



[Table of Contents:](#)

[Table of Contents:](#)

[Step -1 : Update the ubuntu index by using update command](#)

[Jenkins Document Link](#)

[Step -2 : Use the Long term Support Release Command to install the Jenkins.](#)

[Step -3 : Login to Jenkins URL with Default Port 8080](#)

[Jenkins Initial Administrator Password](#)

[Step -4 : Setting up Jenkins](#)

[Jenkins Installation has started](#)

[Jenkins has been installed successfully](#)

[Step -5 : Create a Jenkins Pipeline](#)

[General Template Form](#)

[Build Triggers](#)

[Pipeline](#)

[Success](#)

[Failure](#)

[Step -6 : Setting Up the Agent](#)

[Generate SSH Key](#)

[Install JAVA](#)

[Add the Agent](#)

[Agent Status](#)

[Agent Offline](#)

[Step -7 : Build More Pipelines](#)

[Pipeline Structure](#)

[Build the pipeline](#)

[Pipeline Failed-1](#)

[Solution1](#)

[Pipeline Failed-2](#)

[Solution2](#)

[Verify the Build](#)

[Step -8 : Credential Binding & Image Pushed to DockerHub](#)

[Generate the Personal Access Token on Docker Hub](#)

[Build the Docker Compose pipeline](#)

[Pipeline Succeed](#)

[Step -9 : Jenkins Webhook](#)

[Webhook Connection Status & build Trigger Config](#)

[Commit to Trigger Webhook](#)

[End of Jenkins Fundamentals](#)



Step -1 : Update the ubuntu index by using update command

```
$ sudo apt-get update
```

```
$ sudo apt install fontconfig openjdk-17-jre : Install the JAVA 17 version per Jenkins Document.
```

```
ubuntu@ip-172-31-24-254:~$ sudo apt-get update ←
Hit:1 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [297 kB]
Fetched 423 kB in 1s (645 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-24-254:~$ ←
ubuntu@ip-172-31-24-254:~$ ←
ubuntu@ip-172-31-24-254:~$ ←
ubuntu@ip-172-31-24-254:~$ sudo apt install fontconfig openjdk-17-jre ←
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

Jenkins Document Link

<https://www.jenkins.io/doc/book/installing/linux/#unlocking-jenkins>

```
$ java --version : To check the JAVA version.
```

```
ubuntu@ip-172-31-24-254:~$ ←
ubuntu@ip-172-31-24-254:~$ java --version
openjdk 17.0.11 2024-04-16
OpenJDK Runtime Environment (build 17.0.11+9-Ubuntu-1)
OpenJDK 64-Bit Server VM (build 17.0.11+9-Ubuntu-1, mixed mode, sharing)
ubuntu@ip-172-31-24-254:~$
```



Step -2 : Use the Long term Support Release Command to install the Jenkins.

Long Term Support release

A [LTS \(Long-Term Support\) release](#) is chosen every 12 weeks from the stream of regular releases as the stable release for that time period. It can be installed from the [debian-stable apt repository](#).

```
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
```

\$ 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 : Jenkins has been installed successfully, See below Capture.

```
ubuntu@ip-172-31-24-254:~$ 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
--2024-07-27 17:11:08-- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.122.133, 2a04:4e42:8d::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.122.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins-keyring.asc 100%[=====] 3.10K --.-KB/s in 0s

2024-07-27 17:11:08 (34.0 MB/s) - '/usr/share/keyrings/jenkins-keyring.asc' saved [3175/3175]

Hit:1 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [27.3 kB]
```

\$ **jenkins --version** : To Check the Jenkins Version.



```
ubuntu@ip-172-31-24-254:~$ jenkins --version
2.452.3
ubuntu@ip-172-31-24-254:~$
```

\$ systemctl status jenkins : To check the Jenkins Service Status.

\$ systemctl enable jenkins : Service will come up automatically after Linux Server Reboot, else we need to manually make it UP, by using below:

\$ systemctl start jenkins : To Start the Jenkins Service in case it's not running.

