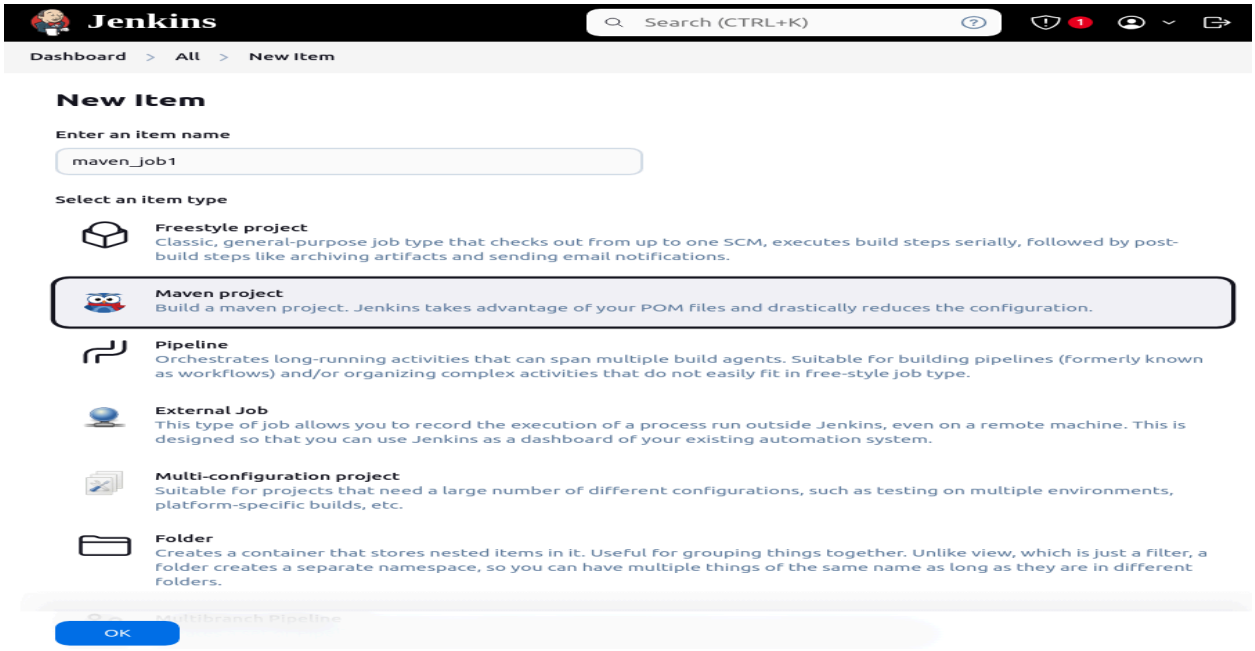
3.3.3. Using Maven Project

- 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

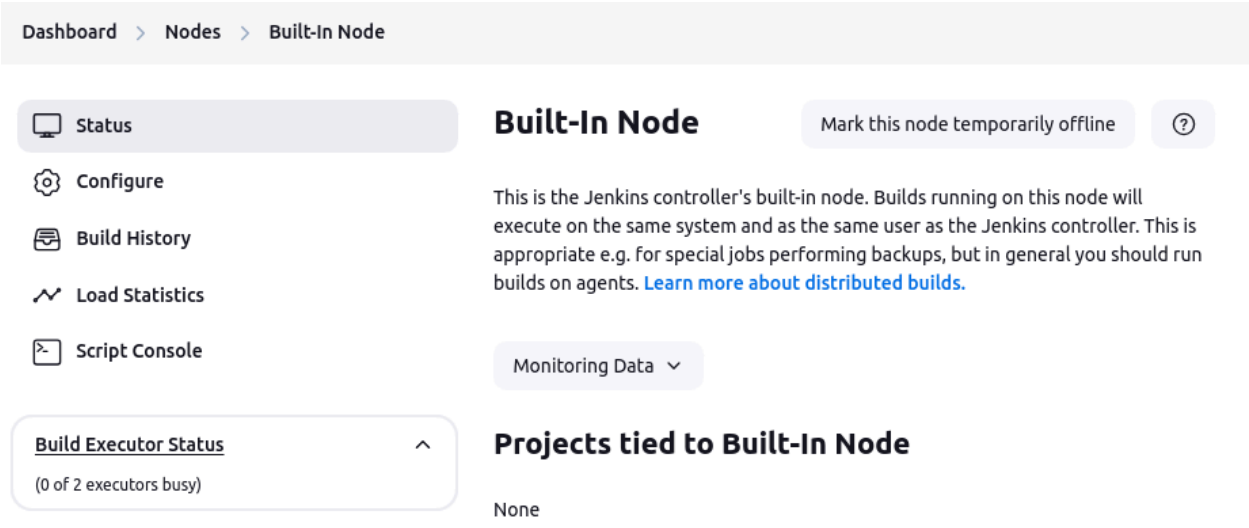




3.4. Nodes

- 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

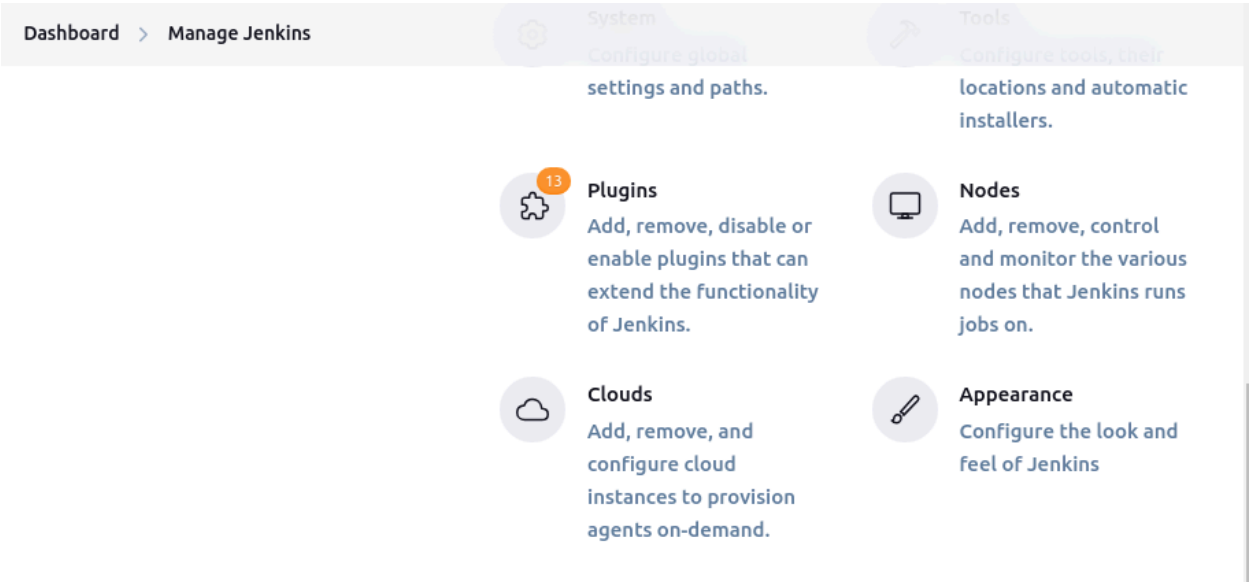


- **Agent (slave) Node**
  - A secondary machine that connects to the master.
  - Executes jobs as assigned by the master.

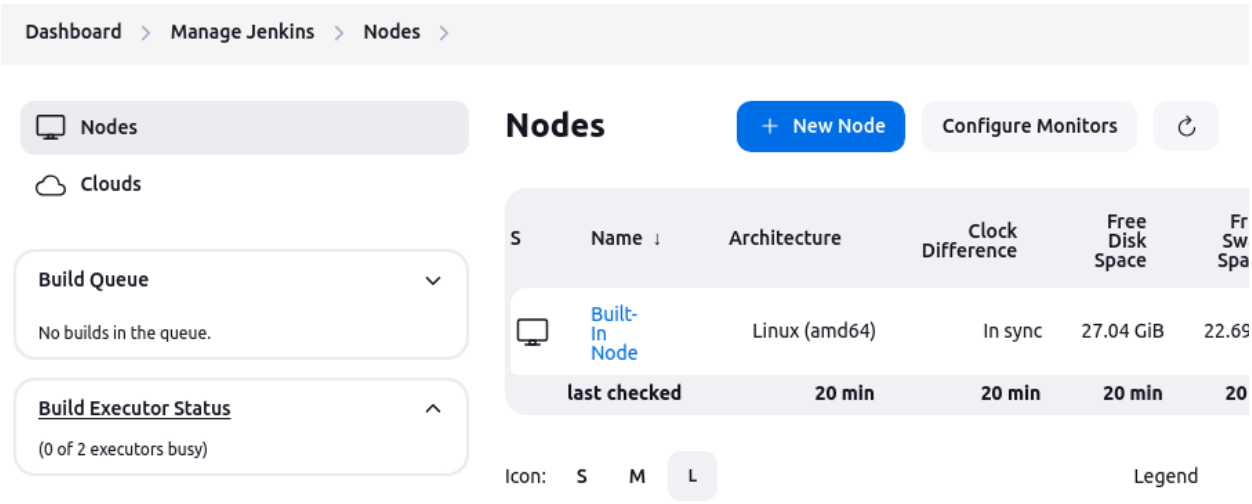
3.4.1. Why ?

- **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



- Goto Manage Jenkins → Nodes



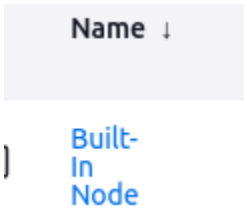
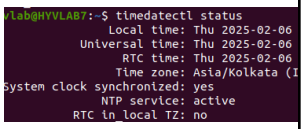
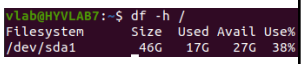
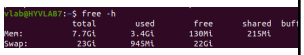
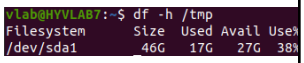
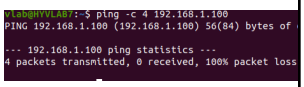
3.4.2.1. Build Queue

- “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	
Architecture	The system architecture (eg linux (adm64) for a 64-bit linux machine	Displays the system architecture	
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.	
Free disk space	Available disk space on the node	Indicates available disk space on the node. Low space may cause build failures.	
Free swap space	Amount of swap memory available	Displays available swap memory, which helps prevent crashes when RAM is full.	
Free temp space	Space available in the temporary directory	Shows available space in /tmp, which is used for temporary build files and logs.	
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.	

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

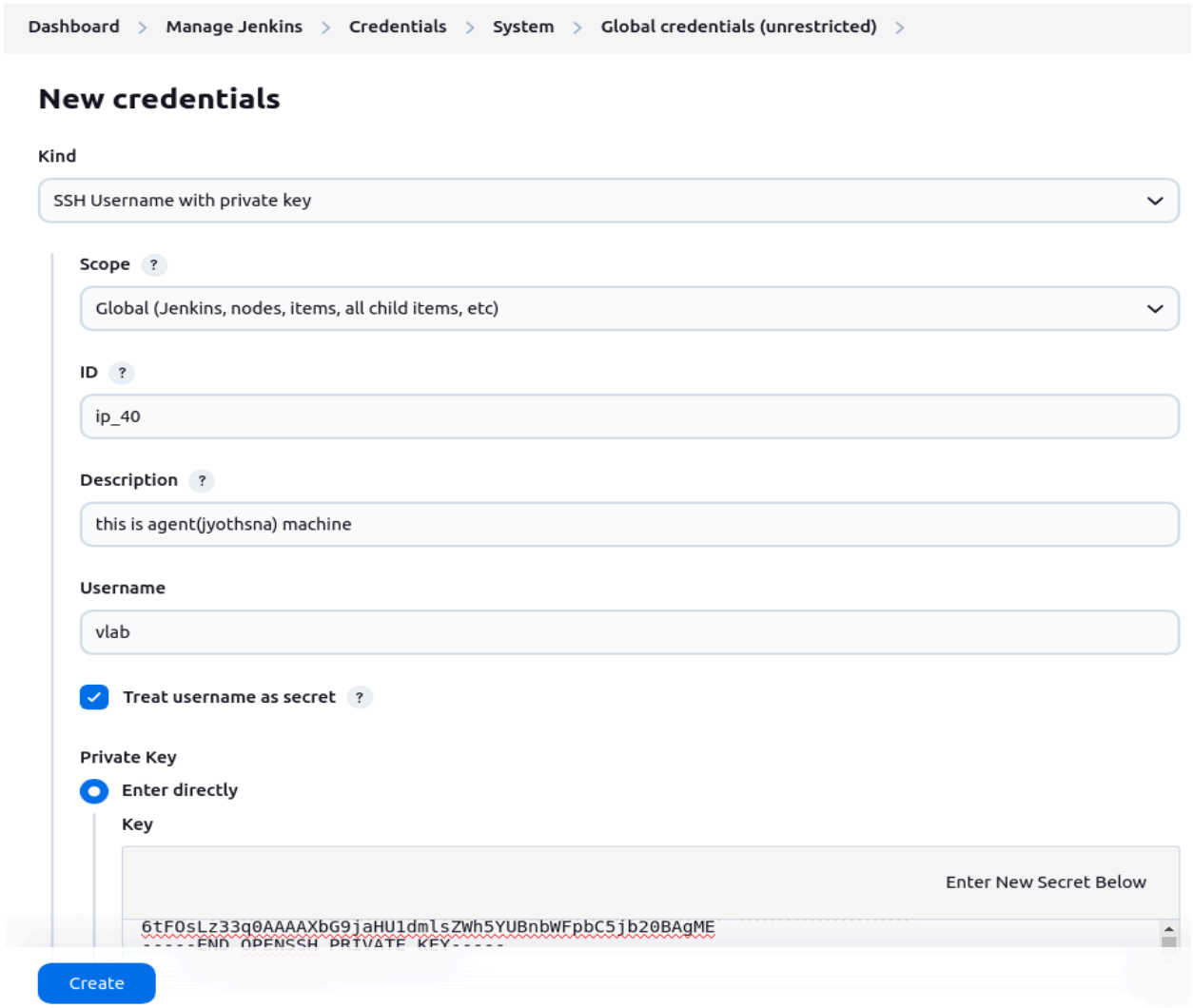
Username and ip address

```
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
```



- **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/var**

```
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
```

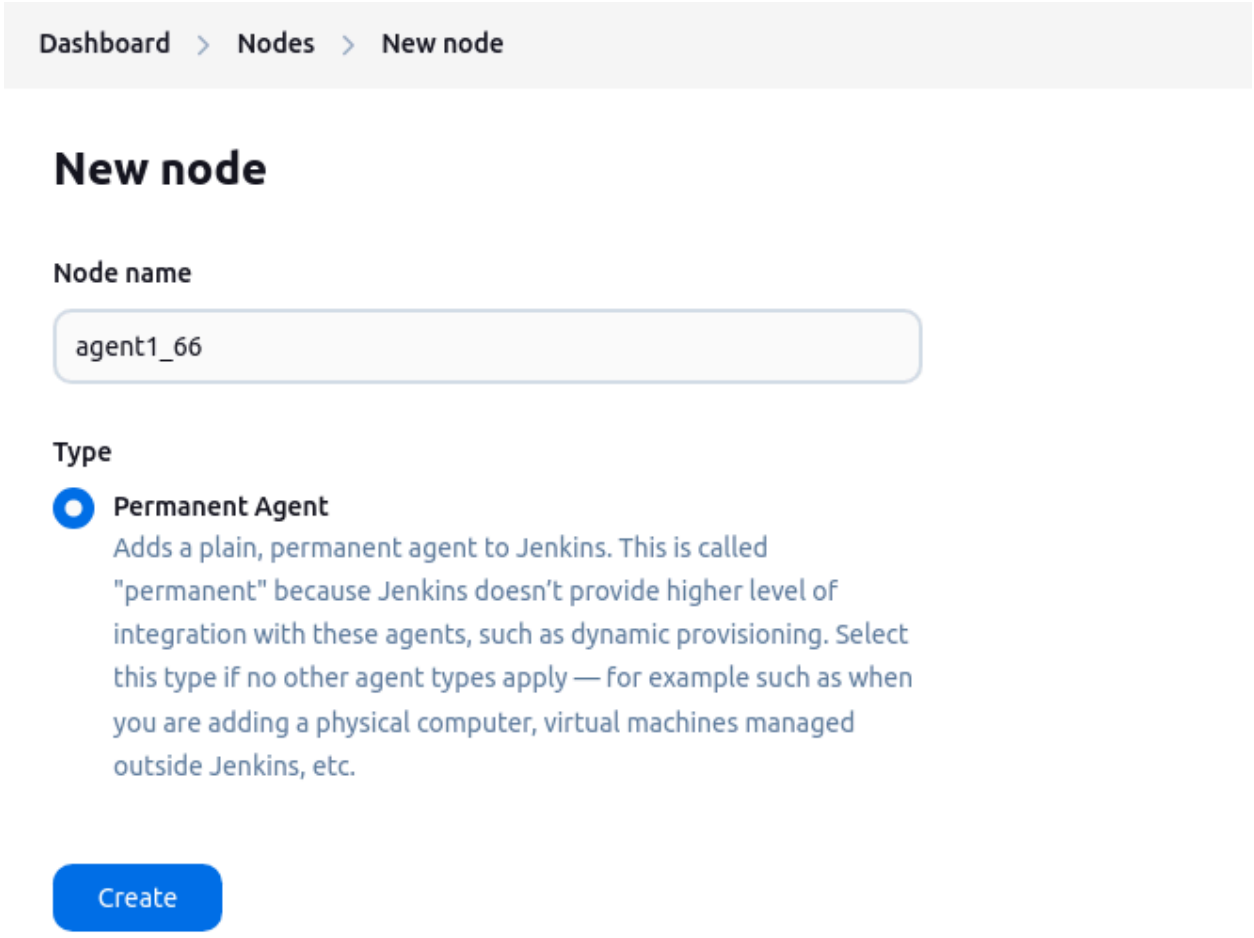
```
vlab@HYVLAB8:~$ sudo useradd -m -d /var/lib/jenkins -s /bin/bash jenkins
vlab@HYVLAB8:~$ getent passwd jenkins
jenkins:x:1001:1001::/var/lib/jenkins:/bin/bash
vlab@HYVLAB8:~$ sudo chown -R jenkins:jenkins /home/vara/jenkins-agent1/
vlab@HYVLAB8:~$ ls -ld /home/vara/jenkins-agent1/
drwxr-xr-x 2 jenkins 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 AAAAB3NzaC1yc2EAAAADAQABAAABgQCaTwR/qJnLfE0jK9AsGVTZTNYiouX7cP8sqpNS25cYlsEHfI9
+PIRb/MU2ADUuFHEnBexSd88AXPi4TFFJTPjOHY3ES/7rOL7uDwX4d2lExm8f/3N7zcpzqeUbfq2ZR2u1lQsWnFXhsX/v1SR3c7hG
VDloOW1Bwr0HDCs5dMRjUS2gqWpjH4/IH0lD4HAM5d2VGoYMLn+NJ6vEvYf2xqDexaEz9SsDG2jsCKhVgeuezq9NbENXdolc6wrdK
Mvbl89cgfSdPH1uoX6LZtA29DIDJoXaI4okuYTrVq/hhZCmLsr44yLotWBx8xTENSE3TAx8sksd92miRLSxaofWrDFIWAFsVLLqBI
1Ly/6gvTjY5W7GNo00Y2iytq9/jsAv7vQ8aGzm3UUXsMBel2j1LIZkYIRLgNVnxPD/1JQCX83W3VEmjTwK9mrVI/YrW+k6X+VX7cS
RscpcWiSqeuUoD0Wdg1g5eOiiapp8ycNk3Aqfj3xerdckZh4YtpnEDXywLtc=
172.16.203.40 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJzyz7FZxdtvVMM
+xgs7bFsKZOgDWO8Rb/9MANONMJC6Swj0VO23kqvJ0ZZi/DLgf8Zkem9TE5R7pJzIoeIEWuI=
172.16.203.40 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBmRCp1OwJs+c7/QfGQpSESX3djXEfoRKBZz9BKq2kEe
```

