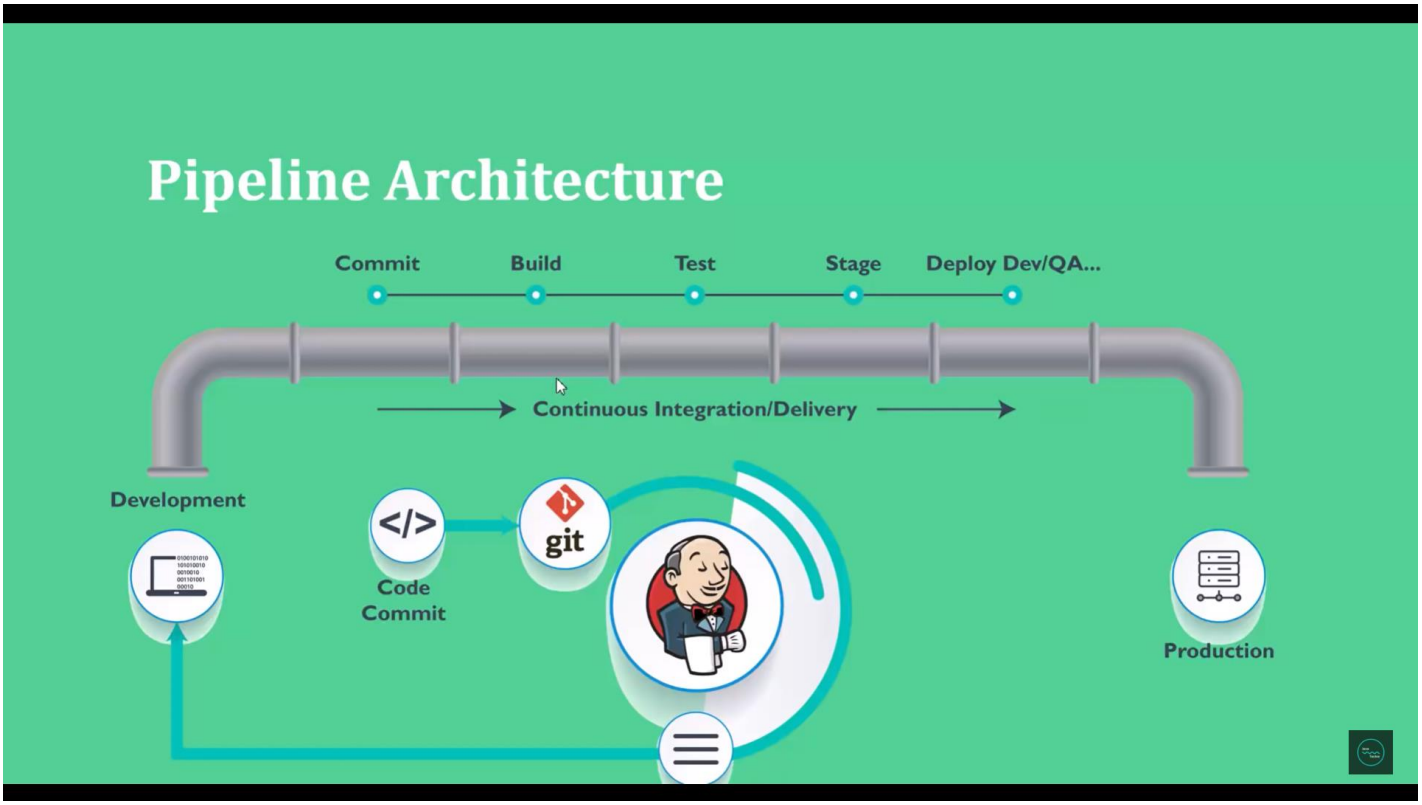
JENKINS

Jenkins is a Java-based open-source automation platform with plugins designed for continuous integration.

It is used to continually create and test software projects, making it easier for developers and

DevOps engineers to integrate changes to the project and for consumers to get a new build.

PIPELINE



- ➔ There is an external trigger that starts the pipeline.
- ➔ The latest code gets fetched from the source code repository.
- ➔ Then the code gets compiled, built, and packaged.
- ➔ Bunch of tests get run.`
- ➔ Then, the compiled output gets published somewhere.
- ➔ Then, if we do continuous delivery, it gets deployed to live servers.
- ➔ The final output is some sort of notification to say whether the build is succeeded or not.

All the features in the pipeline comes from the plugins. We cannot do this with plain installation of Jenkins.

In Jenkins, all the plugins come from a central online catalog, which also gets used for installation and ongoing updates,

which includes both feature updates and to add new functionality to the plugins,

but also security fixes if there are any vulnerabilities in the plugins.

If we want to use some functionality available already in another plugin, we can just import the dependency on that plugin.

Due to this, we ended up with a huge dependency graph where we only installed 5-6 plugins

but ended up with 50-60 that are deployed or with these interrelated dependencies.

INSTALLATION

Basic steps –

1. Install Java Development Kit (JDK) (minimum jdk version 11 required) – **https://www.oracle.com/java/technologies/downloads/#jdk17-windows**