```
ubuntu@ip-172-31-24-254:~$ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
  Active: active (running) since Sat 2024-07-27 17:12:06 UTC; 3min 33s ago
    Main PID: 49918 (java)
       Tasks: 39 (limit: 1130)
      Memory: 270.4M (peak: 290.6M)
        CPU: 45.644s
       CGroup: /system.slice/jenkins.service
               └─49918 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jul 27 17:11:30 ip-172-31-24-254 jenkins[49918]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jul 27 17:11:30 ip-172-31-24-254 jenkins[49918]: ****
Jul 27 17:11:30 ip-172-31-24-254 jenkins[49918]: ****
Jul 27 17:11:30 ip-172-31-24-254 jenkins[49918]: ****
Jul 27 17:12:06 ip-172-31-24-254 jenkins[49918]: 2024-07-27 17:12:06.297+0000 [id=31]      INFO      jenkins.InitReactorRunner$1#onAt
Jul 27 17:12:06 ip-172-31-24-254 jenkins[49918]: 2024-07-27 17:12:06.327+0000 [id=24]      INFO      hudson.lifecycle.Lifecycle#onRea
Jul 27 17:12:06 ip-172-31-24-254 systemd[1]: Started Jenkins Continuous Integration Server.
Jul 27 17:12:06 ip-172-31-24-254 jenkins[49918]: 2024-07-27 17:12:06.833+0000 [id=47]      INFO      h.m.DownloadService$Downloadable>
Jul 27 17:12:06 ip-172-31-24-254 jenkins[49918]: 2024-07-27 17:12:06.834+0000 [id=47]      INFO      hudson.util.Retrier#start: Perfo
Jul 27 17:12:11 ip-172-31-24-254 jenkins[49918]: 2024-07-27 17:12:11.478+0000 [id=60]      WARNING   h.n.DiskSpaceMonitorDescripto
ubuntu@ip-172-31-24-254:~$
```

Step -3 : Login to Jenkins URL with Default Port 8080

If you are using EC2 Instance, please open the Port 8080 on Security Group → Inbound Rules

Go to EC2 Instance—>Select the Instance —> Click Security —> Click Security Group
[hyperLink](#)



EC2 Dashboard EC2 Global View Events Instances Instance Types Launch Templates Spot Requests Savings Plans Reserved Instances Dedicated Hosts Capacity Reservations Images AMIs AMI Catalog Elastic Block Store Volumes Snapshots Lifecycle Manager

Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Linux-Batch-7	i-093f326f5c1503d2d	Running	t2.micro	2/2 checks passed	View alarms +	eu-central-1a

i-093f326f5c1503d2d (Linux-Batch-7)

- Details Status and alarms Monitoring **Security** Networking Storage Tags

Security details

IAM Role	Owner ID	Launch time
-	975049903216	Wed Jul 24 2024 00:04:31 GMT+0200 (Central European Summer Time)

Security groups

- sg-0c8ea4b824d623734 (launch-wizard-6)

Security Group page will Open, Click → Edit Inbound rules

EC2 > Security Groups > sg-0c8ea4b824d623734 - launch-wizard-6

sg-0c8ea4b824d623734 - launch-wizard-6

Actions ▾

Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-6	sg-0c8ea4b824d623734	launched-wizard-6 created 2024-06-30T04:31:47.760Z	vpc-03a3b7d329073a97a
Owner	Inbound rules count	Outbound rules count	
975049903216	1 Permission entry	1 Permission entry	

Inbound rules Outbound rules Tags

Inbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0170dfc0bb3962374	IPv4	SSH	TCP	22

Edit inbound rules

Click → Add Rule → Custom TCP , Port Range 8080.

Note : Please use “**MyIP**” , for this document purpose i have selected “Anywhere-IPV4”



[EC2](#) > [Security Groups](#) > [sg-0c8ea4b824d623734 - launch-wizard-6](#) > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>
sgr-0170dfc0bb3962374	SSH	TCP	22	Custom	Q <input type="text"/> 0.0.0.0/0 <input type="button" value="Delete"/>
-	Custom TCP	TCP	8080	Anyw... ▲ Custom Anywhere-IPv4✓ Anywhere-IPv6 My IP	<input type="text"/> 0.0.0.0/0 <input type="button" value="Delete"/>

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Note : You can install Jenkins on Virtual Box “Ubuntu Linux VM” in your laptop & default url will be in this format —> <http://127.0.0.1:8080> : For Local Ubuntu Linux Machine

For EC2 Instance it will be in this format—> <http://<EC2PublicIP>:8080>

127.0.0.1:8080/



Sign in to Jenkins

Username

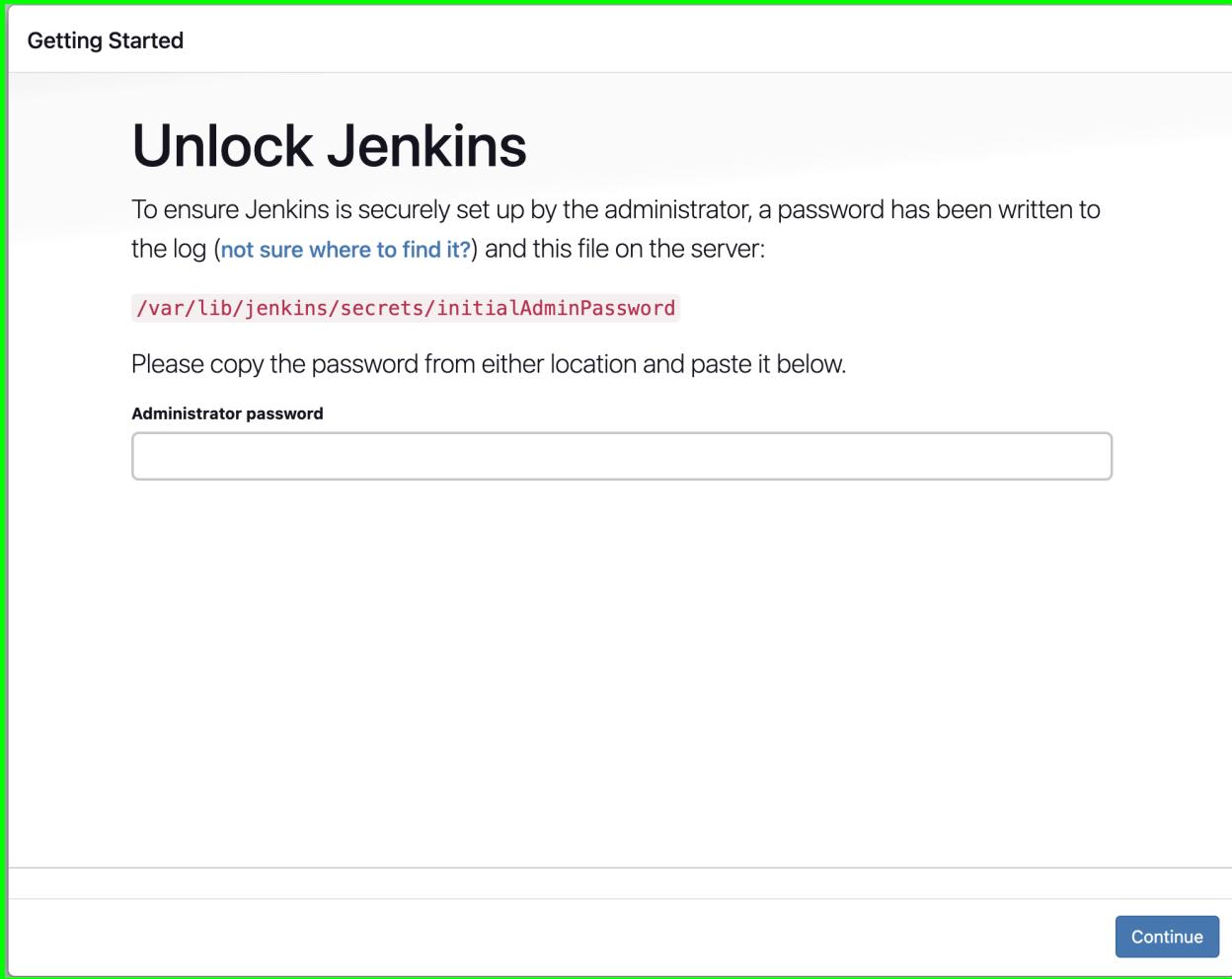
Password

Keep me signed in



Once the above steps are completed, use your browser with below url
http://<yourEC2PublicIP>:8080

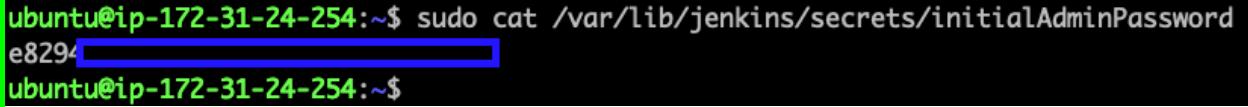
It will Open up with the below “Getting Started” Window.



The screenshot shows the Jenkins 'Unlock Jenkins' setup page. At the top left, it says 'Getting Started'. In the center, there is a large heading 'Unlock Jenkins'. Below the heading, a text block states: 'To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:'. A red highlighted path is shown: '/var/lib/jenkins/secrets/initialAdminPassword'. Below this, a text instruction says: 'Please copy the password from either location and paste it below.' There is a text input field labeled 'Administrator password' with a placeholder 'Enter password here'. At the bottom right of the page, there is a blue 'Continue' button.

Jenkins Initial Administrator Password

As per above Jenkin screen shot, please go to the path below in Linux EC2 Instance Machine.
/var/lib/jenkins/secrets/initialAdminPassword



A terminal window with a black background and white text. It shows the command: 'ubuntu@ip-172-31-24-254:~\$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword'. The password 'e8294' is partially obscured by a blue rectangle. The command is run twice, once with 'sudo' and once without, both resulting in the same output.



Copy the password & enter into the below step.

=====

Step -4 : Setting up Jenkins

After Entering the Password, Click Continue

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

Continue



This will be the first screen, after hitting Continue , Please Select “**Install Suggested plugins**”

It will Start the Installation.

The screenshot shows the Jenkins 'Customize Jenkins' setup screen. At the top left is a 'Getting Started' button and a close ('x') button. The main title 'Customize Jenkins' is centered above a subtitle: 'Plugins extend Jenkins with additional features to support many different needs.' Below this are two options: 'Install suggested plugins' (selected) and 'Select plugins to install'. The 'Install suggested plugins' section contains the text: 'Install plugins the Jenkins community finds most useful.' The 'Select plugins to install' section contains the text: 'Select and install plugins most suitable for your needs.' At the bottom left of the screen, the Jenkins version 'Jenkins 2.452.3' is displayed.



=====

Jenkins Installation has started

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** Bootstrap 5 API ** JQuery3 API ** ECharts API ** Display URL API ** Checks API ** JUnit ** Matrix Project ** Resource Disposer Workspace Cleanup
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle	Ant ** JavaMail API ** Durable Task ** Pipeline: Nodes and Processes ** Instance Identity ** Pipeline: SCM Step ** Pipeline: Groovy ** Pipeline: Job ** Jakarta Activation API ** Jakarta Mail API ** Apache HttpComponents Client 4.x API
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View	Mailer ** Pipeline: Basic Steps Gradle ** Pipeline: Milestone Step ** Pipeline: Build Step ** Pipeline: Groovy Libraries ** Pipeline: Stage Step ** - required dependency
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	
⌚ LDAP	⌚ Email Extension	✓ Mailer	⌚ Dark Theme	

Jenkins 2.452.3



Fill the below details & hit Save and Continue

Username : You can use “admin” as well , but I have used “dheeruyadav54” as my username.

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.452.3

Skip and continue as admin

Save and Continue



Jenkins URL , you can keep the url or can change it

http://<EC2PublicIPAddress>:8080 → **this is the Default URL.**

Hit Save and Finish

Getting Started

Instance Configuration

Jenkins URL:

http://3.79.149.35:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.452.3

Not now

Save and Finish



Click on Start using Jenkins:

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.452.3



Jenkins has been installed successfully

See the below DashBoard:

The screenshot shows the Jenkins dashboard with a green border around the main content area. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user information for 'Dheeraj Yadav'. Below the bar, the dashboard has several sections: 'Welcome to Jenkins!', 'Start building your software project', 'Build Queue' (empty), 'Build Executor Status' (one node listed as 'offline'), and 'Set up a distributed build' with links for 'Create a job', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. A sidebar on the left includes links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'.

Step -5 : Create a Jenkins Pipeline

Click on “New Item”

The screenshot shows the Jenkins dashboard with a green border. The 'New Item' button in the top left corner is highlighted with a red box and a cursor pointing at it. Other visible elements include the Jenkins logo, a 'Dashboard' link, a 'Build History' link, and a 'All' button.



Select Pipeline & give a Project Pipeline Name, Hit OK.

The screenshot shows the Jenkins Pipeline creation interface. A green box highlights the 'Enter an item name' field where 'django-Project' has been typed. Below it, a required field message is visible. A list of project types is shown:

- 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**: Is a set of multibranch project subfolders by scanning for repositories.

A blue 'OK' button is visible at the bottom of the list.

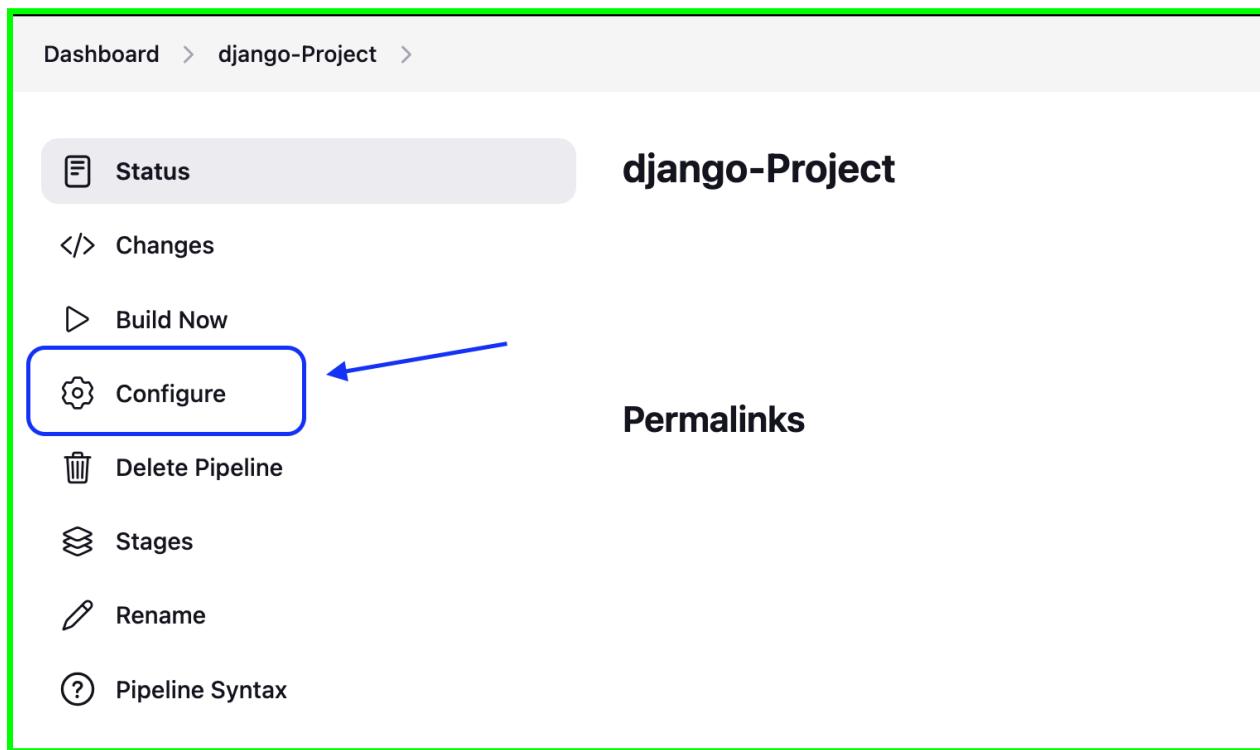
Once you click okay , it will show you like this in your Dashboard.

The screenshot shows the Jenkins Dashboard. A green box highlights the 'Build History' section, which displays the newly created 'django-Project'. The table shows the following details:

S	W	Name ↓	Last Success	Last Failure	Last Duration
...	...	django-Project	N/A	N/A	N/A

Below the table, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node, Offline).

Select the “django-Project” & Click on **Configure** icon.



The screenshot shows the Jenkins Pipeline configuration page for a project named "django-Project". The top navigation bar includes links for "Dashboard" and "django-Project". Below the navigation, there are several buttons and links:

- Status
- </> Changes
- ▷ Build Now
- Configure** (this button is highlighted with a blue border and has a blue arrow pointing to it from the text above)
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax

On the right side of the screen, there is a "Permalinks" section.

General Template Form

It will Open Up a template Form which needs to be filled. Let me explain all the Option 1 by 1.

- Discard old builds** : You can set up the days to keep your Old Build History before Jenkins start Overriding, here in this example i have configured:

- Days to keep builds = 5
- Max # of builds to keep = 7
- Days to keep artifacts = 5
- Max # of builds to keep with artifacts = 7



Description

This is a Django CI/CD Project & it will be a Sample Template how Pipeline will be triggered.

Plain text [Preview](#)

Discard old builds [?](#)

Strategy

Log Rotation

Days to keep builds
if not empty, build records are only kept up to this number of days
5

Max # of builds to keep
if not empty, only up to this number of build records are kept
7

Advanced [^](#) [Edited](#)

Days to keep artifacts
if not empty, artifacts from builds older than this number of days will be deleted, but the logs, history, reports, etc for the build will be kept
5

Max # of builds to keep with artifacts
if not empty, only up to this number of builds have their artifacts retained
7

- b) **Do not allow concurrent builds** : Once you select this option, you can't run a parallel build, meaning it will wait to complete the previous Job.
- c) **Do not allow the pipeline to resume if the controller restarts** : In case, you restart the Jenkins & don't want to resume the pipeline before restart, use this checkbox,

Do not allow concurrent builds

Abort previous builds [?](#)

Do not allow the pipeline to resume if the controller restarts

- d) **GitHub project** : Enter the github Project Repo URL here.

Project URL : <https://github.com/dheeruyadav54/django-notes-app.git>



Display name : It will be visible during Commit Status.

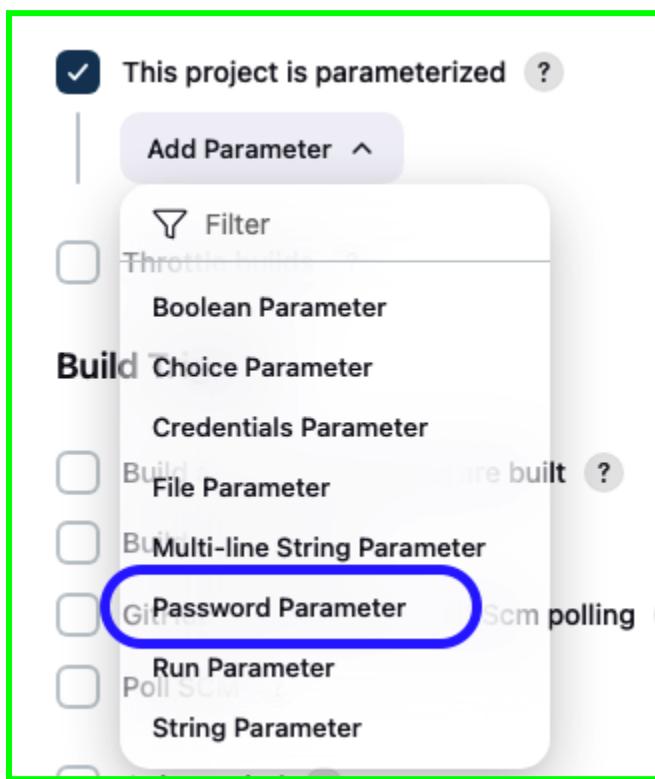
A screenshot of the Jenkins GitHub project configuration screen. A green box highlights the 'Display name' section. It shows a checked checkbox for 'GitHub project', a 'Project url' input field containing 'https://github.com/dheeruyadav54/django-notes-app.git', and a 'Display name' input field containing 'Django CI/CD'.

- e) **Pipeline speed/durability override :** You can configure the Pipeline speed based on the need , we will be using the default option : Maximum Survivability/Durability but Slowest.
- f) **Preserve stashes from completed builds :** If you check this box, meaning you can keep this Project Artifacts up-to 5 latest Pipeline, before Jenkins starts Overriding it.

A screenshot of the Jenkins pipeline configuration screen. A green box highlights the 'Pipeline speed/durability override' and 'Preserve stashes from completed builds' sections. Under 'Pipeline speed/durability override', there are three options: 'Performance-optimized: much faster (requires clean shutdown to save running pipelines)', 'Less durability, a bit faster (specialty use only)', and 'Maximum survivability/durability but slowest', with the last one being selected. Under 'Preserve stashes from completed builds', there is a checked checkbox and a 'Keep this many of the most recent builds' stash input field containing '5'.

A screenshot of the Jenkins pipeline configuration screen. A green box highlights the 'Pipeline speed/durability override' and 'Preserve stashes from completed builds' sections. Under 'Pipeline speed/durability override', there is a 'Custom Pipeline Speed/Durability Level' dropdown menu open, showing the option 'Maximum survivability/durability but slowest'. Under 'Preserve stashes from completed builds', there is a checked checkbox and a 'Keep this many of the most recent builds' stash input field containing '5'.

- g) **This project is parameterized :** We can use the below Parameter to insert into pipeline like Password parameter



This screenshot shows the 'Parameter' configuration dialog for a Jenkins project. A green box highlights the 'Throttle builds' checkbox, which is checked. Below it, a blue oval highlights the 'Password Parameter' option under the 'Build' category.

This project is parameterized ?

Add Parameter ^

Filter

Throttle builds

Boolean Parameter

Build Choice Parameter

Credentials Parameter

Build File Parameter

Multi-line String Parameter

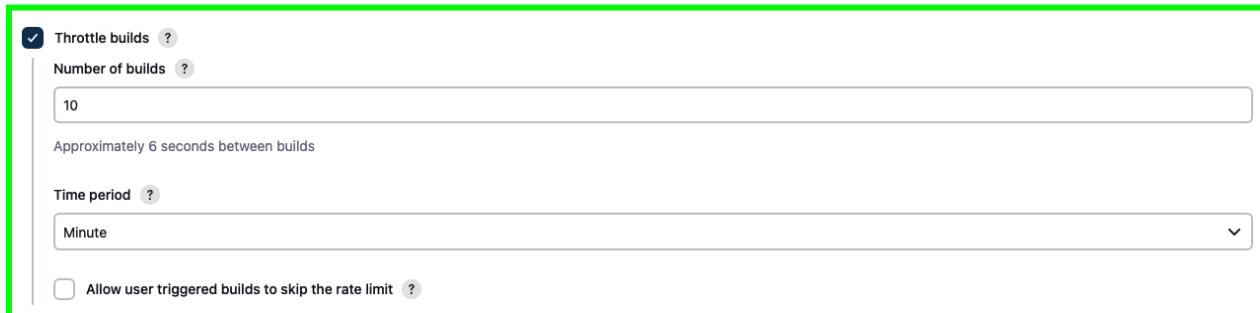
GitHub Password Parameter

Run Parameter

Poll SCM

String Parameter

- h) Throttle builds : We can define the Number of builds per Second, minute, hour, Day, Week, Month, Year etc.
⇒ We have defined 10 Builds per minute



This screenshot shows the 'Throttle builds' configuration dialog. A green box highlights the 'Number of builds' field, which is set to 10. Below it, a dropdown menu shows 'Minute' selected. A checkbox for 'Allow user triggered builds to skip the rate limit' is also visible.

Throttle builds ?

Number of builds ?

10

Approximately 6 seconds between builds

Time period ?

Minute

Allow user triggered builds to skip the rate limit ?



Build Triggers

- a) Build after other projects are built :
- b) Build periodically
- c) GitHub hook trigger for GITSCM polling: Whenever you make changes in the github repository Jenkins receives a GitHub push hook, GitHub Plugin checks to see whether the hook came from a GitHub repository.
- d) Poll SCM
- e) Quiet period
- f) Trigger builds remotely (Script)

Build Triggers

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

Advanced Project Options:

Display name :

If set, the optional display name is shown for the project throughout the Jenkins web GUI. As it is for display purposes only, the display name is not required to be unique amongst projects.

If the display name is not set, the Jenkins web GUI will default to showing the project name.

Advanced Project Options

Advanced ^  Edited

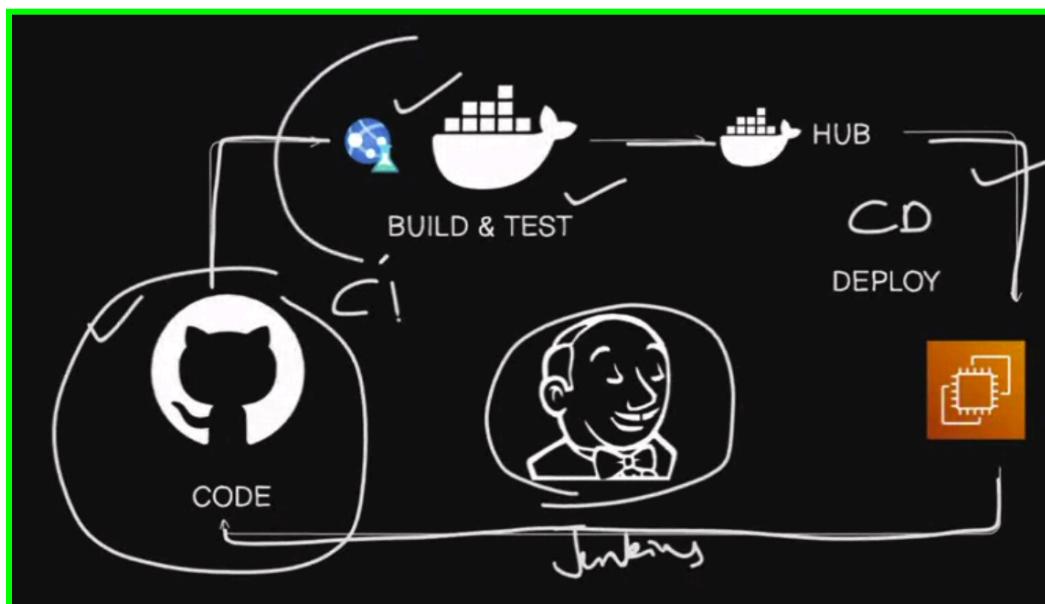
Display Name 

django-Dheeraj-Project

=====

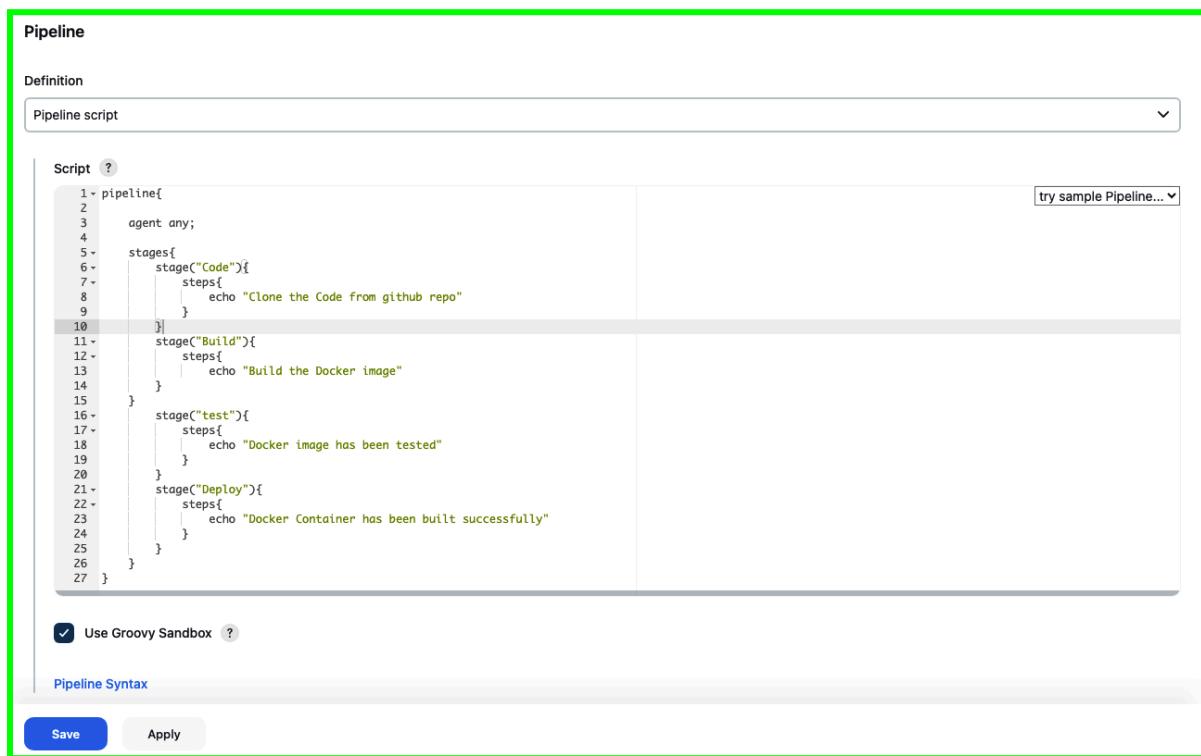
Pipeline

Pipeline Flow





====> Created this Simple Pipeline where we have created 4 stages, Click Save now.



The screenshot shows the Jenkins Pipeline configuration interface. The title bar says "Pipeline". The main area is titled "Definition" and contains a "Pipeline script" section. The script content is:

```
1 pipeline{
2   agent any;
3
4   stages{
5     stage("Code"){
6       steps{
7         echo "Clone the Code from github repo"
8       }
9     }
10    stage("Build"){
11      steps{
12        echo "Build the Docker image"
13      }
14    }
15    stage("test"){
16      steps{
17        echo "Docker image has been tested"
18      }
19    }
20    stage("Deploy"){
21      steps{
22        echo "Docker Container has been built successfully"
23      }
24    }
25  }
26}
27 }
```

Below the script, there is a checkbox "Use Groovy Sandbox" which is checked. At the bottom, there are "Save" and "Apply" buttons.

Once you Save the pipeline, now the next option is to build it.

Click on “Build Now”, you will see the Build History below.



Jenkins

Dashboard > django-Dheeraj-Project >

Status

</> Changes

Build Now

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

django-Dheeraj-Project

Project name: django-Project

Permalinks

- Last build (#1), 42 sec ago
- Last stable build (#1), 42 sec ago
- Last successful build (#1), 42 sec ago
- Last completed build (#1), 42 sec ago

Build History

trend

Filter...

- #3 | Jul 28, 2024, 7:47 PM
- #2 | Jul 28, 2024, 7:47 PM
- #1 | Jul 28, 2024, 7:46 PM

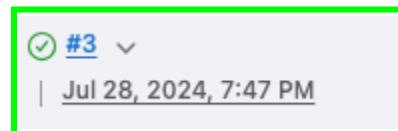
Atom feed for all Atom feed for failures



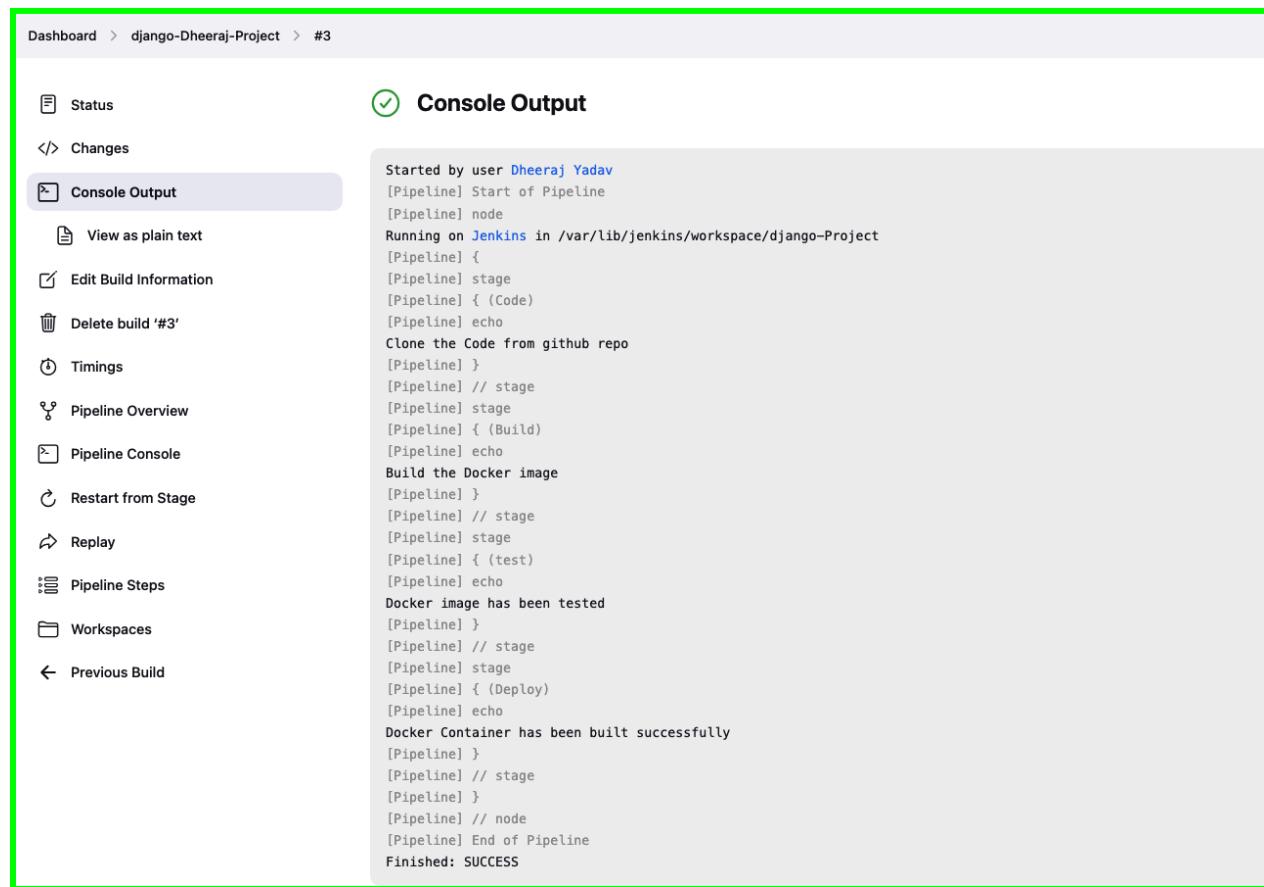
Click on successful pipeline, here in this example click on #03



Success



Click on Console Output to see more details about pipeline steps

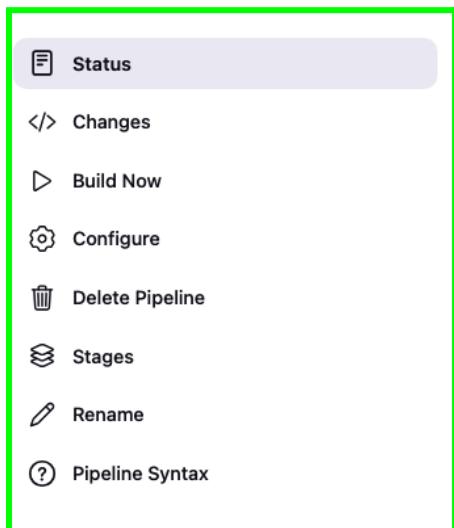


A screenshot of the Jenkins Pipeline Console Output page. The URL in the address bar is 'Dashboard > django-Dheeraj-Project > #3'. The left sidebar has a 'Console Output' tab selected, which is highlighted with a green background. Other tabs include 'Status', 'Changes', 'Edit Build Information', 'Delete build #3', 'Timings', 'Pipeline Overview', 'Pipeline Console', 'Restart from Stage', 'Replay', 'Pipeline Steps', 'Workspaces', and 'Previous Build'. The main content area is titled 'Console Output' with a green checkmark icon. It displays the following log output:

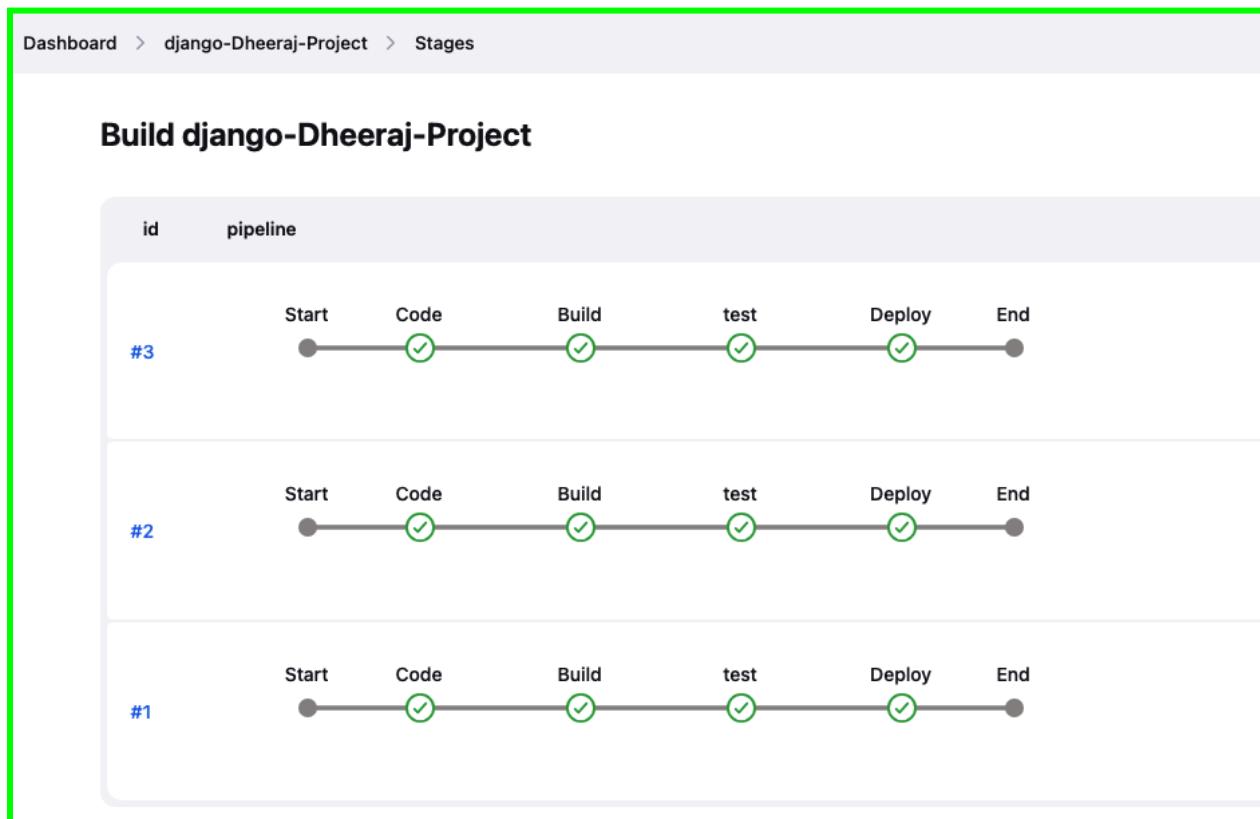
```
Started by user Dheeraj Yadav
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/django-Project
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] echo
Clone the Code from github repo
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Build the Docker image
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (test)
[Pipeline] echo
Docker image has been tested
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Docker Container has been built successfully
[Pipeline] }
[Pipeline] // stage
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Click on Stages tab to see each stage status.