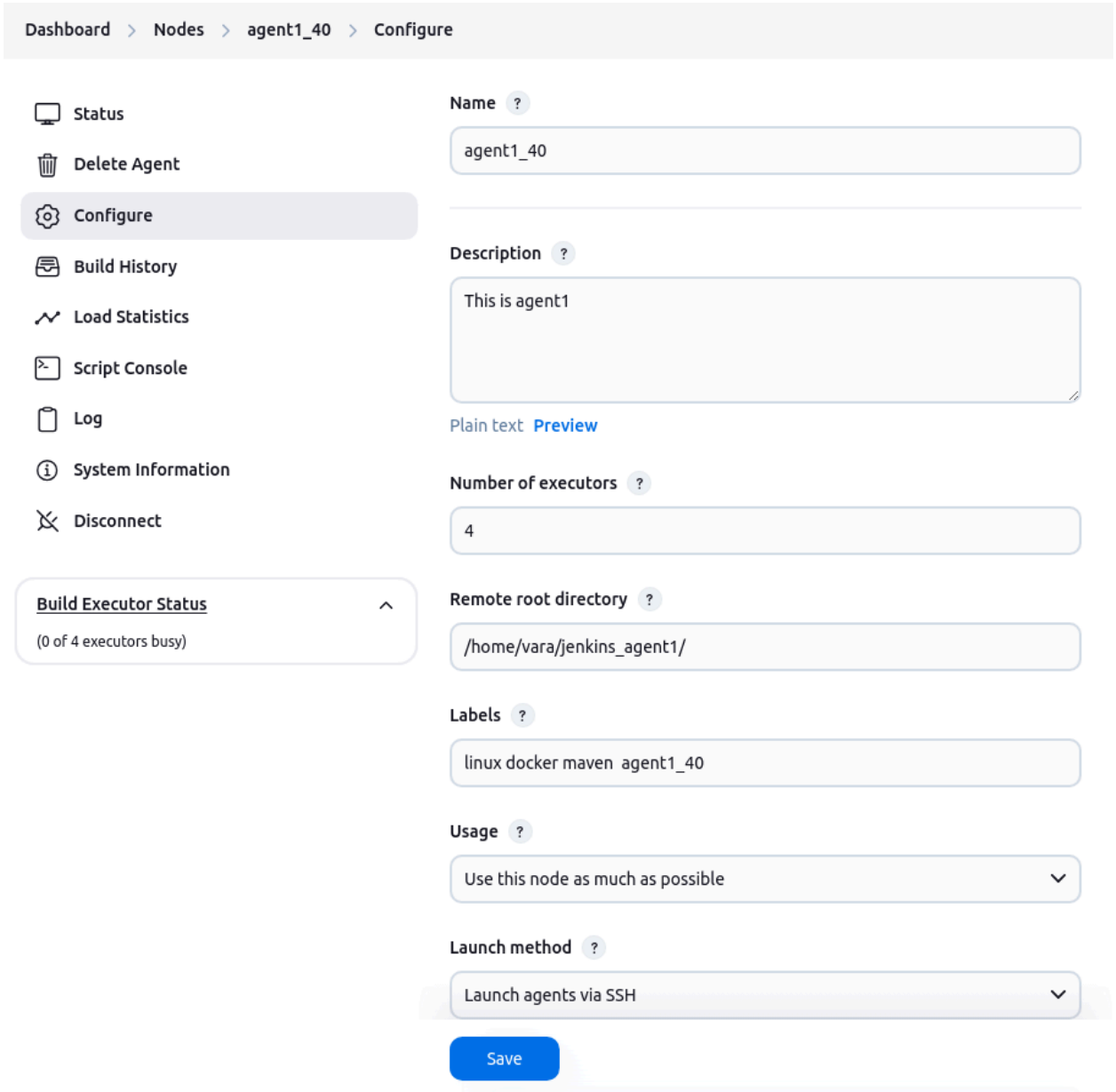
```
vlab@HYVLAB7:/var/lib/jenkins$ sudo ssh-keygen -R 172.16.203.40 -f .ssh/known_hosts
# Host 172.16.203.40 found: line 1
# Host 172.16.203.40 found: line 2
# Host 172.16.203.40 found: line 3
.ssh/known_hosts updated.
Original contents retained as .ssh/known_hosts.old
```

3.4.2.4.2. Set up

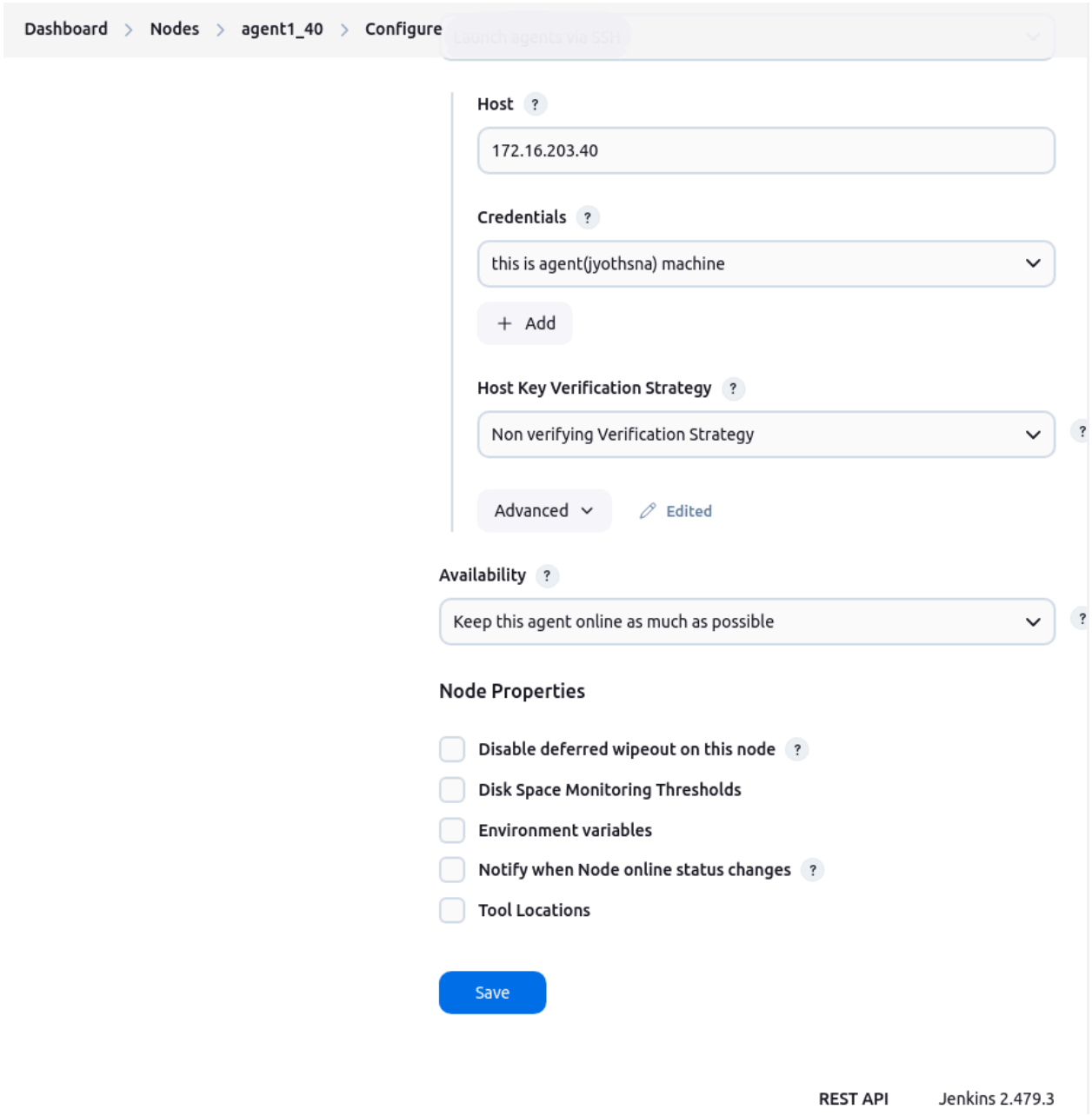
- Click on new node and enter the node name



- **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.



- Host : this should be the IP address or hostname of the agent machine.
- Jenkins requires **SSH credentials** to connect the agent machine.
- Check the Know 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**



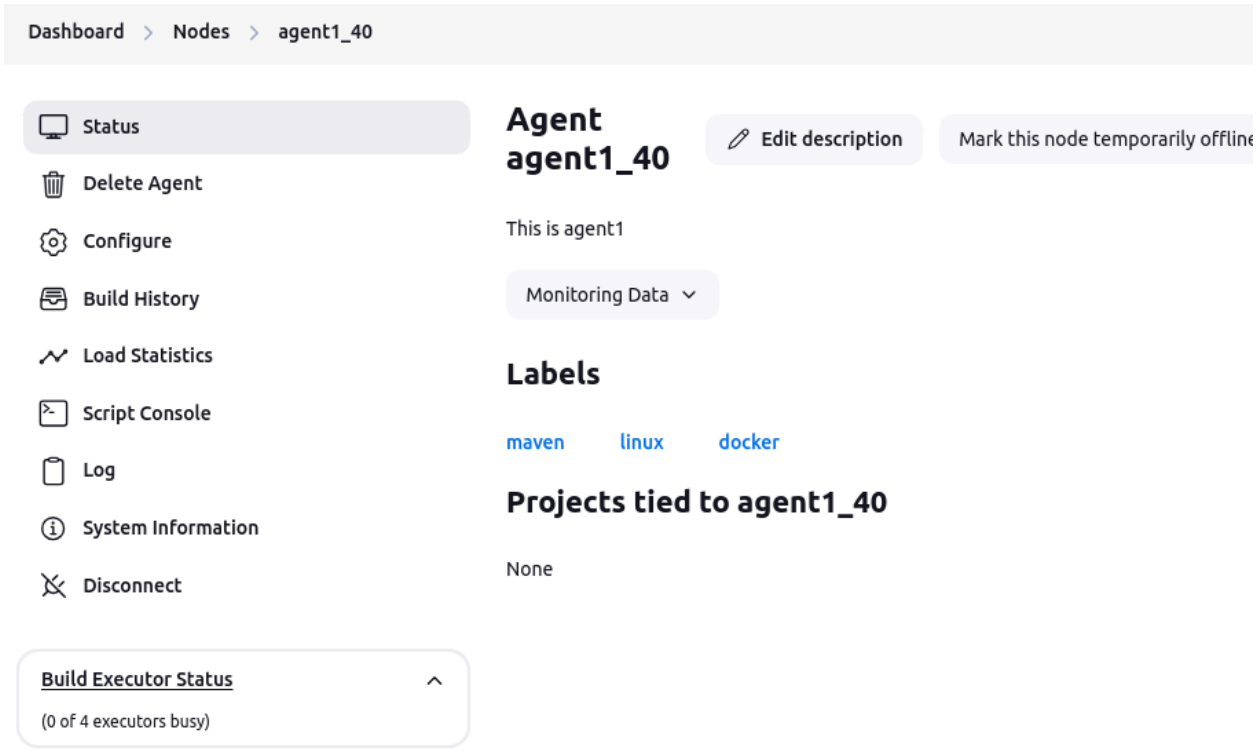
- **Host Key Verification Strategy :**
  - **Non verifying verification strategy:**
    - Disables host key verification and is less secure because it does not validate the server's identity.
  - **Known Hosts File Verification Strategy:**
    - Uses a known_hosts file (typically located in the Jenkins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

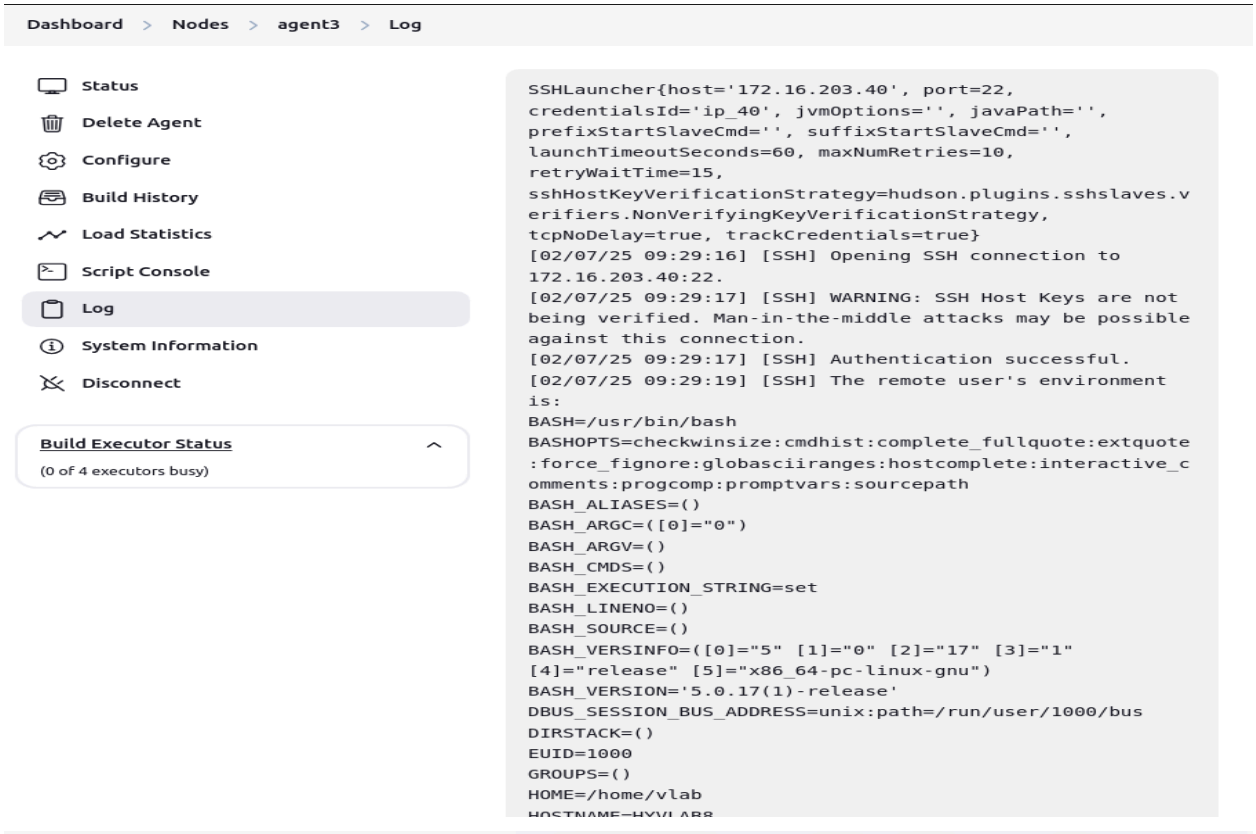
```
vlab@HYVLAB7:/var/lib/jenkins$  ssh-keyscan 172.16.203.40 | sudo tee -a .ssh/known_hosts
# 172.16.203.40:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.11
# 172.16.203.40:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.11
# 172.16.203.40:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.11
# 172.16.203.40:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.11
# 172.16.203.40:22 SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.11
172.16.203.40 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCaTwR/qJnLfE0jK9AsGVTZTNYiouX7cP8sqpNS25cYlsEHfI9
+PIRb/MU2ADUuFHEnBexSd88AXPi4TFFJTPjOHY3ES/7rOL7uDwX4d2lExm8f/3N7zcpzqeUbfq2ZR2u1lQsWnFXhsX/v1SR3c7hG
VDloOW1Bwr0HDCs5dMRjUS2gqWpjH4/IH0lD4HAM5d2VGoYMLn+NJ6vEvYf2xqDexaEz9SsDG2jsCKhVgeuezq9NbENXdolc6wrdK
Mvbl89cgfSdPH1uoX6LZtA29DIDJoXaI4okuYTrVq/hhZCmLsr44yLotWBx8xTENSE3TAx8sksd92miRLSxaofWrDFIWAFsVLLqBI
1Ly/6gvTjY5W7GNo00Y2iytq9/jsAv7vQ8aGzm3UUXsMBel2j1LIZkYIRLgNVnxPD/1JQCX83W3VEmjTwK9mrVI/YrW+k6X+VX7cS
RscpcWiSqeuUoD0Wdg1g5eOiiapp8ycNk3Aqfj3xerdckZh4YtpnEDXywLtc=
172.16.203.40 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJzyz7FZxdtvVMM
+xgs7bFsKZOgDWO8Rb/9MANONMJC6Swj0VO23kqvJ0ZZi/DLgf8Zkem9TE5R7pJzIoeIEWuI=
172.16.203.40 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBmRCp1OwJs+c7/QfGQpSESX3djXEfoRKBZz9BKq2kEe
```

