Jenkins:

Jenkins is an open source, Java-based automation server that offers an easy way to set up a continuous integration and continuous delivery (CI/CD) pipeline.

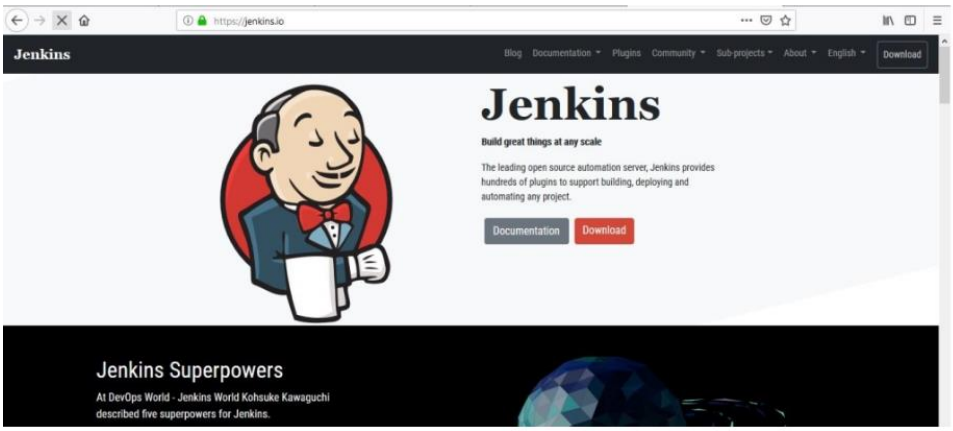
Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control. Continuous Delivery (CD) is the process to build, test, configure and deploy from a build to a production environment.

Jenkins is a tool that is used for automation, and it is an open-source server that allows all the developers to build, test and deploy software. It works or runs on java as it is written in java. By using Jenkins, we can make a continuous integration of projects(jobs) or end-to-endpoint automation.

Steps to Install Jenkins:

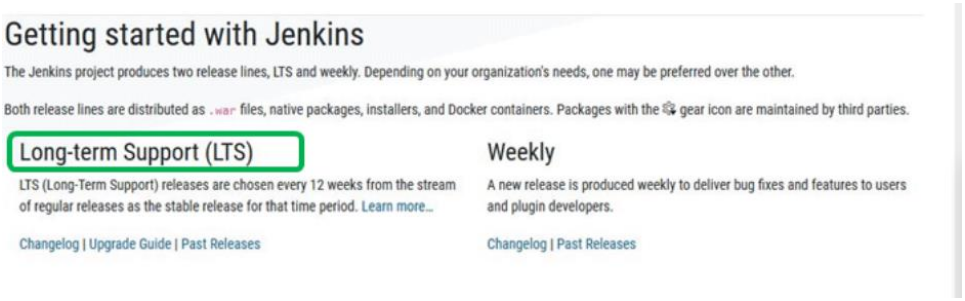
Step 1: go to official jenkins website and click on downloads

https://jenkins.io



Step 2: Once we are on the Jenkins website, you will see the ‘Download’ option available in the dashboard. There are 2 types of releases available.

- Long-term support release
- Weekly release



We will use the long term support link for the ubuntu platform

Follow provided command instructions to download and run jenkins.

Step 3: Bydefault jenkins runs with port 8080 . Search hostname/ip:8080

Step 4: Now the first thing to unlock Jenkins is 'Password'; yes, we need to provide the initial admin password.

