

Jenkins

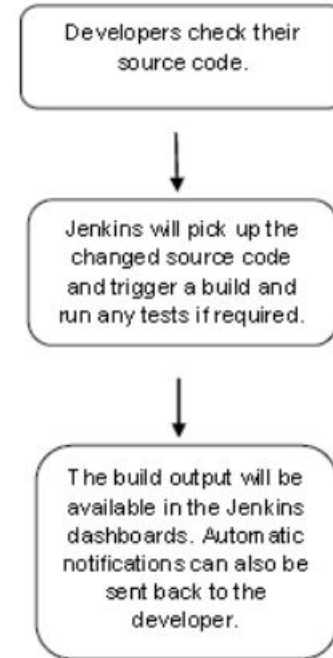


What is Jenkins

- Jenkins is an open source automation tool written in Java programming language that allows continuous integration.
- Jenkins builds and tests our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.
- It also allows us to continuously deliver our software by integrating with a large number of testing and deployment technologies.

Steps in Jenkins

- Possible steps executed by Jenkins are for example:
- Perform a software build using a build system like Gradle or Maven Apache
- Execute a shell script
- Archive a build result
- Running software tests.



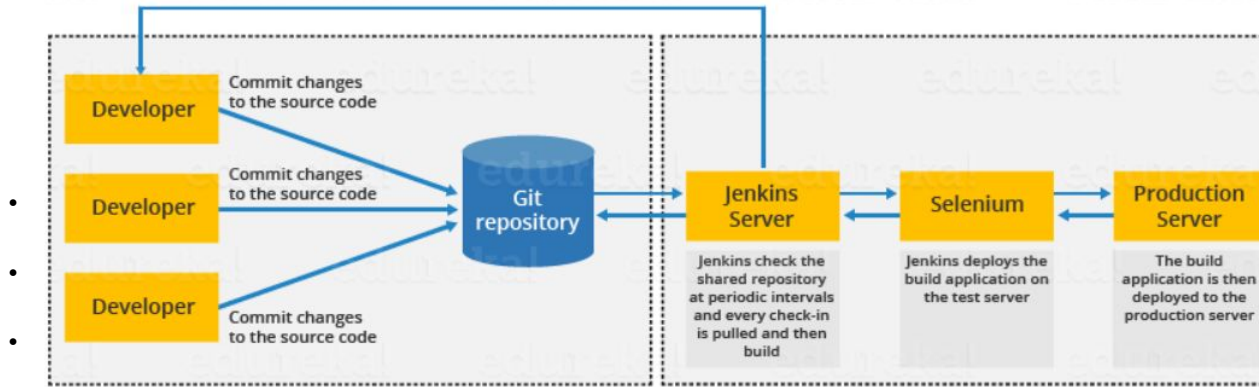
History of Jenkins

- Kohsuke Kawaguchi, who is a Java developer, working at SUN Microsystems, was tired of building the code and fixing errors repetitively. In 2004, he created an automation server called Hudson that automates build and test task.
- In 2011, Oracle who owned Sun Microsystems had a dispute with Hudson open source community, so they forked Hudson and renamed it as Jenkins.
- Both Hudson and Jenkins continued to operate independently. But in short span of time, Jenkins acquired a lot of contributors and projects while Hudson remained with only 32 projects. Then with time, Jenkins became more popular, and Hudson is not maintained anymore.

What is Continuous Integration (CI)

- Continuous Integration (CI) is a development practice in which the developers are needed to commit changes to the source code in a shared repository at regular intervals. Every commit made in the repository is then built. This allows the development teams to detect the problems early.
- Continuous integration requires the developers to have regular builds. The general practice is that whenever a code commit occurs, a build should be triggered.

How Jenkins builds the code.

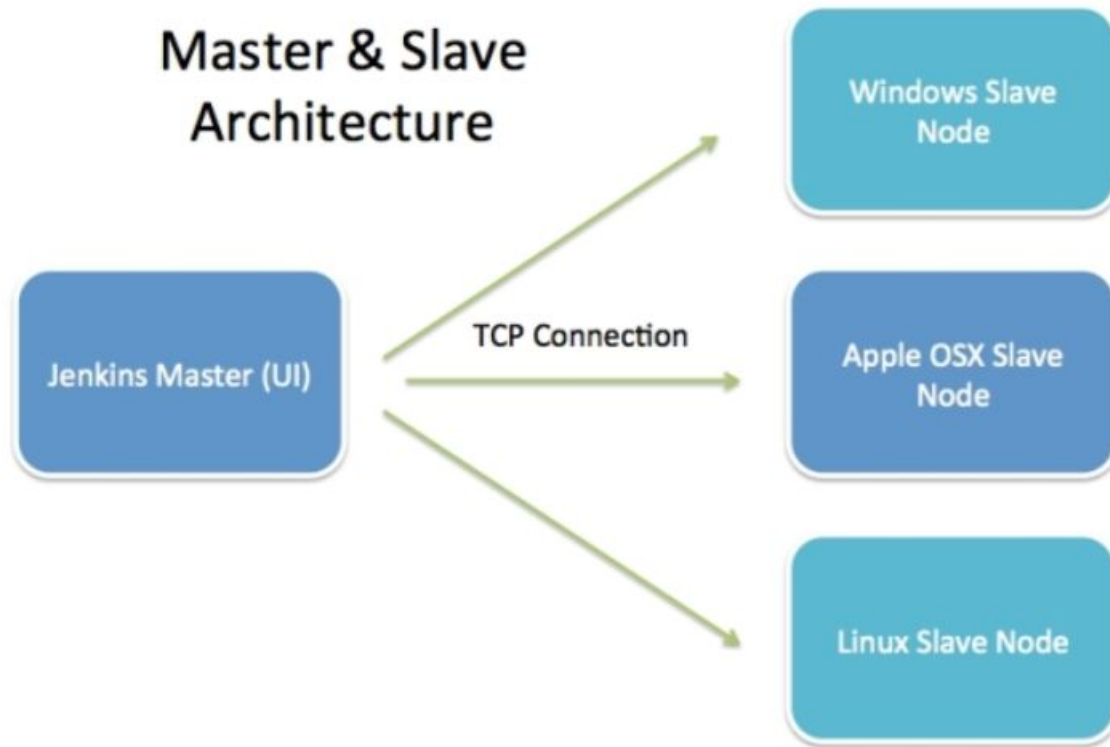


- First of all, a developer commits the code to the source code repository. Meanwhile, the Jenkins checks the repository at regular intervals for changes.
- Soon after a commit occurs, the Jenkins server finds the changes that have occurred in the source code repository. Jenkins will draw those changes and will start preparing a new build.
- If the build fails, then the concerned team will be notified.
- If built is successful, then Jenkins server deploys the built in the test server.
- After testing, Jenkins server generates a feedback and then notifies the developers about the build and test results.
- It will continue to verify the source code repository for changes made in the source code and the whole process keeps on repeating.

Benefits of Jenkins

- It is an open source tool.
- It is free of cost.
- It does not require additional installations or components. Means it is easy to install.
- Easily configurable.
- It supports 1000 or more plugins to ease your work. If a plugin does not exist, you can write the script for it and share with community.
- It is built in java and hence it is portable.
- It is platform independent. It is available for all platforms and different operating systems. Like OS X, Windows or Linux.
- Easy support, since it open source and widely used.
- Jenkins also supports cloud based architecture so that we can deploy Jenkins in cloud based platforms.

Jenkins Architecture



Jenkins Master

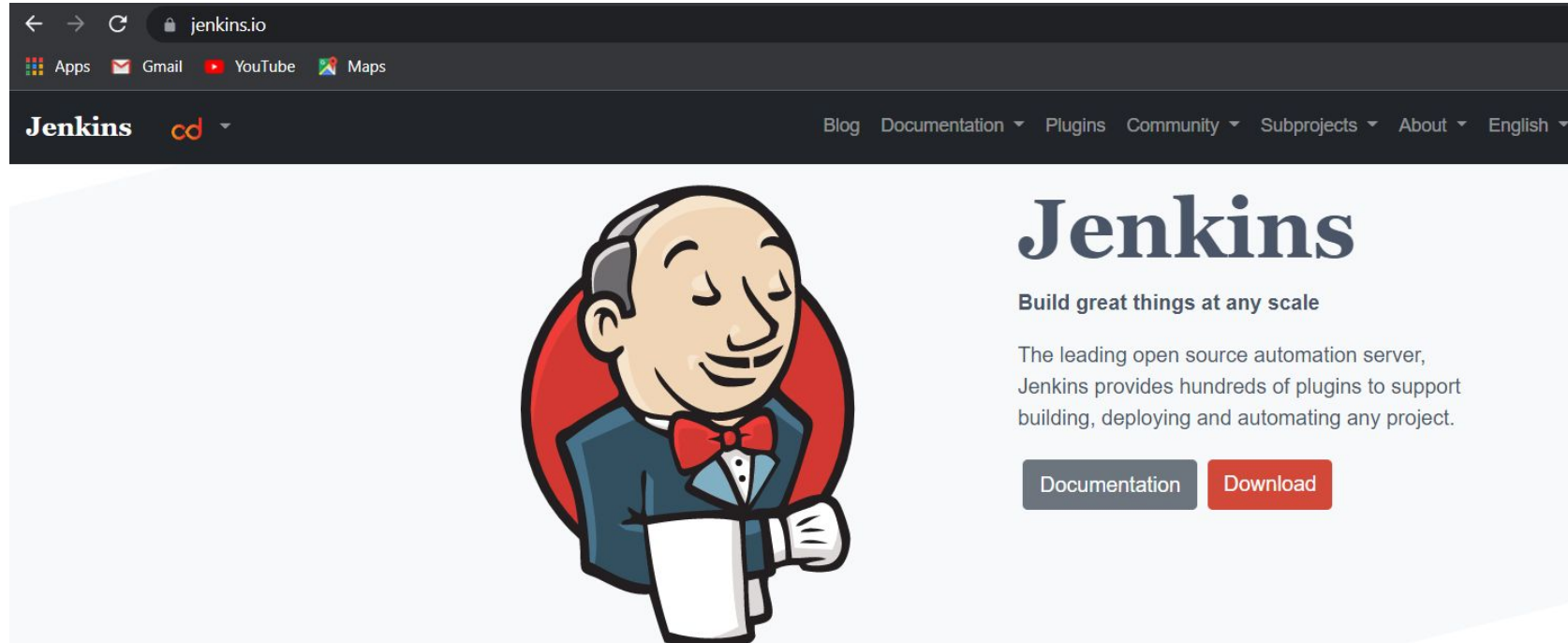
- The main server of Jenkins is the Jenkins Master. It is a web dashboard which is nothing but powered from a war file. By default it runs on 8080 port. With the help of Dashboard, we can configure the jobs/projects but the build takes place in Nodes/Slave. By default one node (slave) is configured and running in Jenkins server.
- **The server's job or master's job is to handle:**
 - Scheduling build jobs.
 - Dispatching builds to the nodes/slaves for the actual execution.
 - Monitor the nodes/slaves (possibly taking them online and offline as required).
 - Recording and presenting the build results.
 - A Master/Server instance of Jenkins can also execute build jobs directly.