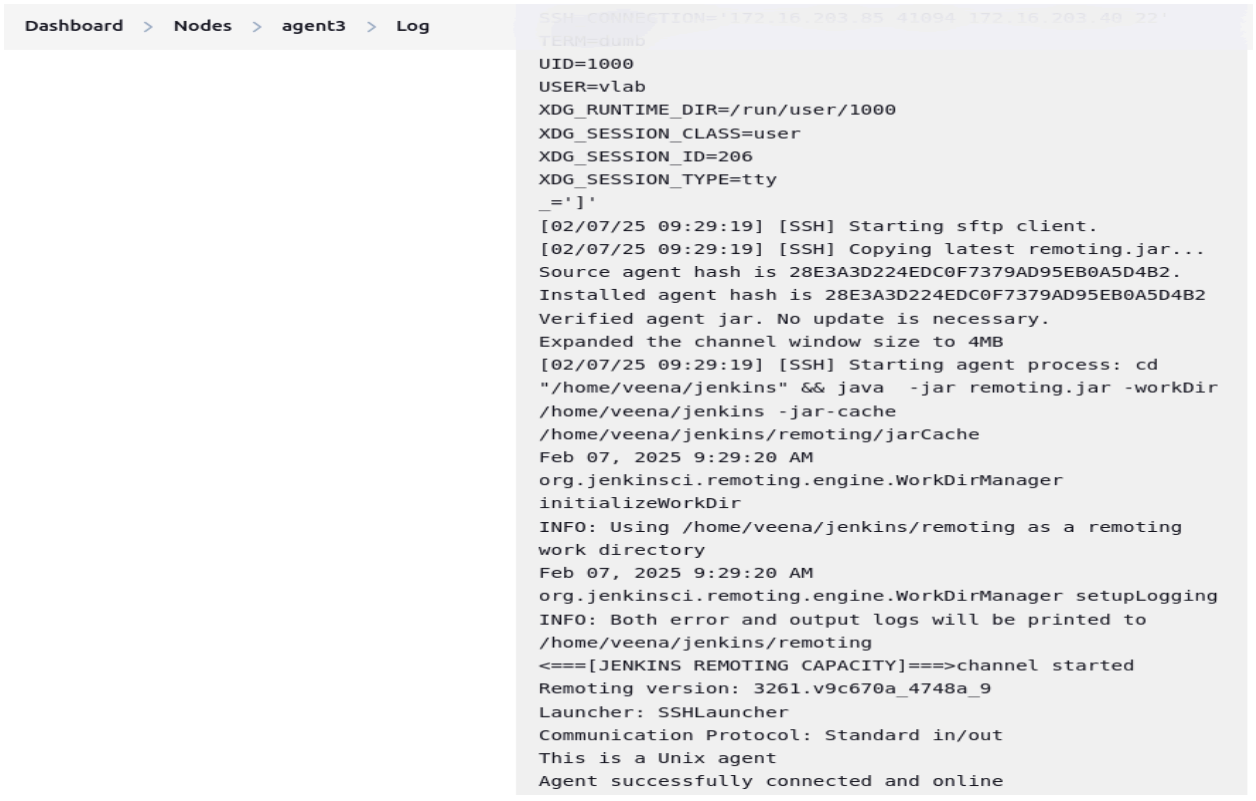
- When the agent connects, Jenkins compares the host key provided by the agent with the entries in the known_hosts file. If there's a match, the connection proceeds; if not, it fails.
- Requires maintenance of the known_hosts file, and if the remote host's key changes (for example, due to a server rebuild),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

Check if the system is online or offline at status:

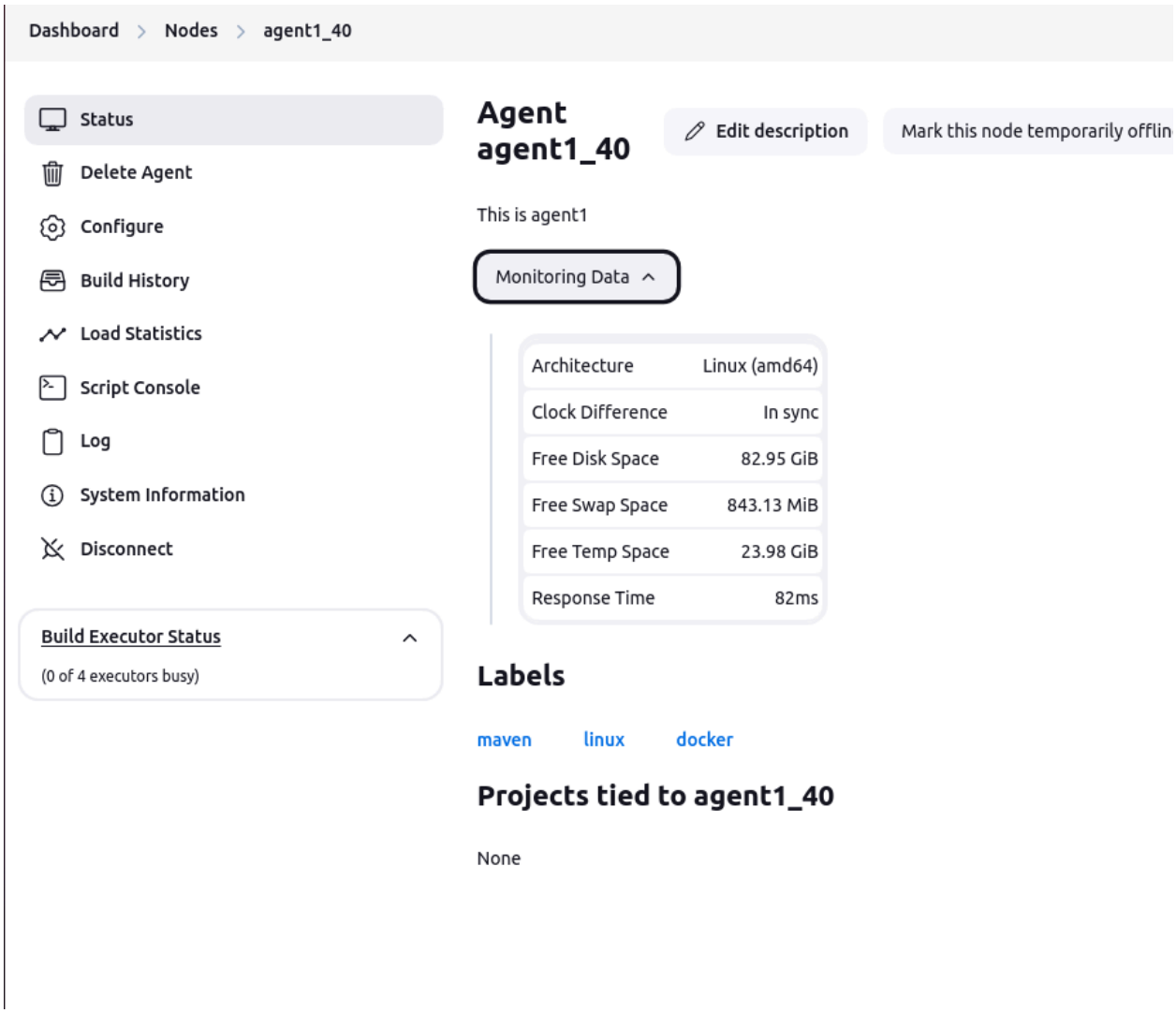


3.4.2.4.3. Log


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.v
erifiers.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_c
omments: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

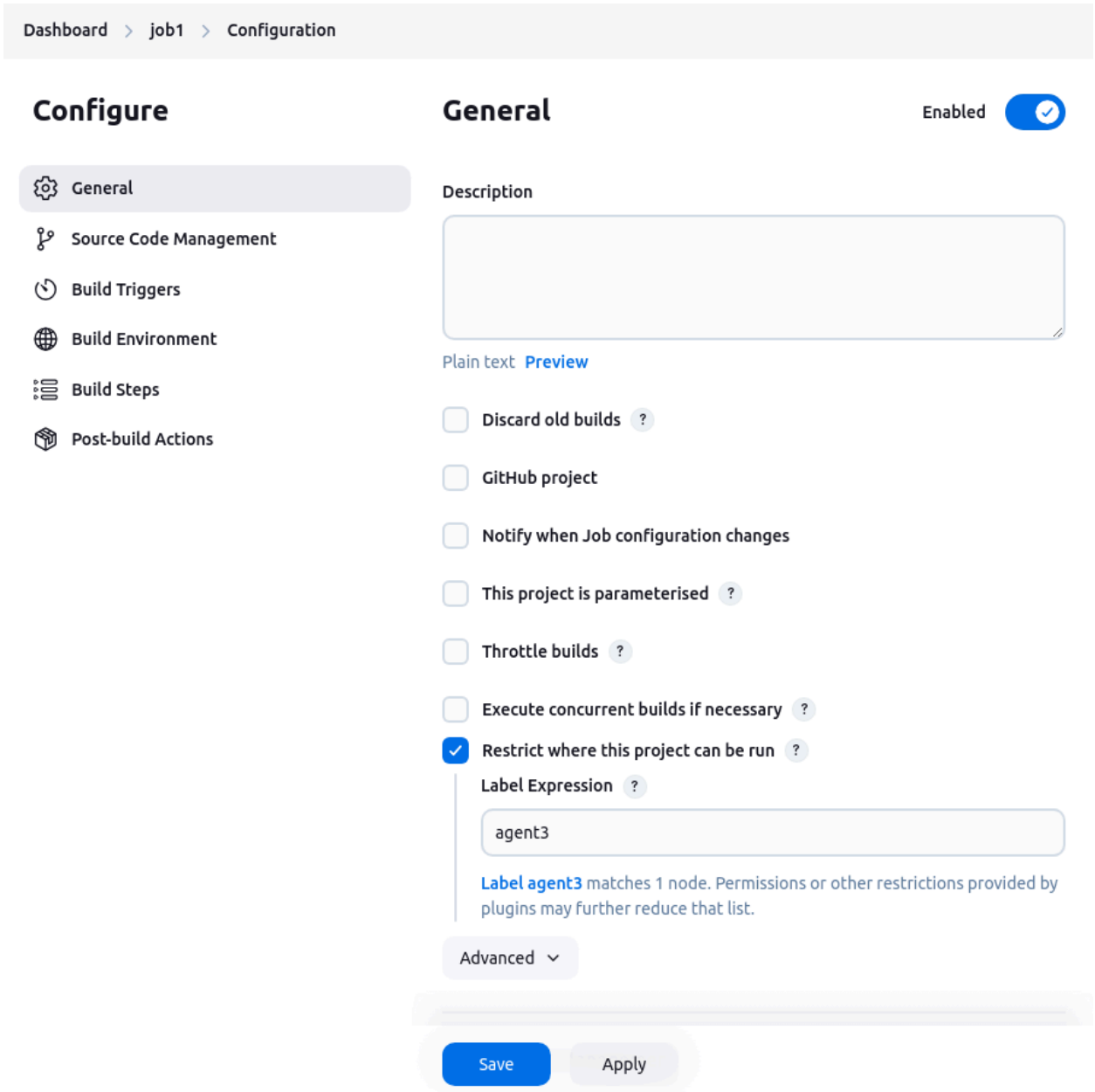

Dashboard > Nodes > agent3 > Log
SSH_CONNECTION='172.16.203.85 41094 172.16.203.40 22'
TERM=dumb
UID=1000
USER=vlab
XDG_RUNTIME_DIR=/run/user/1000
XDG_SESSION_CLASS=user
XDG_SESSION_ID=206
XDG_SESSION_TYPE=tty
_=']'
[02/07/25 09:29:19] [SSH] Starting sftp client.
[02/07/25 09:29:19] [SSH] Copying latest 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

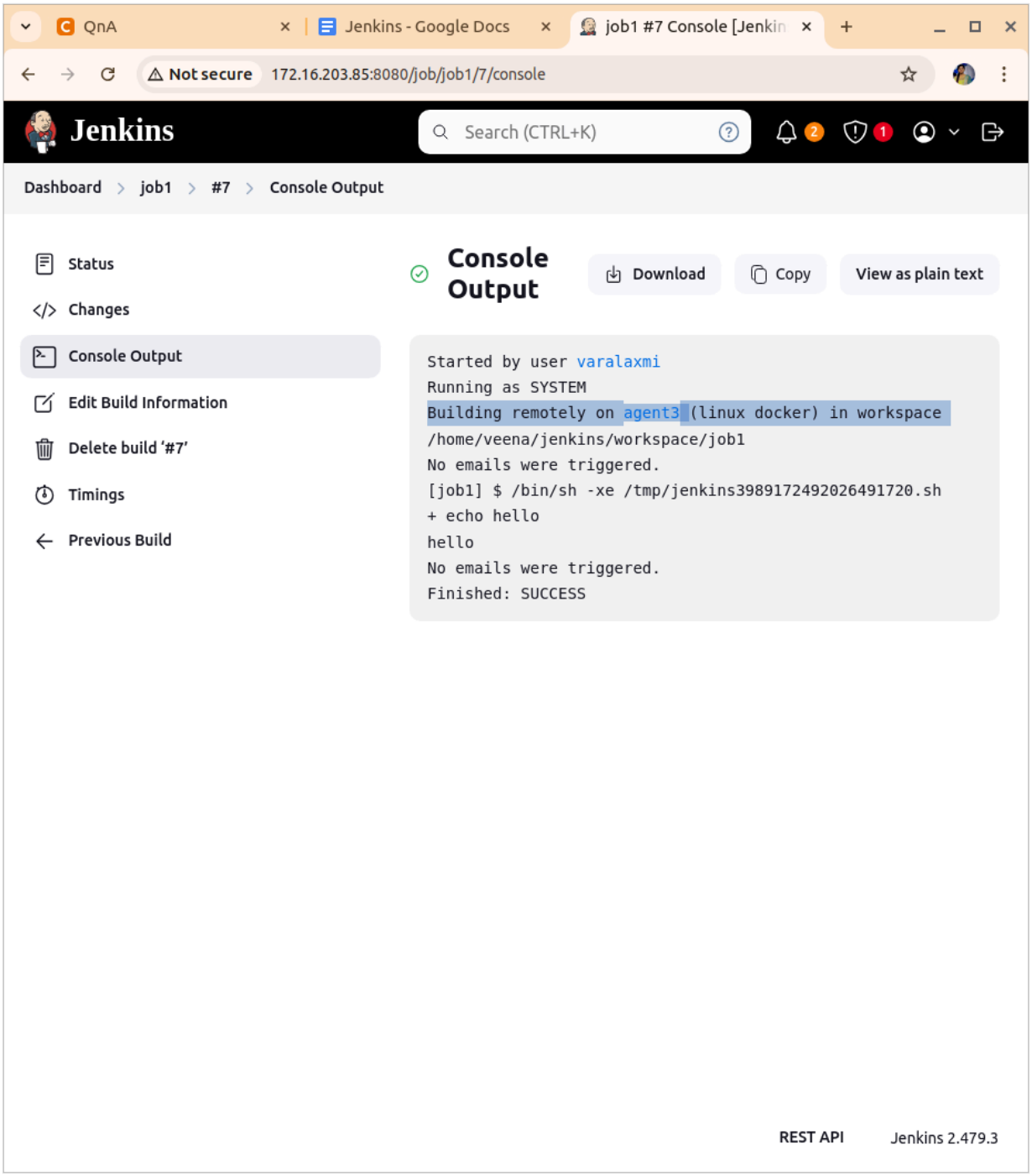


3.4.2.5. Run a job on a specific node in jenkins

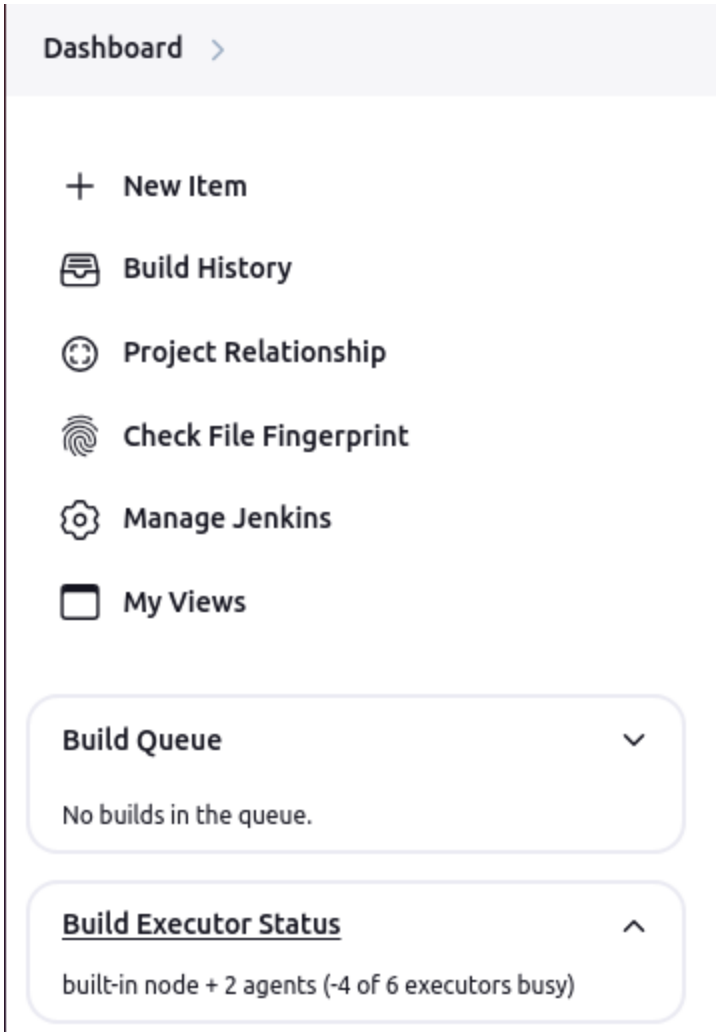
- From the Jenkins dashboard, create a new Freestyle project or open an existing one.
- In the job configuration, check the option **"Restrict where this project can be run"**.
- In the field that appears, enter the **label** that you assigned to your node
- Save the configuration.
- Click **"Build Now"**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



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

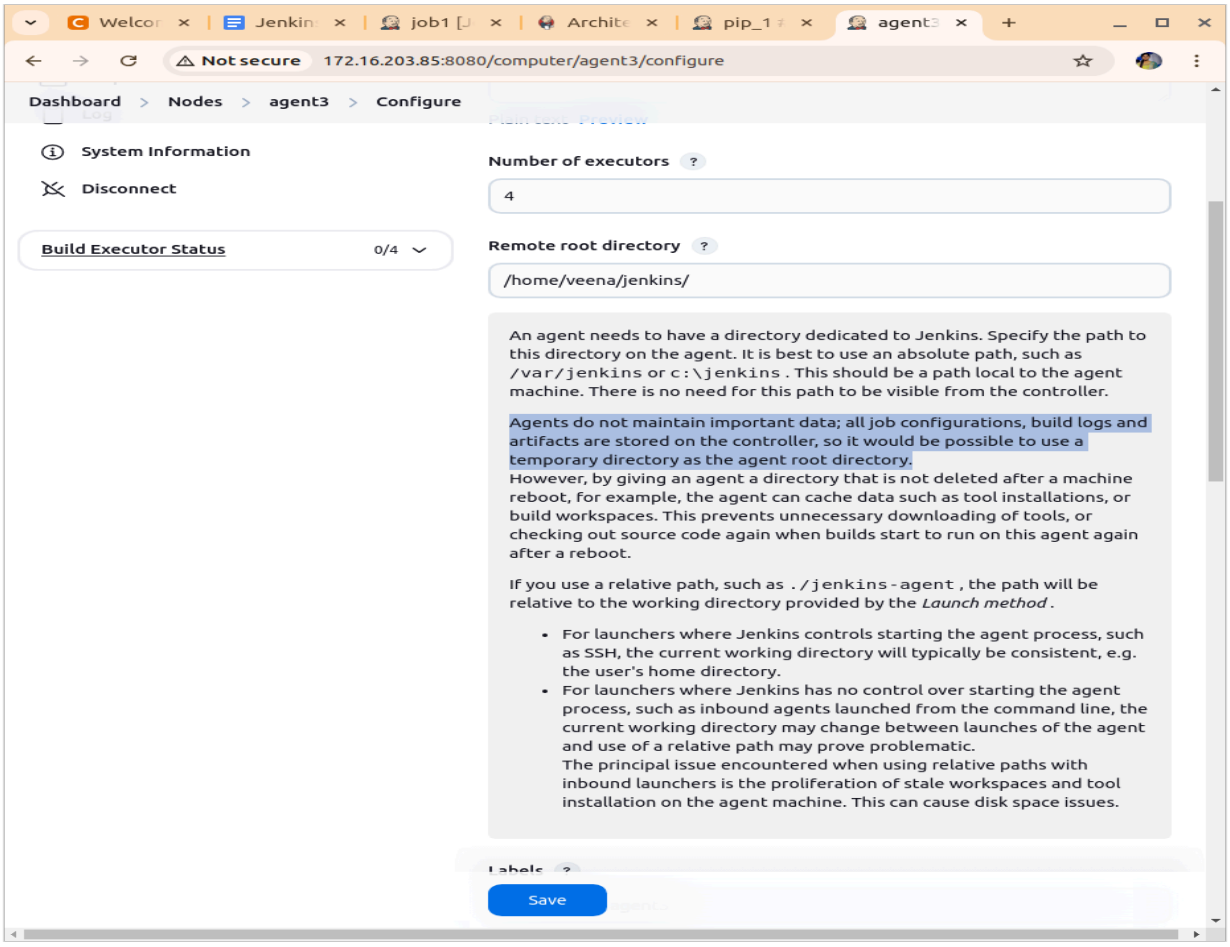


Check Build Executor status:



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.