Directory: /var/lib/Jenkins/secrets/initialAdminPassword

Step 5: copy the following details and paste them into ubuntu server

[] Cat /var/lib/Jenkins/secrets/initialAdminPassword

[] 123jdsjfdsfmkf3

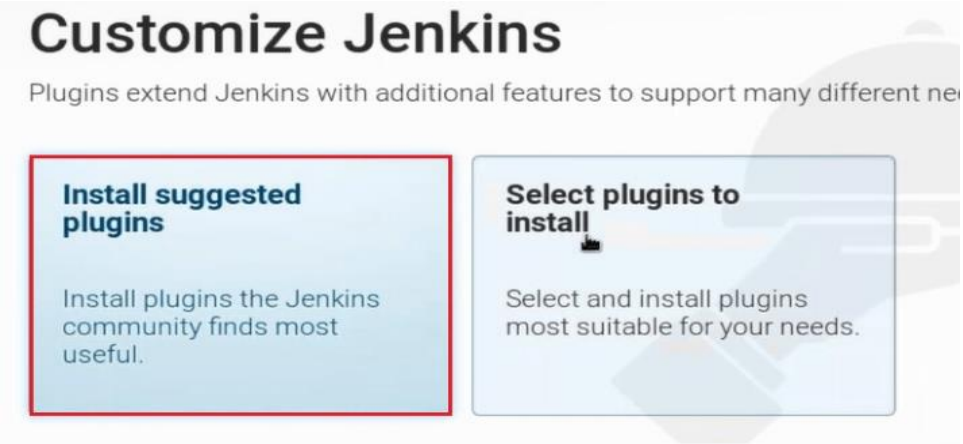
Step 6: Copy this password and paste it into the Jenkins window opened in the browser.

I have pasted my password and clicked Continue.

Step 7: Now, on the next screen, we will get the 'Customize Jenkins' screen. Again, there are two options for the users to prefer.

- Install suggested plugins. (I'll go with suggested plugins)
- Select plugins to install.

We will choose the first option, as it is the most used list of plugins preferred by the community members. Also, if we have any specific purpose, then only prefer the 2nd option.

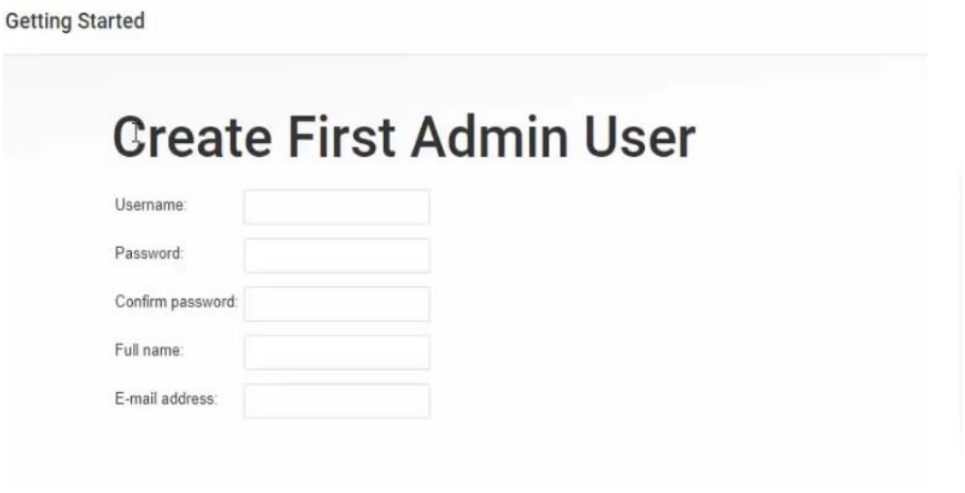


With this, the recommended plugins will start to download.

Step 9: Fill the following fields:

- Username
- Password
- Confirm Password
- Full Name

- E-mail address

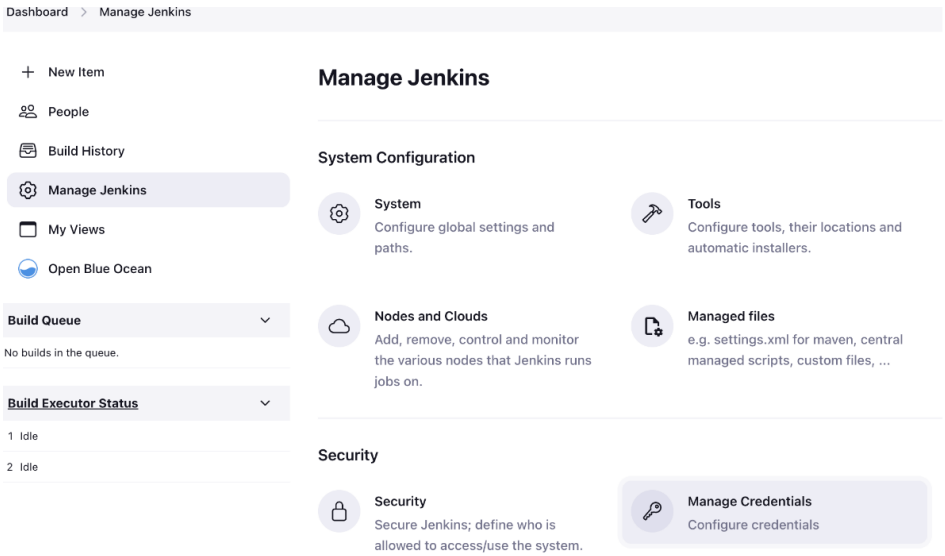


Once the details are provided, click on the ‘Save and Finish’ button, and we will get the screen. This means Jenkins is installed properly and it is ready to start.

Adding new global credentials:

1) If required, ensure you are logged in to Jenkins (as a user with the Credentials > Create permission).

2) From the Jenkins home page (i.e., the Dashboard of the Jenkins classic UI), click Manage Jenkins > Manage Credentials.



3) Under Stores scoped to Jenkins on the right, click on Jenkins.

4) Under System, click the Global credentials (unrestricted) link to access this default domain.

5)Click **Add Credentials** on the left.
Note: If there are no credentials in this default domain, you could also click the **add some credentials** link (which is the same as clicking the **Add Credentials** link).

A) From the **Kind** field, choose the type of credentials to add.

B) From the **Scope** field, choose either:

- **Global** - if the credential/s to be added is/are for a Pipeline project/item. Choosing this option applies the scope of the credential/s to the Pipeline project/item "object" and all its descendant objects.
- **System** - if the credential/s to be added is/are for the Jenkins instance itself to interact with system administration functions, such as email authentication, agent connection, etc. Choosing this option applies the scope of the credential/s to a single object only.

C) Add the credentials themselves into the appropriate fields for your chosen credential type:

- **Secret text** - copy the secret text and paste it into the **Secret** field.
- **Username and password** - specify the credential's **Username** and **Password** in their respective fields.
- **Secret file** - click the **Choose file** button next to the **File** field to select the secret file to upload to Jenkins.
- **SSH Username with private key** - specify the credentials **Username**, **Private Key** and optional **Passphrase** into their respective fields.
  Note: Choosing **Enter directly** allows you to copy the private key's text and paste it into the resulting **Key** text box.
- **Certificate** - specify the **Certificate** and optional **Password**. Choosing **Upload PKCS#12 certificate** allows you to upload the certificate as a file via the resulting **Upload certificate** button.
- **Docker Host Certificate Authentication** - copy and paste the appropriate details into the **Client Key**, **Client Certificate** and **Server CA Certificate** fields.

D) In the ID field, specify a meaningful credential ID value - for example, `jenkins-user-for-xyz-artifact-repository`. The inbuilt (default) credentials provider can use uppercase or lowercase letters for the credential ID, as well as any valid separator character, other credential providers may apply further restrictions on allowed characters or lengths. However, for the benefit of all users on your Jenkins instance, it is best to use a single and consistent convention for specifying credential IDs.

E) Specify an optional **Description** for the credential/s.

D) Click **OK** to save the credentials.