x64 Installer	159.94 MB

2. Set the Path for the Environmental Variable for JDK

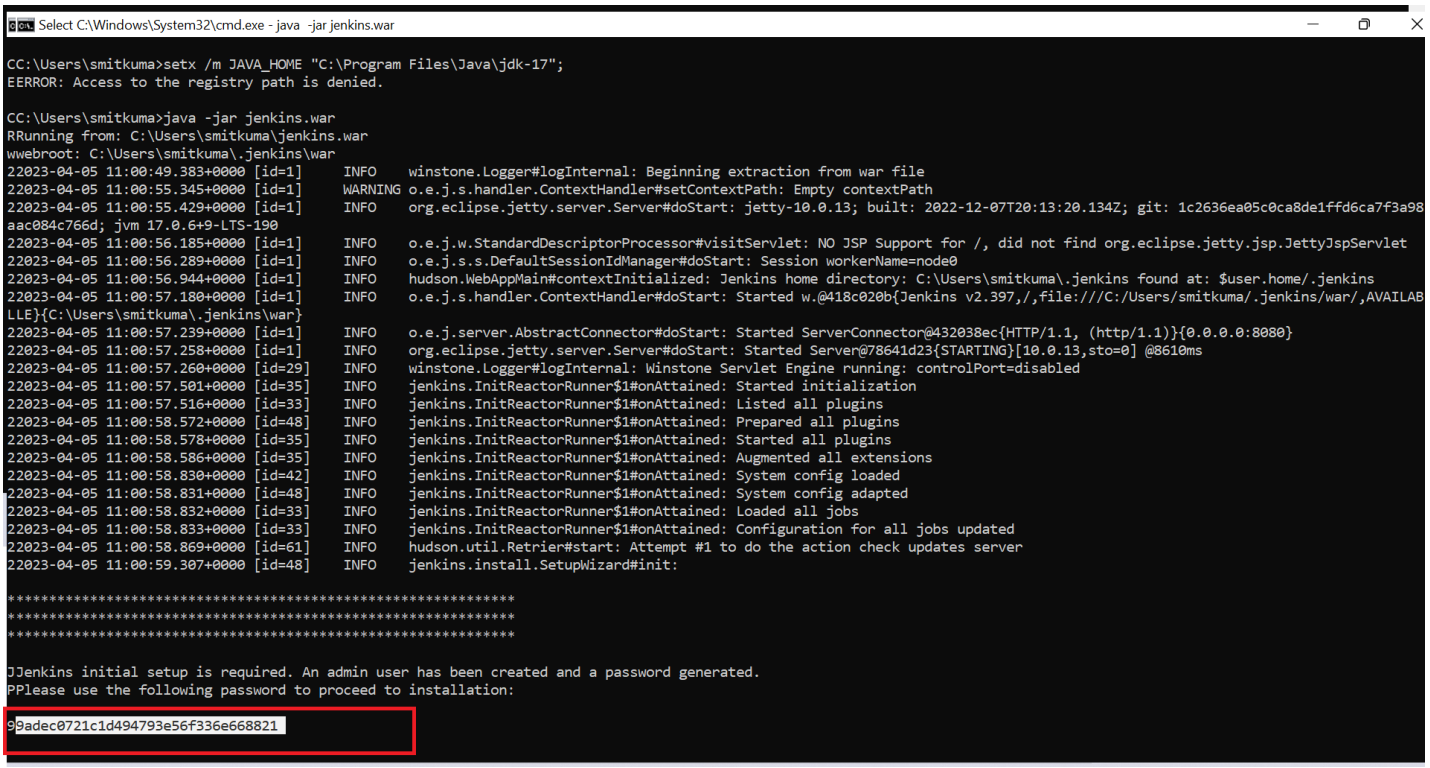
Download and run Jenkins

3. [Download Jenkins Generic Java package (.war)]
4. Open up a terminal in the download directory
5. Run java -jar jenkins.war --httpPort=8080
6. Browse to http://localhost:8080
7. Follow the instructions to complete the installation

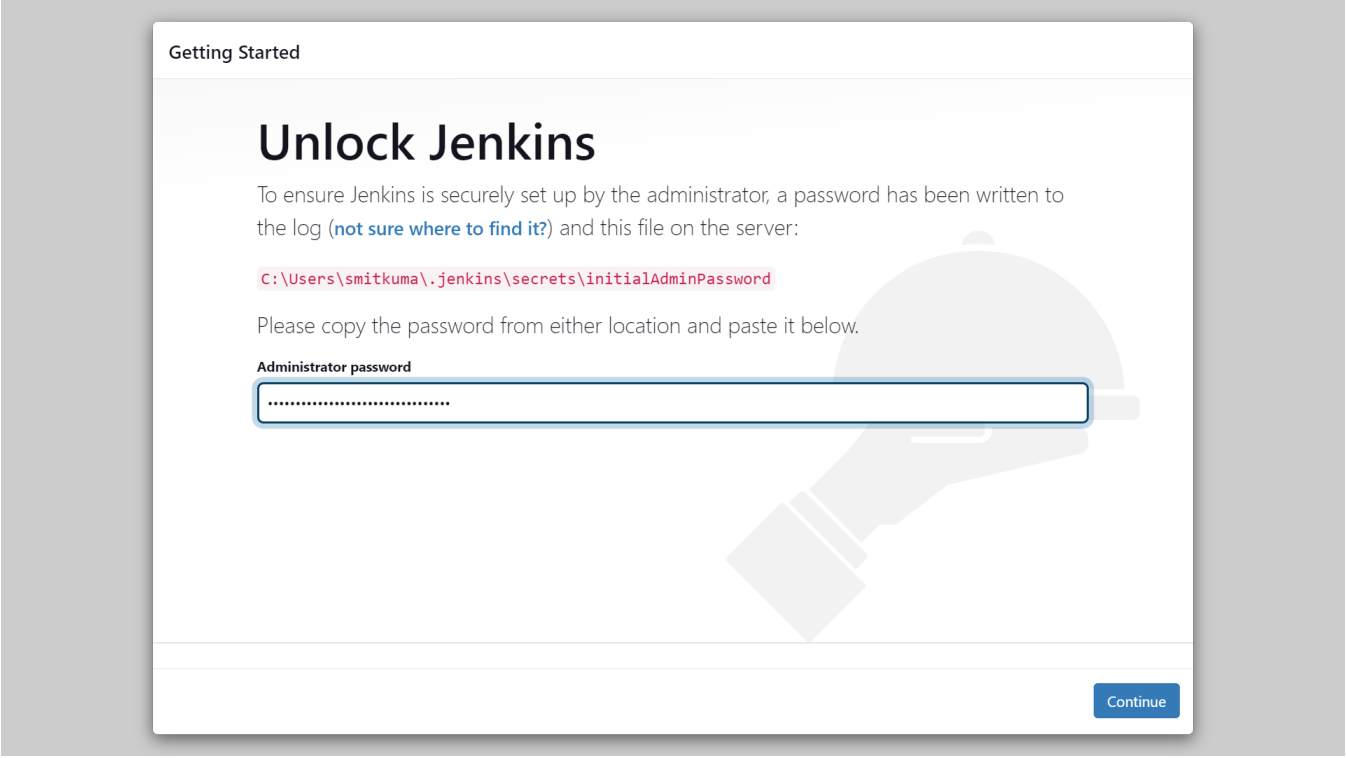
When the installation is complete, you can start putting Jenkins to work!