Jenkins Slave

- Jenkins slave is used to execute the build jobs dispatched by the master. We can configure a project to always run on a particular slave machine, or particular type of slave machine, or simple let the Jenkins to pick the next available slave/node.
- As we know Jenkins is developed using Java is platform independent thus Jenkins Master/Servers and Slave/nodes can be configured in any servers including Linux, Windows, and Mac.

Jenkins Installation

- Please ensure Java is already installed on your system.
- Download jenkins from this site: <http://www.jenkins.io> and click on download button,.



Jenkins Installation

- Select

 Download Jenkins 2.332.1 LTS for:

Generic Java package (.war)

SHA-256: 5a14b379574419abb14123b45d0c6c32276c3198bf694239c93585ef78c33977

- And it starts downloading automatically.
- Copy this war file into a newly created folder named jenkins and run the following command : `java -jar jenkins.war`

```
C:\Java\Jenkins>dir
Volume in drive C is Windows
Volume Serial Number is D87D-3DAA

Directory of C:\Java\Jenkins

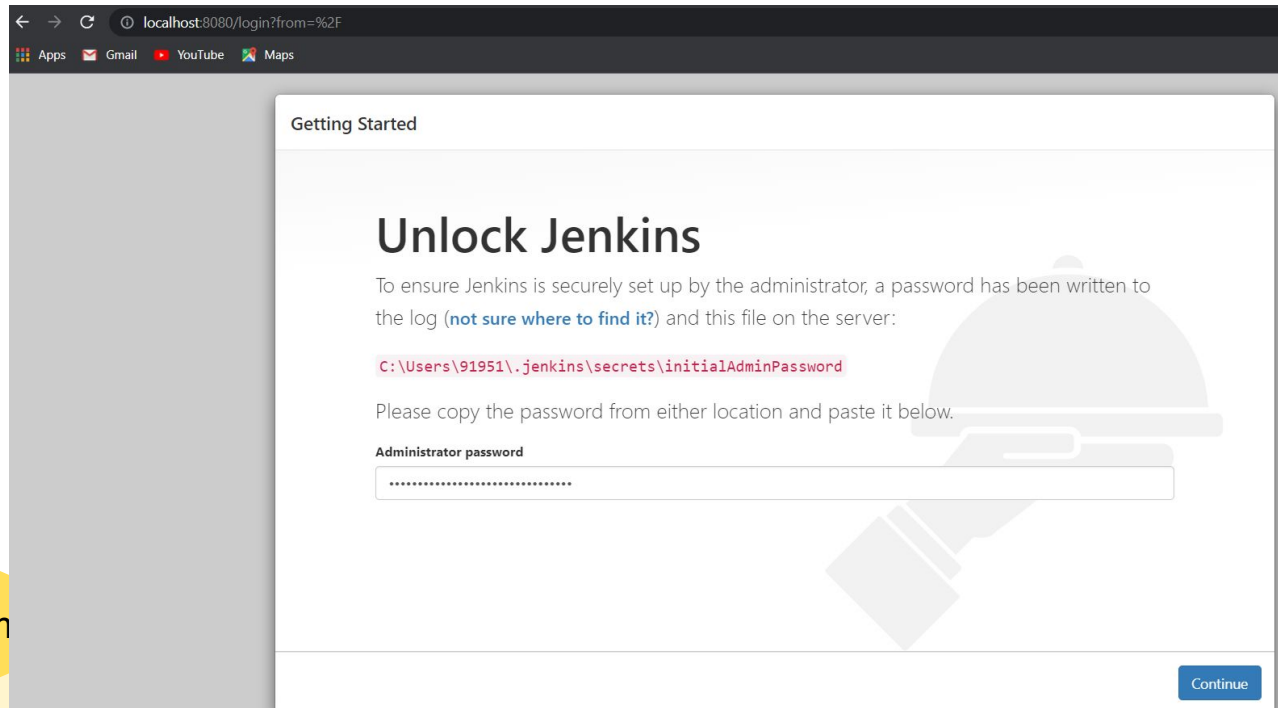
14-03-2022  23:29    <DIR>          .
14-03-2022  23:29    <DIR>          ..
14-03-2022  23:28         94,952,903 jenkins.war
               1 File(s)         94,952,903 bytes
               2 Dir(s)  404,899,405,824 bytes free

C:\Java\Jenkins>java -jar jenkins.war
Running from: C:\Java\Jenkins\jenkins.war
webroot: $user.home/.jenkins
2022-03-14 18:00:20.435+0000 [id=1]    INFO    org.eclipse.jetty.util.log.Log#initialized: Logging initialized @472ms t
o org.eclipse.jetty.util.log.JavaUtilLog
2022-03-14 18:00:20.497+0000 [id=1]    INFO    winstone.Logger#logInternal: Beginning extraction from war file
2022-03-14 18:00:22.144+0000 [id=1]    WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2022-03-14 18:00:22.191+0000 [id=1]    INFO    org.eclipse.jetty.server.Server#doStart: jetty-9.4.43.v20210629; built:
2021-06-30T11:07:22.254Z; git: 526006ecfa3af7f1a27ef3a288e2bef7ea9dd7e8; jvm 1.8.0_321-b07
2022-03-14 18:00:22.583+0000 [id=1]    INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /,
did not find org.eclipse.jetty.jsp.JettyJspServlet
2022-03-14 18:00:22.614+0000 [id=1]    INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager worke
rName=node0
2022-03-14 18:00:22.614+0000 [id=1]    INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, usin
```

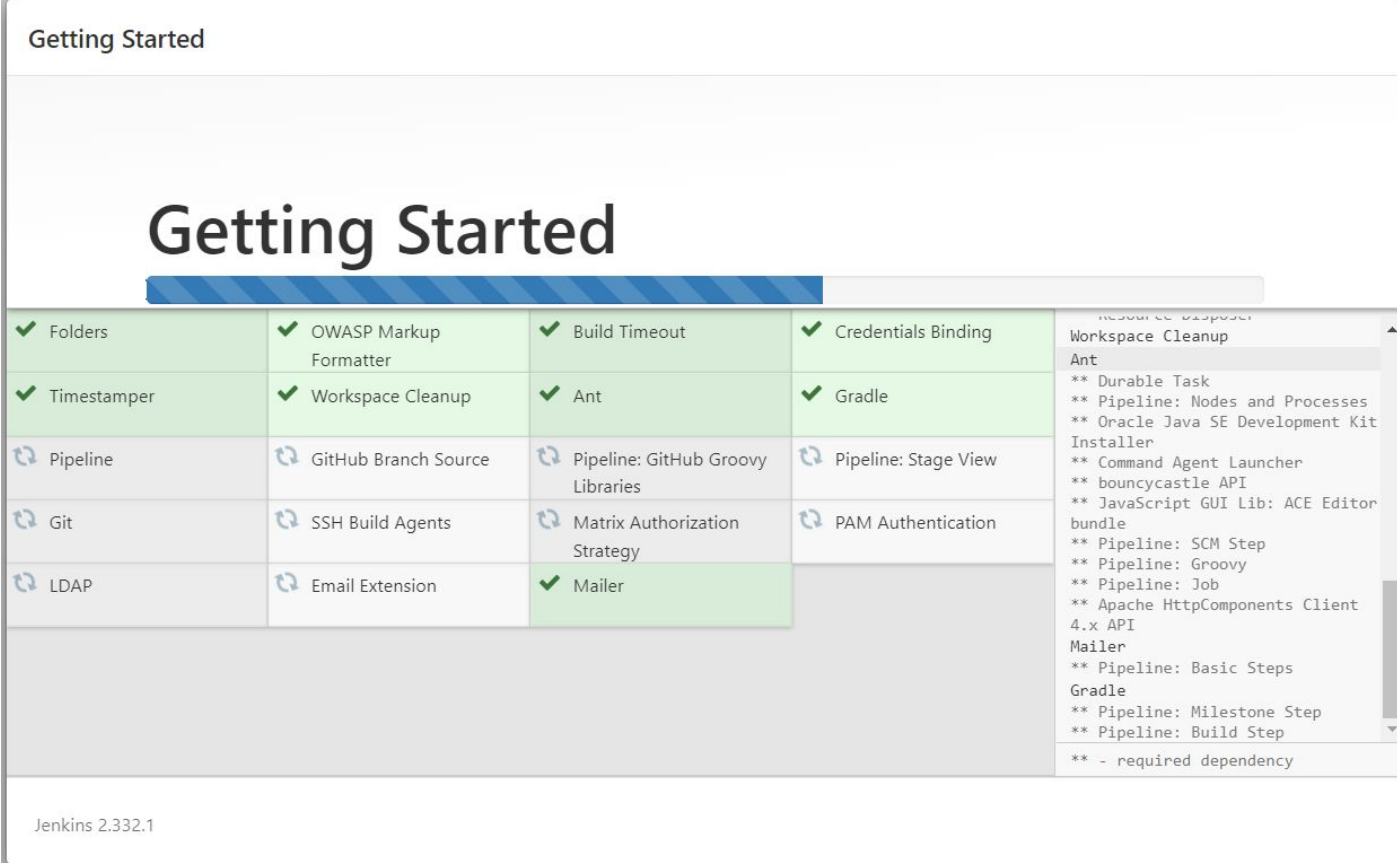
<date/time>

Jenkins Installation

- When the Jenkins is opened in the browser, it shows the following. Enter the password generated by the installation and select the first option of installation that is “most suited for industry”



- Jenkins will install these modules...