All the Pipelines were successful.





Let's make a mistake to see pipeline fail status.



Failure

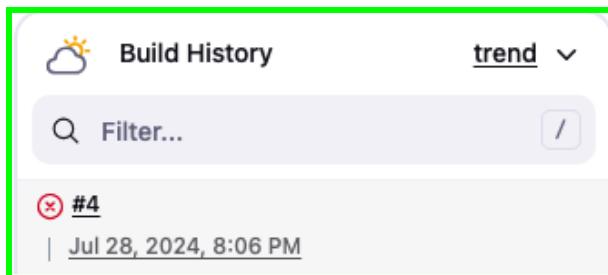
Make a small mistake, now the pipeline will fail, Just to show you how fail pipeline looks.

A screenshot of the Jenkins Pipeline Script Editor. The title bar says "Pipeline". Under "Definition", there is a dropdown menu set to "Pipeline script". The main area shows a Groovy script:

```
4
5 stages{
6   stage("Code"){
7     steps{
8       echo "Clone the Code from github repo"
9     }
10  }
11 stage("Build"){
12   steps{
13     echo "Build the Docker image"
14     ech "making this typo deliberately for pipeline to fail"
15   }
16 }
17 stage("test"){
18   steps{
```

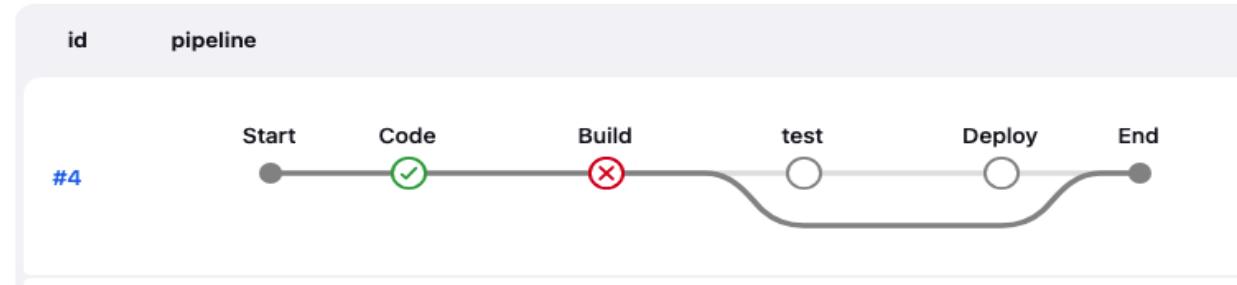
A blue arrow points to the line "ech "making this typo deliberately for pipeline to fail"" to highlight the error. At the bottom left, there is a checked checkbox labeled "Use Groovy Sandbox".

Build number #4 failed, Let's check the stages Output.





Build django-Dheeraj-Project



Click on Build Failed step to see more details about Failure.

The screenshot shows the Jenkins Pipeline Console for Build #4. The pipeline failed 7 minutes 6 seconds ago in 2.6 seconds. The stages are: Code (Success), Build (Failure), test (Not Started), and Deploy (Not Started). The 'Build' stage details show it started 7 min 5 sec ago, queued 0 ms, took 0.34 sec, and failed. It includes a link to 'View as plain text'. The 'Build the Docker image' step took 0.29 sec and was successful. The 'Pipeline error' step took 0.34 sec and failed.

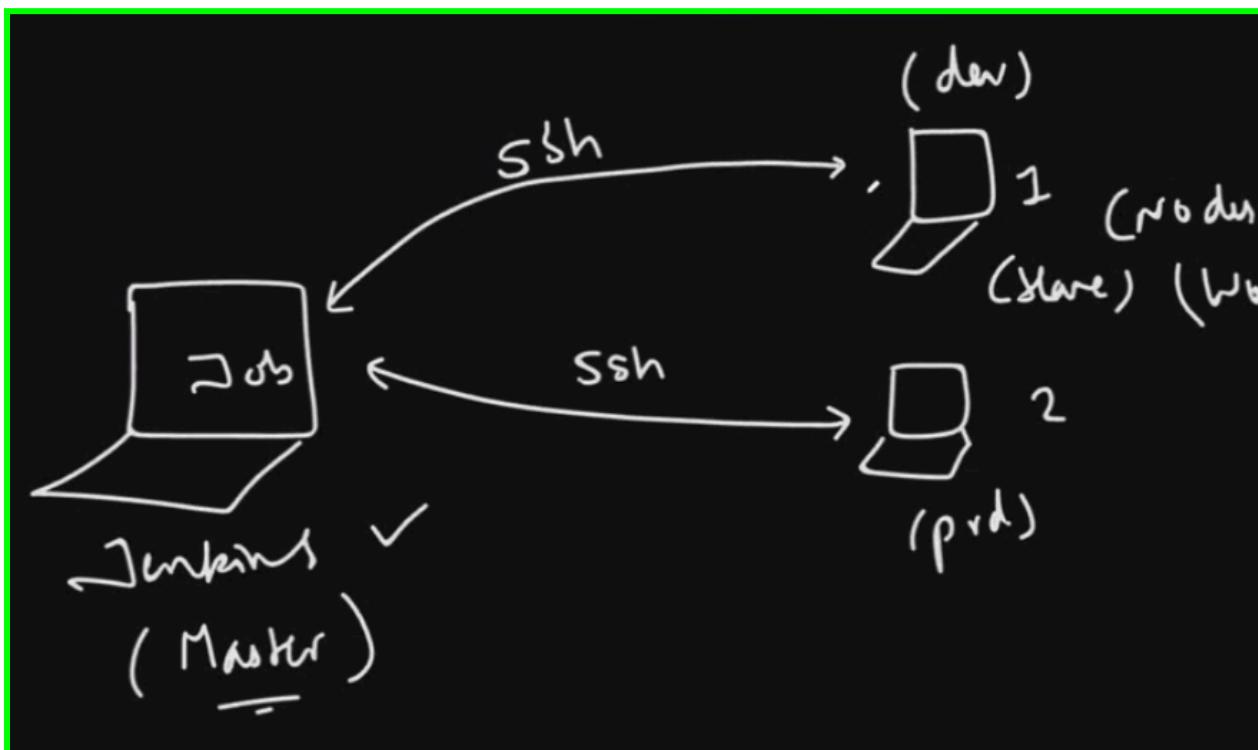
Step -6 : Setting Up the Agent

==> We will Create one more EC2 instance to be running as AGENT.

Note : Agent doesn't need Jenkins to be running, if it does then it is running as MASTER.

Prerequisites : Need to install JAVA only on Agent Machine.

This is how Master → Agent will be communicating “through Private , Public key”



====> Created a New EC2 Instance with Name “Jenkins-Agent”

Instances (2) Info							Actions	Launch instances
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	
<input type="checkbox"/>	Jenkins-Agent	i-022648426d2e4db04	<input checked="" type="radio"/> Running	t2.micro	<input checked="" type="radio"/> 2/2 checks passed	View alarms +	eu-central-1b	
<input type="checkbox"/>	Linux-Batch-7	i-093f326f5c1503d2d	<input checked="" type="radio"/> Running	t2.micro	<input checked="" type="radio"/> 2/2 checks passed	View alarms +	eu-central-1a	

=====

Generate SSH Key

Let's Generate SSH Key so jenkins's Agent can connect to Jenkins Master Server with Public key.