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

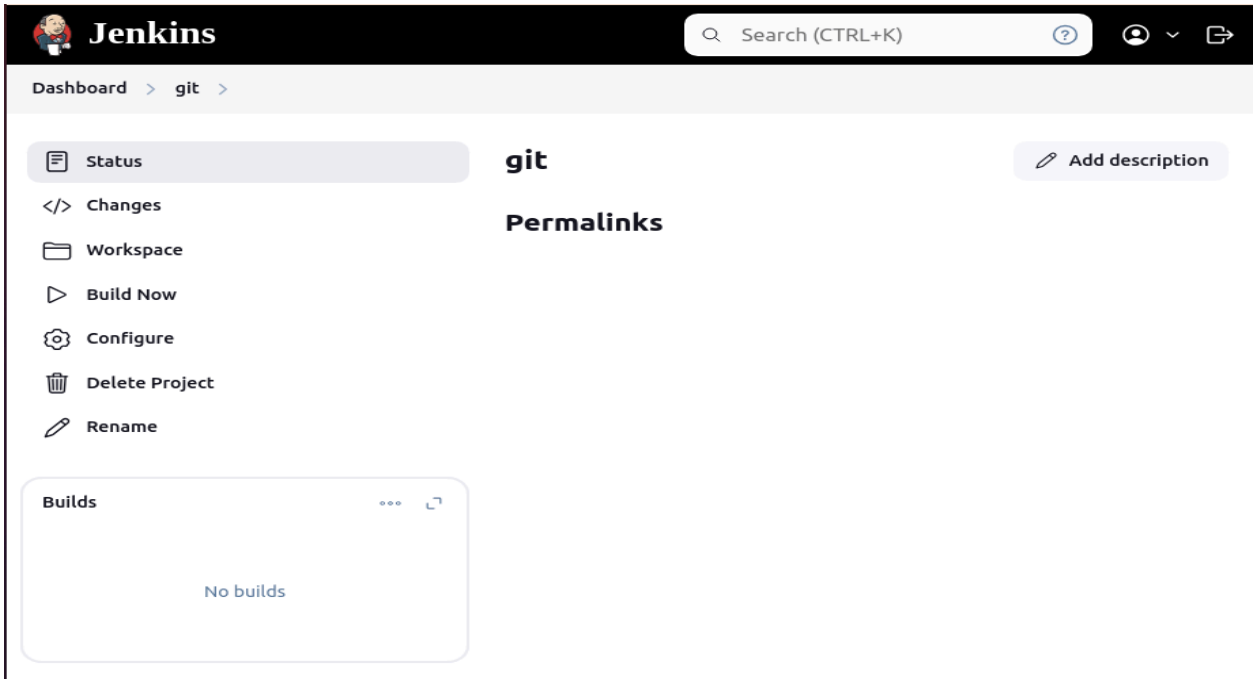
- Management of jobs based on the repositories and branches within a source code management (SCM) organization.
- Automatically create and delete jobs as repositories or branches are added or removed from the SCM system.

4.1.1. Git Polling

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

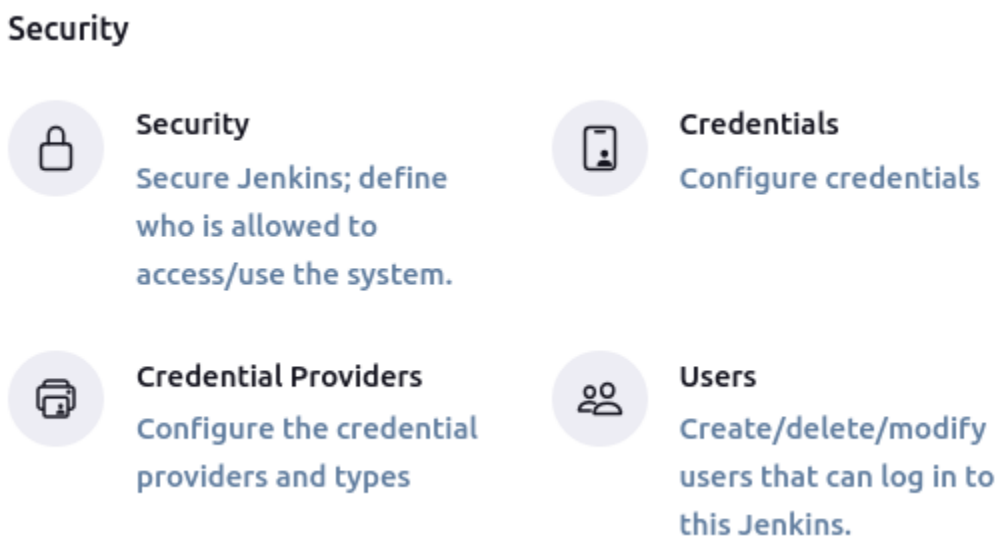
4.1.1.1. Using HTTP

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

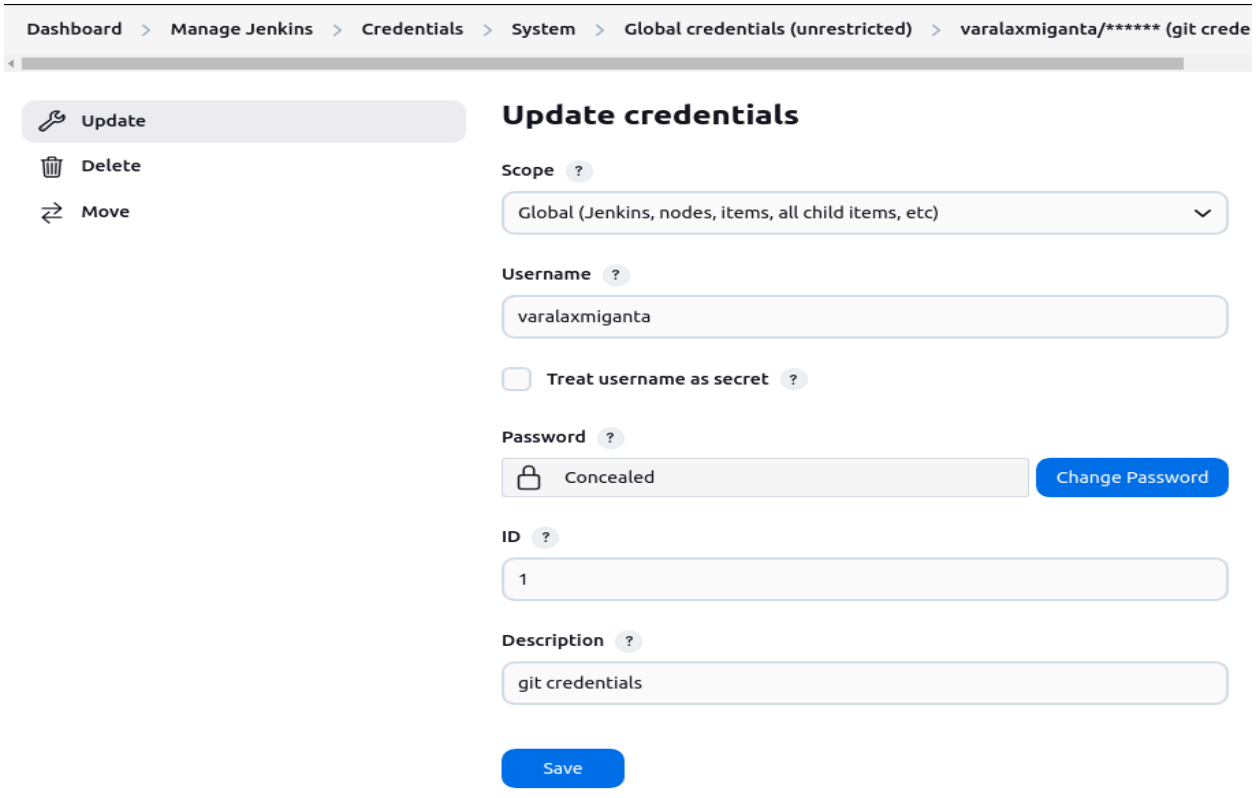


- **Add the Token to Jenkins Credentials:**

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

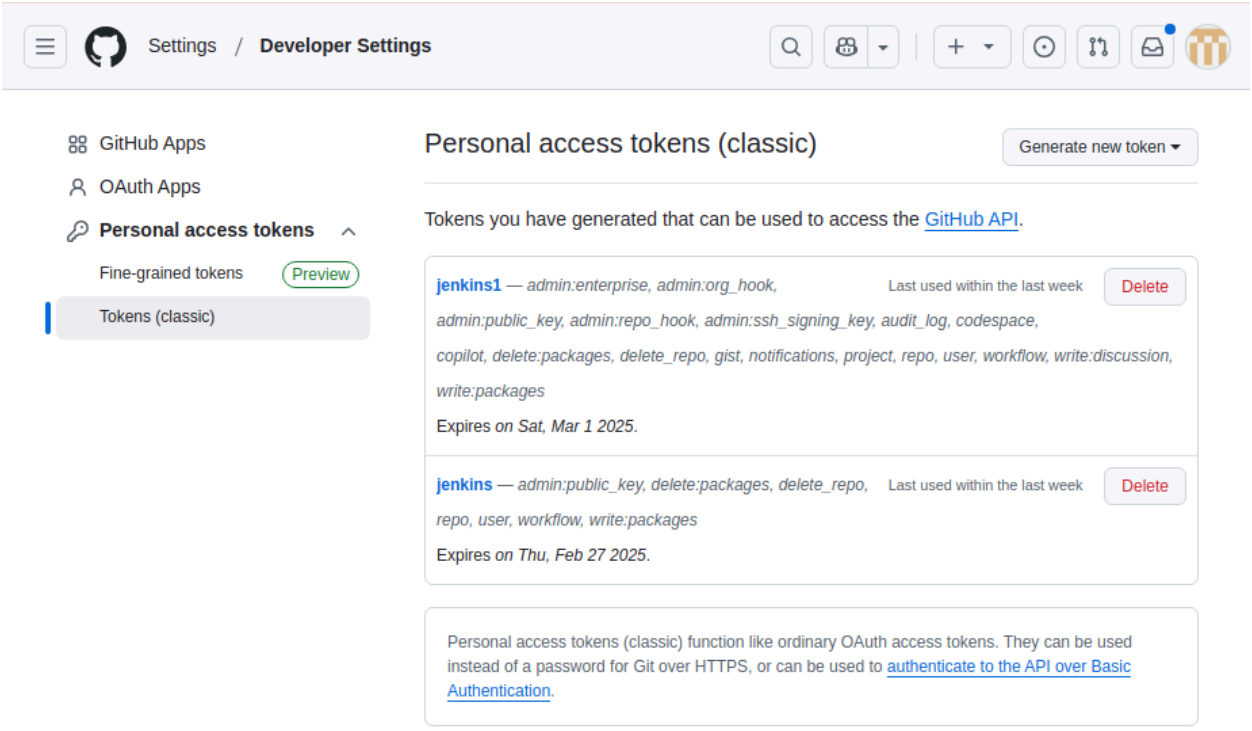


- Set the **Kind** to **Username with password**.

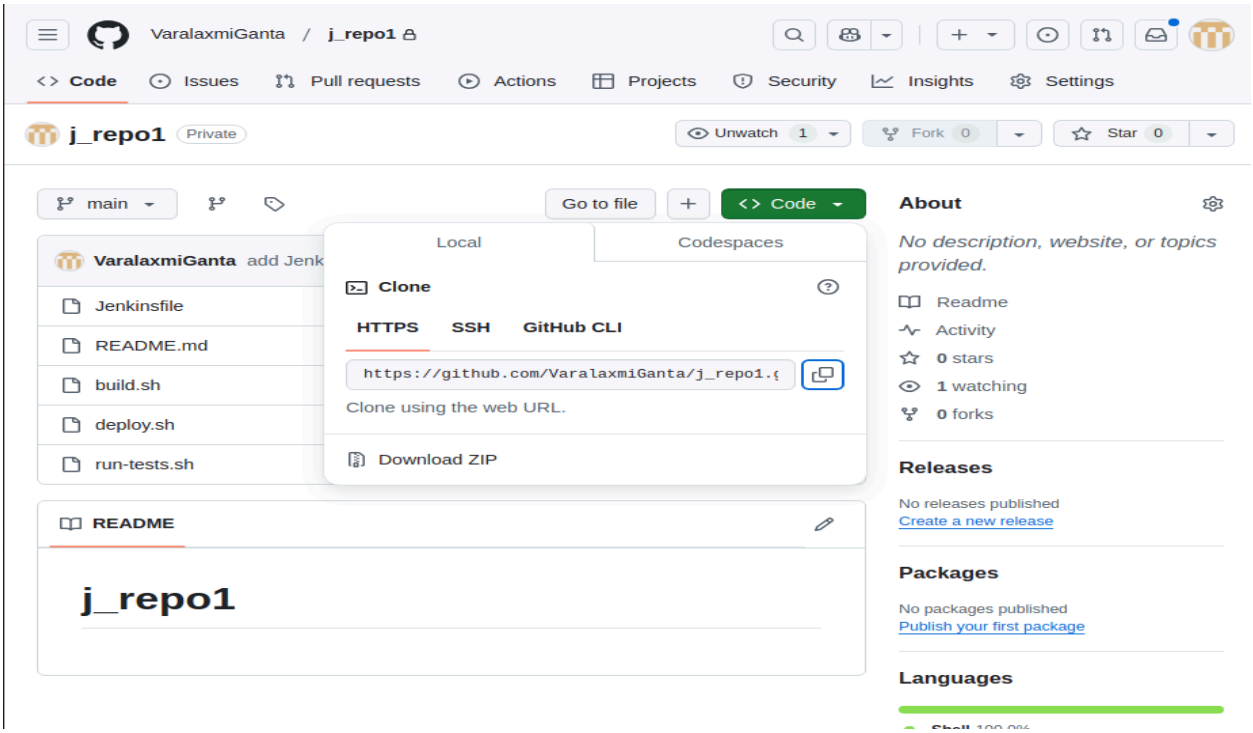


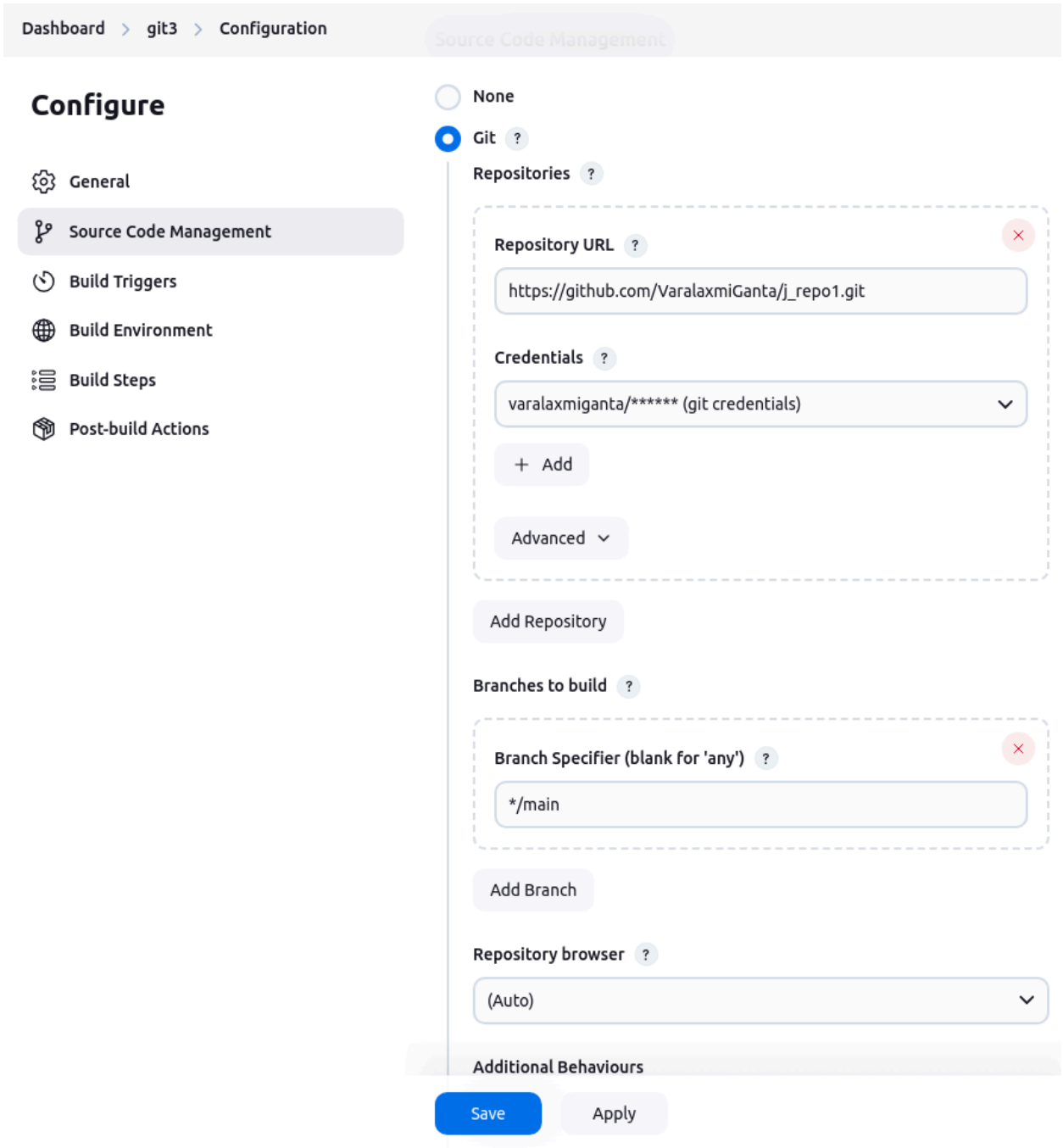
- In **Username**, put your **GitHub username**.