- 

Getting Started

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

Workspace Cleanup
Ant
** Durable Task
** Pipeline: Nodes and Processes
** Oracle Java SE Development Kit
Installer
** Command Agent Launcher
** bouncycastle API
** JavaScript GUI Lib: ACE Editor
bundle
** Pipeline: SCM Step
** Pipeline: Groovy
** Pipeline: Job
** Apache HttpComponents Client
4.x API
Mailer
** Pipeline: Basic Steps
Gradle
** Pipeline: Milestone Step
** Pipeline: Build Step
** - required dependency

Jenkins 2.332.1

- Registering a new user in Jenkins.
- Click on Save and Finish on configuration screen.

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.332.1

[Skip and continue as admin](#) [Save and Continue](#)

- Jenkins installation is done and it is ready to use.

-

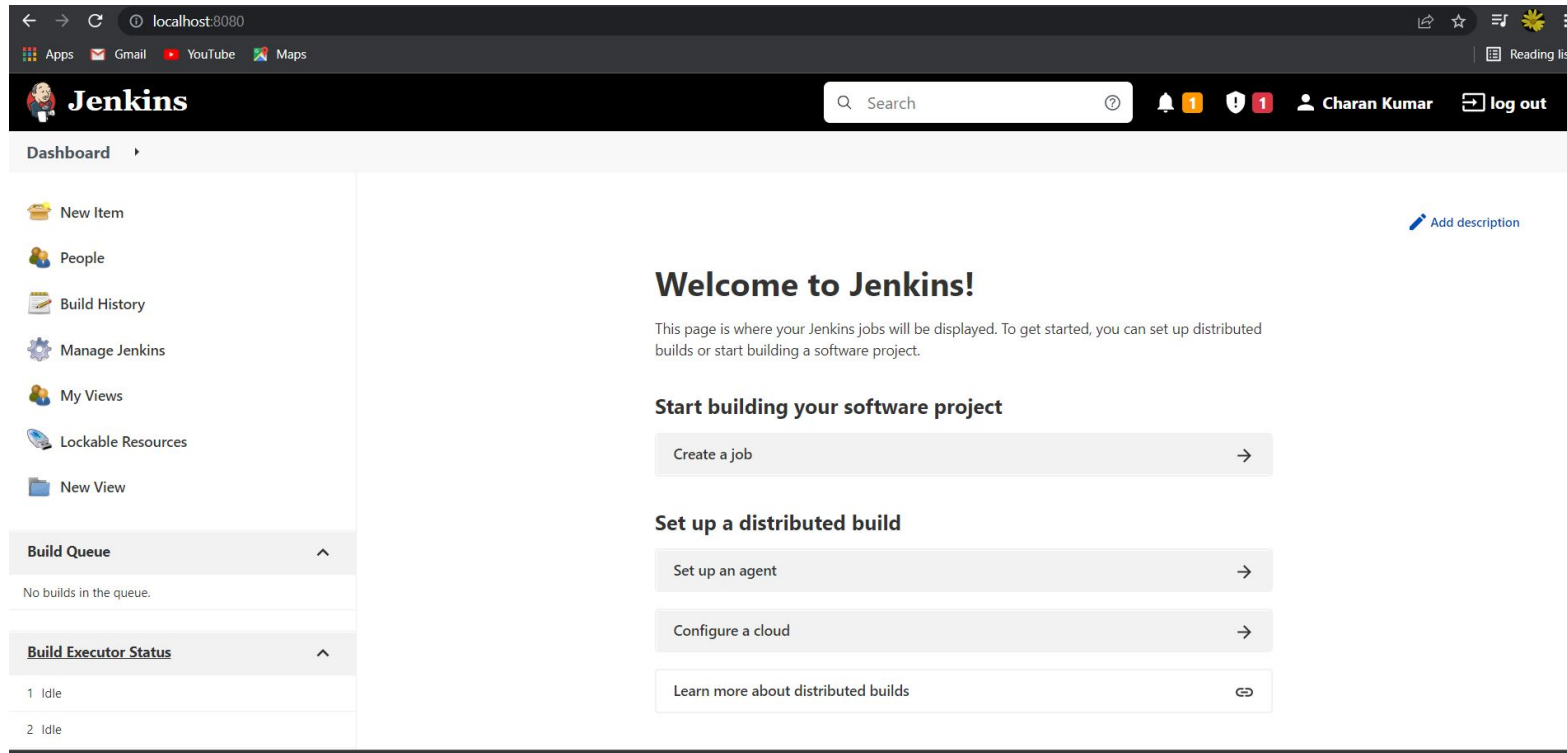
Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

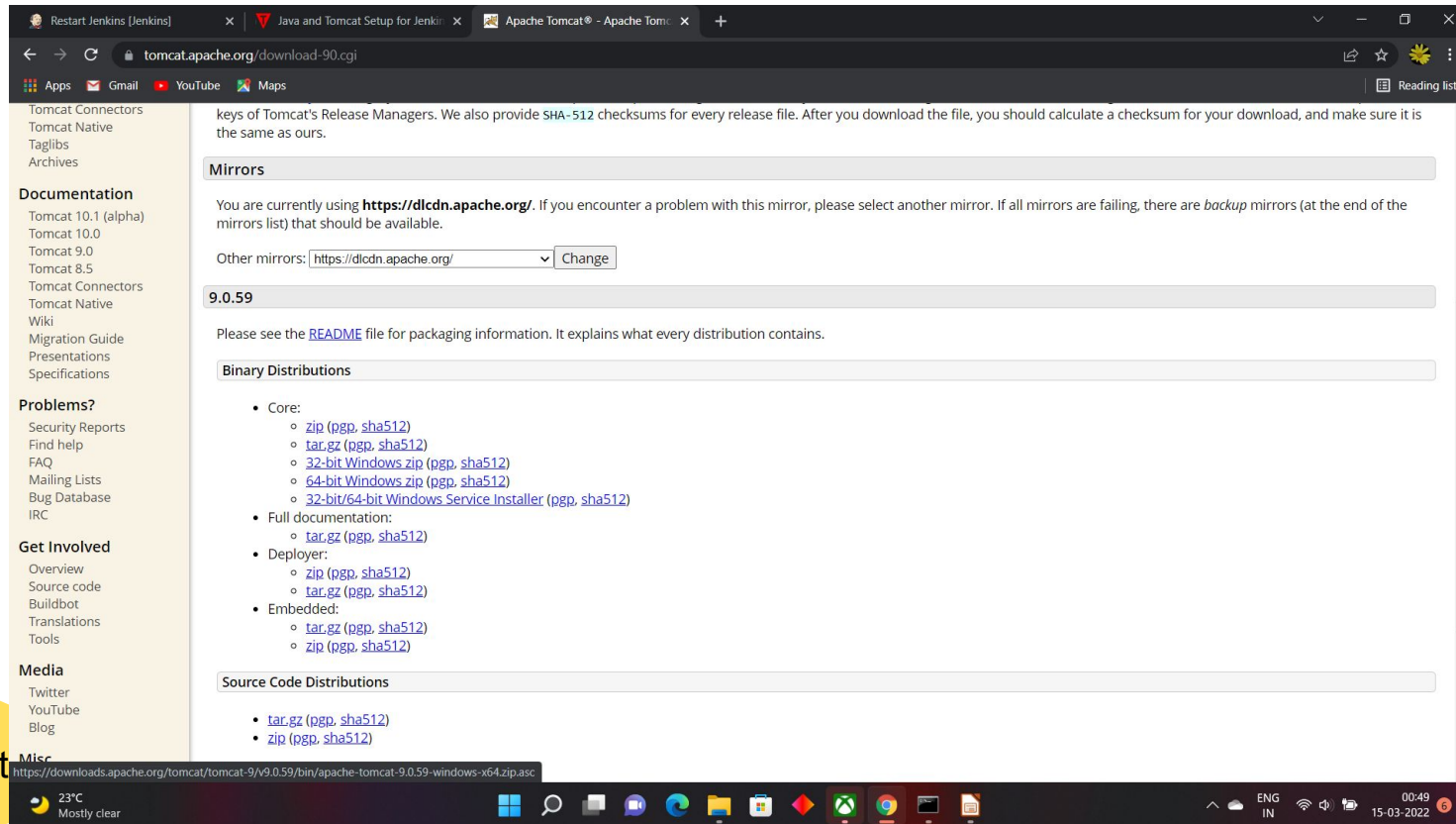
Start using Jenkins

- Jenkins dashboard looks like this.



Install Tomcat for Jenkins

- Go to tomcat site <https://tomcat.apache.org/>
- and download tomcat binaries 64 bit zip.