\$ ssh-keygen : Run this command to generate the SSH Private & Public key on Jenkins Master Server.

Note : if you keep the default name id_ed25519 for your Private key then Public Key Name would be id_ed25519.pub & both will be saved to default location
`/home/ubuntu/.ssh`



In this example, I have changed the name hence the key saved into my /home/ubuntu not into .ssh.

```
ubuntu@ip-172-31-24-254:~$ ssh-keygen
ubuntu@ip-172-31-24-254:~$ Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519): id_jenkins
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_jenkins
Your public key has been saved in id_jenkins.pub
The key fingerprint is:
SHA256:B97Q5EQ5xDQuUb9YrjY2wLMSSH5XiTJ9r4+8UiSugZ8 ubuntu@ip-172-31-24-254
The key's randomart image is:
---[ED25519 256]---
|       .B0o..
|   o .oB*+o
|   o *o=oo.
|   o B.+...
|   .S.=oo.
|   ...+..o
|   .++*o o
|   E=.oo .
|   ...
|-----[SHA256]-----
ubuntu@ip-172-31-24-254:~$ 
ubuntu@ip-172-31-24-254:~$ 
```

```
ubuntu@ip-172-31-24-254:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-24-254:~$ ls -ltr id_jenkins*
-rw-r--r-- 1 ubuntu ubuntu 105 Jul 28 20:34 id_jenkins.pub
-rw----- 1 ubuntu ubuntu 419 Jul 28 20:34 id_jenkins
ubuntu@ip-172-31-24-254:~$ 
ubuntu@ip-172-31-24-254:~$ 
```

====> Copy this Public key & paste into Jenkins-Agent under **authorized_keys** file.

```
ubuntu@ip-172-31-24-254:~$ cat id_jenkins.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDGn1vsy3oNTfCBGvvIu+PWPZgnBui4JWYH6bTSCueAy ubuntu@ip-172-31-24-254
ubuntu@ip-172-31-24-254:~$ 
```



```
ubuntu@ip-172-31-35-243:~/.ssh$ pwd
/home/ubuntu/.ssh
ubuntu@ip-172-31-35-243:~/.ssh$ 
ubuntu@ip-172-31-35-243:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-35-243:~/.ssh$
```

Paste the Pub Key , See below:

```
ubuntu@ip-172-31-35-243:~/.ssh$ cat authorized_keys
ssh-rsa
myd7d8PP
M8xZ1DZH
b3gX2iEOK0B Batch-7-Key
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDGn1vsy3aNTfCBGvvIu+PWPZgnBui4JWYH6bTSCueAy ubuntu@ip-172-31-24-254
ubuntu@ip-172-31-35-243:~/.ssh$
```

From Master i did the SSH to Jenken's Agent using the id_jenkins Private key.

Master Jenkins is able to connect Jenkins Agent through SSH Key.

```
Jenkins-EC2-Master Jenkins-EC2-Agent
ubuntu@ip-172-31-24-254:~$ ssh -i id_jenkins ubuntu@ec2-3-127-68-109.eu-central-1.compute.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

System information as of Sat Aug  3 15:22:12 UTC 2024

 System load:  0.0          Processes:           103
 Usage of /:   37.3% of 6.71GB   Users logged in:      0
 Memory usage: 19%           IPv4 address for enX0: 172.31.35.243
 Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

27 updates can be applied immediately.
6 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat Aug  3 15:22:13 2024 from 172.31.24.254
ubuntu@ip-172-31-35-243:~$
```



Install JAVA

Install the JAVA into Jenkin's Agent Machine.

```
$ sudo apt-get update
```

```
$ sudo apt install fontconfig openjdk-17-jre : Install the JAVA 17 version per Jenkins Document.
```

```
ubuntu@ip-172-31-24-254:~$ sudo apt-get update ←
Hit:1 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [297 kB]
Fetched 423 kB in 1s (645 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-24-254:~$ ←
ubuntu@ip-172-31-24-254:~$ ←
ubuntu@ip-172-31-24-254:~$ sudo apt install fontconfig openjdk-17-jre ←
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

```
$ java --version : To check the JAVA version.
```

```
ubuntu@ip-172-31-24-254:~$ ←
ubuntu@ip-172-31-24-254:~$ java --version
openjdk 17.0.11 2024-04-16
OpenJDK Runtime Environment (build 17.0.11+9-Ubuntu-1)
OpenJDK 64-Bit Server VM (build 17.0.11+9-Ubuntu-1, mixed mode, sharing)
ubuntu@ip-172-31-24-254:~$
```

Note : This SSH Connection is a EC2-EC2 instance connection , we need to add the Private Key into Jenkin's Master GUI Page as well.



=====

Add the Agent

Click on **Manage Jenkins**:

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with options: '+ New Item', 'Build History', 'Manage Jenkins' (which is highlighted with a green border), and 'My Views'. Below the sidebar are two dropdown menus: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). On the right, there's a 'Builds' section with a table:

S	W	Name ↓
✓	☁️	django-Dheeraj-Project
...	☀️	nodeJs-Project

Below the table, there's a 'Icon:' section with buttons for 'S', 'M', and 'L'.

Manage Jenkins—> Set Up Agent

In case if you don't see this "Set Up agent" then click on —> Nodes—> New Nodes



Manage Jenkins

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

System Configuration

- Build Queue**: No builds in the queue.
- Build Executor Status**: 1 idle, 2 idle
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Tools**: Configure global settings and paths.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Appearance**: Configure the look and feel of Jenkins
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

New Node → Name → Create

New node

Node name
DheerajYadav

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

====> Number of executors : 1 “Only 1 Pipeline can run at a time”

Remote Root Directory : Create a Directory on Agent’s machine, using “**mkdir apps**” command

```
ubuntu@ip-172-31-35-243:~$ mkdir apps
ubuntu@ip-172-31-35-243:~$ ls
apps
ubuntu@ip-172-31-35-243:~$
```



Labels : Agent name “dev” Script will use this name to find the agent to build the pipeline. We can give the same Name to multiple Agents to increase the efficiency. More Agents means stability & speed to execute the pipeline.

The screenshot shows the Jenkins Node configuration page. A green box highlights the 'Labels' section. The 'Name' field contains 'DheerajYadav'. The 'Description' field contains 'This is a Jenkin's Dev Agent Node'. The 'Number of executors' field is set to '1'. The 'Remote root directory' field is set to '/home/ubuntu/apps'. The 'Labels' field contains 'dev', with a tooltip explaining it's an Agent Label Name. The 'Usage' field has the value 'Use this node as much as possible'.

=====>**Launch Method :** Please use “**agents with SSH**”

Host : IP address of **Jenkin's Agent**

Credentials: User→ ubuntu , use **Jenkins Master Private key**



Launch method ?
 Launch agents via SSH ←

Host ?
 18.184.253.198 Jenken's Agent EC2 Public IP

Credentials ?
 Click on Add, Follow the below screen shot to add it
 ubuntu (This is a Key to connect Agent Node)

Host Key Verification Strategy ? Jenkins Master will connect to Slave Agent with Private Key
 Non verifying Verification Strategy

Advanced ?

Availability ?
 Keep this agent online as much as possible

Node Properties

- Disable deferred wipeout on this node ?
- Disk Space Monitoring Thresholds
- Environment variables
- Tool Locations

====> **SSH Username with Private key** : Select this option

Jenkins Credentials Provider: Jenkins

Add Credentials

Kind
 SSH Username with private key ←

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

ID ?
 node-DheerajYadav-ssh-Key

Description ?
 This is a Key to connect Agent Node

Username
 ubuntu

====> Enter the Master Jenkin Private key here from
cat /home/ubuntu/.ssh/id_ed25519



Private Key

Enter directly

Key

```
AAAEBLser01w31LW+E/v+f5CnZZyfRXDl0aeq3PxEj1ErsDGn1vsy3aNTfCBGvvIU+Pwp
ZgnBu14JWYH6bTSCueAyAAAF3VidW50dUBpcC0xNzItMjQtMjU0AQIDBAUG
-----END OPENSSH PRIVATE KEY-----
```

Enter New Secret Below

Passphrase

Add

=====

Agent Status

Agent-Node is in-sync means it's UP & running.

Dashboard > Manage Jenkins > Nodes >

Nodes

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	2.28 GiB	0 B	2.28 GiB	0ms
	DheerajYadav	Linux (amd64)	In sync	4.46 GiB	0 B	4.46 GiB	5ms

Build Queue

No builds in the queue.

Build Executor Status

- Built-In Node
 - 1 Idle
 - 2 Idle
- DheerajYadav
 - 1 Idle

Icon: S M L

Legend



====> Add a Label "dev" & test the pipeline.

Definition

Pipeline script

Script ?