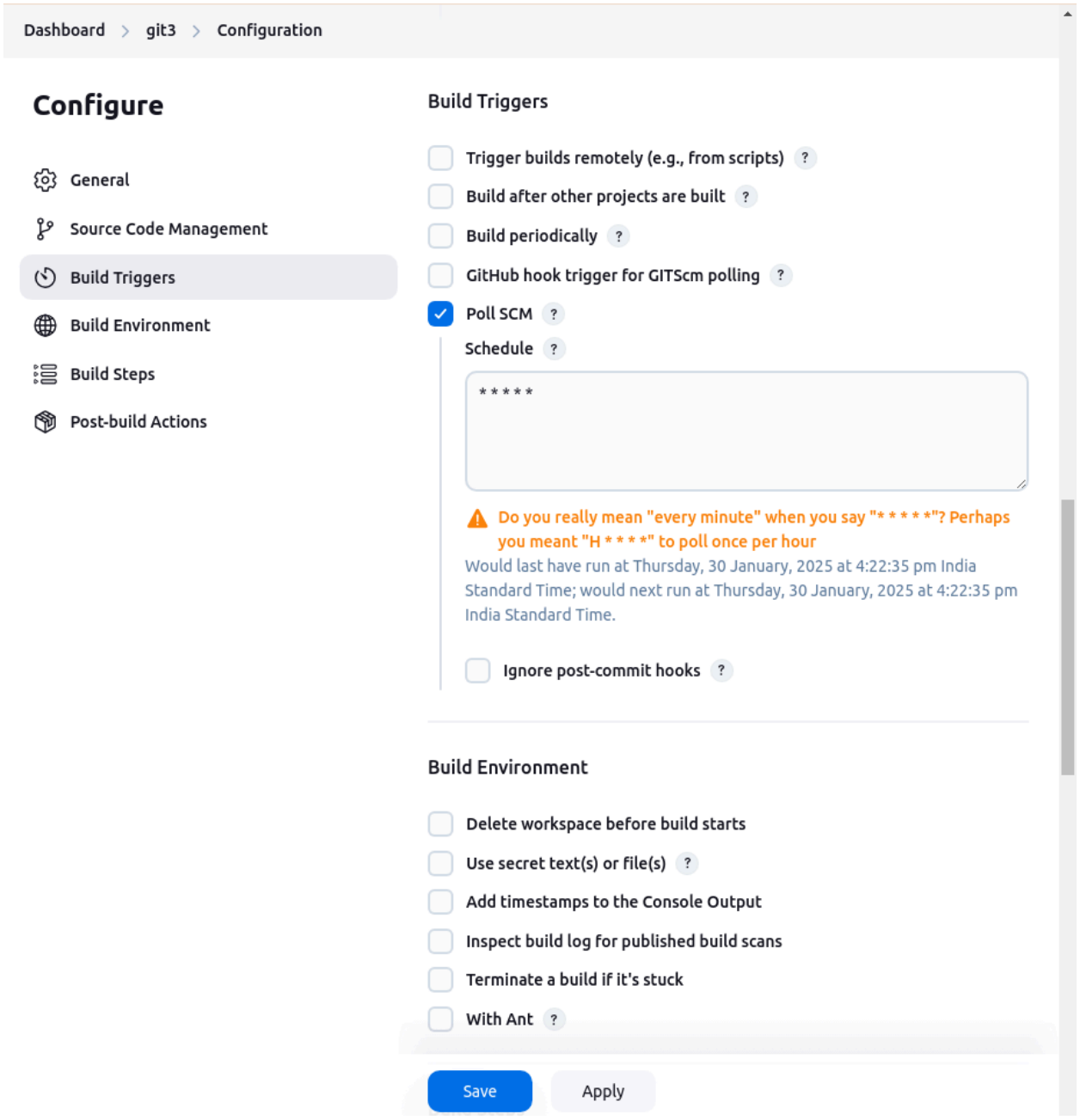
- In **Password**, paste the **Personal Access Token (PAT)** you generated.



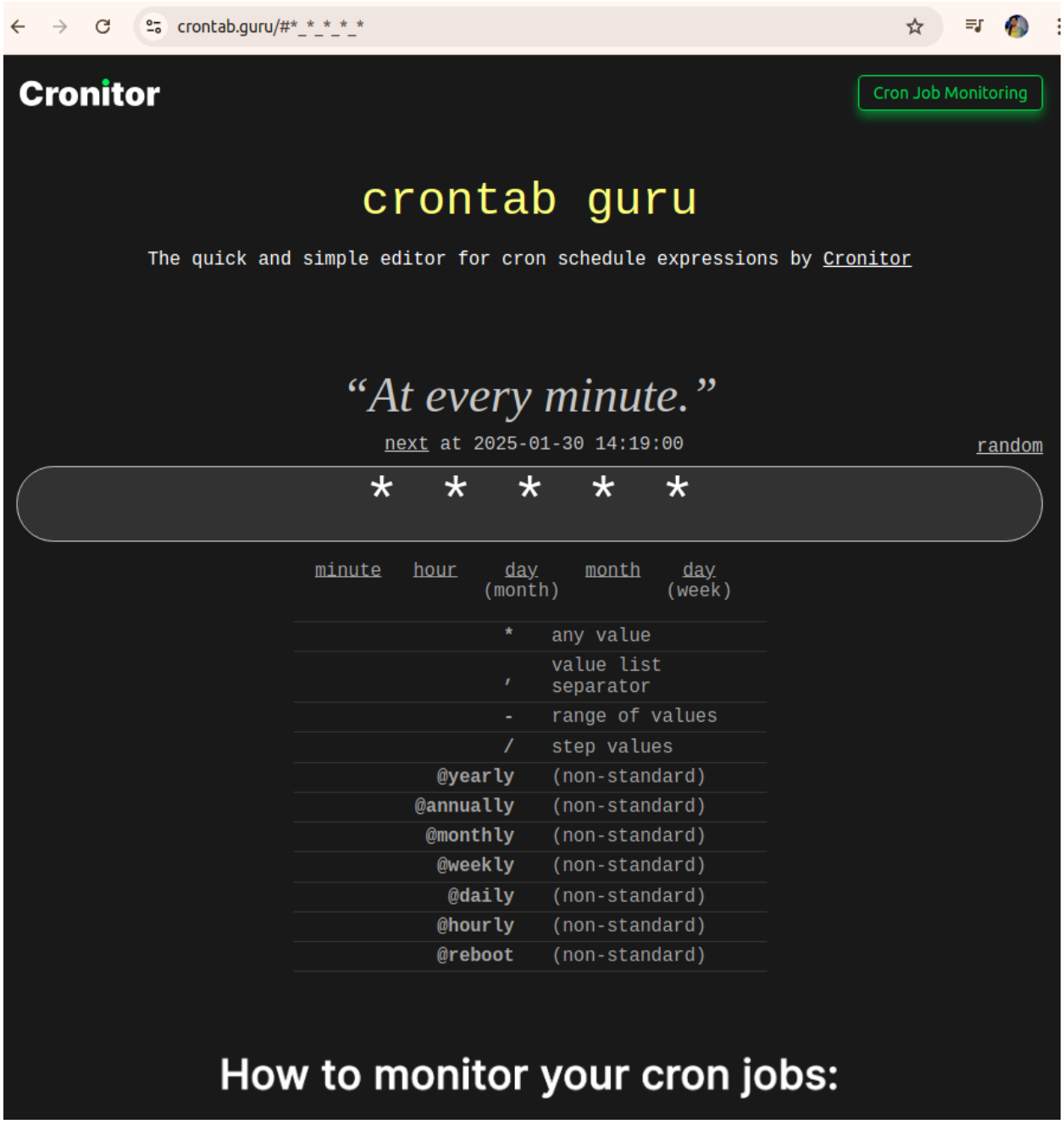
- **Update Git Repository URL in Jenkins Job:**

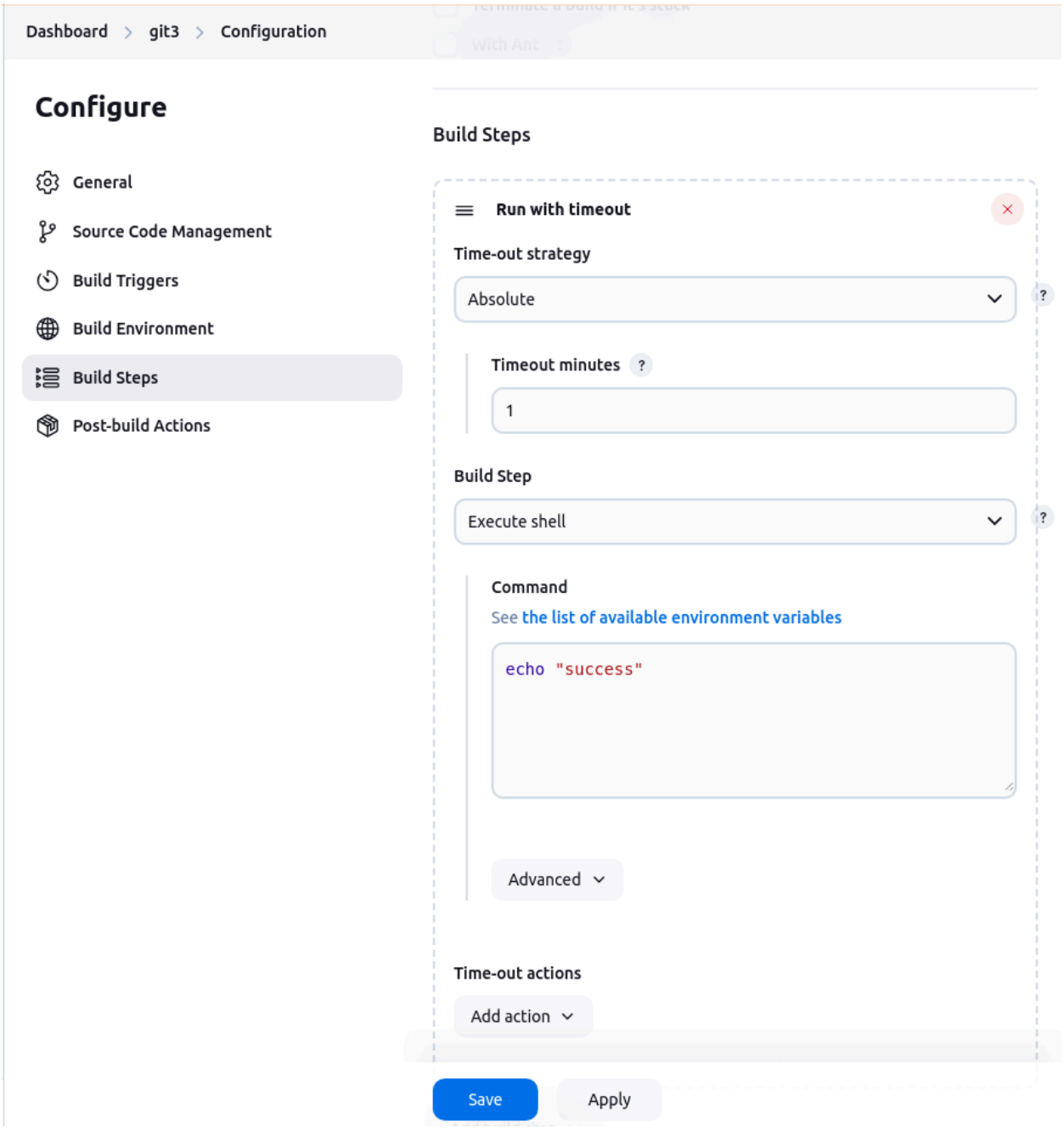



Dashboard
git3
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
None
Git
Repositories
Repository URL
https://github.com/VaralaxmiGanta/j_repo1.git
Credentials
varalaxmiganta/****** (git credentials)
Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/main
Add Branch
Repository browser
(Auto)
Additional Behaviours
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:



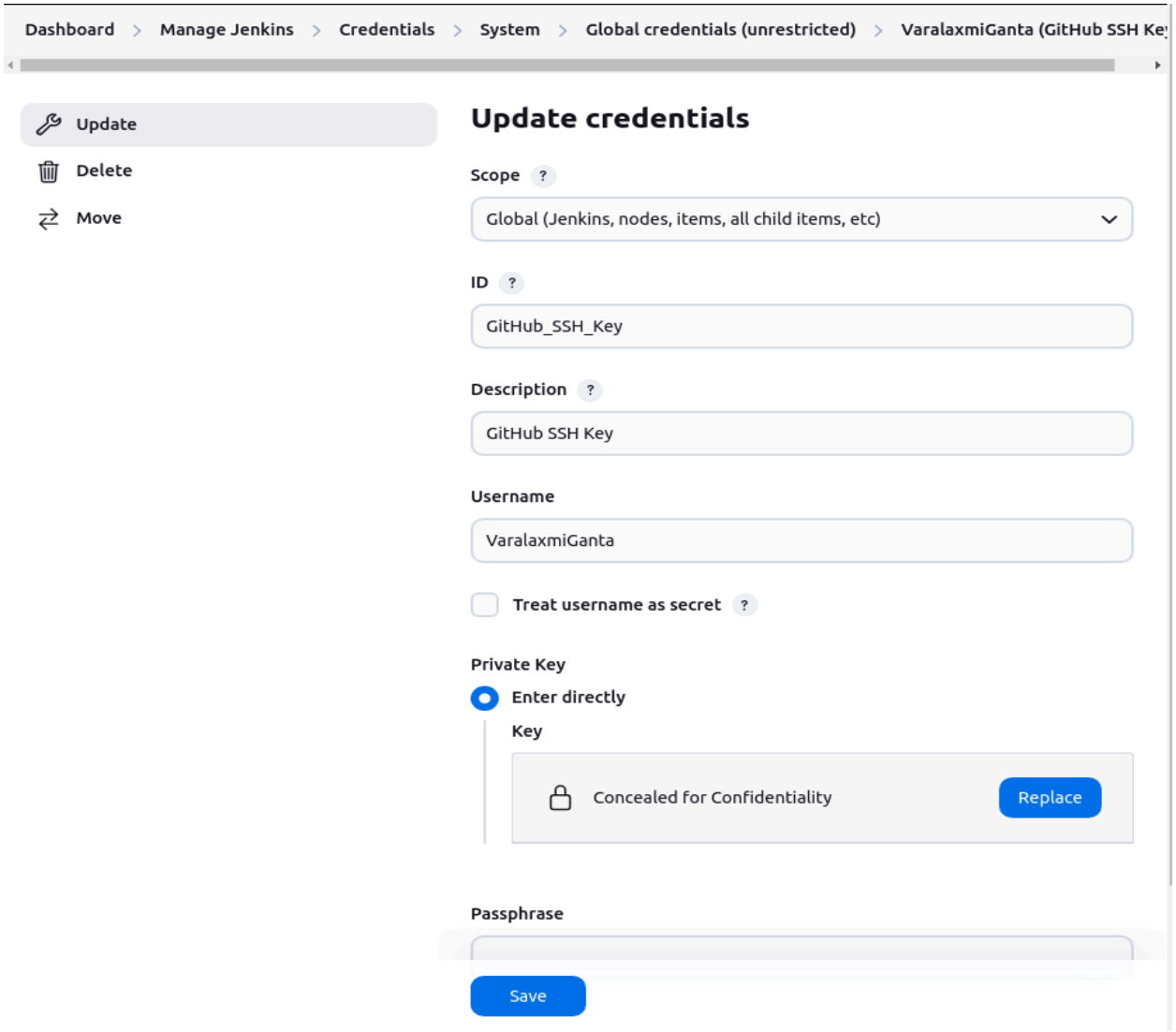
when you change the code in a Git repository, a new commit is created. If you have configured Jenkins to trigger a build upon detecting changes in the Git repository, Jenkins will automatically start a build when it detects this new commit.