Freestyle Project:

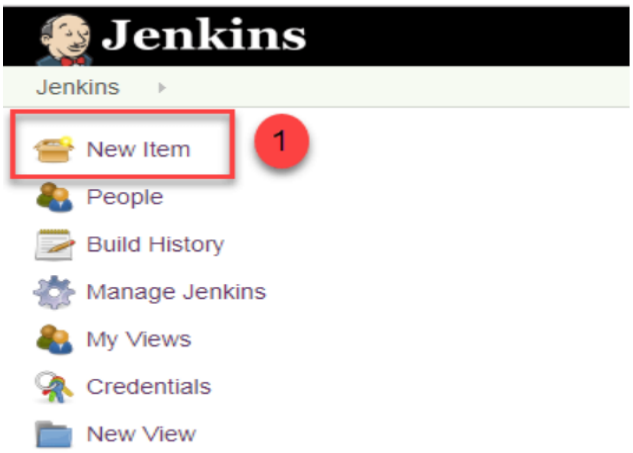
Jenkins Freestyle Project is a repeatable build job, script, or pipeline that contains steps and post-build actions. It is an improved job or task that can span multiple operations. It allows you to configure build triggers and offers project-based security for your Jenkins project. It also offers plugins to help you build steps and post-build actions.

The freestyle build job is a highly flexible and easy-to-use option. You can use it for any type of project; it is easy to set up, and many of its options appear in other build jobs. Below is a step-by-step process to create job in Jenkin

Steps to freestyle projects:

Step 1: Create New Item

Click on "**New Item**" at the top left-hand side of your dashboard.



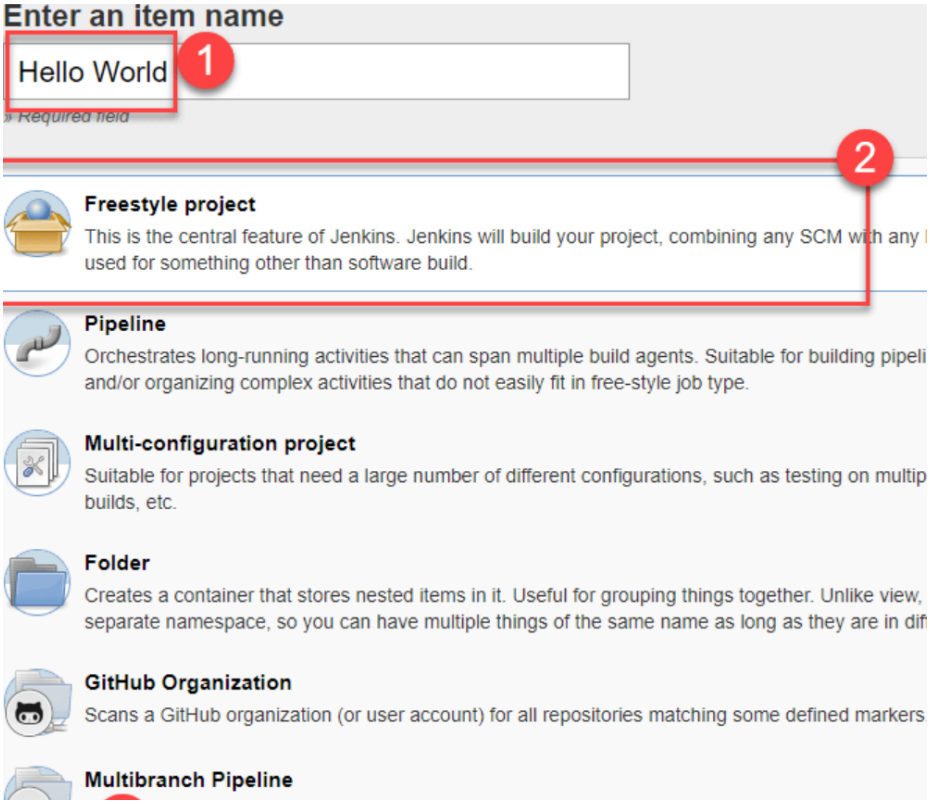
Step 2: Enter Item details

In the next screen,

A) Enter the name of the item you want to create.

B) Select Freestyle project

C) Click Okay



Step 3: Enter Project details

Enter the details of the project you want to test.

Step 4: Enter repository URL

Under Source Code Management, Enter your repository URL. We have a test repository located at https://github.com/kriru/firstJava.git

It is also possible for you to use a local repository.

If your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then pull the source code from your GitHub repository.

Step 5: Save the project

When you have entered all the data,

A) Click **Apply**
B) **Save** the project.