Install Tomcat for Jenkins

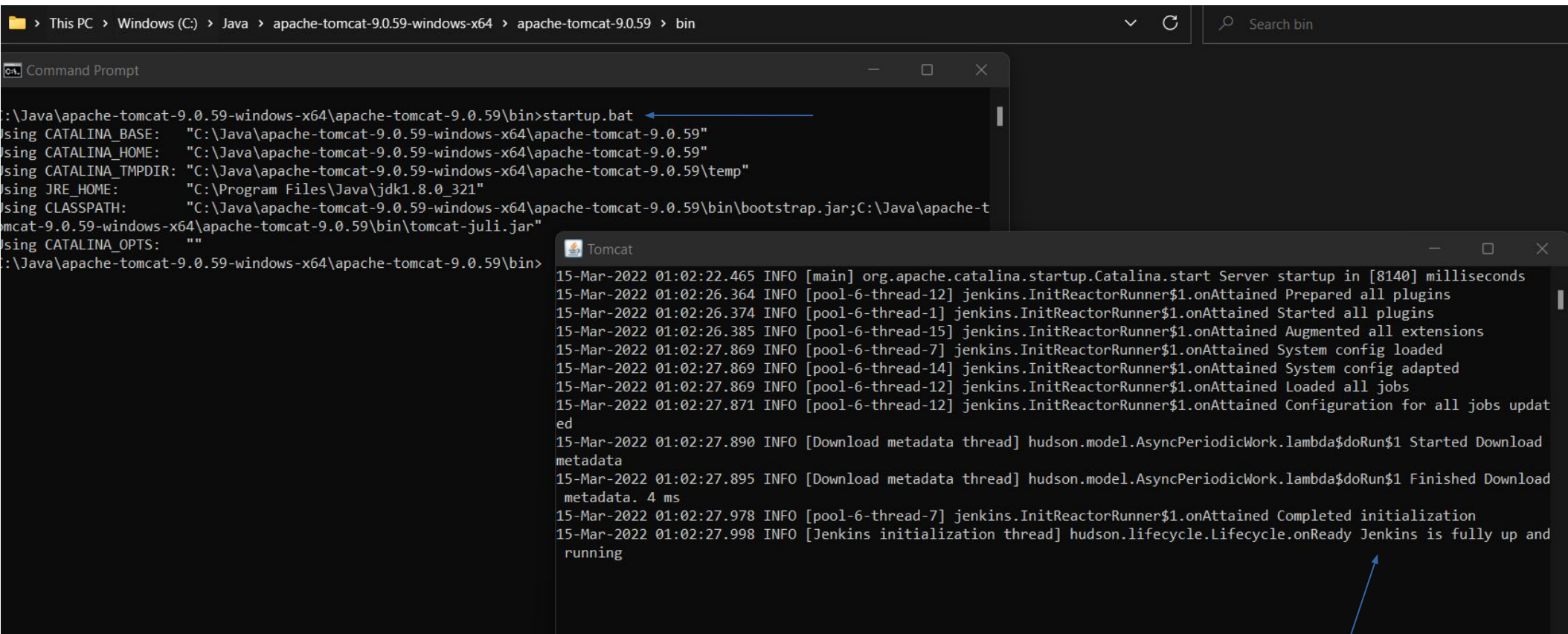
- Unzip apache tomcat into a directory.
- Go to WebApps directory of Apache Tomcat and copy jenkins.war over there.

The screenshot shows a Windows File Explorer window with the following path: This PC > Windows (C:) > Java > apache-tomcat-9.0.59-windows-x64 > apache-tomcat-9.0.59 > webapps. The file list is as follows:

Name	Date modified	Type	Size
docs	15-03-2022 00:51	File folder	
examples	15-03-2022 00:51	File folder	
host-manager	15-03-2022 00:51	File folder	
manager	15-03-2022 00:51	File folder	
ROOT	15-03-2022 00:51	File folder	
jenkins.war	14-03-2022 23:28	WAR File	92,728 KB

- Ensure the environmental variable `JAVA_HOME` is set. If not, set it.
- From the bin directory, give the command **startup.bat**. With that apache tomcat is started and along with that jenkins also is started.

Install Tomcat for Jenkins



The screenshot shows a Windows file explorer window at the top with the path: This PC > Windows (C:) > Java > apache-tomcat-9.0.59-windows-x64 > apache-tomcat-9.0.59 > bin. Below it are two command prompt windows. The first window shows the execution of startup.bat and the setting of environment variables. The second window shows the Tomcat startup logs, which indicate that Jenkins is successfully initialized and running.

```
C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59\bin>startup.bat
Using CATALINA_BASE:   "C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59"
Using CATALINA_HOME:   "C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59"
Using CATALINA_TMPDIR: "C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0_321"
Using CLASSPATH:       "C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59\bin\bootstrap.jar;C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59\bin>

15-Mar-2022 01:02:22.465 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [8140] milliseconds
15-Mar-2022 01:02:26.364 INFO [pool-6-thread-12] jenkins.InitReactorRunner$1.onAttained Prepared all plugins
15-Mar-2022 01:02:26.374 INFO [pool-6-thread-1] jenkins.InitReactorRunner$1.onAttained Started all plugins
15-Mar-2022 01:02:26.385 INFO [pool-6-thread-15] jenkins.InitReactorRunner$1.onAttained Augmented all extensions
15-Mar-2022 01:02:27.869 INFO [pool-6-thread-7] jenkins.InitReactorRunner$1.onAttained System config loaded
15-Mar-2022 01:02:27.869 INFO [pool-6-thread-14] jenkins.InitReactorRunner$1.onAttained System config adapted
15-Mar-2022 01:02:27.869 INFO [pool-6-thread-12] jenkins.InitReactorRunner$1.onAttained Loaded all jobs
15-Mar-2022 01:02:27.871 INFO [pool-6-thread-12] jenkins.InitReactorRunner$1.onAttained Configuration for all jobs updated
15-Mar-2022 01:02:27.890 INFO [Download metadata thread] hudson.model.AsyncPeriodicWork.lambda$doRun$1 Started Download metadata
15-Mar-2022 01:02:27.895 INFO [Download metadata thread] hudson.model.AsyncPeriodicWork.lambda$doRun$1 Finished Download metadata. 4 ms
15-Mar-2022 01:02:27.978 INFO [pool-6-thread-7] jenkins.InitReactorRunner$1.onAttained Completed initialization
15-Mar-2022 01:02:27.998 INFO [Jenkins initialization thread] hudson.lifecycle.Lifecycle.onReady Jenkins is fully up and running
```

Install Tomcat for Jenkins

- Jenkins dashboard can be started with <http://localhost:8080/jenkins>

The screenshot shows the Jenkins dashboard in a web browser. The address bar displays `localhost:8080/jenkins/`. The page features a dark header with the Jenkins logo, a search bar, and a user profile for "Charan Kumar" with a "log out" button. The main content area is titled "Welcome to Jenkins!" and includes a description: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this, there are two sections: "Start building your software project" with a "Create a job" button, and "Set up a distributed build" with buttons for "Set up an agent", "Configure a cloud", and a link to "Learn more about distributed builds". On the left sidebar, there are links for "New Item", "People", "Build History", "Manage Jenkins", "My Views", "Lockable Resources", and "New View". At the bottom of the sidebar, there are sections for "Build Queue" (showing "No builds in the queue.") and "Build Executor Status" (showing two idle executors).

Git hub setup in Jenkins.

- Click on Manage Jenkins in the main dashboard and click on Manage Plugins

The screenshot displays the Jenkins Dashboard interface. On the left sidebar, the 'Manage Jenkins' option is highlighted with a yellow arrow. The main content area is titled 'Manage Jenkins' and is divided into two sections: 'System Configuration' and 'Security'. Under 'System Configuration', there are four options: 'Configure System' (gear icon), 'Global Tool Configuration' (wrench icon), 'Manage Nodes and Clouds' (laptop icon), and 'Install as Windows Service' (CD icon). Under 'Security', there is one option: 'Manage Plugins' (puzzle piece icon), which is also highlighted with a yellow arrow. The 'Manage Plugins' option includes a description: 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.'

Dashboard ▾

- New Item
- People
- Build History
- Manage Jenkins**
- My Views
- Lockable Resources
- New View





Build Queue ^

No builds in the queue.


Build Executor Status ^

Manage Jenkins

System Configuration

-  **Configure System**
Configure global settings and paths.
-  **Global Tool Configuration**
Configure tools, their locations and automatic installers.
-  **Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
-  **Install as Windows Service**
Installs Jenkins as a Windows service to this system, so that Jenkins starts automatically when the machine boots.

Security

-  **Manage Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Git hub setup in Jenkins.

- Click on Available tab and enter github and

Plugin Manager

🔍 gitHub

Updates

Available

Installed

Advanced

•

GitHub Integration 0.4.0



emailx

Build Triggers

2 mo 3 days ago

GitHub Integration Plugin for Jenkins

Install without restart

Download now and install after restart

Update information obtained: 51 min ago

Check now

<date/time>

<footer>

23

Git hub setup in Jenkins.

- From the jenkins dashboard, select create a job.

- **Start building your software project**

Create a job



- Enter a name and select Free Style Project.