4.1.1.2. Using SSH

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

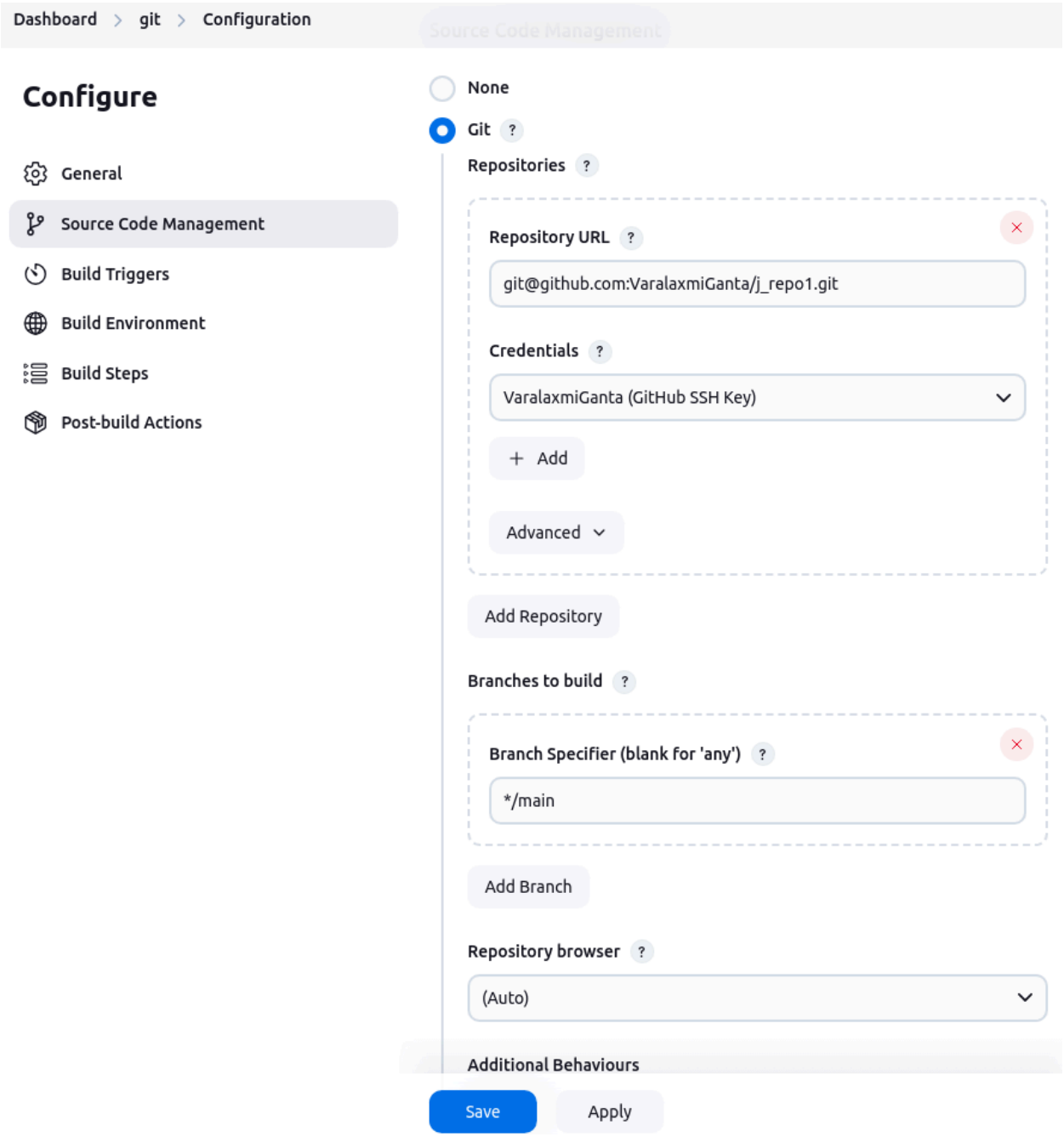
- **Key Pair Created**: After completing the above steps, your SSH key pair will be created:
  - The **private key** will be saved in the file: `~/.ssh/id_rsa`.
  - The **public key** will be saved in the file: `~/.ssh/id_rsa.pub`.
    $ ls .ssh/
    config id_rsa id_rsa.pub known_hosts known_hosts.old veena veena.pub
- Copy the private key and it to the global credentials in jenkins.

- **Test the SSH key:**
  - $ ssh -T git@github.com

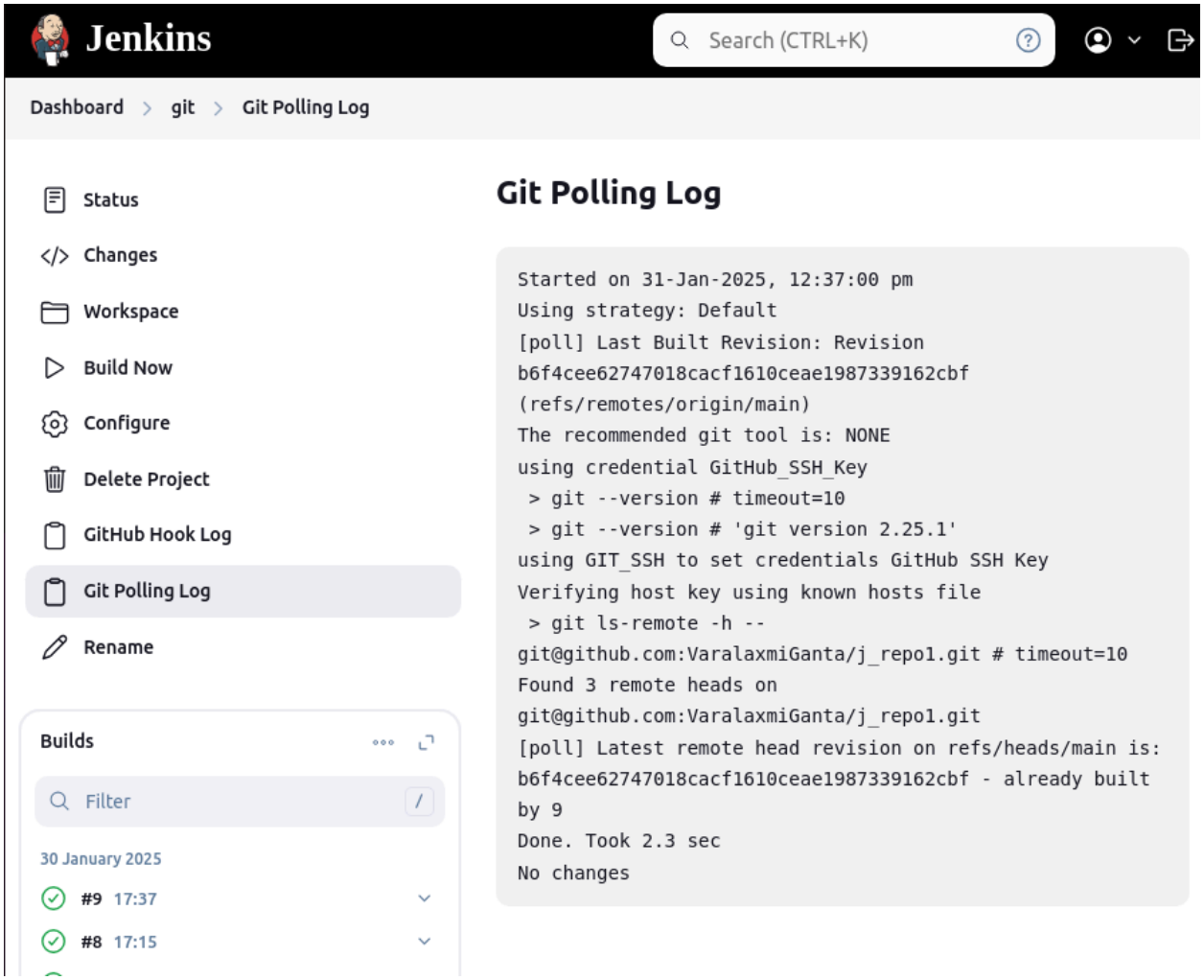


Note : user name must be exact

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



- **Git Polling Log**



4.1.2. Git Webhook

4.1.2.1. Required Plugin (Generic Webhook Trigger Plugin)

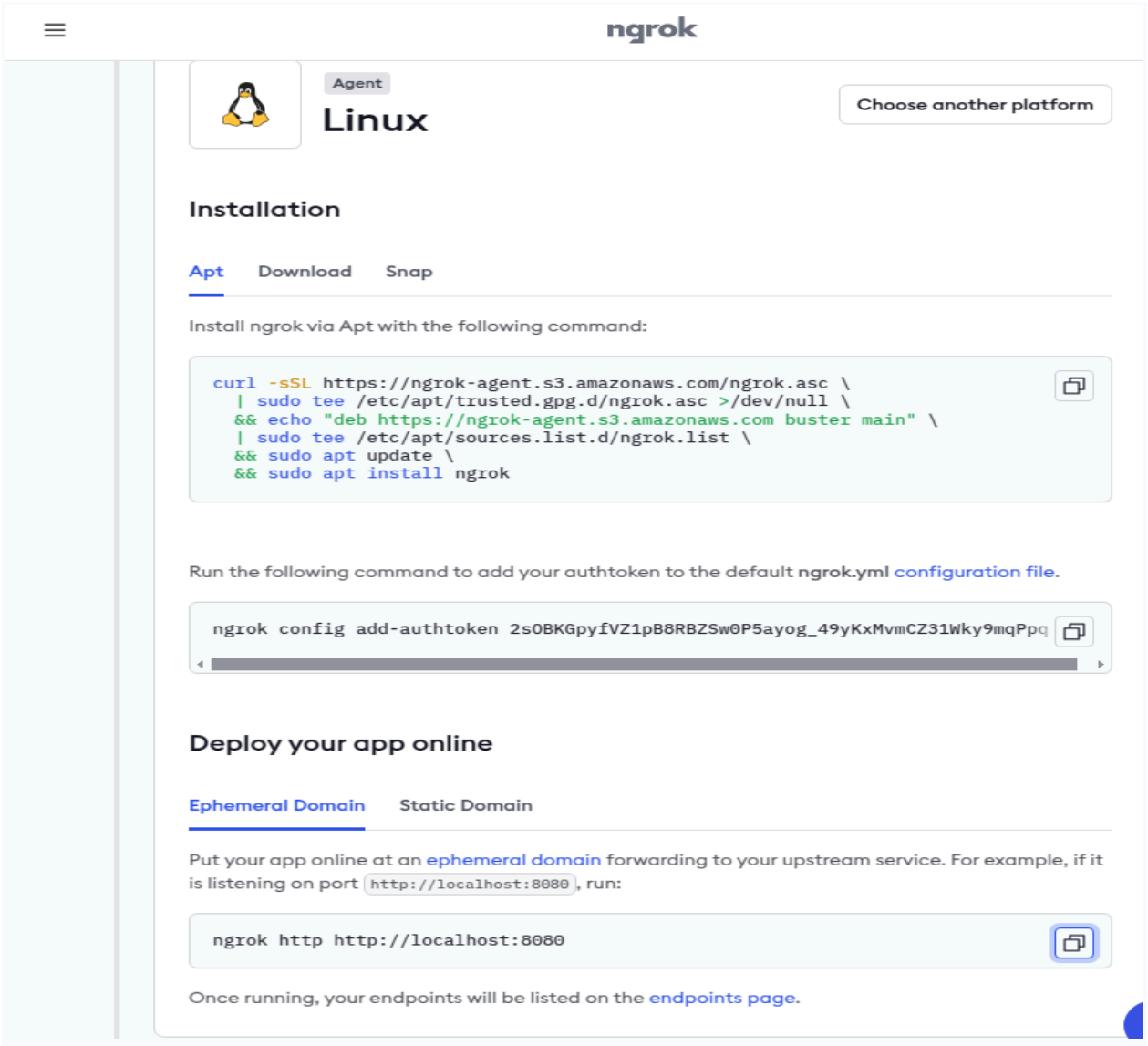
- Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.
  https://plugins.jenkins.io/generic-webhook-trigger

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
```



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

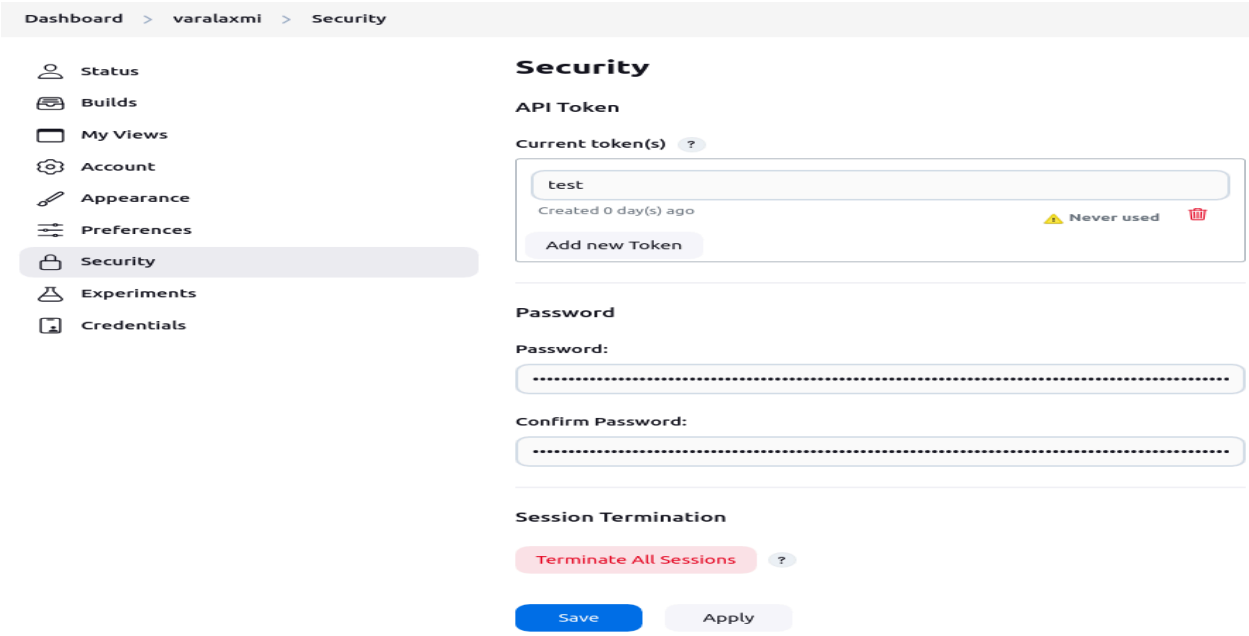
$ sudo ngrok config add-authtoken 2sOBKGpyfVZ1pB8RBZSw0P5ayog_49yKxMvmCZ31Wky9mqPpq

- **Open terminal**

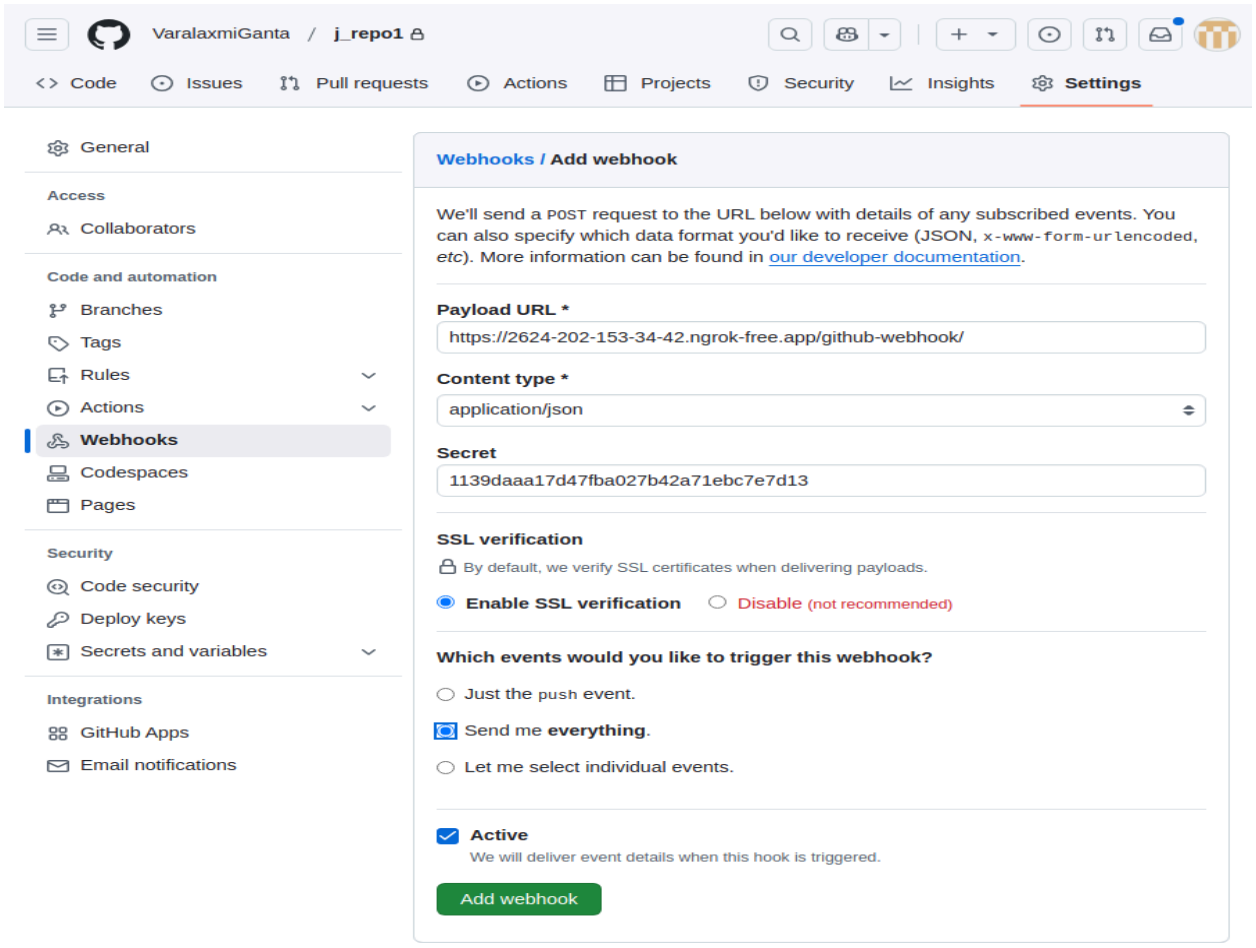
`$ sudo ngrok http http://localhost:8080`