Step 6: Build Source code

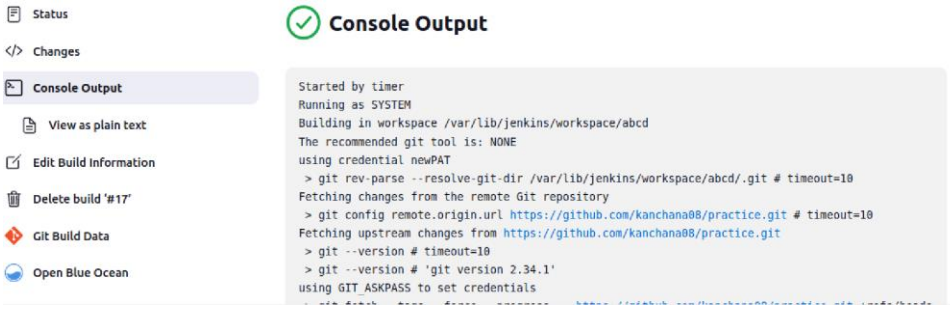
Now, in the main screen, Click the **Build Now** button on the left-hand side to build the source code.

Step 7: Check the status

After clicking on **Build now,** you can see the status of the build you run under **Build History**.

Step 8: See the console output

Click on the **build number** and then Click on **console output** to see the status of the build you run. It should show you a message of success, provided you have followed the setup properly as shown in the Jenkins create new job example below.



Pipeline project:

Jenkins Pipeline (or simply "Pipeline") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins. A continuous delivery pipeline is an automated expression of your process for getting software from version control right through to your users and customers.

There are two types of Jenkins pipeline syntax

1) Declarative
2) Scripted

Declarative:

Declarative pipeline syntax offers an easy way to create pipelines. It contains a predefined hierarchy to create Jenkins pipelines. It gives you the ability to control all aspects of a pipeline execution in a simple, straight-forward manner.

Scripted:

Scripted Jenkins pipeline runs on the Jenkins master with the help of a lightweight executor. It uses very few resources to translate the pipeline into atomic commands. Both declarative and scripted syntax are different from each other and are defined totally differently.

Steps to pipeline project:

Step 1: Install the **Pipeline plugin** through the **Manage Jenkins > Plugins** page

Step 2: After installing the plugin, restart Jenkins so that the plugin is ready to use

Step 3: Click the **New Item** menu within Jenkins

Step 4:

A) Enter the name of the item you want to create.

B) Select **Build a pipeline view** under **options**

C) Click **ok**

Step 5: In the next page, you will be asked for some more details to configure your Jenkins pipeline. Just accept the default settings, and make sure you choose the first job under the settings.

Click on **Apply** and then **OK**.

Step 6: Click the **Save** button and watch your first Pipeline run

Example : code for git checkout

```
pipeline{
agent any
stages{
stage('checkout'){
step{
git branch : 'master',
credentialID:'PAT'
url:github url which rep what to clone
}
}
}
}
```

Jenkins Build Triggers:

1. Trigger builds remotely

If you want to trigger your project built from anywhere anytime then you should select **Trigger builds remotely** option from the build triggers.

You'll need to provide an authorization token in the form of a string so that only those who know it would be able to remotely trigger this project's builds. This provides the predefined URL to invoke this trigger remotely.

[] predefined URL to trigger build remotely: JENKINS_URL/job/JobName/build?token=TOKEN_NAME JENKINS_URL: the IP and PORT which the Jenkins server is running TOKEN_NAME: You have provided while selecting this build trigger.

2. Build after other projects are built

If your project depends on another project build, then you should select **Build after other projects are built** option from the build triggers.

In this, you must specify the project (Job) names in the **Projects to watch** field section and select one of the following options:

a. Trigger only if the build is stable
Note: A build is stable if it was built successfully, and no publisher reports it as Unstable

b. Trigger even if the build is unstable

Note: A build is unstable if it was built successfully, and one or more publishers report it unstable

c. Trigger even if the build fails

3.Build periodically

If you want to schedule your project build periodically then you should select the **Build periodically** option from the build triggers.