```
1 > pipeline{
2
3 >     agent{
4 >         node{
5 |             label "dev"
6
7 }
8 }
```

====> Jenkin's Agent is running & executing the pipeline with Agent Node

Dashboard > django-Dheeraj-Project > #7

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#7'

Timings

Pipeline Overview

Pipeline Console

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

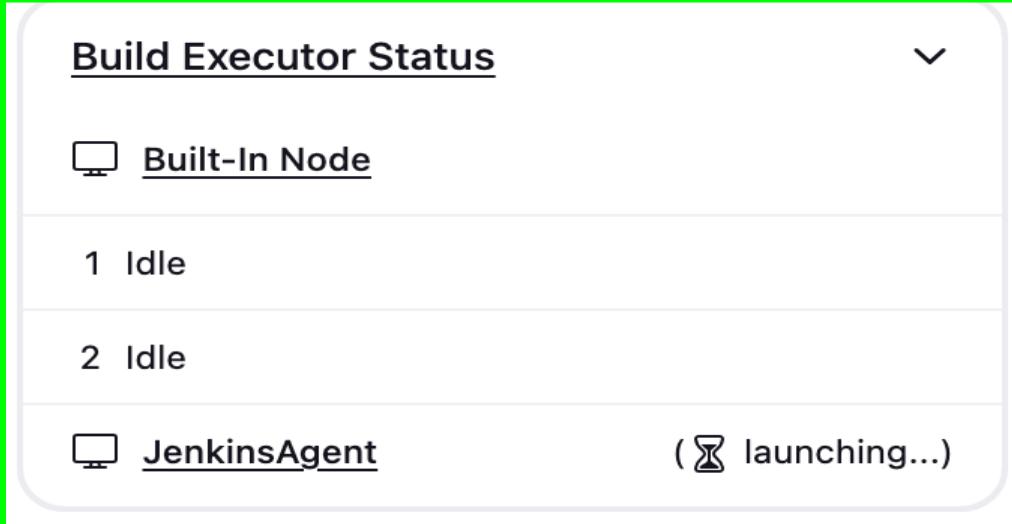
Console Output

Started by user Dheeraj Yadav
[Pipeline] Start of Pipeline
[Pipeline] node
Running on JenkinsAgent in /home/ubuntu/apps/workspace/django-Project
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] echo
Clone the Code from github repo
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Build the Docker image
[Pipeline] echo
making this typo deliberately for pipeline to fail
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (test)
[Pipeline] echo
Docker image has been tested
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Docker Container has been built successfully
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS



Agent Offline

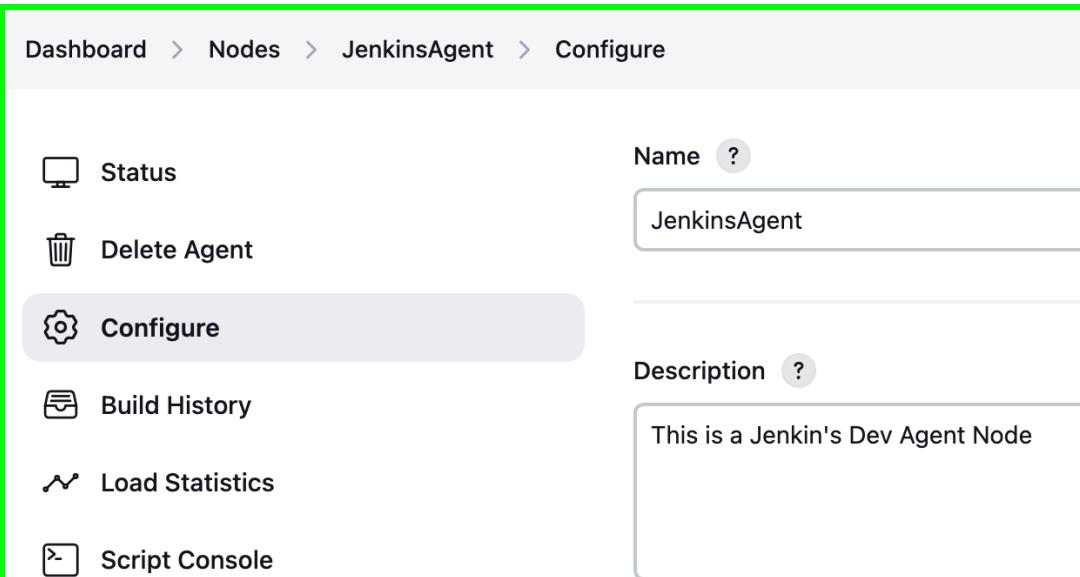
In case, you stop your Agent instance, Public IP will be changed & Either Jenkin's Agent Status will be showing “**Offline**” Or “**Launching**”



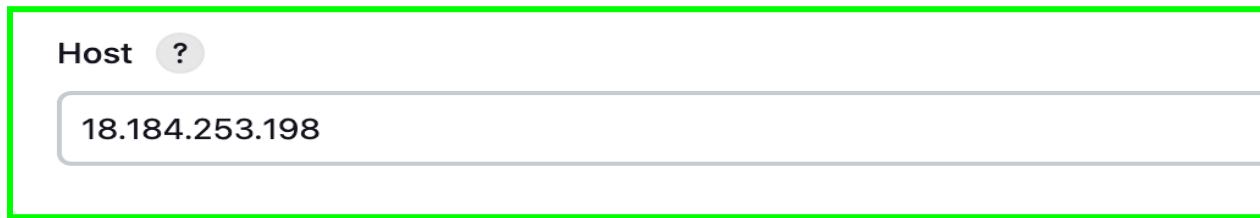
A screenshot of the Jenkins Build Executor Status page. A green box highlights the 'Built-In Node' section. It shows two nodes: '1 Idle' and '2 Idle'. Below them is a node named 'JenkinsAgent' with a PC icon, followed by '(⏳ launching...)'. There is a dropdown arrow in the top right corner of the main status area.

Click on JenkinsAgent PC icon showing above —> It will redirect you on the status page —> Select Configure & Update the Host IP with New IP address.

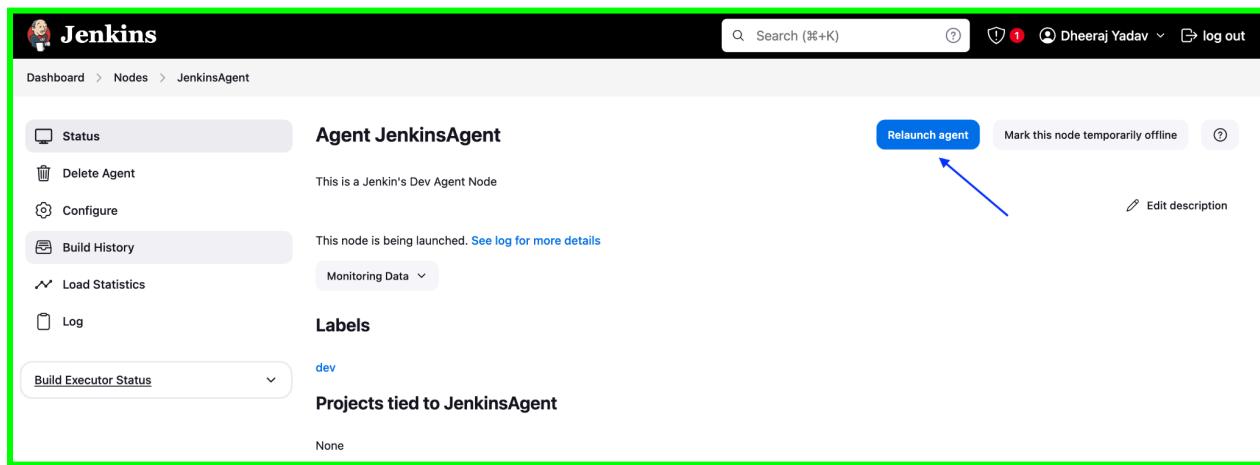
Host IP : Change with New Public IP Address.



A screenshot of the Jenkins Node configuration page for 'JenkinsAgent'. A green box highlights the 'Configure' button in the sidebar. The main panel shows the 'Name' field set to 'JenkinsAgent' and the 'Description' field containing 'This is a Jenkin's Dev Agent Node'. The sidebar also lists other options: Status, Delete Agent, Build History, Load Statistics, and Script Console.



Once you update the IP address, Click on the “Relaunch agent” icon, it will try to re-establish the connection.





Agent Connection has been Connected and online.

The screenshot shows the Jenkins Log page for the JenkinsAgent node. The log output is as follows:

```

Checking Java version in the PATH
openjdk version "17.0.12" 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu224.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu224.04, mixed mode, sharing)
[08/03/24 15:44:18] [SSH] Checking java version of /home/ubuntu/apps/jdk/bin/java
Couldn't figure out the Java version of /home/ubuntu/apps/jdk/bin/java
bash: line 1: /home/ubuntu/apps/jdk/bin/java: No such file or directory

[08/03/24 15:44:18] [SSH] Checking java version of java
[08/03/24 15:44:18] [SSH] java -version returned 17.0.12.
[08/03/24 15:44:18] [SSH] Starting sftp client.
[08/03/24 15:44:18] [SSH] Copying latest remoting.jar...
Source agent hash is 2E3A82C4A7B2851BCA33BDFC24E223C. Installed agent hash is 2E3A82C4A7B2851BCA33BDFC24E223C
Verified agent jar. No update is necessary.
Expanded the channel window size to 4MB
[08/03/24 15:44:19] [SSH] Starting agent process: cd "/home/ubuntu/apps" && java -jar remoting.jar -workDir /home/ubuntu/apps -jar-cache /home/ubuntu/apps/remoting/jarCache
Aug 03, 2024 3:44:19 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/apps/remoting as a remoting work directory
Aug 03, 2024 3:44:19 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/apps/remoting
<===[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 3206.vb_15dcf73f6a_9
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
Agent successfully connected and online

```

Agent is Online now.

The screenshot shows the Jenkins Nodes page. The table displays the following information:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	2.19 GiB	! 0 B	2.19 GiB	0ms
	JenkinsAgent	Linux (amd64)	In sync	4.22 GiB	! 0 B	4.22 GiB	59ms
Data obtained		27 sec	27 sec	27 sec	27 sec	27 sec	27 sec

Icon: S M L Legend



Jenkins Agent is showing **Idle** now.

Build Executor Status

 <u>Built-In Node</u>	▼
1 Idle	
2 Idle	
 <u>JenkinsAgent</u>	▼
1 Idle	



=====

Step -7 : Build More Pipelines

====> Created the below pipeline to build the notes-app docker image.

Pipeline Structure

```
Script ?  
1 - pipeline{  
2  
3 -   agent{  
4 -     node{  
5 -       label "dev"  
6 -     }  
7 -   }  
8  
9 -   stages{  
10 -     stage("Code"){  
11 -       steps{  
12 -         git url: "https://github.com/dheeruyadav54/django-notes-app.git", branch: "main"  
13 -         echo "Clone the Code from github repo"  
14 -       }  
15 -     }  
16 -     stage("Build"){  
17 -       steps{  
18 -         echo "Build the Docker image"  
19 -         sh "docker build -t notes-app:latest ."  
20 -         echo "making this typo deliberately for pipeline to fail"  
21 -       }  
22 -     }  
23 -     stage("test"){  
24 -       steps{  
25 -         echo "We don't have any test steps available, it will be skip"  
26 -       }  
27 -     }  
28 -     stage("Deploy"){  
29 -       steps{  
30 -         sh "docker run -d -p 8000:8000 --name notes-app notes-app:latest"  
31 -         echo "Docker Container has been built successfully"  
32 -       }  
33 -     }  
34 -   }  
35 }
```



=====

Build the pipeline

Click —< Build Now

I see the New Pipeline has been triggered but failed.

The screenshot shows the Jenkins Pipeline page for the project "django-Dheeraj-Project". The pipeline status is failed, indicated by a red circle with a white "X". The pipeline name is "django-Dheeraj-Project" and the project name is "django-Project".

On the left, there is a sidebar with the following options:

- Status
- </> Changes
- ▷ Build Now (highlighted with a blue arrow)
- ⚙ Configure
- Delete Pipeline
- ☷ Stages
- ✍ Rename
- (?) Pipeline Syntax

On the right, there is a "Permalinks" section listing recent builds:

- Last build (#8), 41 sec ago
- Last stable build (#7), 5 days 18 hr ago
- Last successful build (#7), 5 days 18 hr ago
- Last failed build (#8), 41 sec ago
- Last unsuccessful build (#8), 41 sec ago
- Last completed build (#8), 41 sec ago

Below this is a "Build History" section:

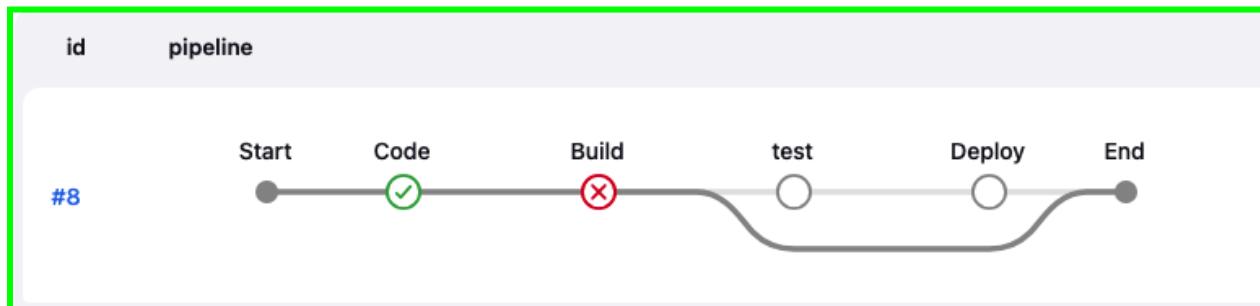
trend	Build #	Date
trend dropdown	✖ #8 (highlighted with a blue arrow)	Aug 3, 2024, 4:36 PM
trend dropdown	✓ #7	Jul 28, 2024, 9:52 PM



Pipeline Failed-1

====> Let's check the reason for failure.

Click —> Stages → Build stage has failed.



====> Click on Build Stage, it will open up the Pipeline Console Window.

1. Code has been cloned from github successfully.
2. Build stage has failed due to **docker not found**.

The Pipeline Console window for build #8. The 'Code' stage is successful. The 'Build' stage failed with a 'Failure' message. The 'Git' step 'Clone the Code from github repo' was successful. The 'Print Message' step was also successful.

====> Jenkins Agent Machine doesn't have docker installed, let's install & re-run the build.

The Pipeline Console window for build #8. The 'Build' stage failed initially. After installing Docker, the 'Build' stage was successful. The 'Build the Docker image' step was successful. The 'docker build -t notes-app:latest .' step failed with an error message: 'script returned exit code 127'.



=====

Solution1

Install Docker into Agent-Linux Machine.

```
$ sudo apt-get update  
$ sudo apt-get install docker.io
```

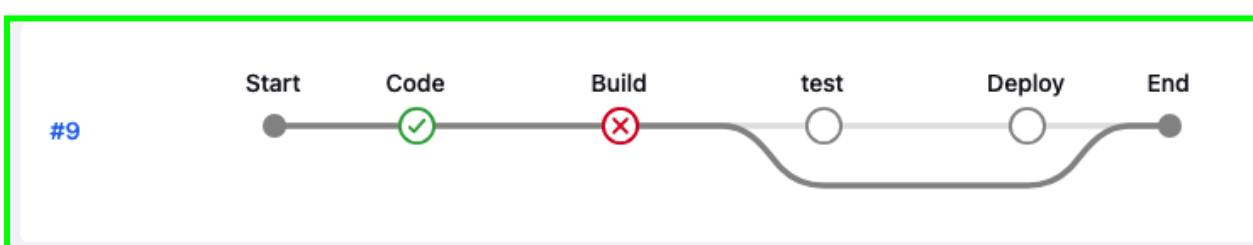
```
ubuntu@ip-172-31-35-243:~$ sudo apt-get update  
Hit:1 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Hit:2 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease  
Hit:3 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease  
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease  
Reading package lists... Done  
ubuntu@ip-172-31-35-243:~$  
ubuntu@ip-172-31-35-243:~$  
ubuntu@ip-172-31-35-243:~$  
ubuntu@ip-172-31-35-243:~$ sudo apt-get install docker.io  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done
```

Since docker has been installed, let's re-run the build stage again.

=====

Pipeline Failed-2

====> The New Pipeline has also failed, let's check it again.





Failure Reason :--> Build has failed due to docker Permission Denied.

The screenshot shows the Jenkins interface for a build that failed. The build was triggered 2 min 21 sec ago and took 3.2 sec. The stages are: Code (green), Build (yellow, marked with an 'X'), test (green), Deploy (green). The Build stage details show it started 2 min 18 sec ago, queued 0 ms, took 0.43 sec, and failed. The step 'Build the Docker image' (Shell Script) failed with the command 'docker build -t notes-app:latest'. The output shows a permission denied error: 'permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock'. The Jenkins version is v1.24.

Solution2

1. We will check which user is having a permission denied.
2. Add a shell command in the build stage “whoami” in the script pipeline.