Enter an item name

» Required field

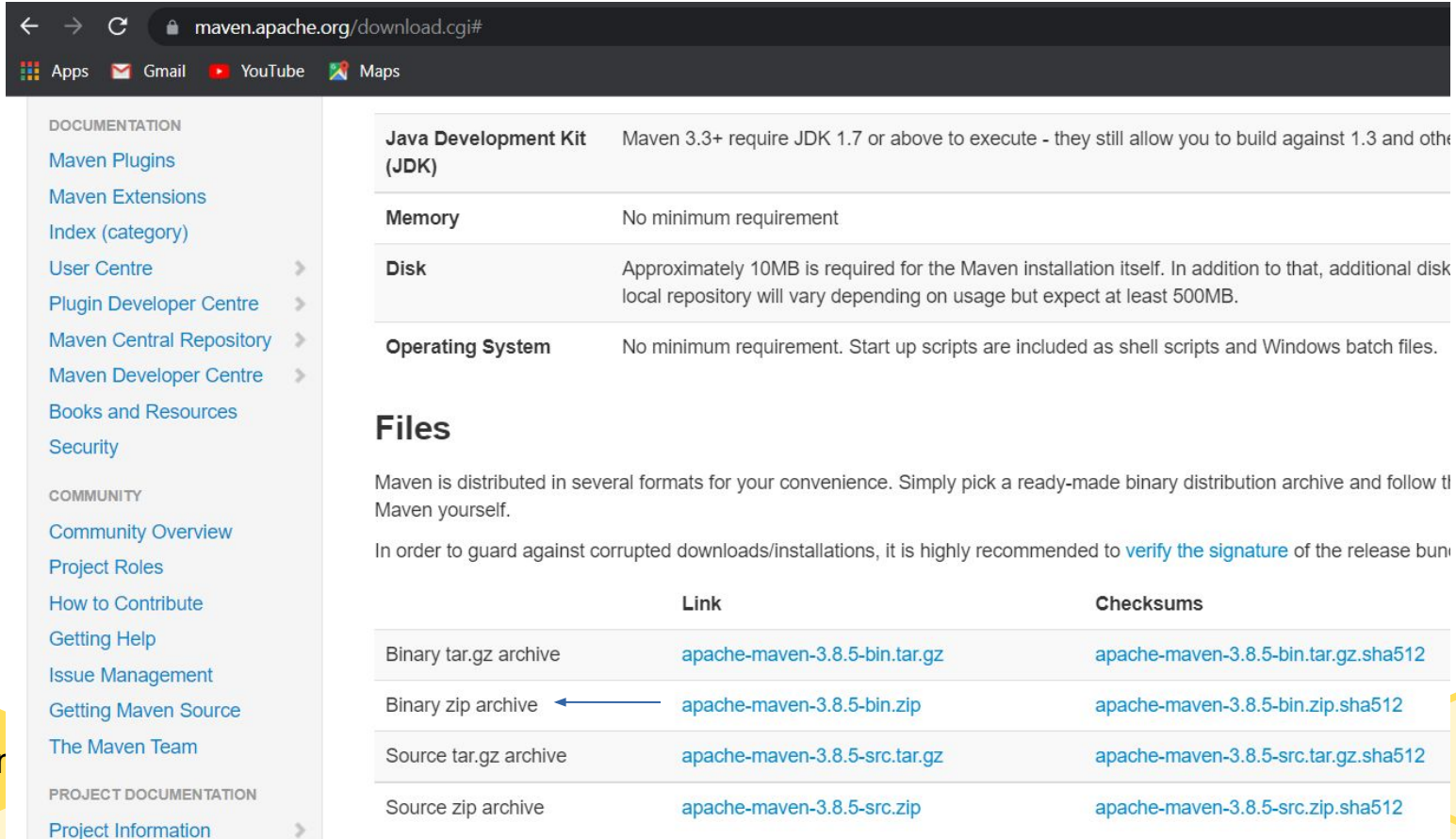


Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven setup in Jenkins.

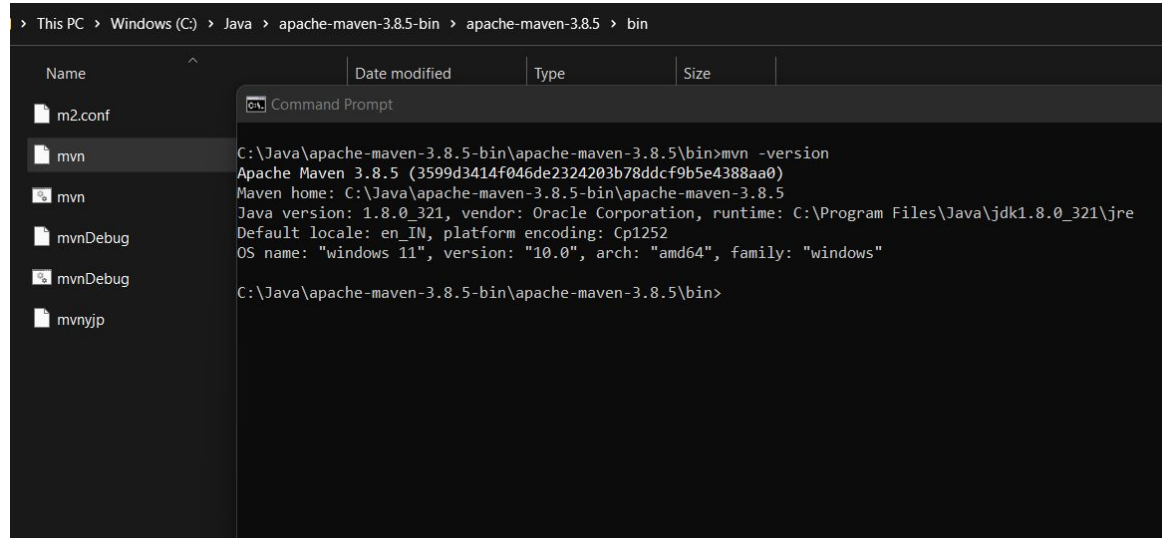
- Goto the url [maven.apache.org](https://maven.apache.org/download.cgi#) and click on download and download the binaries in zip format. After downloading, extract to maven directory.

- 
-
-

	Link	Checksums
Binary tar.gz archive	apache-maven-3.8.5-bin.tar.gz	apache-maven-3.8.5-bin.tar.gz.sha512
Binary zip archive	apache-maven-3.8.5-bin.zip	apache-maven-3.8.5-bin.zip.sha512
Source tar.gz archive	apache-maven-3.8.5-src.tar.gz	apache-maven-3.8.5-src.tar.gz.sha512
Source zip archive	apache-maven-3.8.5-src.zip	apache-maven-3.8.5-src.zip.sha512

Maven setup in Jenkins.

- Given mvn -version in the command prompt by going to bin directory of maven.



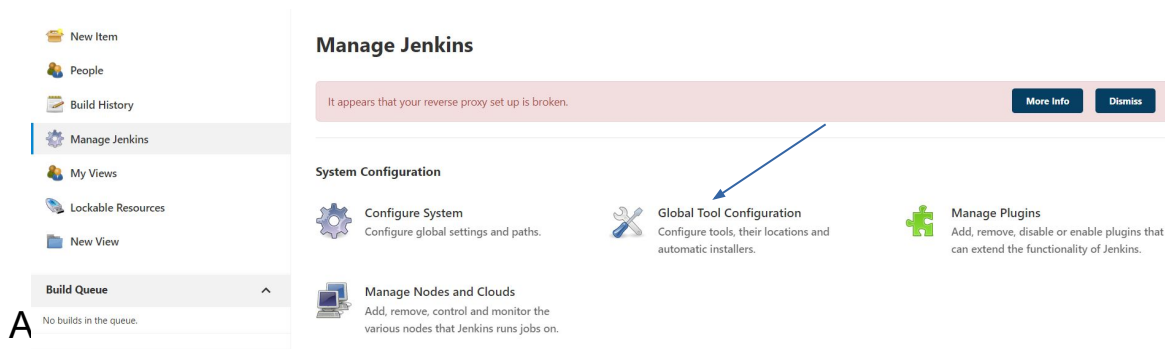
The screenshot shows a Windows File Explorer window with the path `This PC > Windows (C:) > Java > apache-maven-3.8.5-bin > apache-maven-3.8.5 > bin`. The file list includes `m2.conf`, `mvn`, `mvn`, `mvnDebug`, `mvnDebug`, and `mvnyjp`. A Command Prompt window is open, displaying the output of the `mvn -version` command. The output shows the Maven version (3.8.5), Maven home, Java version (1.8.0_321), vendor (Oracle Corporation), runtime, default locale, platform encoding, and OS name (windows 11).

```
C:\Java\apache-maven-3.8.5-bin\apache-maven-3.8.5\bin>mvn -version
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: C:\Java\apache-maven-3.8.5-bin\apache-maven-3.8.5
Java version: 1.8.0_321, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_321\jre
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Java\apache-maven-3.8.5-bin\apache-maven-3.8.5\bin>
```

Maven setup in Jenkins.

Select Global Tool Configuration in Main Dashboard.



Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

JDK

JDK installations

Add JDK

JDK

Name

JDK 1.8

JAVA_HOME

C:\Program Files\Java\jdk1.8.0_321

☐ Install automatically ?

<date/time>

27

Maven setup in Jenkins.

Add git information:

Git

Git installations

Git

Name

Default

Path to Git executable ?

C:\Program Files\Git\bin\git.exe

☐ Install automatically ?

Delete Git

Add Git ▾

Maven

Name

Maven

MAVEN_HOME

C:\Java\apache-maven-3.8.5-bin\apache-maven-3.8.5

☐ Install automatically ?

Delete Maven

Add Maven

List of Maven installations on this system

<date/time>

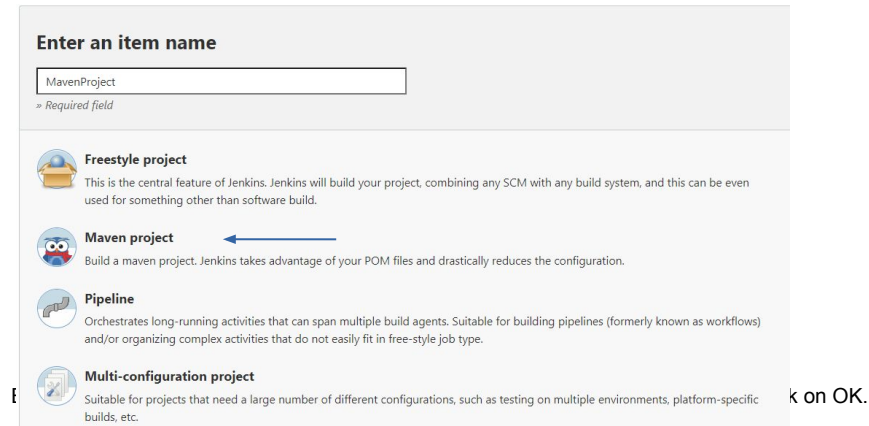
<footer>

28

Maven setup in Jenkins.

If Maven is not seen as part of build then install maven plug-in by going to Manage Plugins --> Available --> Enter Maven in find --> Select the Maven plugin and install it.

Select New Item from Dashboard and enter Name as MavenProject and Select Maven Project is the option.