You must specify the periodical duration of the project build in the scheduler field section

Syntax:

MINUTE HOUR DOM MONTH DOW

Examples:

```
every fifteen minutes (perhaps at :07, :22, :37, :52)
H/15 * * * *
```

4.GitHub webhook trigger for GITScm polling

A webhook is an HTTP callback, an HTTP POST that occurs when something happens through a simple event-notification via HTTP POST.

GitHub webhooks in Jenkins are used to trigger the build whenever a developer commits something to the branch.

Let's see how to add build a webhook in GitHub and then add this webhook in Jenkins.

A) Go to your project repository.
B) Go to "settings" in the right corner.
C) Click on "webhooks."
D) Click "Add webhooks."

E) Write the Payload URL as

```
http://e330c73d.ngrok.io/github-webhook
//This URL is a public URL where the Jenkins server is running
```

5.Poll SCM

Poll SCM periodically polls the SCM to check whether changes were made (i.e., new commits) and builds the project if new commits were pushed since the last build.

You must schedule the polling duration in the scheduler field. As we explained above in the Build periodically section. You can see the Build periodically section to know how to schedule.

After successfully scheduled, the scheduler polls the SCM according to your specified duration in scheduler field and builds the project if new commits were pushed since the last build.

Email Notification in Jenkins:

Step 1: Click the 'Manage Jenkins' menu option displayed at the right side of the screen. You will be redirected to the 'Manage Jenkins' page, where you need to select the 'Manage Plugin' option.

Step 2: Click the 'Available' tab present at the top of the 'Manage Plugin' page.

Step 3: Start typing ‘Notification’ in the ‘Filter’ field displayed at the top-right side of the ‘Manage Plugin’ page. Click the checkbox next to the ‘Email-ext plugin’ option. Click the ‘Install without restart’ button.

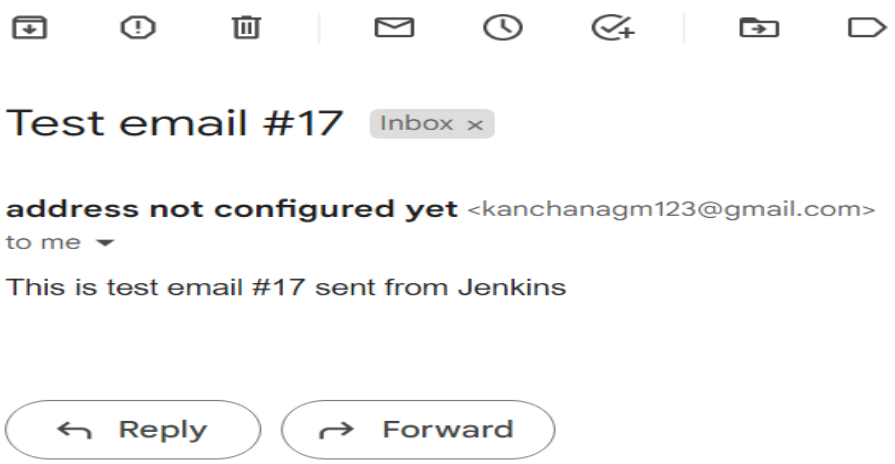
Step 4: Now, click the checkbox next to the ‘Email-ext Template Plugin’ option. Click the ‘Install without restart’ button.

Step 5: Go to the Jenkins home page and click the ‘Manage Jenkins’ menu option. Then, select the ‘Configure System’ option

Step 6: Enter the SMTP server name under ‘**Email Notification**’. Click the ‘Advanced’ button and then click the checkbox next to the ‘Use SMTP Authentication’ option. Now, set the following fields.

-
    - **SMTP server name** : smtp.gmail.com
    - **User name**: user_email_id@gmail.com
    - **Password**: 123456 (app psswd generated in google account)
    - **Use SSL** : Checked
    - **SMTP Port**: 456

Step 7: Check the email notification functionality by clicking the checkbox next to the ‘Test configuration by sending Test e-mail recipient’ option. Enter a valid email id and click the ‘Test configuration’ button to check whether the email id is valid or not.