```
ngrok

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
```

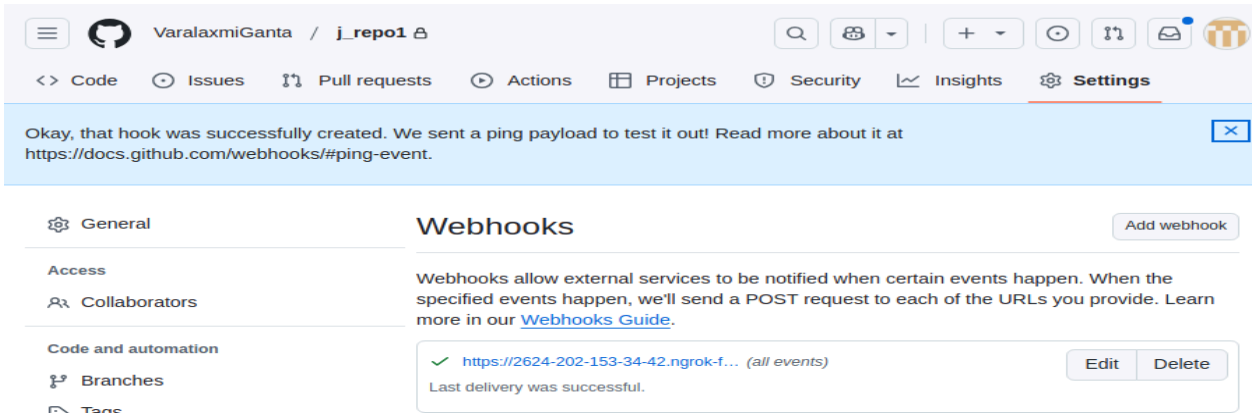
- **In github add webhook for one required repo:**
  - Copy the forwarding url into the payload URL.
  - Select content type is **application/json**
  - Secret key is get from the jenkins:



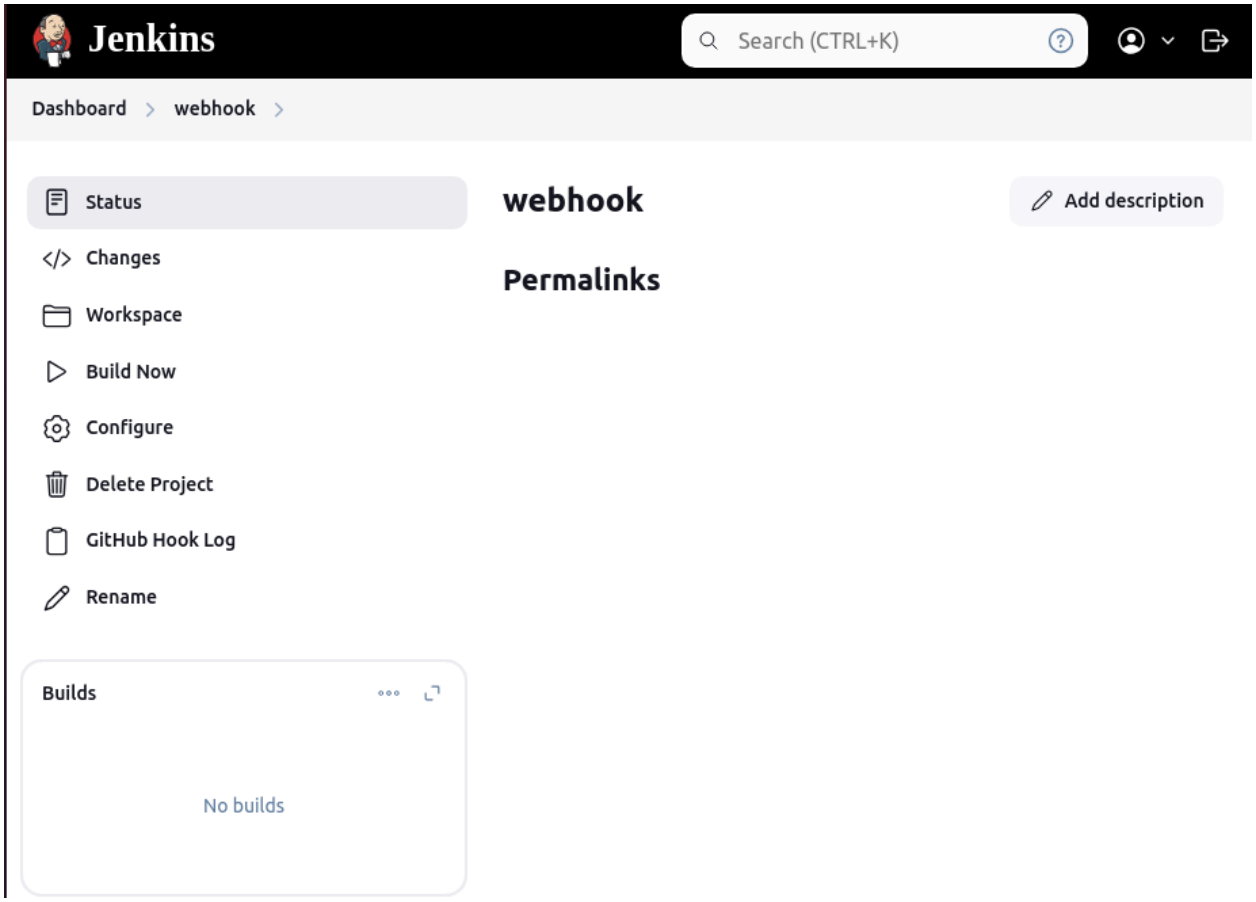
  - Open jenkins dashboard → click admin icon →security
  - In the API Token add new token → first enter name and click on generate token copy it and paste it on the jenkins.



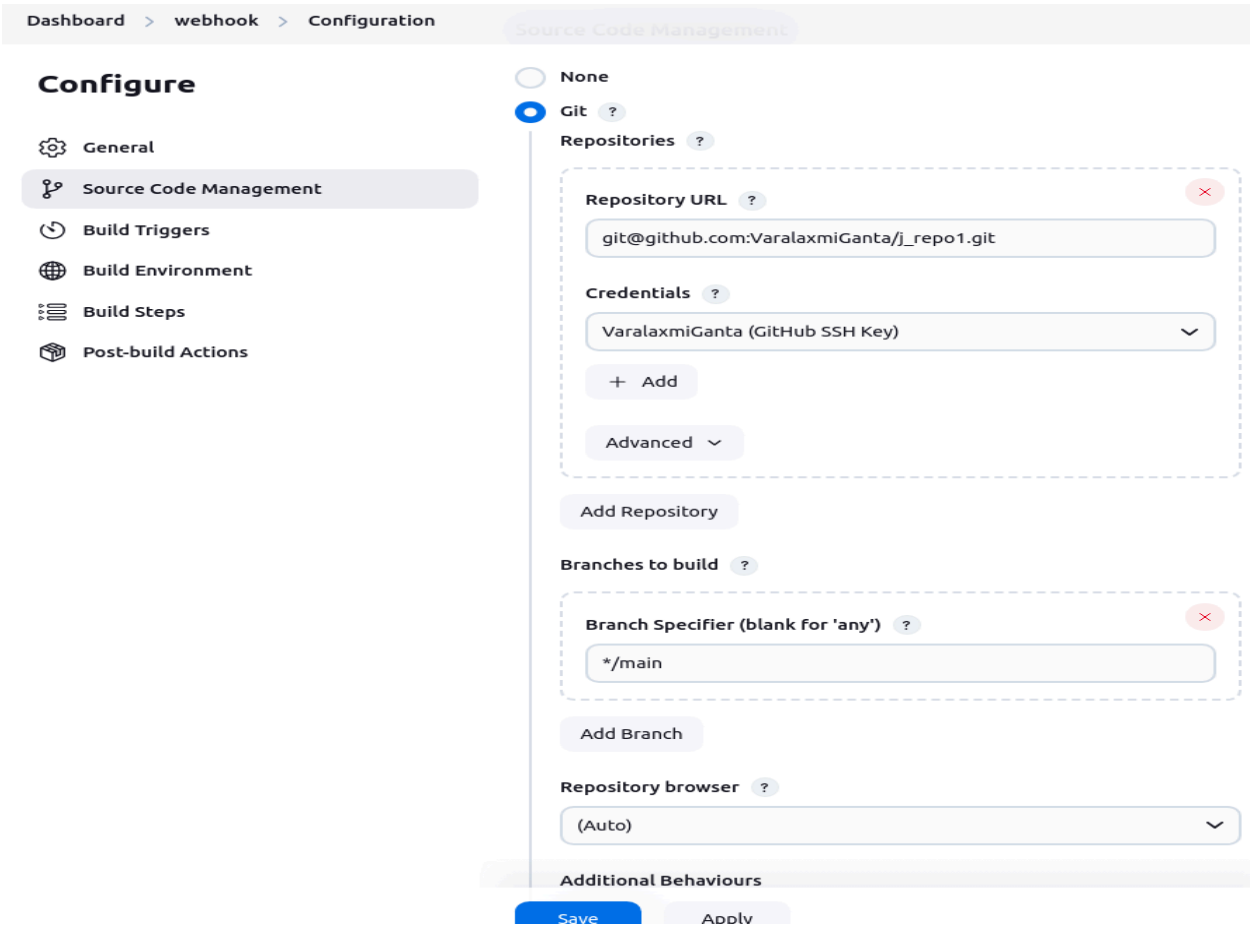
- Add the webhook it looks like this



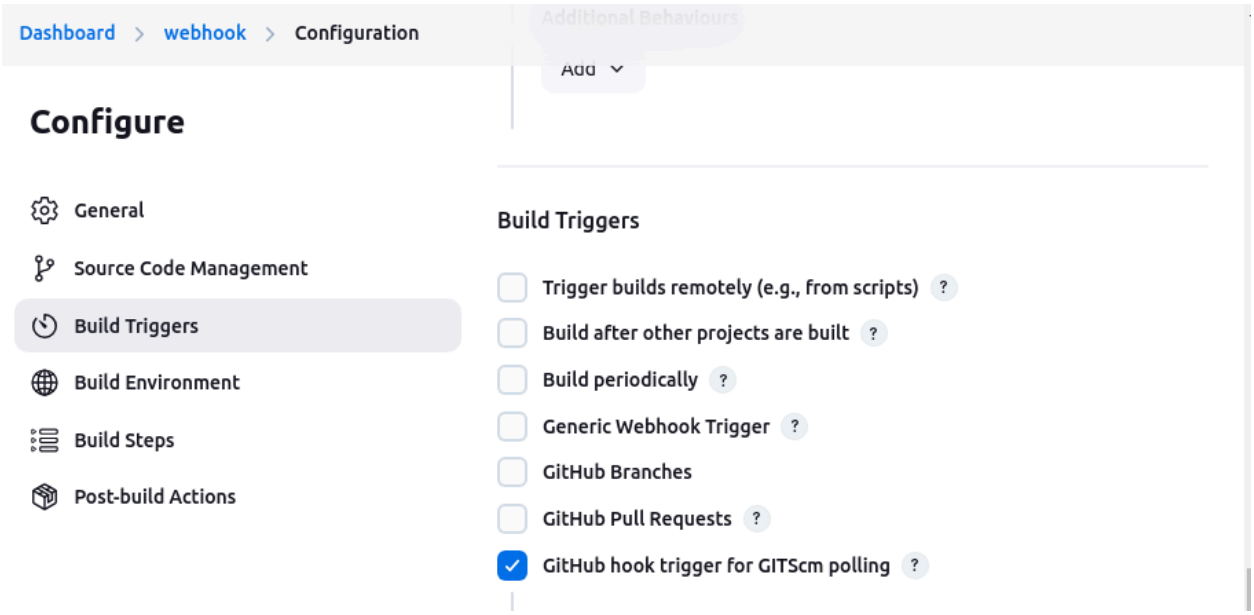
- Goto jenkins dashboard create a job for Freestyle



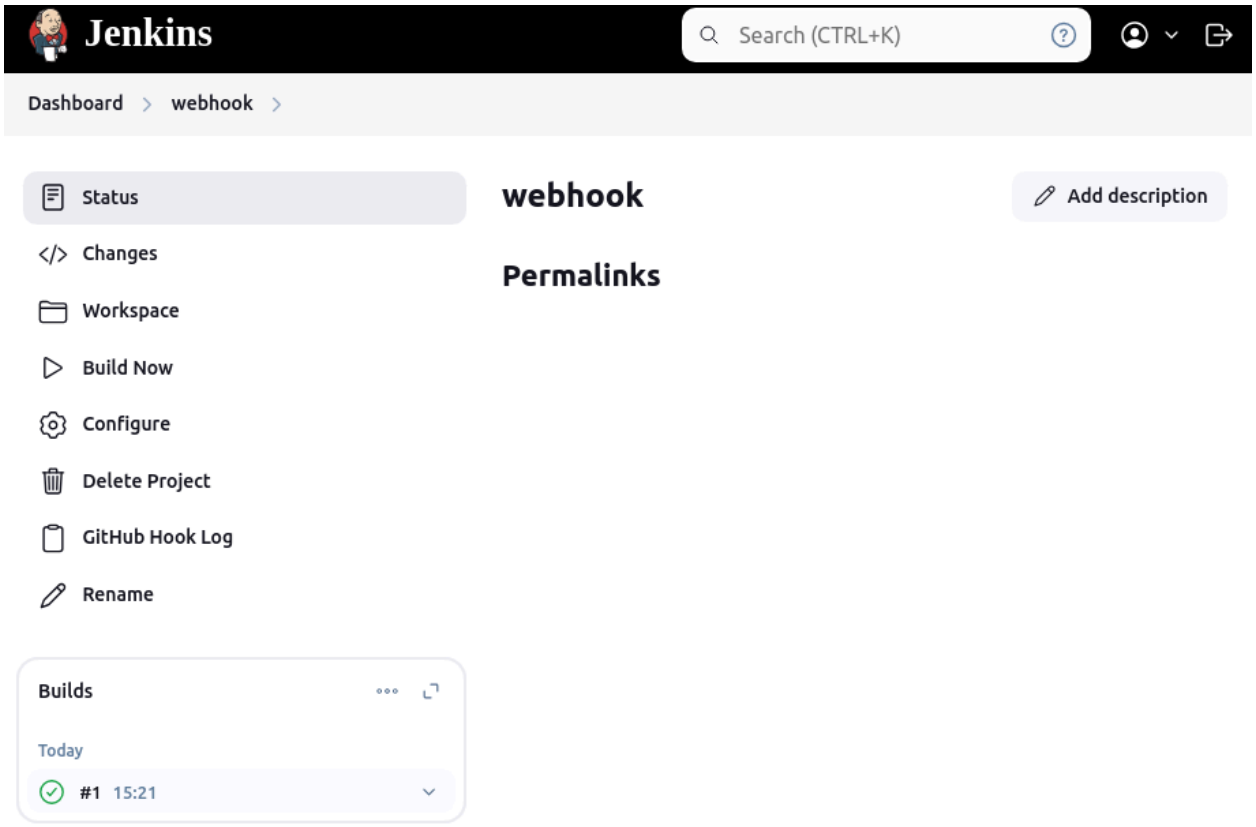
- Change the configure



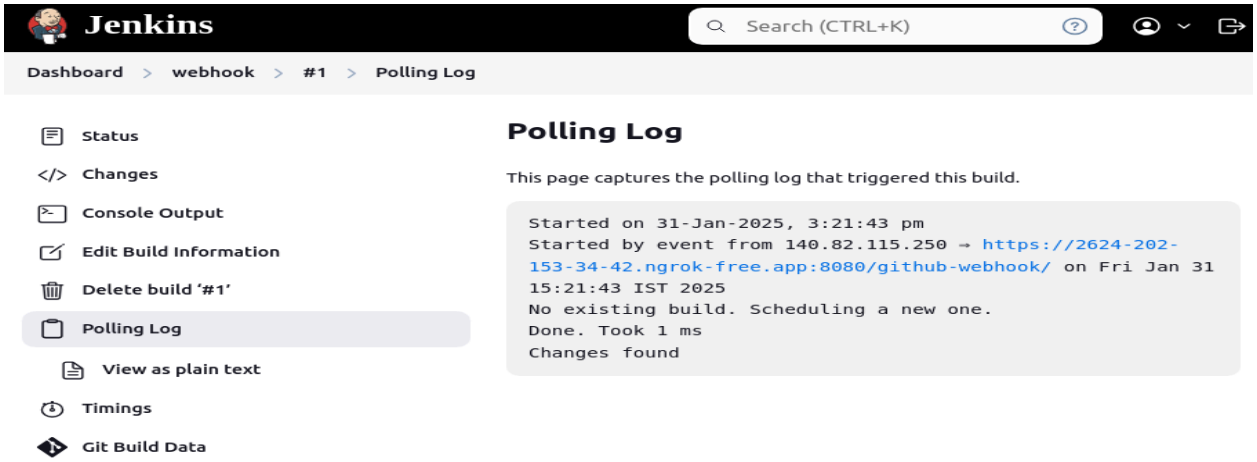
- Select build triggers is **GitHub hook trigger for GITScm polling**



- Change the script in the git repo



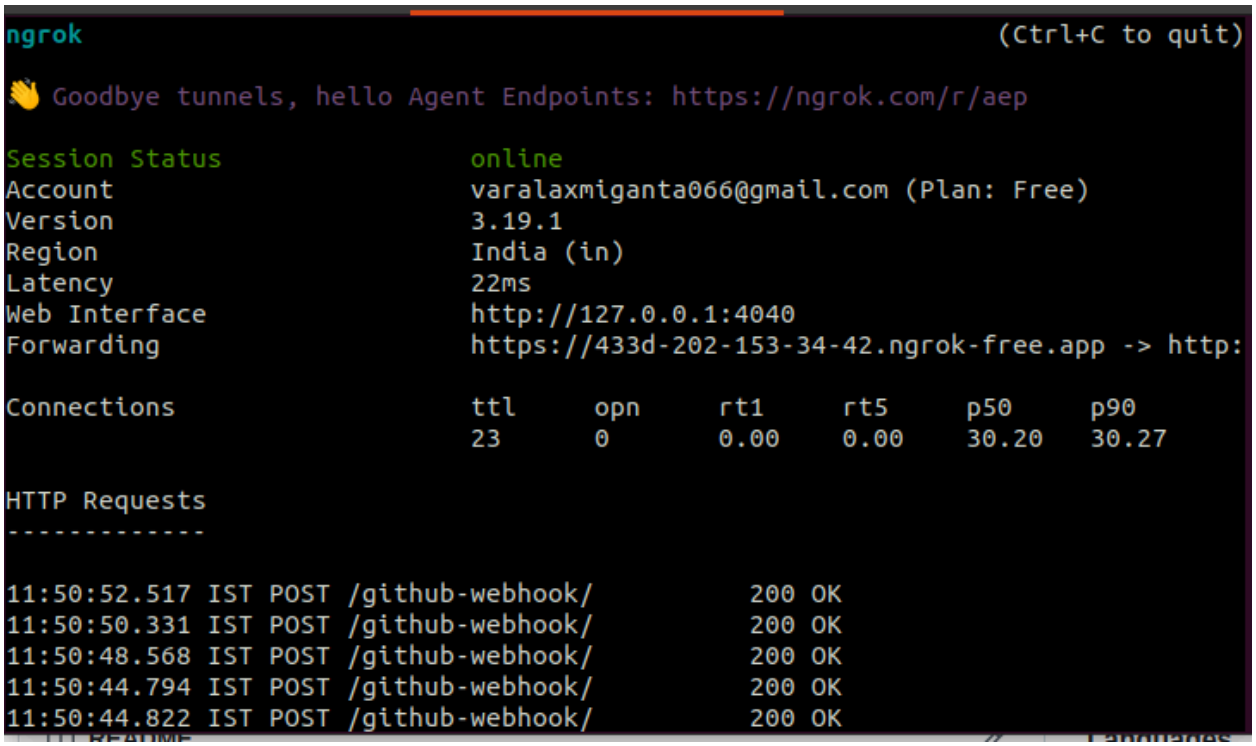
- Jenkins will be trigger the changes



4.1.3. Webhook Trigger for Multibranch Pipeline

Note :

- Check the agent endpoints if it online or not



- 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 :

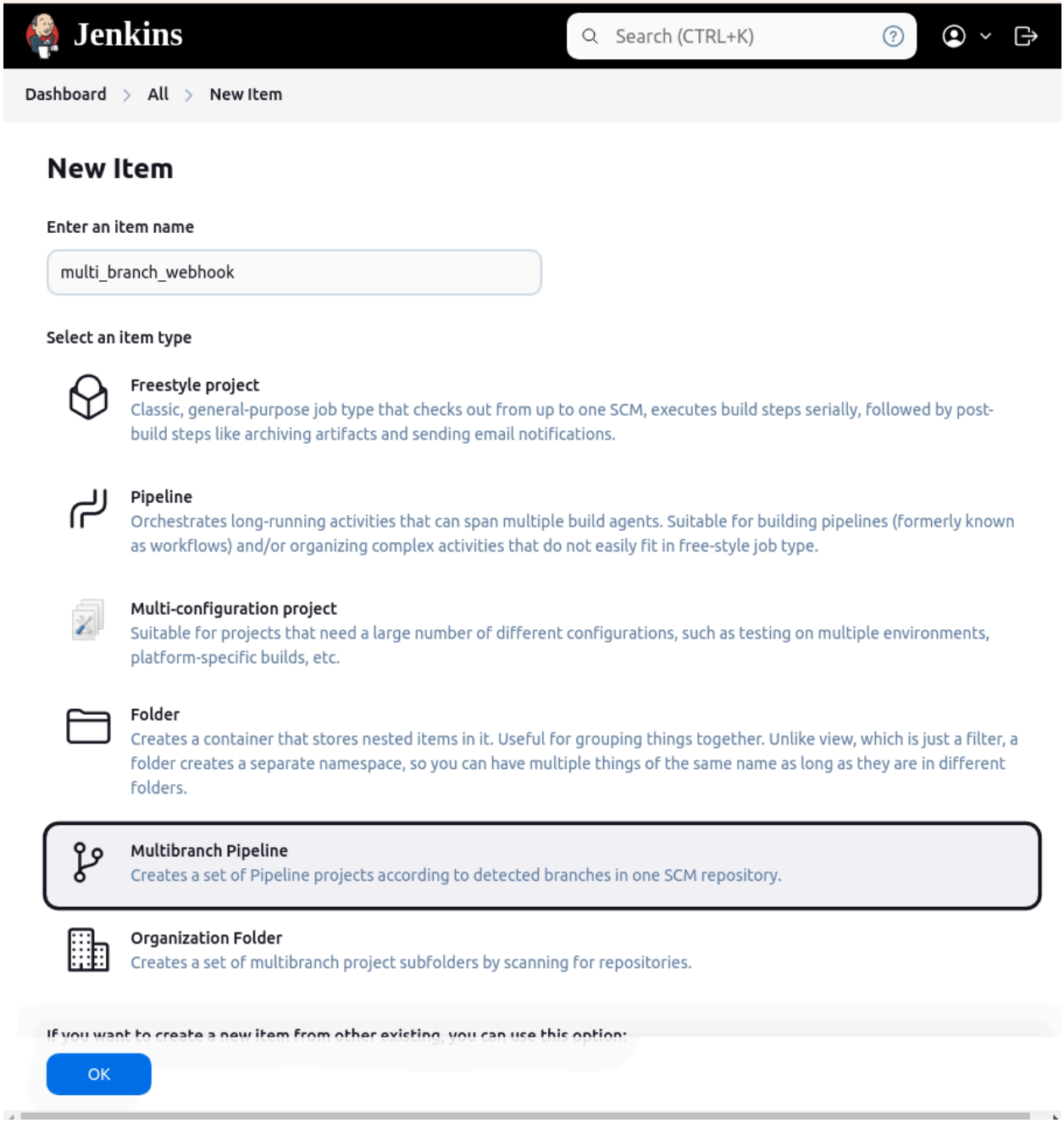
- List all the branches for both remote and local

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

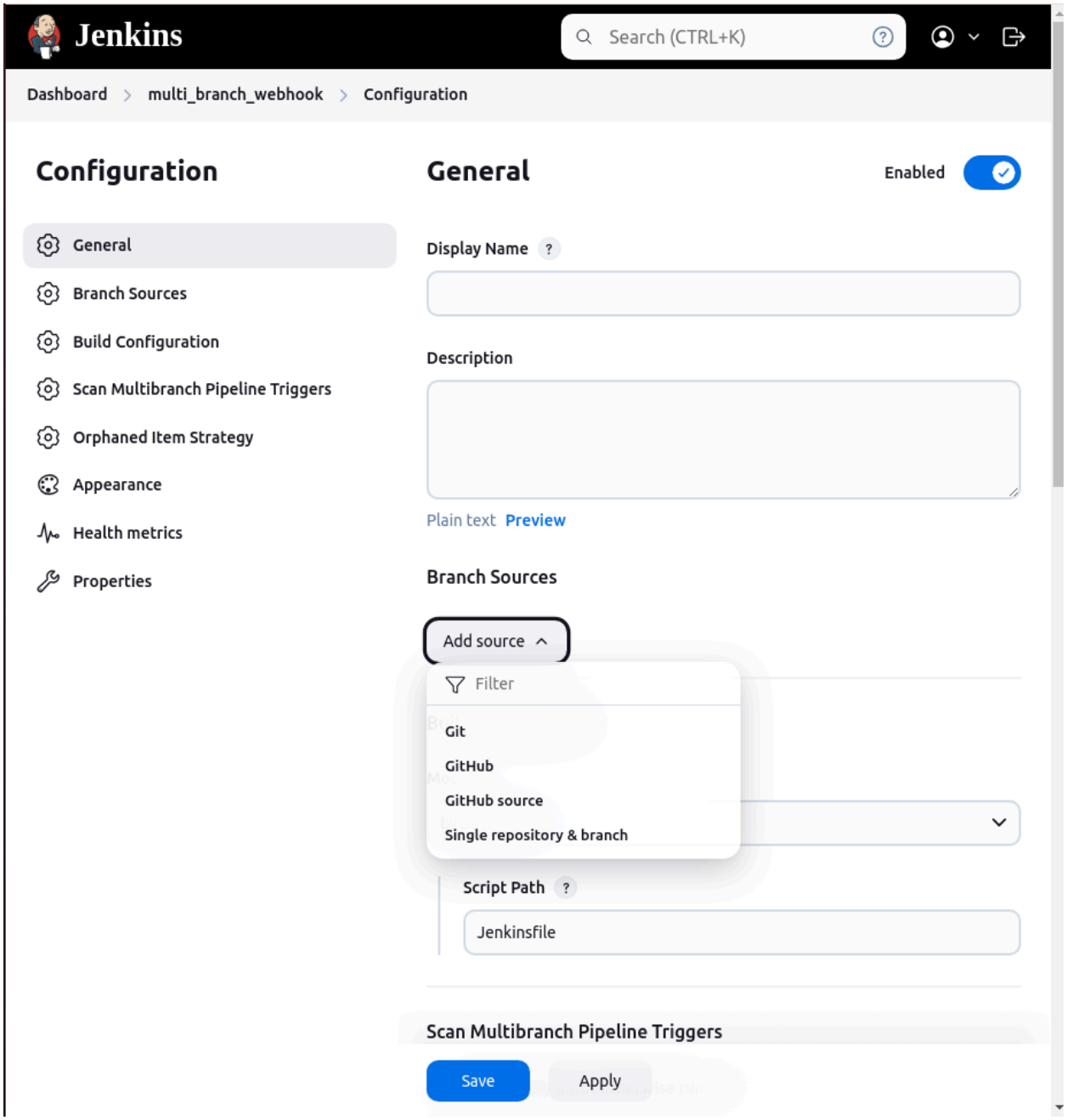
```
vlab@HYVLAB7:~/vara/j_repo1$ git ls-tree -r remotes/origin/dev --name-only
Jenkinsfile
README.md
vlab@HYVLAB7:~/vara/j_repo1$ git ls-tree -r remotes/origin/feature-1 --name-only
README.md
vlab@HYVLAB7:~/vara/j_repo1$ git ls-tree -r main --name-only
Jenkinsfile
README.md
build.sh
deploy.sh
run-tests.sh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