[Continue to "Create your first Pipeline"]

1. Initialize the jenkins.war file with the command “java -jar jenkins.war” and install it.

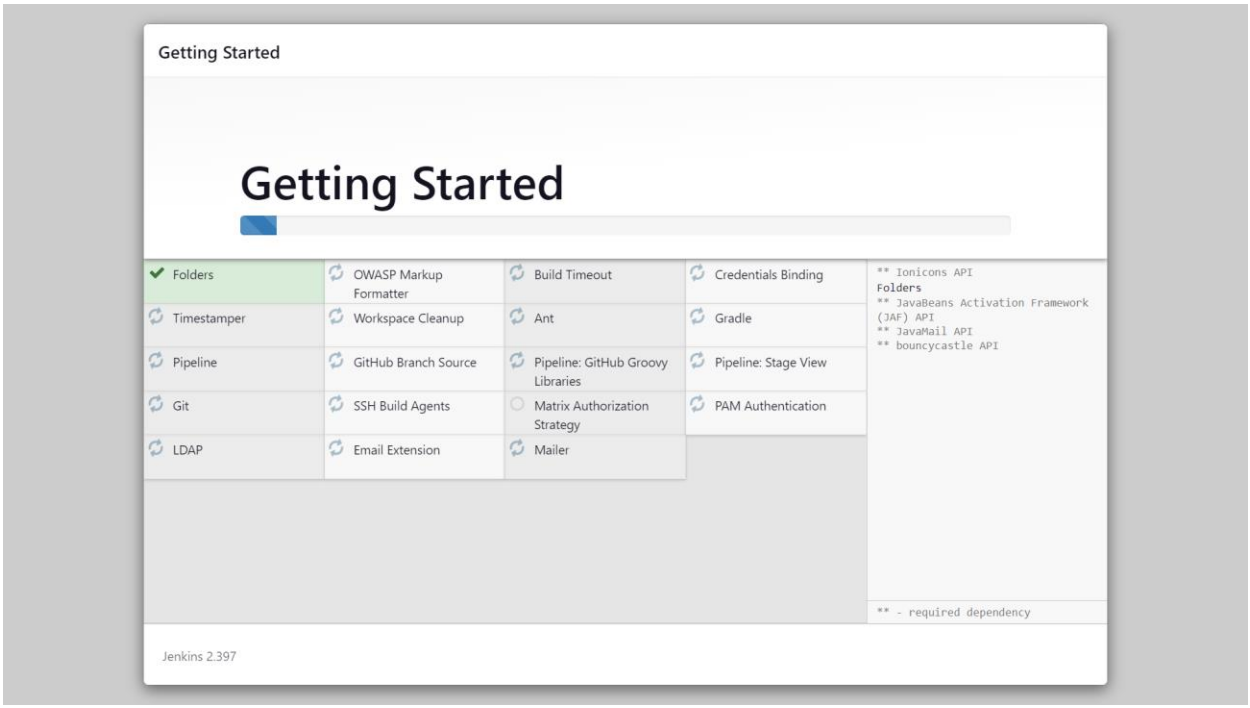


2. Make sure you copy paste the admin password which has been generated- 99adec0721c1d494793e56f336e668821
3. (This may also be found at: C:\Users\smitkuma\.jenkins\secrets\initialAdminPassword)
4. To use Jenkins run https://localhost:8080 (the default port) and
   enter the key obtained after installing Jenkins as shown above to unlock Jenkins. Save this key this is our admin password.