```

14
15
16
17
18
19
20
21
22
23
  }
}
stage("Build"){
    steps{
        echo "Build the Docker image"
        sh "whoami"
        sh "docker build -t notes-app:latest ."
        echo "making this typo deliberately for pipeline to fail"
    }
}

```

Jenkins Agent User Ubuntu is getting the Permission denied, Let's add Ubuntu User into Docker Group.



Dashboard > Dheeraj Yadav > My Views > All > django-Dheeraj-Project > #10

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#10'](#)

[Timings](#)

[Git Build Data](#)

[Pipeline Overview](#)

[Pipeline Console](#)

[Restart from Stage](#)

[Replay](#)

[Pipeline Steps](#)

[Workspaces](#)

[Previous Build](#)

```

Running on JenkinsAgent in /home/ubuntu/apps/workspace/django-Project
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Fetching changes from the remote Git repository
Checking out Revision 66b22abf8fb7d753fe8f9c25b77d167af4c96cc4 (refs/remotes/origin/main)
Commit message: "Update Jenkinsfile"
> git rev-parse --resolve-git-dir /home/ubuntu/apps/workspace/django-Project/.git # timeout=10
> git config remote.origin.url https://github.com/dheeruyadav54/django-notes-app.git # timeout=10
Fetching upstream changes from https://github.com/dheeruyadav54/django-notes-app.git
> git -version # git version 2.43.0*
> git fetch --tags --force --progress -- https://github.com/dheeruyadav54/django-notes-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f 66b22abf8fb7d753fe8f9c25b77d167af4c96cc4 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D main # timeout=10
> git checkout -b main 66b22abf8fb7d753fe8f9c25b77d167af4c96cc4 # timeout=10
> git rev-list --no-walk 66b22abf8fb7d753fe8f9c25b77d167af4c96cc4 # timeout=10
> git [Pipeline] echo
Clone the Code from github repo
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Build the Docker image
[Pipeline] sh
+ whoami
ubuntu
[Pipeline] sh
+ docker build -t notes-app:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?
```

\$ cat /etc/group : It will show you if an ubuntu user has docker Group access.

ubuntu@ip-172-31-35-243:~\$ cat /etc/group

Ubuntu User does not have access , Let's add into Docker Group.

\$ sudo usermod -aG docker ubuntu : This Command will add User into Docker Group.

ubuntu:x:1000:
docker:x:113:
ubuntu@ip-172-31-35-243:~\$ sudo usermod -aG docker ubuntu

\$ cat /etc/group : Ubuntu user has access now.

docker:x:113:ubuntu



\$ docker ps : tried running this command , still getting permission denied, which is kind of a common issue. **Let's fix this.**

```
ubuntu@ip-172-31-35-243:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-35-243:~$
```

\$ newgrp docker : This command will Activate the changes to groups. Now the "**docker ps**" command is running,

```
ubuntu@ip-172-31-35-243:~$ newgrp docker
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$ docker ps
CONTAINER ID        IMAGE               COMMAND       CREATED          STATUS          PORTS          NAMES
ubuntu@ip-172-31-35-243:~$
```

Note : if you still get the permission error, we can reboot the EC2 machine once.

```
ubuntu@ip-172-31-35-243:~$ sudo reboot
Broadcast message from root@ip-172-31-35-243 on pts/0 (Sat 2024-08-03 19:37:25 UTC):
The system will reboot now!
```

Let's build our pipeline again

Nice, it has worked this time, Container has been deployed.

Let's Verify it.



id pipeline

```

graph LR
    Start((Start)) --> Code((Code))
    Code --> Build((Build))
    Build --> test((test))
    test --> Deploy((Deploy))
    Deploy --> End((End))

```

#13

Jenkins

Dashboard > Dheeraj Yadav > My Views > All > django-Dheeraj-Project > #13 > Pipeline Console

Build #13

Success 16 min ago in 47 sec

Code, Build, test, Deploy

Stage 'Deploy'
 Started 15 min ago
 Queued 0 ms
 Took 0.97 sec
 Success
[View as plain text](#)

docker run -d -p 8000:8000 --name notes-app notes-app:latest
 Shell Script
 0.91 sec

Docker Container has been built successfully
 Print Message
 Docker Container has been built successfully
 13 ms

Verify the Build

\$ docker ps : It is showing that Container has been Deployed Successfully.

```

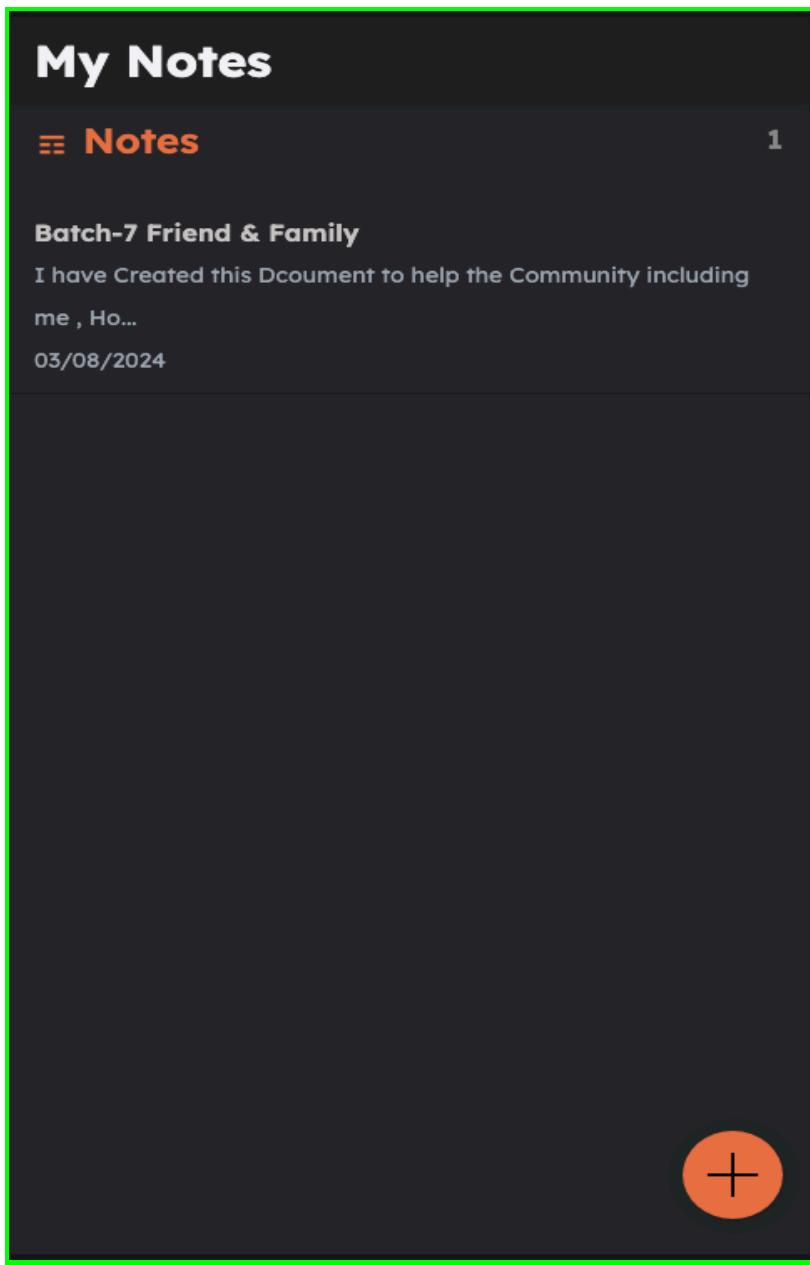
ubuntu@ip-172-31-35-243:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
8eac590723ff        notes-app:latest   "/bin/sh -c 'python ..."   16 minutes ago   Up 16 minutes   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   notes-app
ubuntu@ip-172-31-35-243:~$
```

Note: Notes application is running on port 8000, please make sure port 8000 has been opened under **"inbound rules"** Jenkins Agent EC2 Machine.

sgr-0c5f9785b36e92799	Custom TCP	TCP	8000	Custom	0.0.0.0/0	Delete



Notes app Project has been deployed through Jenkins Pipeline.



The screenshot shows a mobile application titled "My Notes". At the top, there is a navigation bar with the title "My Notes" and a subtitle "Notes" followed by a number "1". Below the navigation bar, there is a section titled "Batch-7 Friend & Family" containing the text: "I have Created this Dcoument to help the Community including me , Ho...". Below this text, the date "03/08/2024" is displayed. At the bottom right of the screen, there is a large orange circular button with a white plus sign (+) in the center, used for adding new notes.



Step -8 : Credential Binding & Image Pushed to DockerHub

Manage Jenkins—> Security —> Credentials : To Add the user & DockerHub Token here.

The screenshot shows the Jenkins Manage Jenkins dashboard. In the center, under the 'System Configuration' heading, there is a 'Security' section. A blue arrow points from the text 'Click on the global icon.' to the 'Domain' field, which contains '(global)' and is highlighted with a blue border.

Click on the global icon.

The screenshot shows the Jenkins Credentials page. It displays a single credential entry in the 'Stores scoped to Jenkins' section. The entry has the following details: ID 'node-DheerajYadav-ssh-Key', Name 'ubuntu (This is a Key to connect Agent Node)', and Description 'ubuntu (This is a Key to connect Agent Node)'. The 'Domain' field is also visible above the entry, containing '(global)'. A blue arrow points to the '(global)' text in the Domain field.

Click on Add Credentials

The screenshot shows the Jenkins Global credentials (unrestricted) page. It lists a single credential entry: 'node-DheerajYadav-ssh-Key' with the name 'ubuntu (This is a Key to connect Agent Node)', kind 'SSH Username with private key', and description 'This is a Key to connect Agent Node'. A blue arrow points to the '+ Add Credentials' button at the top right of the page.



Kind : Username with password

Username : dockerHub Username

Password : generate the Personal Access token per below steps.

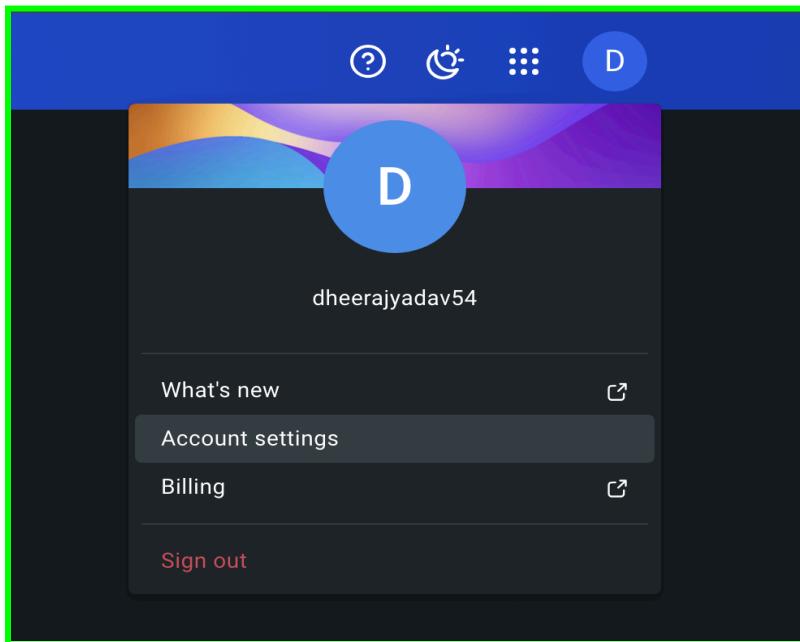
ID : It will be representing username & password so in the script pipeline we will use this ID.

The screenshot shows the Jenkins 'New credentials' configuration page. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Username' field contains 'dheeruyadav54'. The 'Password' field is empty and has a placeholder '.....'. There is a note above the password field: 'Generate DockerHub Personal Acces Token & enter it'. A red arrow points from this note to the password input field. The 'ID' field is filled with 'dockerHubCreds'. The 'Description' field contains 'This is a Credentials for DockerHub'. At the bottom is a 'Create' button.



Generate the Personal Access Token on Docker Hub

- Docker Hub Account —> Account Setting



- Personal access tokens → Generate new token

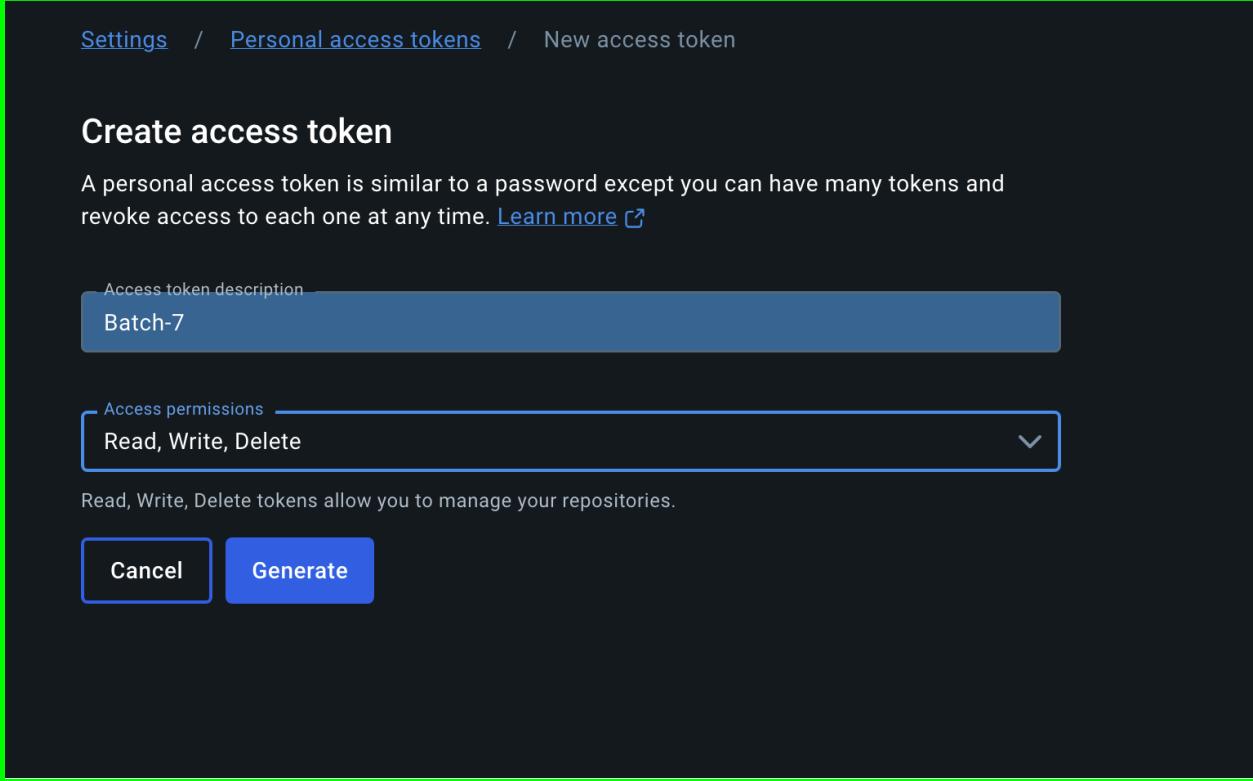
A screenshot of the Docker Hub personal access tokens page. It shows a table with one token entry and a 'Generate new token' button.

Description	Scope	Status	Source	Created	Last Used	More
Batch-7	Read, Write, Delete	Active	Manual	Jul 25, 2024 at 01:10:36	Jul 25, 2024 at 01:10:43	⋮

Rows per page: 10 | 1–1 of 1



Click on Generate & Copy the token. Once you Copied the Token, go to Jenkins Credentials page & paste under Password Section.



The screenshot shows the Jenkins 'Create access token' dialog. At the top, it says 'Settings / Personal access tokens / New access token'. Below that is the title 'Create access token'. A descriptive text states: 'A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more ↗](#)'. There are two input fields: 'Access token description' containing 'Batch-7' and 'Access permissions' set to 'Read, Write, Delete'. A note below says 'Read, Write, Delete tokens allow you to manage your repositories.' At the bottom are 'Cancel' and 'Generate' buttons.

Now we have Credentials configured, let's build the Pipeline & push the image to docker HUB.



The screenshot shows the Jenkins 'Credentials' configuration screen. It lists a credential named 'dockerHubCreds' of type 'Username with password'. The details shown are 'dheeruyadav54*****' (This is a Credentials for DockerHub) and 'This is a Credentials for DockerHub'. There is also a small gear icon for editing.



Build the Docker Compose pipeline

Note : we need to have docker compose installed, I have added the command in the pipeline itself. Also stopping & removing the existing docker container, Marked in RED Box.