Enter an item name

MavenProject
= Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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

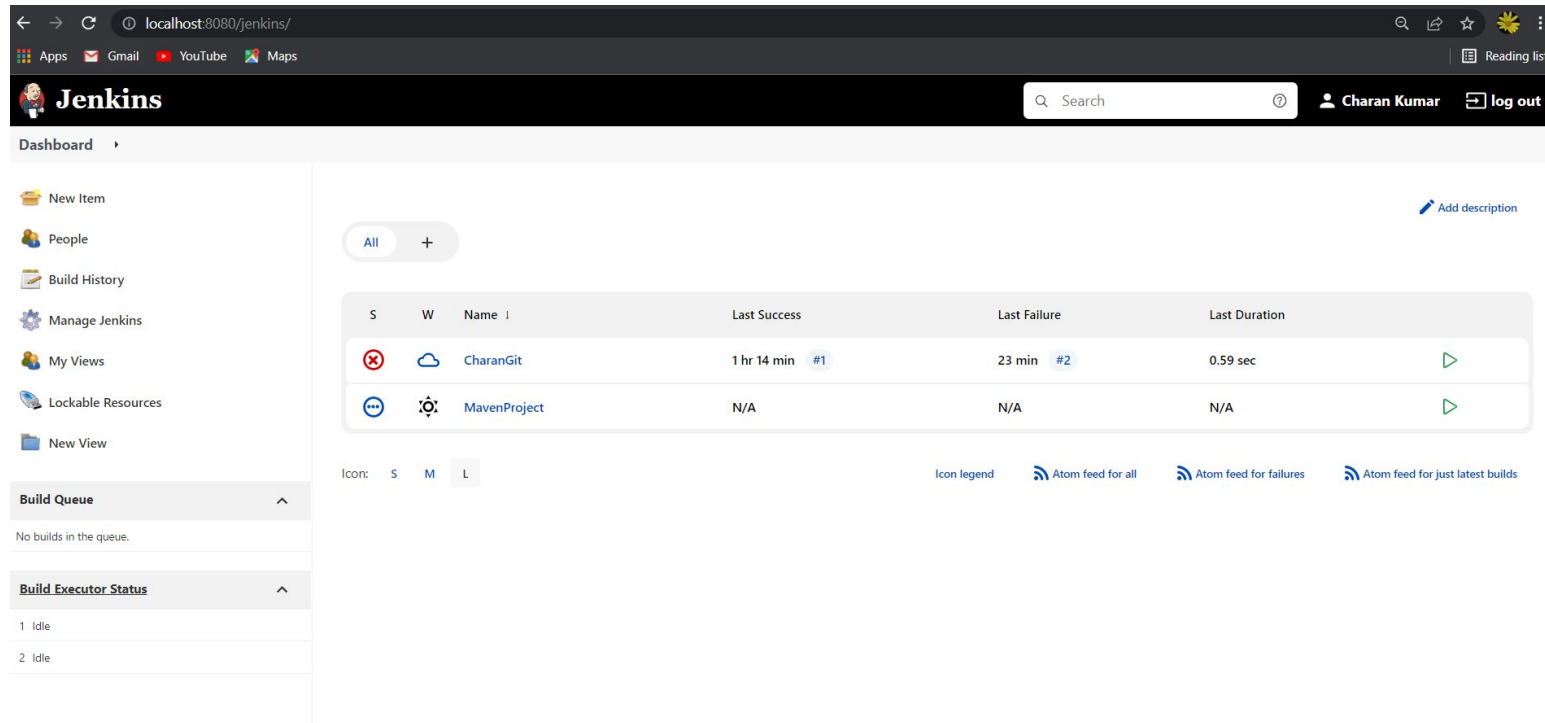
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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

Click on OK.

Jenkins Dashboard.

After adding the projects for Git and Maven, the dashboard looks like this:



The screenshot shows the Jenkins Dashboard interface. The top navigation bar includes the Jenkins logo, a search bar, and the user name 'Charan Kumar' with a 'log out' button. The left sidebar contains a 'Dashboard' menu and a list of links: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. Below these are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two 'Idle' executors). The main content area displays a table of build jobs. The table has columns for 'S' (Status), 'W' (Icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. Two jobs are listed: 'CharanGit' and 'MavenProject'. 'CharanGit' has a status of 'Failed' (red X icon), a last success of '1 hr 14 min #1', and a last failure of '23 min #2'. 'MavenProject' has a status of 'Success' (blue checkmark icon), a last success of 'N/A', and a last failure of 'N/A'. Both jobs have a 'Last Duration' of '0.59 sec'. Below the table, there is an 'Icon legend' and four Atom feed links: 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

S	W	Name	Last Success	Last Failure	Last Duration
❌	☁	CharanGit	1 hr 14 min #1	23 min #2	0.59 sec
✅	⚙	MavenProject	N/A	N/A	N/A

Jenkins Configuration.

- From the dashboard, click on **Manage Jenkins** and select **Configure System**.

-  Build History
-  Manage Jenkins
-  My Views
-  Lockable Resources

System Configuration





Configure System


Configure global settings and paths.


- This has all the configuration information about Jenkins.


Dashboard > configuration


 New Item


 People

 Build History

 Manage Jenkins

 My Views

 Lockable Resources

 New View

Build Queue ^
No builds in the queue.

Build Executor Status ^

1 Idle

2 Idle

Home directory ?

C:\Users\91951\jenkins

System Message ?

[Plain text] [Preview](#)

Maven Project Configuration

Global MAVEN_OPTS ?

Local Maven Repository ?

Default ("~/m2/repository", or the value of 'localRepository' in Maven's settings file, if defined)

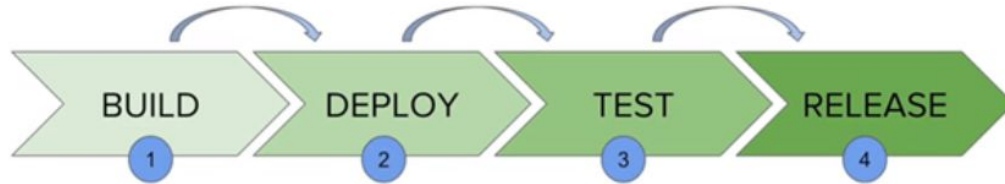
of executors

Jenkins Management.

- When Manage Jenkins is clicked on the left hand side of the dashboard, following options are displayed.
- (1) Configuration System – It is used to configure Jenkins Location, Setting up variables, Setting up Email Servers and many more.
- (2) Global Tool Configuration – It sets up the installed location for JDK, GIT and Maven.
- (3) Manage Plugin – New Plugins can be added/deleted and updated.
- (4) Manage Node and clouds – For the management of new Nodes.
- (5) Security – Configure Global Security -
- (6) Security – Manage Credentials -
- (7) Security – Configure credential Providers.
- (8) Security – Manage Users
- (9) Status – System Information – Provides the environmental information like env variables etc., Helps for troubleshooting.
- (10) Status – System Log – Logs the information from java.util.Logging.
- (11) Status – Load Statistics – Gives the information about how much the system is loaded.
- (12) Status – About Jenkins.
- (13) Manage Old Data.

Jenkins - Pipeline

- In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.
- It is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins.



- These are phases of the build. i.e., the build the project, deploy on server, Test it with selenium and Release it to the customer. In each phase some action to be taken. A build script or Jenkins script file is created so that each phase action is taken and goes to the next action / next phase of action.
- A pipeline can be created in the following manner.
- Select the create a new job, give a name and select a pipeline.

Jenkins - Pipeline

Enter an item name

CharanPipeLine1

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



Maven project

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



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pip and/or organizing complex activities that do not easily fit in free-style job type.

Jenkins pipeline script for hello world is given :

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition

Pipeline script

Script ?

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

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

<date/time>

<foo

Jenkins - Pipeline

- Format of the Jenkins pipeline file is

- pipeline {
- agent any
- stages {
- stage ('Build') {
- ...
- }
- stage ('Test') {
- ...
- }
- stage ('QA') {
- ...
- }
- }

<date/time>

stage ('Deploy') {

<footer>

35

Jenkins - Pipeline

- Sample build script for github and maven is

Script ?

try sample Pipeline... ▼

```
1 pipeline {
2   agent any
3
4   tools {
5     // Install the Maven version configured as "M3" and add it to the path.
6     maven "M3"
7   }
8
9   stages {
10    stage('Build') {
11      steps {
12        // Get some code from a GitHub repository
13        git 'https://github.com/jglick/simple-maven-project-with-tests.git'
14
15        // Run Maven on a Unix agent.
16        sh "mvn -Dmaven.test.failure.ignore=true clean package"
17
18        // To run Maven on a Windows agent, use
19        // bat "mvn -Dmaven.test.failure.ignore=true clean package"
20      }
21
22      post {
23        // If Maven was able to run the tests, even if some of the test
24        // failed, record the test results and archive the jar file.
25        success {
26          junit '**/target/surefire-reports/TEST-*.xml'
27          archiveArtifacts 'target/*.jar'
28        }
29      }
30    }
31  }
32 }
33
```

<define

testen

se

Jenkins – Build setup

- Create a GitRepo directory in Git.
- Create HelloWorld.java and compile it.

```
C:\Program Files\Git>cd \java
C:\Java>cd GitRepo
C:\Java\GitRepo>dir
Volume in drive C is Windows
Volume Serial Number is D87D-3DAA

Directory of C:\Java\GitRepo

15-03-2022  17:39    <DIR>          .
15-03-2022  17:38    <DIR>          ..
15-03-2022  17:39                  121 HelloWorld.java
                1 File(s)                121 bytes
                2 Dir(s)  404,194,033,664 bytes free

C:\Java\GitRepo>javac HelloWorld.java

C:\Java\GitRepo>java HelloWorld
Hello World...
```


- Create a New Repository as HelloWorld in GitHub.