Step 8: Go to the home page and click on a created job, like Homes. Then, click the ‘Configure’ option.

Step 9: Click the ‘Add post-build action’ drop-down.

Step 10: Select the ‘E-mail Notification’ value.

Step 11: Enter the recipient email id in the ‘E-mail Notification’ box and select the checkbox next to the ‘Send e-mail for every unstable build’ option.

Step 12: Click the ‘Add post-build action’ drop-down and select the ‘Editable Email Notification’ value.

ill the ‘Editable Email Notification’ fields.

- Project Recipient List : email_id@gmail.com

Step 13: Click the ‘Advance Settings…’ button in the ‘Editable Email Notification’ box.

Step 14: Click the ‘Add Trigger’ drop-down and select the ‘Always’ option.

Step 15: Click the ‘Save’ button.

Step 16: Go to the home page and click on the job, like Homes.

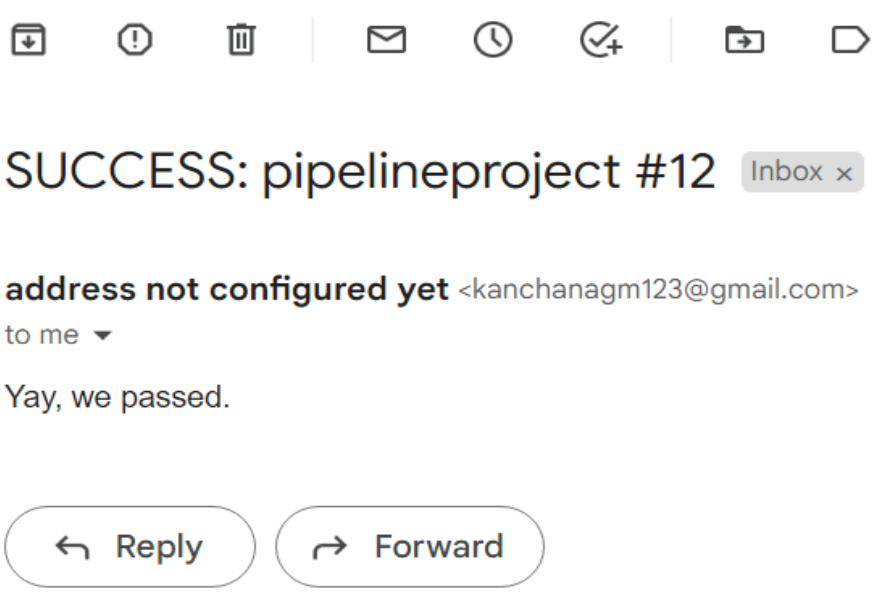
Step 17: Click the ‘Build now’ link and check the email id after the job execution.

Declarative pipeline code for email notification:

```
pipeline{
   agent any
   stages{
      stage('checkout'){
         steps{
            git credentialsId: 'newPAT', url:
'https://github.com/kanchana08/docker-compose-sample.git'
            }

      }
      stage('docker build'){
         steps{
            sh 'docker build -t kanchana/new:v1 .'
```

```
        }
    }


}


post {
    success {
        mail to:"kanchanagm123@gmail.com", subject:"SUCCESS: ${currentBuild.fullDisplayName}", body: "Yay, we passed."
    }
    failure {
        mail to:"kanchanagm123@gmail.com", subject:"FAILURE: ${currentBuild.fullDisplayName}", body: "Boo, we failed."
    }
}
}
```

above code we get email notification if build is success and failure



CI/CD pipeline for docker build and docker push:

Provided docker hub credentials and git hub credentials.