```
Script ?  
1 pipeline{  
2     agent{  
3         node{  
4             label "dev"  
5         }  
6     }  
7     stages{  
8         stage("Code"){  
9             steps{  
10                git url: "https://github.com/dheeruyadav54/django-notes-app.git", branch: "main"  
11                echo "Clone the Code from github repo"  
12            }  
13        }  
14        stage("Build"){  
15            steps{  
16                echo "Build the Docker image"  
17                sh "whoami"  
18                sh "docker build -t notes-app-jenkins:latest ."  
19                echo "making this typo deliberately for pipeline to fail"  
20            }  
21        }  
22    }  
23    stage("Push to DockerHub"){  
24        steps{  
25            echo "Install the Docker Compose package into Jenkins Agent Machine"  
26            sh "sudo apt-get install docker-compose-v2"  
27            echo "Push the image to DockerHub"  
28            withCredentials{  
29                [usernamePassword(  
30                    credentialsId:"dockerHubCreds",  
31                    passwordVariable:"dockerHubpass",  
32                    usernameVariable:"dockerHubuser"  
33                )  
34            ]  
35        }  
36        steps{  
37            sh "docker image tag notes-app-jenkins:latest ${env.dockerHubuser}/notes-app-jenkins:latest"  
38            sh "docker login -u ${env.dockerHubuser} -p ${env.dockerHubpass}"  
39            sh "docker push ${env.dockerHubuser}/notes-app-jenkins:latest"  
40        }  
41    }  
42    stage("Deploy"){  
43        steps{  
44            sh "docker stop 8eac590723ff && docker rm 8eac590723ff && docker compose up -d"  
45            echo "Docker Container has been built successfully"  
46        }  
47    }  
48}  
49}  
50}
```



Let's Build the Pipeline & push the image to DockerHub:

Pipeline Succeed

Console Output

```
Started by user Dheeraj Yadav
[Pipeline] Start of Pipeline
[Pipeline] node
Running on JenkinsAgent in /home/ubuntu/apps/workspace/django-Project
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Fetching changes from the remote Git repository
Checking out Revision d678d955ad1b68a775a5751c1630c7e422cea5c5 (refs/remotes/origin/main)
Commit message: "Added my DockerHub UserName"
> git rev-parse --resolve-git-dir /home/ubuntu/apps/workspace/django-Project/.git # timeout=10
> git config remote.origin.url https://github.com/dheeruyadav54/django-notes-app.git # timeout=10
Fetching upstream changes from https://github.com/dheeruyadav54/django-notes-app.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/dheeruyadav54/django-notes-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f d678d955ad1b68a775a5751c1630c7e422cea5c5 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D main # timeout=10
> git checkout -b main d678d955ad1b68a775a5751c1630c7e422cea5c5 # timeout=10
> git rev-list --no-walk d678d955ad1b68a775a5751c1630c7e422cea5c5 # timeout=10
[Pipeline] echo
Clone the Code from github repo
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Build the Docker image
[Pipeline] sh
+ whoami
ubuntu
[Pipeline] sh
+ docker build -t notes-app-jenkins:latest .
```



Image has been pushed to DockerHub

```
Push the image to DockerHub ←  
[Pipeline] withCredentials  
Masking supported pattern matches of $dockerHubpass  
[Pipeline] {  
[Pipeline] sh  
+ docker image tag notes-app-jenkins:latest dheerajyadav54/notes-app-jenkins:latest  
[Pipeline] sh  
Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.  
Affected argument(s) used the following variable(s): [dockerHubpass]  
See https://jenkins.io/redirect/groovy-string-interpolation for details.  
+ docker login -u dheerajyadav54 -p **** ←  
WARNING! Using --password via the CLI is insecure. Use --password-stdin.  
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
  
Login Succeeded ←  
[Pipeline] sh  
+ docker push dheerajyadav54/notes-app-jenkins:latest ←  
The push refers to repository [docker.io/dheerajyadav54/notes-app-jenkins]  
ab23de4977a9: Preparing ←  
e88bdbea9065: Preparing ←
```

Pipeline Succeed & removed the existing Docker container too.

```
ab23de4977a9: Pushed ←  
latest: digest: sha256:7ef6bfff5ece5643a0da29da3fa47932706711dc4d8cefcd24cefcd23e10b85d31 size: 2844  
[Pipeline] } ←  
[Pipeline] // withCredentials  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Deploy)  
[Pipeline] sh  
+ docker stop 8eac590723ff  
8eac590723ff  
+ docker rm 8eac590723ff  
8eac590723ff  
+ docker compose up -d  
Network django-project_default Creating  
Network django-project_default Created  
Container django-project-web-1 Creating  
Container django-project-web-1 Created  
Container django-project-web-1 Starting  
Container django-project-web-1 Started  
[Pipeline] echo  
Docker Container has been built successfully  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline ←  
Finished: SUCCESS ←
```



Step -9 : Jenkins Webhook

Go to your Github repository —> Setting —> webhooks

The screenshot shows the GitHub repository settings page for 'dheeruyadav54 / django-notes-app'. The 'Settings' tab is selected. On the left, there's a sidebar with various options like General, Access, Collaborators, etc. The 'Actions' section is expanded, and the 'Webhooks' option is highlighted with a green box and a red arrow pointing to it. The main content area shows the 'General' settings for the repository, including fields for 'Repository name' (django-notes-app) and 'Default branch' (main). There are also sections for 'Template repository' and 'Require contributors to sign off on web-based commits'. At the bottom, there's a 'Social preview' section with a placeholder for an image.

Add Webhook —> it might ask your github password to login again.

The screenshot shows the 'Webhooks' configuration page. The title is 'Webhooks'. A button labeled 'Add webhook' is visible in the top right corner. Below the title, there's a descriptive text: '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](#)'. The entire screenshot is framed by a thick green border.



Payload URL : This is a Jenkins Master URL

http://<EC@PublicIP>:8080/github-webhook

SSL verification : We have disabled it, since it will require certificate handshake.

Just the Push Event : Selected this Option, meaning whenever there is a push in the repository.

Webhooks / Add webhook

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 *
http://3.76.204.109:8080/github-webhook/ Jenkins Master URL

Content type *
application/x-www-form-urlencoded

Secret

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



Webhook Connection Status & build Trigger Config

It will take a few seconds to become Active.

The screenshot shows the Jenkins 'Webhooks' configuration page. At the top, there is a button labeled 'Add webhook'. Below it, a descriptive text explains what webhooks are: '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](#)'. A green arrow points from the text to the first webhook entry. The first entry is for GitHub, with the URL 'http://3.76.204.109:8080/github-webhook' and the status '(push)'. It includes 'Edit' and 'Delete' buttons. Below this, a message says 'Last delivery was successful.'

Enable the GitHub hook Trigger & Poll SCM Configuration in Jenkins Project Template.

The screenshot shows the Jenkins 'Build Triggers' configuration page. A blue box highlights the 'GitHub hook trigger for GITScm polling' and 'Poll SCM' options, both of which are checked. Below this, a section titled 'Schedule' contains a note: 'No schedules so will only run due to SCM changes if triggered by a post-commit hook'. Other options shown include 'Ignore post-commit hooks', 'Quiet period', and 'Trigger builds remotely (e.g., from scripts)'. At the bottom, there are 'Save' and 'Apply' buttons, with 'Save' being highlighted by a blue box.



Commit to Trigger Webhook

Now as soon as you make changes in the GitHub repository it will create a GitHub webhook to Jenkins & pipeline will be triggered.

Let's make a small change & commit & see pipeline triggers or not.

As soon you push a commit it has triggered a new **Pipeline #19**

The screenshot displays two side-by-side browser windows. The left window is the Jenkins dashboard for the 'django-Dheeraj-Project'. It shows a build history with three entries: '#18' (Aug 4, 2024, 2:35 PM), '#17' (Aug 4, 2024, 12:30 PM), and '#19' (highlighted with a blue oval). The right window is a GitHub repository page for 'dheeruyadav54/django-notes-app'. It shows a file named 'docker-compose.yml' with a recent commit by 'dheeruyadav54' titled 'Added my DockerHub UserName' made 3 hours ago. The commit message includes the Jenkins build number '#19'.

End of Jenkins Fundamentals