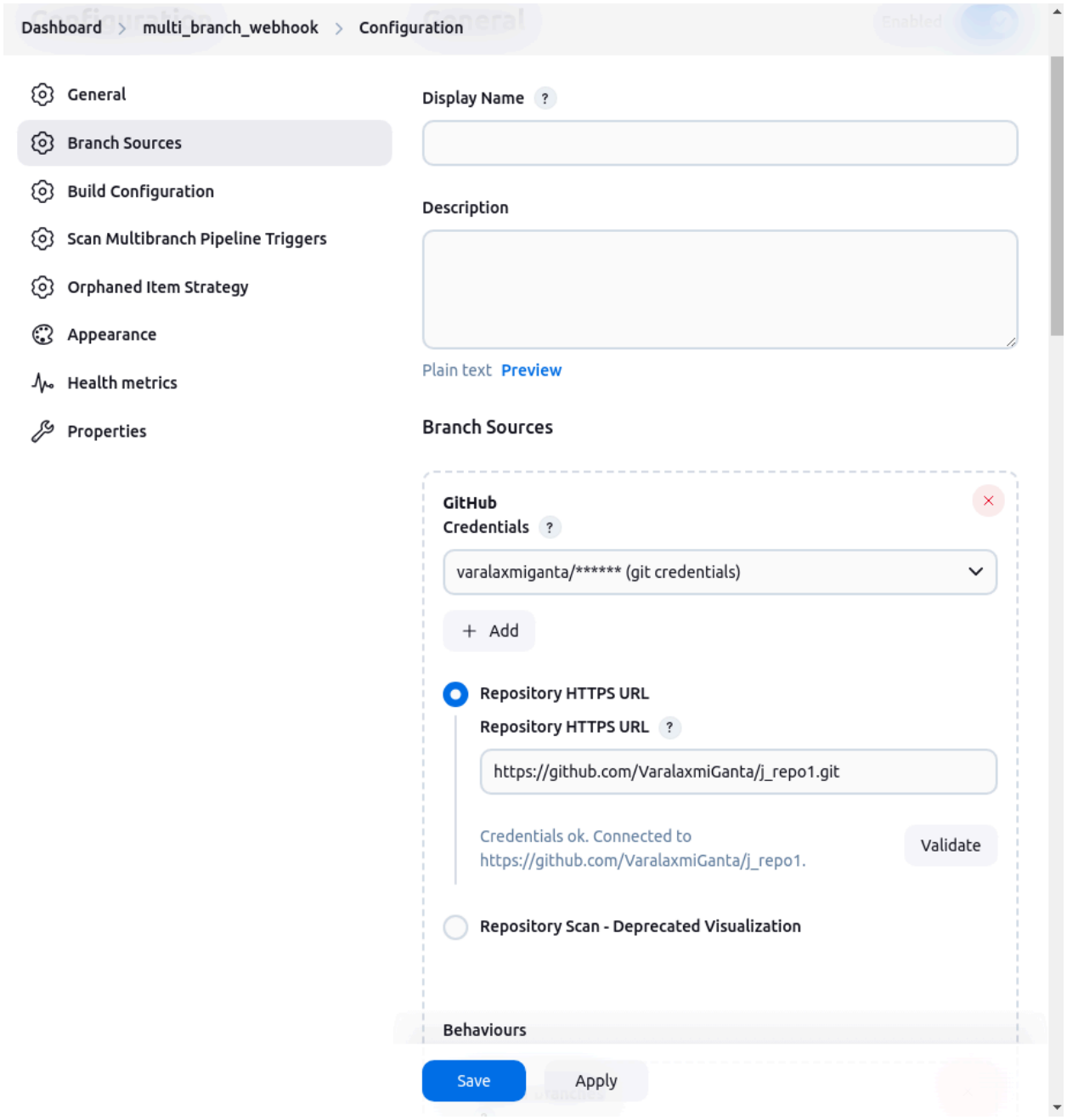
- In the above dev and main branches are having Jenkinsfile so if getting any new commit from dev or main branches then webhook triggers but not for feature-1 because there is no jenkinsfile in that branch.

- 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.

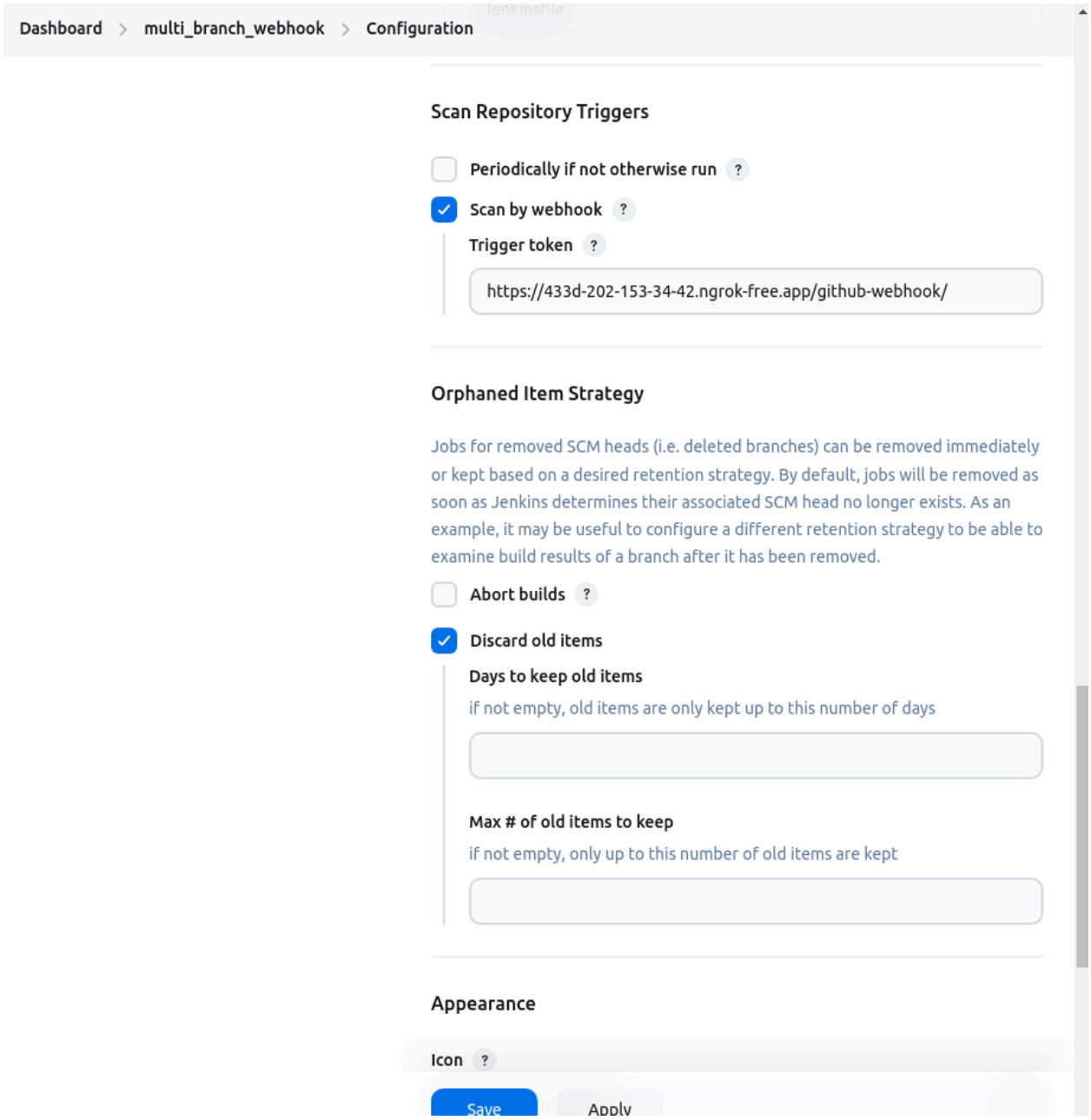


- Configure the Multibranch Pipeline Job:
    - Under the **Branch Sources** section:
        - Select your source control management **GitHub**
        - Provide the repository URL.
        - Configure credentials if needed.

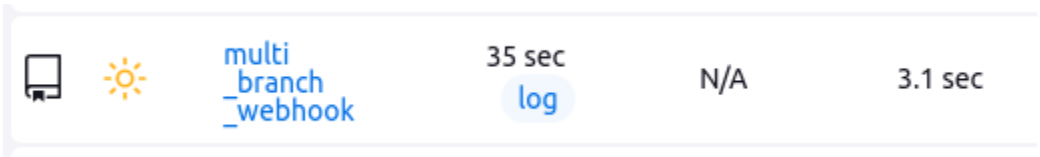

Jenkins
Search (CTRL+K)
Dashboard
multi_branch_webhook
Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
General
Enabled
Display Name
Description
Plain text
Preview
Branch Sources
Add source
Filter
Git
GitHub
GitHub source
Single repository & branch
Script Path
Jenkinsfile
Scan Multibranch Pipeline Triggers
Save
Apply



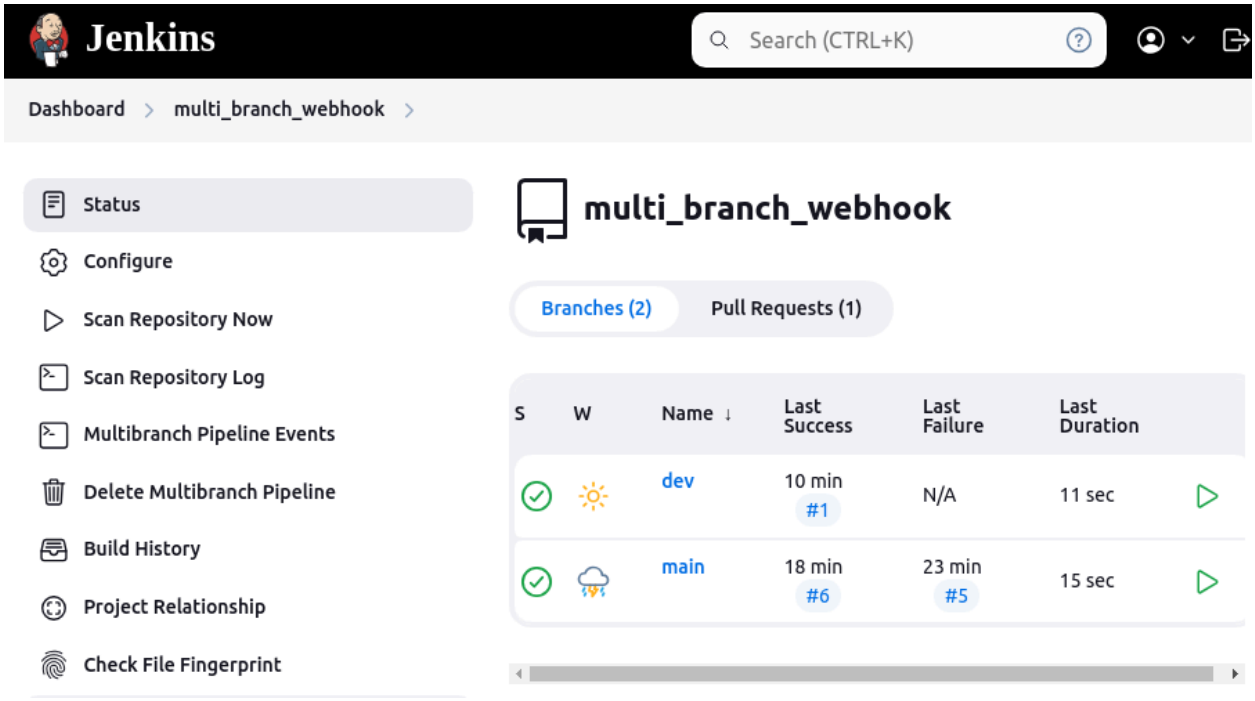
This is from ngrok endpoint



Save the configurations



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



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
```