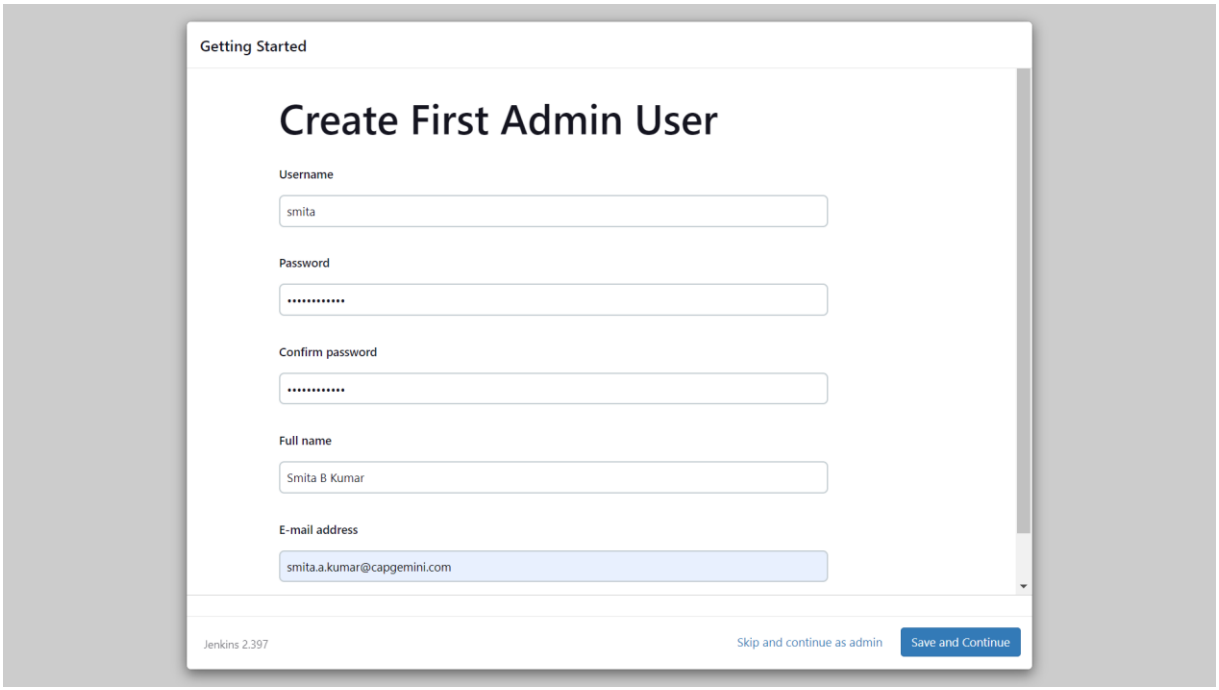


5. Select the required plugins and initiate your jenkins profile

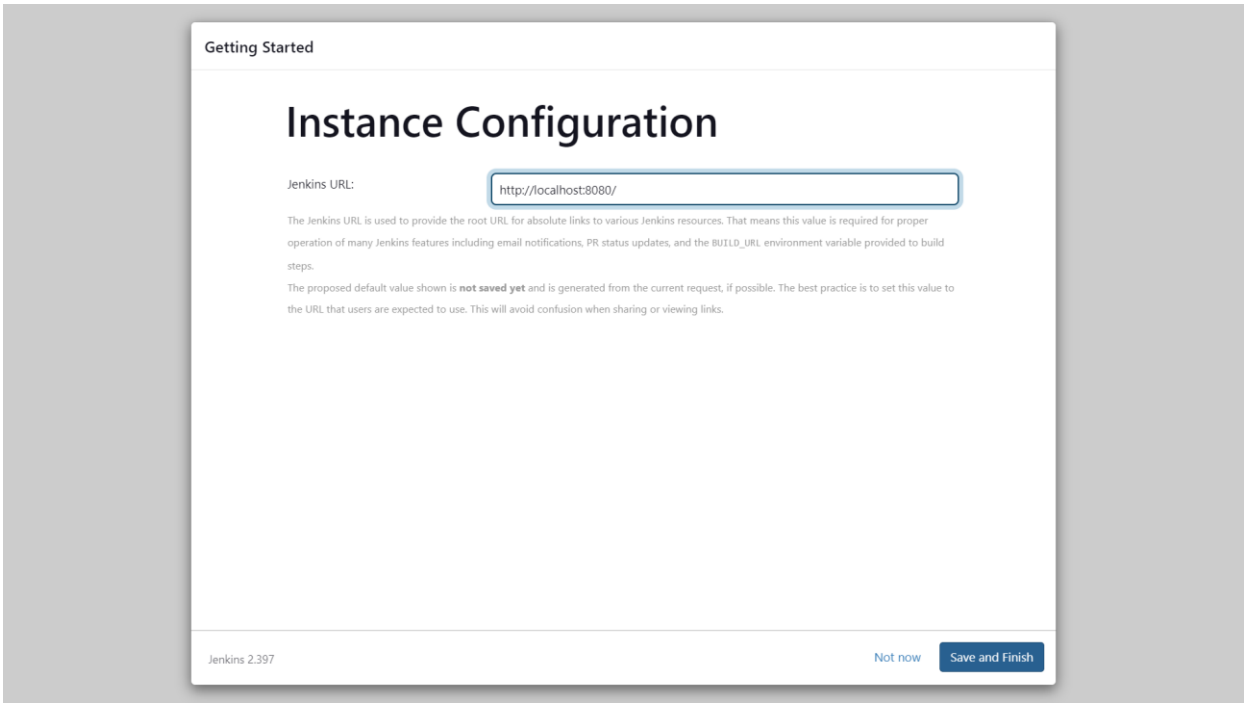
6. Now it will take few minutes to install the plugins.



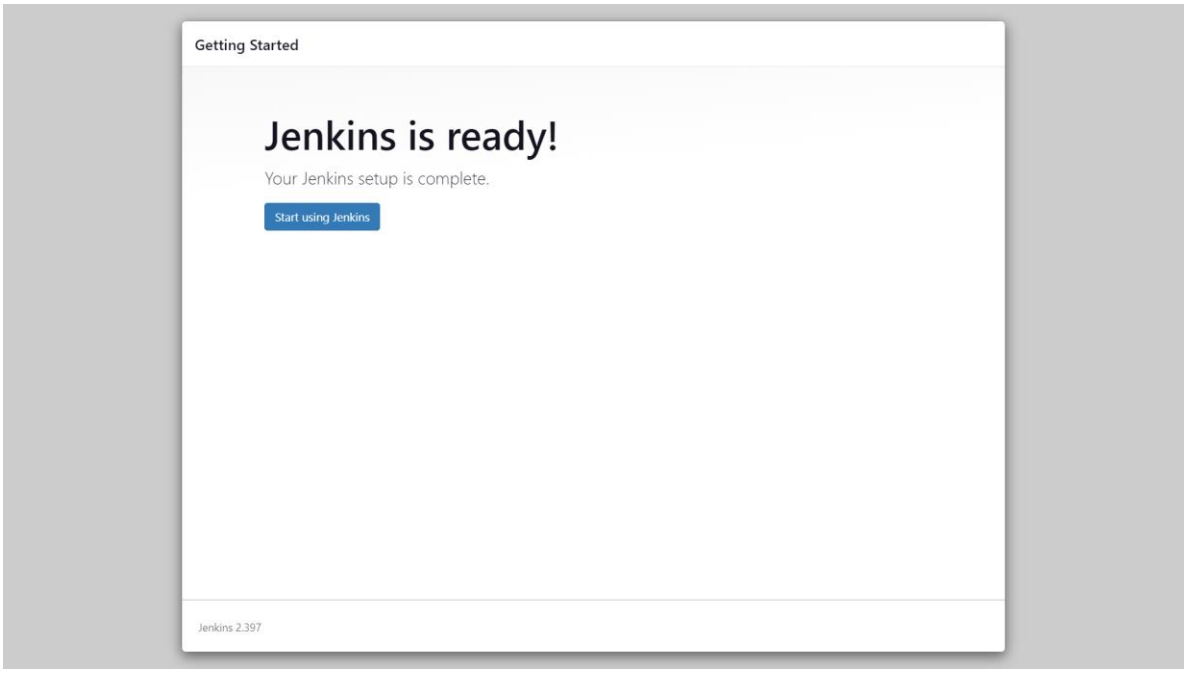
7. Now we can create a personal profile or login as admin – give a simple password which you can remember ex: Password@123