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner ^{*}

Repository name ^{*}


 kcharankumar ▾

/


HelloWorld 

Great repository names are short and memorable. Need inspiration? How about [crispy-robot](#)?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Jenkins – Build setup

- Open the git_cmd and execute the following commands. This will create a repository, adds the files in the repository and pushes them to github.
- Git init
- Git status
- Git add . --> Adds both the files in the repository
- Git status
- git config --global user.email "kcharankumar@gmail.com"
- git config --global user.name "K.Charan Kumar"
- Git commit -m "HelloWorld Program is added"
- git remote add origin <https://github.com/kcharankumar/HelloWorld.git>
- git push -u origin master

Jenkins – Build setup

```
CA\Program Files\Git\git-cmd.exe

C:\Java\GitRepo>git init
Initialized empty Git repository in C:/Java/GitRepo/.git/

C:\Java\GitRepo>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        HelloWorld.class
        HelloWorld.java

nothing added to commit but untracked files present (use "git add" to track)

C:\Java\GitRepo>git add .

C:\Java\GitRepo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   HelloWorld.class
        new file:   HelloWorld.java

C:\Java\GitRepo>git config --global user.email "kcharankumar@gmail.com"

C:\Java\GitRepo>git config --global user.name "K.Charan Kumar"

C:\Java\GitRepo>git commit -m
error: switch `m' requires a value

C:\Java\GitRepo>git commit -m "Added HelloWorld Program"
[master (root-commit) 95076a9] Added HelloWorld Program
 2 files changed, 10 insertions(+)
 create mode 100644 HelloWorld.class
 create mode 100644 HelloWorld.java

C:\Java\GitRepo>git remote add origin https://github.com/kcharankumar/HelloWorld.git

C:\Java\GitRepo>git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
```

Jenkins – Build setup

-

```
C:\Java\GitRepo>git remote add origin https://github.com/kcharankumar/HelloWorld.git

C:\Java\GitRepo>git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 660 bytes | 660.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kcharankumar/HelloWorld.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

C:\Java\GitRepo>
```


Jenkins – Build setup

- From the Jenkins dashboard, select New Item
- Enter a **name** and select **Free Style Project** and click on **OK**.

-
-
-
-

- Enter details in General Section :

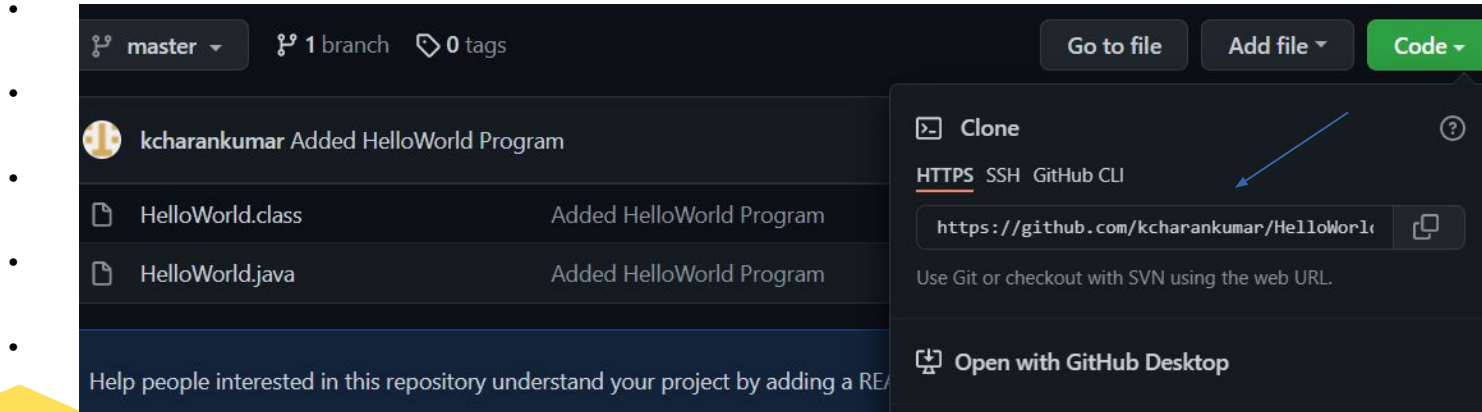
-

Jenkins – Build setup

- Enter git details in Source Code Repositories.

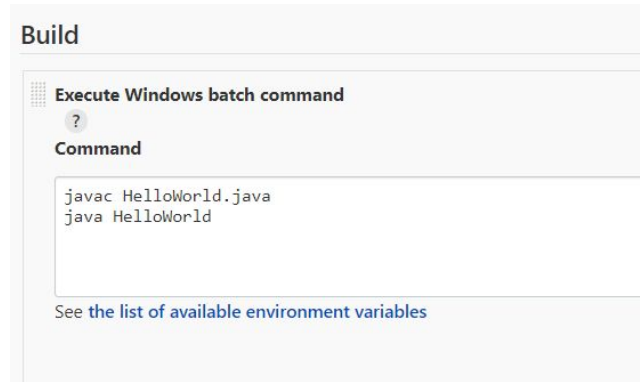
- 

- Git location can be picked up from GitHub from here.

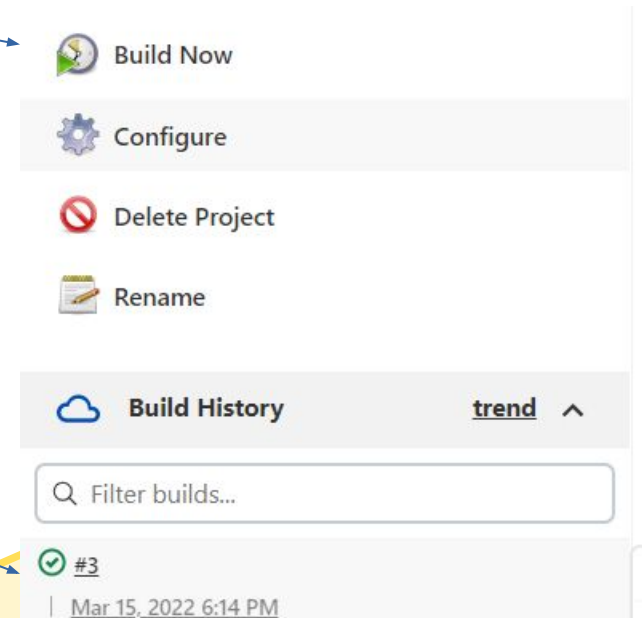


Jenkins – Build setup

- Under Build --> Select **Execute Windows Batch command** and enter the following and click on **Save and Apply**.

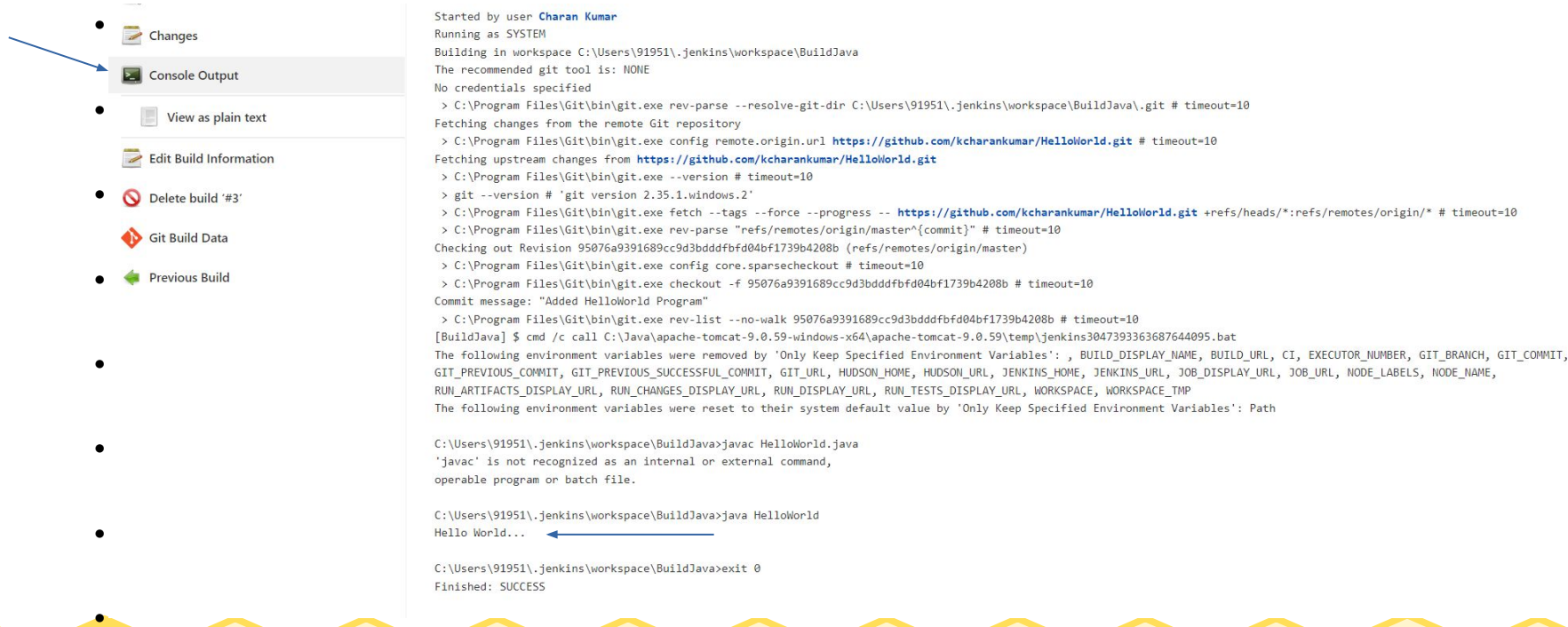


- From Dashboard of BuildJava Project, click on Build Now.
- It will build and gives you the result.



Jenkins – Build setup

- After clicking on the link #3 or #2 or #1 as per the number of times the build is made, it opens another dashboard. In that select Console output. It shows the following.