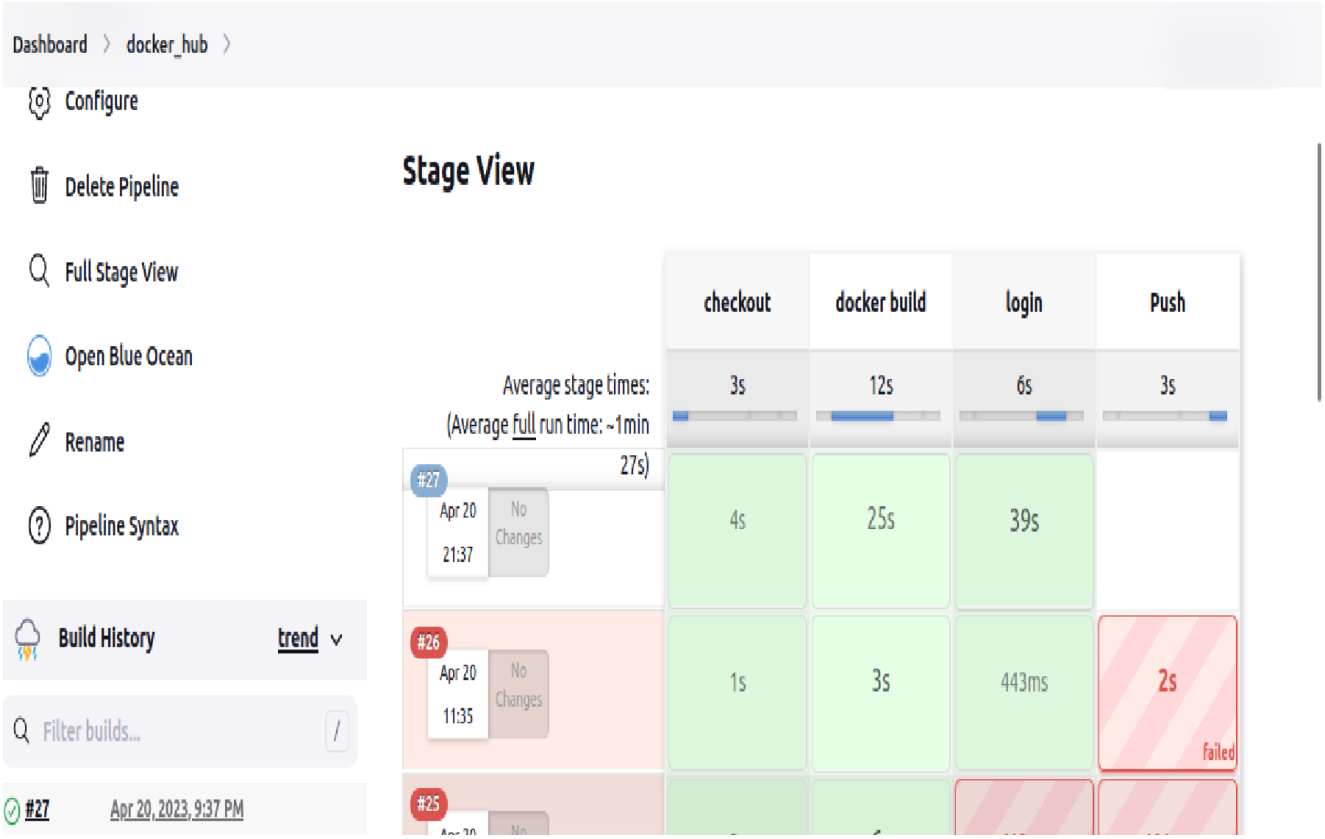
Installed docker related plugins and git related plugins.

Using declarative pipeline by mentioning stages and step accordingly.

Build the job successfully and docker image build successfully

Pushed to Docker hub.

Output:


Dashboard
docker_hub
Configure
Delete Pipeline
Full Stage View
Open Blue Ocean
Rename
Pipeline Syntax
Build History
trend
Filter builds...
#27
Apr 20, 2023, 9:37 PM
Stage View
checkout
docker build
login
Push
Average stage times:
(Average full run time: ~1min 27s)
3s
12s
6s
3s
#27
Apr 20
21:37
No Changes
4s
25s
39s
#26
Apr 20
11:35
No Changes
1s
3s
443ms
2s
failed
#25