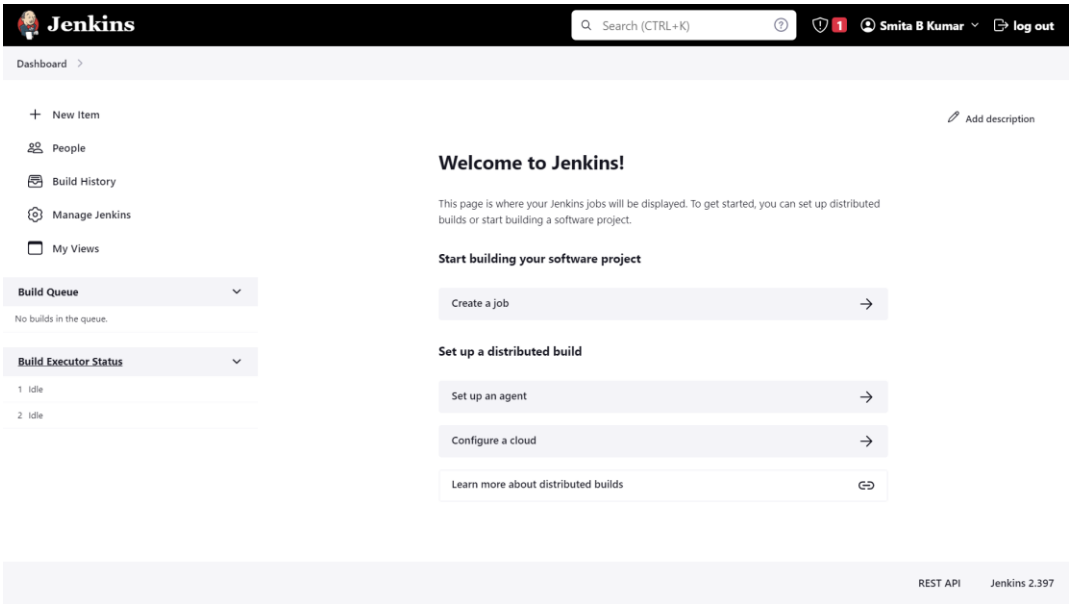
8. We can change the instance if needed at this step



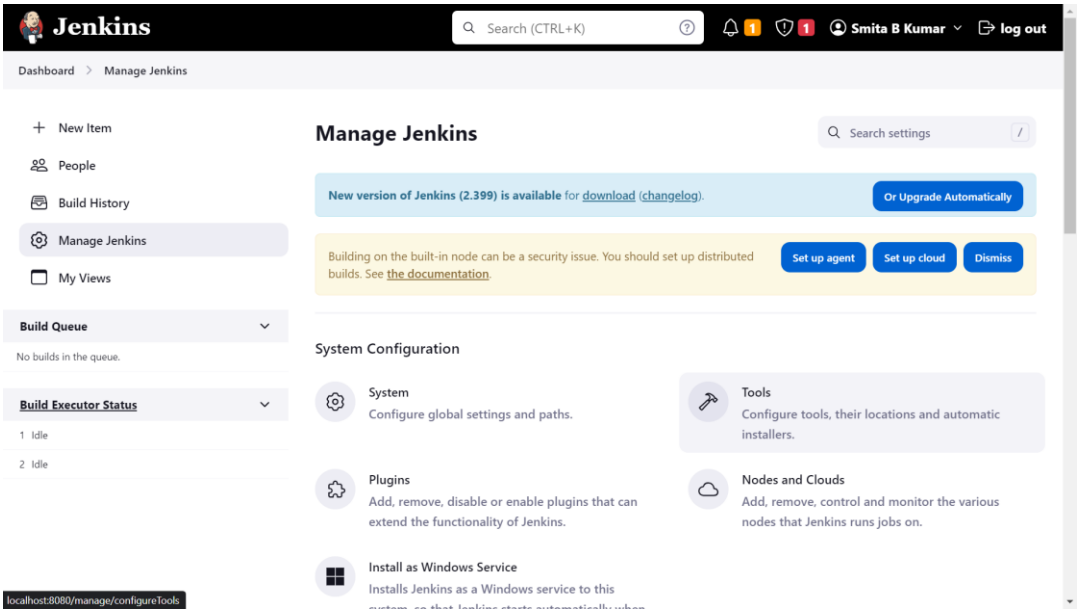
9. Jenkins is installed successfully.

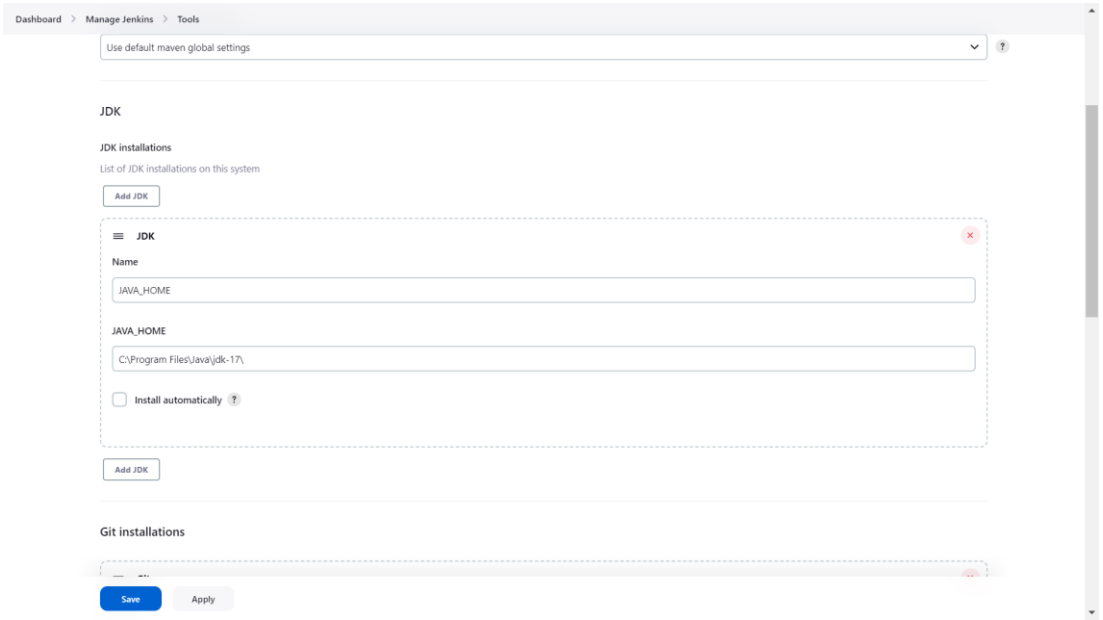


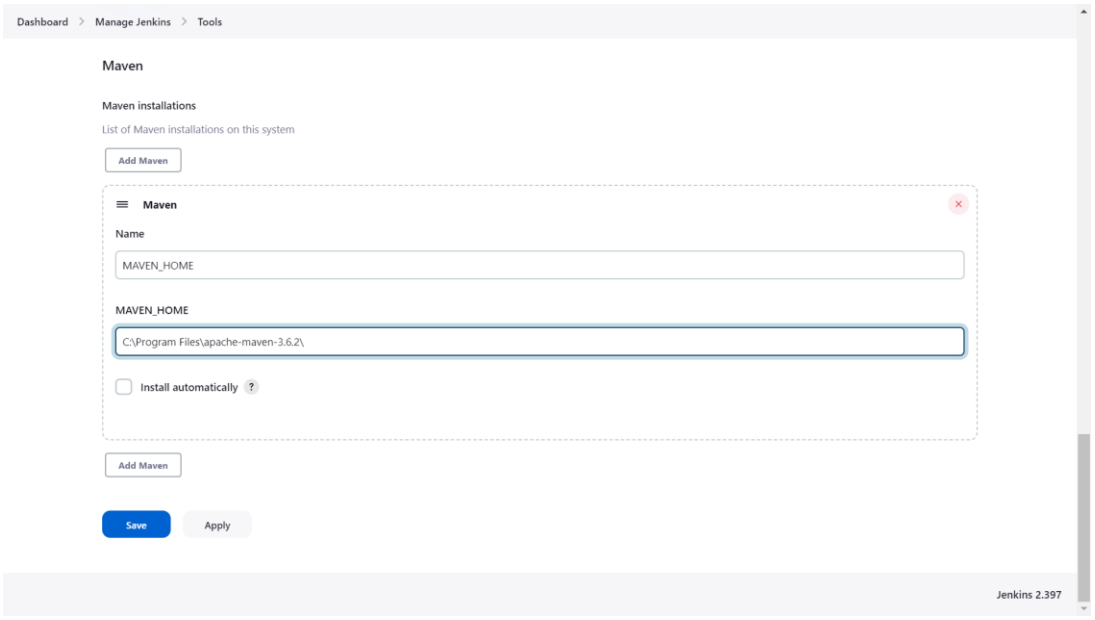
10. Now we can create a job my following these steps and cloning any project.

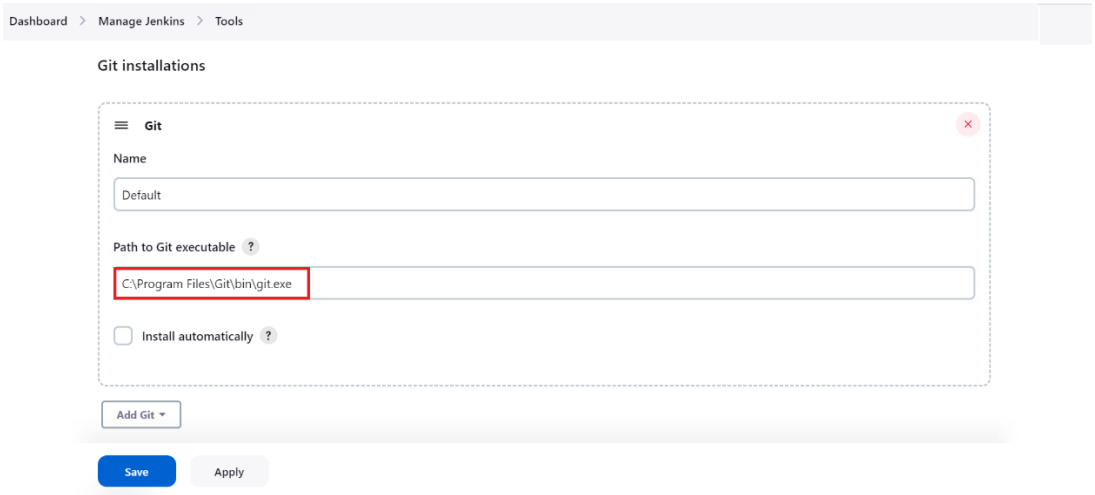


11. Now Click on **Manage Jenkins**-> **Tool** -> Script the **Java Home Path** & **Maven Home Path** and repository URL and the project will be ready to build.

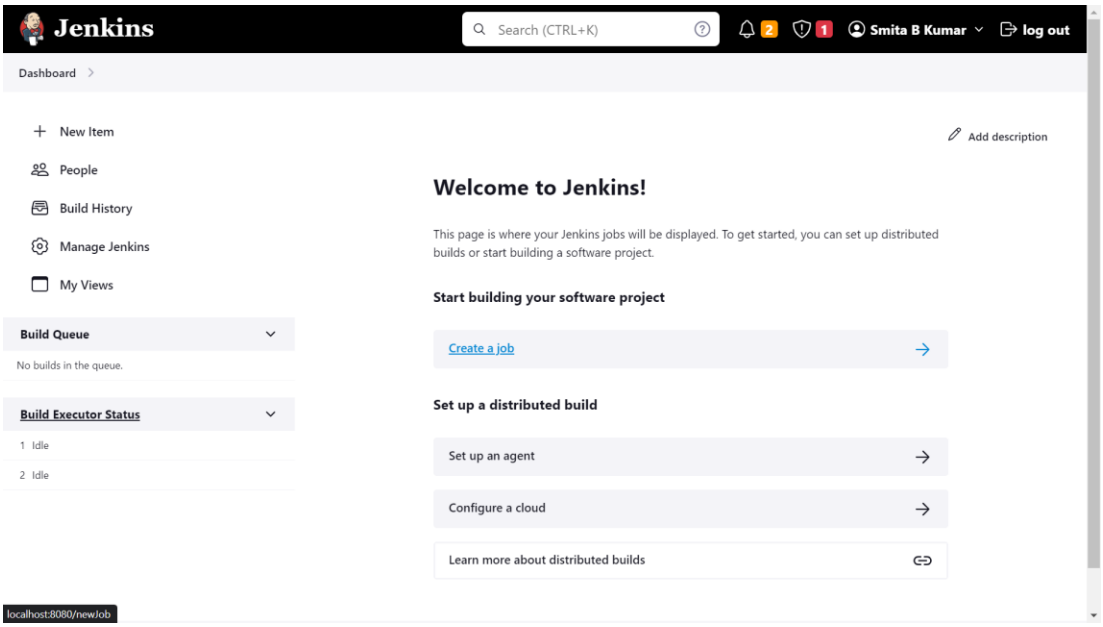






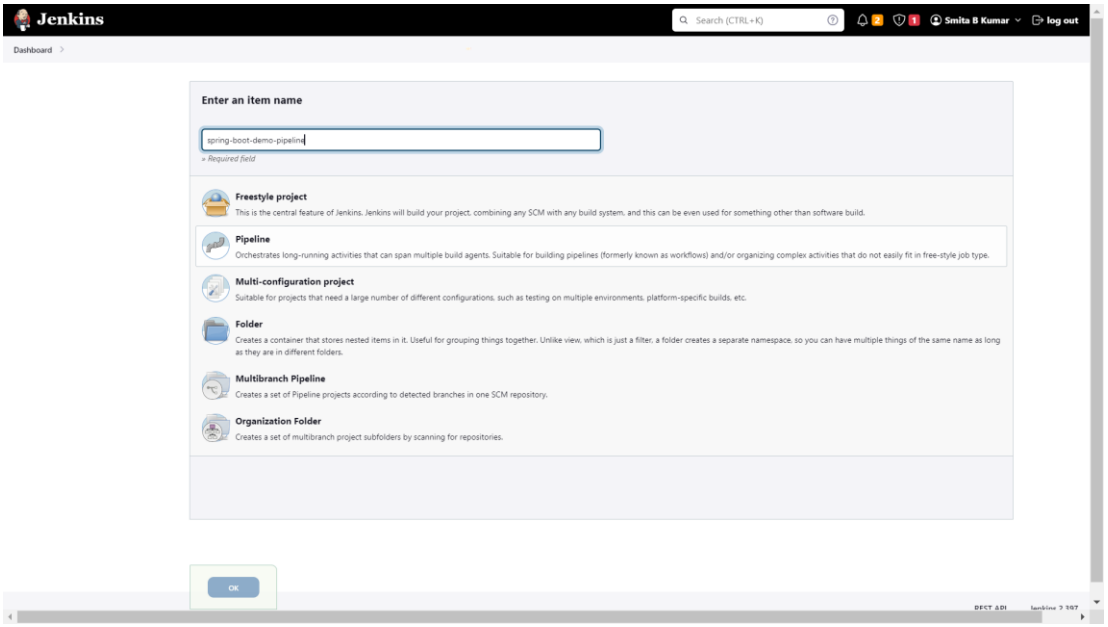


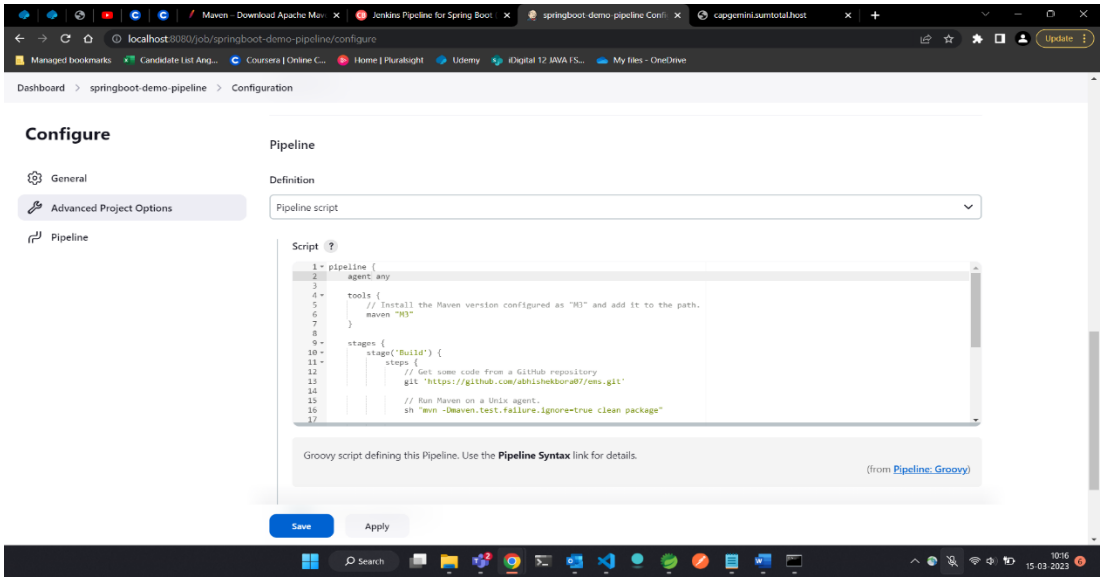
Create a new job

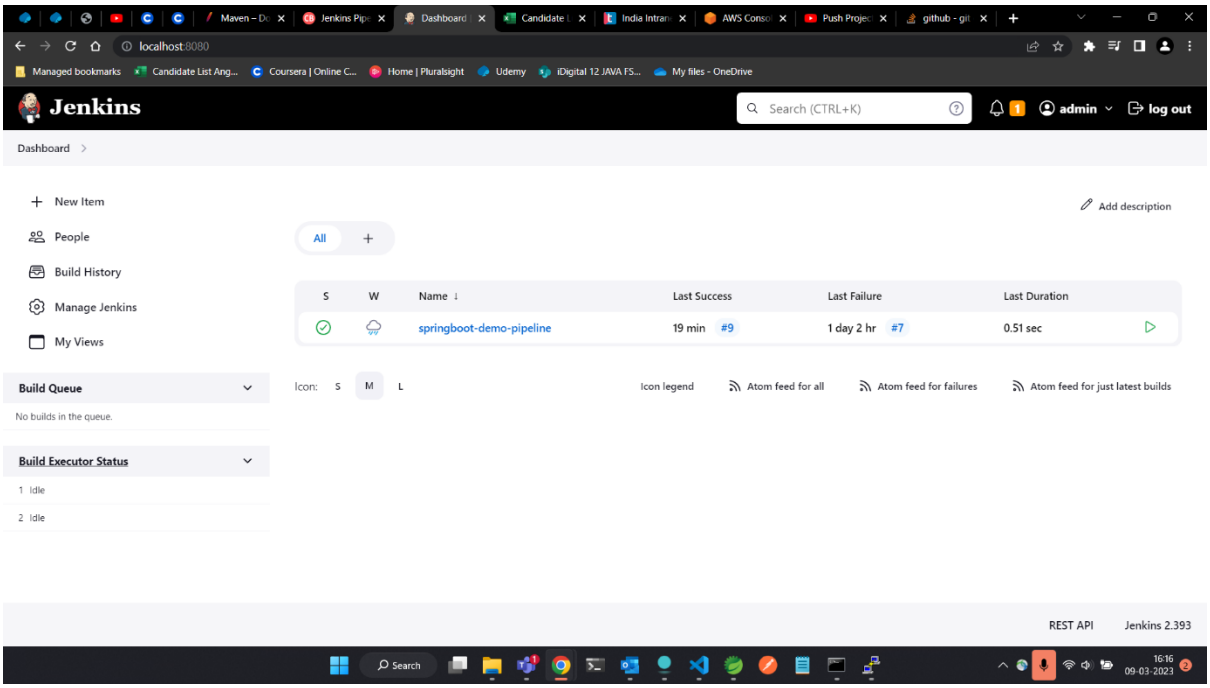


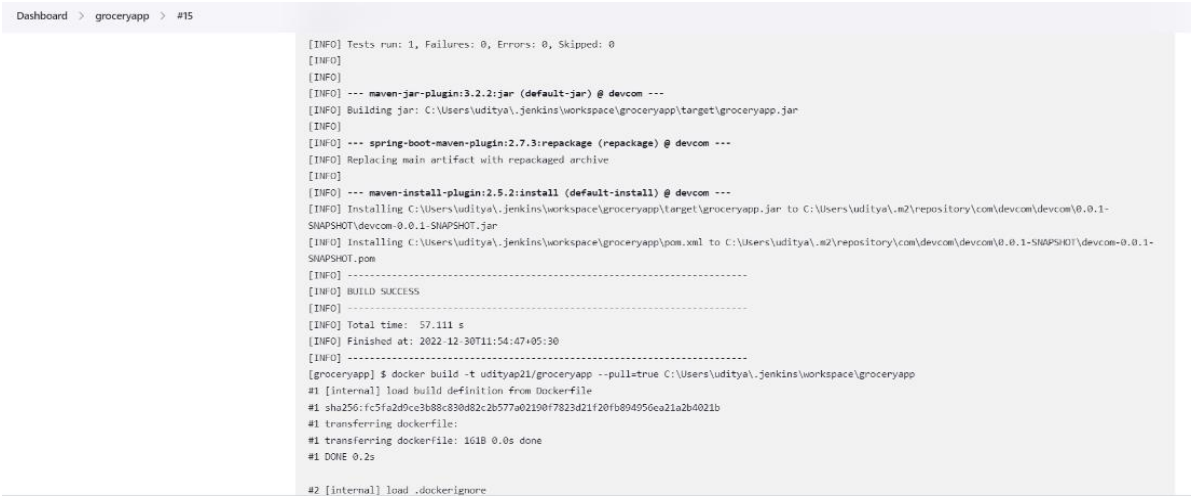
Enter the name - **spring-boot-demo-pipeline**

Select **Pipeline**


Jenkins
Search (CTRL+K)
Smita B Kumar
log out
Dashboard
Enter an item name
spring-boot-demo-pipeline
» 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.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.
Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.
OK


localhost:8080/job/springboot-demo-pipeline/configure
Dashboard
springboot-demo-pipeline
Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script
Script
pipeline {
agent any
tools {
// Install the Maven version configured as "M3" and add it to the path.
maven "M3"
}
stages {
stage('Build') {
steps {
// Get some code from a GitHub repository
git 'https://github.com/abhishekbora07/cms.git'
// Run Maven on a Unix agent.
sh "mvn -Dmaven.test.failure.ignore=true clean package"
Groovy script defining this Pipeline. Use the Pipeline Syntax link for details.
(from Pipeline: Groovy)
Save
Apply


localhost:8080
Jenkins
Search (CTRL+K)
admin
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
springboot-demo-pipeline
19 min #9
1 day 2 hr #7
0.51 sec
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Icon: S M L
Icon legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds
REST API
Jenkins 2.393



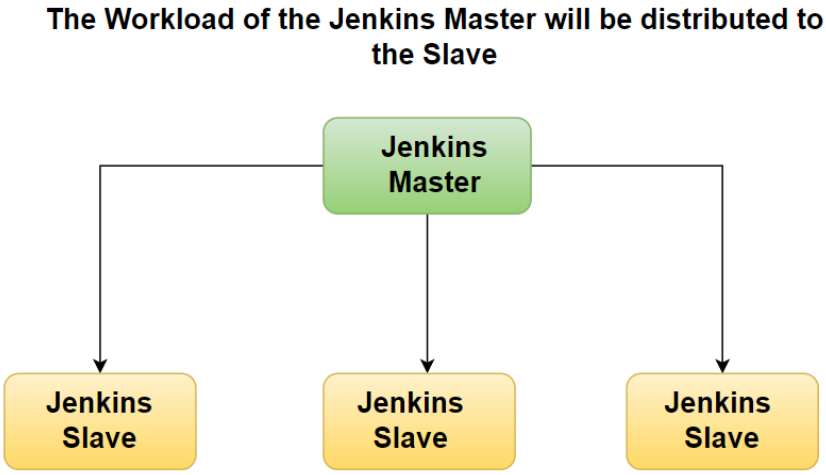
ARCHITECTURE

- ➔ Developers commit changes to the source code, found in the repository.
- ➔ The Jenkins CI server checks the repository at regular intervals and pulls any newly available code.
- ➔ The Build Server builds the code into an executable file. In case the build fails, feedback is sent to the developers.
- ➔ Jenkins deploys the build application to the test server. If the test fails, the developers are alerted.
- ➔ If the code is error-free, the tested application is deployed on the production server.

The files can contain different code and be very large, requiring multiple builds. However,

a single Jenkins server cannot handle multiple files and builds simultaneously; for that,

a distributed Jenkins architecture is necessary.



- ➔ Jenkins Master

  The Jenkins master is in charge of scheduling the jobs, assigning slaves, and .

  sending builds to slaves to execute the jobs. It'll also keep track of the slave state (offline or online)

and retrieve the build result responses from slaves and display them on the console output.

➔ Jenkins Slave

It runs on the remote server. The Jenkins server follows the requests of the Jenkins master and is compatible with all operating systems. Building jobs dispatched by the master are executed by the slave.

Also, the project can be configured to choose a specific slave machine.