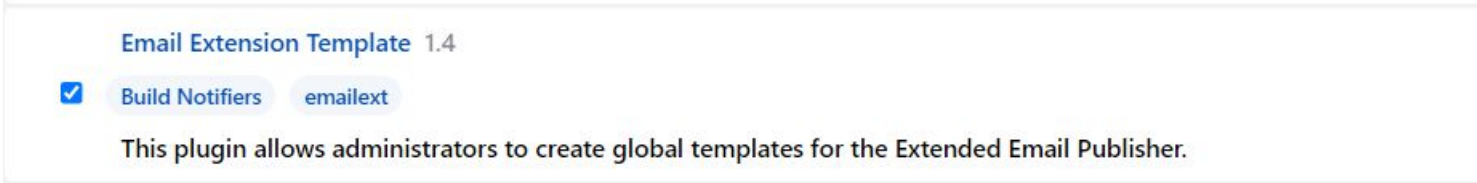
The screenshot displays the Jenkins build dashboard for a build named 'BuildJava'. The 'Console Output' tab is selected, showing the following log:


```
Started by user Charan Kumar
Running as SYSTEM
Building in workspace C:\Users\91951\.jenkins\workspace\BuildJava
The recommended git tool is: NONE
No credentials specified
> C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\Users\91951\.jenkins\workspace\BuildJava\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/kcharankumar/HelloWorld.git # timeout=10
Fetching upstream changes from https://github.com/kcharankumar/HelloWorld.git
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> git --version # 'git version 2.35.1.windows.2'
> C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/kcharankumar/HelloWorld.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 95076a9391689cc9d3bddd04bf1739b4208b (refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f 95076a9391689cc9d3bddd04bf1739b4208b # timeout=10
Commit message: "Added HelloWorld Program"
> C:\Program Files\Git\bin\git.exe rev-list --no-walk 95076a9391689cc9d3bddd04bf1739b4208b # timeout=10
[BuildJava] $ cmd /c call C:\Java\apache-tomcat-9.0.59-windows-x64\apache-tomcat-9.0.59\temp\jenkins3047393363687644095.bat
The following environment variables were removed by 'Only Keep Specified Environment Variables': , BUILD_DISPLAY_NAME, BUILD_URL, CI, EXECUTOR_NUMBER, GIT_BRANCH, GIT_COMMIT,
GIT_PREVIOUS_COMMIT, GIT_PREVIOUS_SUCCESSFUL_COMMIT, GIT_URL, HUDSON_HOME, HUDSON_URL, JENKINS_HOME, JENKINS_URL, JOB_DISPLAY_URL, JOB_URL, NODE_LABELS, NODE_NAME,
RUN_ARTIFACTS_DISPLAY_URL, RUN_CHANGES_DISPLAY_URL, RUN_DISPLAY_URL, RUN_TESTS_DISPLAY_URL, WORKSPACE, WORKSPACE_TMP
The following environment variables were reset to their system default value by 'Only Keep Specified Environment Variables': Path

C:\Users\91951\.jenkins\workspace\BuildJava>javac HelloWorld.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\91951\.jenkins\workspace\BuildJava>java HelloWorld
Hello World...
C:\Users\91951\.jenkins\workspace\BuildJava>exit 0
Finished: SUCCESS
```

Jenkins – Notification

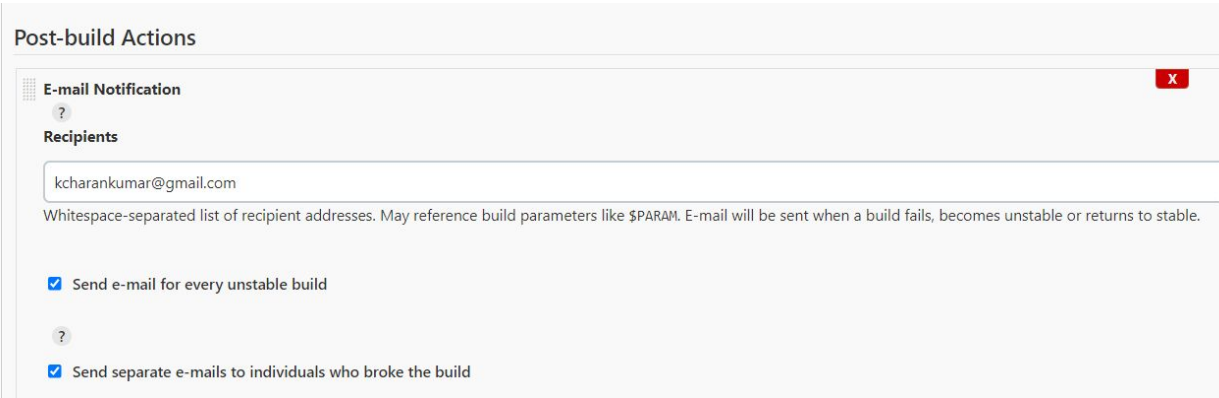
- Email notification from Jenkins
 - 1. Go to **Manage Jenkins** and click on **Manage Plug-ins**
 - 2. Click on **Available** and enter **Email** in the search box and install the selected one.
- 
- Select the **BuildJava Project** --> Click on **Configure** --> select post build operations as **Email Notification**.



- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- GitHub PR: add labels
- GitHub PR: close PR
- GitHub PR: post comment
- GitHub PR: remove labels
- GitHub PR: set PR status
- Publish JUnit test result report
- Publish Javadoc
- Record fingerprints of files to track usage
- Git Publisher
- E-mail Notification
- Editable Email Notification
- Editable Email Notification Templates
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done

Jenkins – Notification

- Enter the email id and click on when the build is broken and click on Apply and Save.

- 

The screenshot shows the 'Post-build Actions' section in Jenkins. Under 'E-mail Notification', there is a 'Recipients' text box containing 'kcharankumar@gmail.com'. Below this, there are two checked checkboxes: 'Send e-mail for every unstable build' and 'Send separate e-mails to individuals who broke the build'. A red 'X' icon is visible in the top right corner of the configuration box.
-
-
-
-
-

Jenkins – Notification

Click on **Manage Jenkins** from Dashboard --> **Configure System**.

- Go to the section Email Notification --> click on Advanced and enter the following information.
- SMTP Server : smtp.gmail.com
- UserName : kcharankumar@gmail.com
- Password : <password>
- Use SSL : Check this.
- Smtplib port : 465
-
-

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

☒ Use SMTP Authentication ?

User Name

kcharankumar@gmail.com

Password

.....

☒ Use SSL ?

☐ Use TLS

SMTP Port ?

465

Jenkins – Notification

Click on test email notification. This will send test email to the given mail id.

☒ Test configuration by sending test e-mail

Test e-mail recipient

kcharankumar@gmail.com

Email was successfully sent

Test configuration

- Make HelloWorld.java generate errors and make the build fail in local git as
- Removed the quotes in s.o.p.
-

```
class HelloWorld{  
    public static void main (String args[]){  
        System.out.println (Hello World...);  
    }  
}
```


Jenkins – Notification

Run the following git commands.

Git add HelloWorld.java

Git commit -m "Removed the quotes"

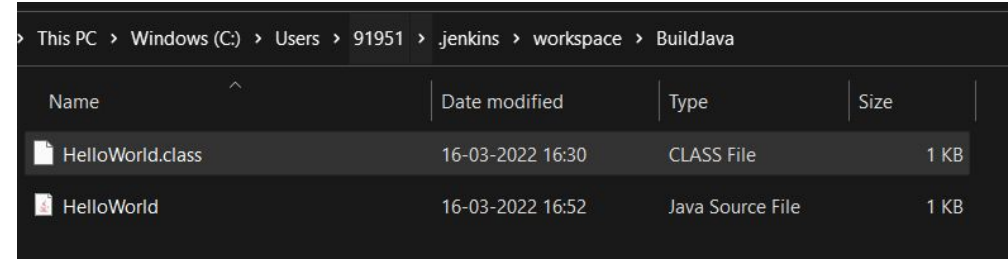
Git push -u origin master



Ensure the modified code is in github.

```
10 lines (5 sloc) | 110 Bytes  
1  class HelloWorld{  
2  
3      public static void main (String args[]){  
4  
5          System.out.println (Hello World...);  
6  
7      }  
8  
9  
10 }
```

Jenkins – Notification

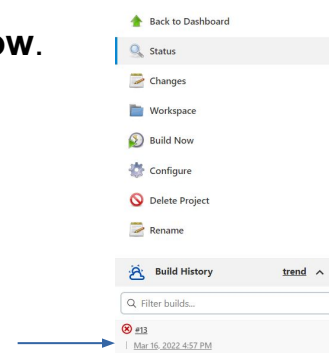
Remove the previously created HelloWorld.class:



This PC > Windows (C:) > Users > 91951 > .jenkins > workspace > BuildJava			
Name	Date modified	Type	Size
 HelloWorld.class	16-03-2022 16:30	CLASS File	1 KB
 HelloWorld	16-03-2022 16:52	Java Source File	1 KB

Open Dashboard, click on **BuildJava Project** and click on **Build Now**.

Build Failed as shown in here.



Jenkins – Notification




Can click on the build link and click on Console Output. Following screen will be seen.

```
C:\Users\91951\.jenkins\workspace\BuildJava>javac HelloWorld.java
HelloWorld.java:5: error: ')' expected
    System.out.println (Hello World...);
                        ^
HelloWorld.java:5: error: illegal start of expression
    System.out.println (Hello World...);
                        ^
HelloWorld.java:5: error: ';' expected
    System.out.println (Hello World...);
                        ^
3 errors

C:\Users\91951\.jenkins\workspace\BuildJava>java HelloWorld
Error: Could not find or load main class HelloWorld

C:\Users\91951\.jenkins\workspace\BuildJava>exit 1
Build step 'Execute Windows batch command' marked build as failure
Sending e-mails to: kcharankumar@gmail.com
Finished: FAILURE
```

Email will be sent to the given email id:

Build failed in Jenkins: BuildJava #13   



address not configured yet <kcharankumar@gmail.com>

to me 

See <<http://localhost:8080/job/BuildJava/13/display/redirect>>

Changes: