## A picture containing text, clipart, sign Description automatically generatedMahavir Education Trust's

## Shah & Anchor Kutchhi Engineering College,

**Chembur, Mumbai 400 088**

## UG Program in Information Technology

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment No: 4** | | | | | |
| **Date of Performance:** |  | | | | |
| **Date of Submission:** |  | | | | |
| **Program formation/ Execution/**  **ethical practices (07)** | **Documentation (02)** | **Timely Submission (03)** | **Viva Answer (03)** | **Experiment Marks (15)** | **Teacher Signature with date** |
|  |  |  |  |  |  |

**Experiment No.: 4**

**Aim:** To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build Job.

**Lab Outcome:** To understand the importance of Jenkins to Build and deploy Software Applications on server environment

|  |  |
| --- | --- |
| **Theory:** |  |
|  | |

**Jenkins** is an open-source Continuous Integration server written in Java for orchestrating a chain of actions to achieve the Continuous Integration process in an automated fashion. Jenkins supports the complete development life cycle of software from building, testing, documenting the software, deploying, and other stages of the software development life cycle.

Jenkins is a widely used application around the world that has around 300k installations and growing day by day. By using Jenkins, software companies can accelerate their software development process, as Jenkins can automate build and test at a rapid rate.

It is a server-based application and requires a web server like Apache Tomcat. The reason Jenkins software became so popular is that of its monitoring of repeated tasks which arise during the development of a project. For example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development. **Why Jenkins?**

Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.

Diagram, text

Description automatically generated

Along with Jenkins, sometimes, one might also see the association of Hudson. Hudson is a very popular open-source Java-based continuous integration tool developed by Sun Microsystems which was later acquired by Oracle. After the acquisition of Sun by Oracle, a fork was created from the Hudson source code, which brought about the introduction of Jenkins.

**What is Continuous Integration?**

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to have frequent builds.

Diagram

Description automatically generated

It is the practice of automating the integration of code changes from multiple contributors into a single software project. It’s a primary [DevOps best practice,](https://www.atlassian.com/devops/what-is-devops/devops-best-practices) allowing developers to frequently merge code changes into a central repository where builds and tests then run.

Automated tools are used to assert the new code’s correctness before integration.

A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more.

# Importance of continuous integration

In order to understand the importance of CI, it’s helpful to first discuss some pain points that often arise due to the absence of CI. Without CI, developers must manually coordinate and communicate when they are contributing code to the end product. This coordination extends beyond the development teams to operations and the rest of the organization. Product teams must coordinate when to sequentially launch features and fixes and which team members will be responsible.

The communication overhead of a non-CI environment can become a complex and entangled synchronization chore, which adds unnecessary bureaucratic cost to projects. This causes slower code releases with higher rates of failure, as it requires developers to be sensitive and thoughtful towards the integrations. These risks grow exponentially as the engineering team and codebase sizes increase.

Without a robust CI pipeline, a disconnect between the engineering team and the rest of the organization can form. Communication between product and engineering can be cumbersome. Engineering becomes a black box which the rest of the team inputs requirements and features and maybe gets expected results back. It will make it harder for engineering to estimate time of delivery on requests because the time to integrate new changes becomes an unknown risk.

# What CI does

CI helps to scale up headcount and delivery output of engineering teams. Introducing CI to the aforementioned scenario allows software developers to work independently on features in parallel. When they are ready to merge these features into the end product, they can do so independently and rapidly. CI is a valuable and well-established practice in modern, high performance software engineering organizations.

# How CI can be used

CI is generally used alongside an agile software development workflow. An organization will compile list of tasks that comprise a product roadmap. These tasks are then distributed amongst software engineering team members for delivery. Using CI enables these software development tasks to be developed independently and in parallel amongst the assigned developers. Once one of these tasks is complete, a developer will introduce that new work to the CI system to be integrated with the rest of the project.

# Benefits and challenges of continuous integration

Continuous integration is an essential aspect of [DevOps](https://www.atlassian.com/devops/what-is-devops) and high-performing software teams. Yet CI benefits are not limited to the engineering team but greatly benefit the overall organization. CI enables better transparency and insight into the process of software development and delivery. These benefits enable the rest of the organization to better plan and execute go to market strategies. The following are some of the overall organizational benefits of CI. Enable scaling

Improve the feedback loop

Enhance communication

Adoption and installation

Technology learning curve

# Advantages of Jenkins

* It is open source and it is user-friendly, easy to install and does not require additional installations or components.
* It is free of cost.
* Easily Configurable. Jenkins can be easily modified and extended. It deploys code instantly, generates test reports. Jenkins can be configured according to the requirements for continuous integrations and continuous delivery.
* Platform Independent. Jenkins is available for all platforms and different operating systems, whether OS X, Windows or Linux.
* Rich Plugin ecosystem. The extensive pool of plugins makes Jenkins flexible and allows building, deploying and automating across various platforms.
* Easy support. Because it is open source and widely used, there is no shortage of support from large online communities of [agile teams.](https://apiumhub.com/?p=56666)
* Developers write the tests to detect the errors of their code as soon as possible. So the developers don’t waste time on large-scale error-ridden integrations.
* Issues are detected and resolved almost right away which keeps the software in a state where it can be released at any time safely.
* Most of the integration work is automated. Hence fewer integration issues. This saves both time and money over the lifespan of a project.

# Disadvantages of Jenkins

* Its interface is out dated and not user friendly compared to current user interface trends.
* Not easy to maintain it because it runs on a server and requires some skills as server administrator to monitor its activity.
* CI regularly breaks due to some small setting changes. CI will be paused and therefore requires some developer's team attention.

# Jenkins Architecture

Diagram

Description automatically generated

This single Jenkins server was not enough to meet certain requirements like:

* Sometimes you might need several different environments to test your builds. This cannot be done by a single Jenkins server.
* If larger and heavier projects get built on a regular basis then a single Jenkins server cannot simply handle the entire load.

To address the above-stated needs, Jenkins distributed architecture came into the picture.

Jenkins Distributed Architecture

Jenkins uses a Master-Slave architecture to manage distributed builds. In this architecture, Master and Slave communicate through TCP/IP protocol.

# Jenkins Master

Your main Jenkins server is the Master. The Master’s job is to handle:

* Scheduling build jobs.
* Dispatching builds to the slaves for the actual execution.
* Monitor the slaves (possibly taking them online and offline as required).
* Recording and presenting the build results.
* A Master instance of Jenkins can also execute build jobs directly.

# Jenkins Slave

A Slave is a Java executable that runs on a remote machine. Following are the characteristics of Jenkins Slaves:

* It hears requests from the Jenkins Master instance.
* Slaves can run on a variety of operating systems.
* The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.
* You can configure a project to always run on a particular Slave machine or a particular type of Slave machine, or simply let Jenkins pick the next available Slave.

The diagram below is self-explanatory. It consists of a Jenkins Master which is managing three Jenkins Slave.

Graphical user interface, website

Description automatically generated

**How Jenkins Master and Slave Architecture works?**

Now let us look at an example in which we use Jenkins for testing in different environments like Ubuntu, MAC, Windows, etc.

The diagram below represents the same:

A picture containing diagram

Description automatically generated

The above image represents the following functions:

* Jenkins checks the Git repository at periodic intervals for any changes made in the source code.
* Each builds requires a different testing environment which is not possible for a single Jenkins server. In order to perform testing in different environments, Jenkins uses various Slaves as shown in the diagram.
* Jenkins Master requests these Slaves to perform testing and to generate test reports.

**Steps for installing Jenkins:**

**Step 1:** Got to<https://www.jenkins.io/download/>and select the platform. In our case Windows.

A screenshot of a computer

Description automatically generated

**Step 2:** Go to download location from local computer and unzip the downloaded package. Doubleclick on unzipped jenkins.msi. You can also Jenkin using a WAR (Web application ARchive) but that is not recommended.

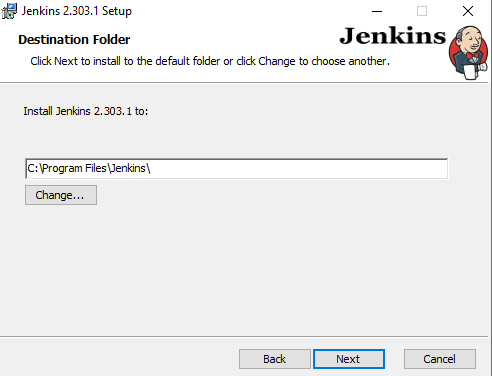
Graphical user interface, text, application

Description automatically generated

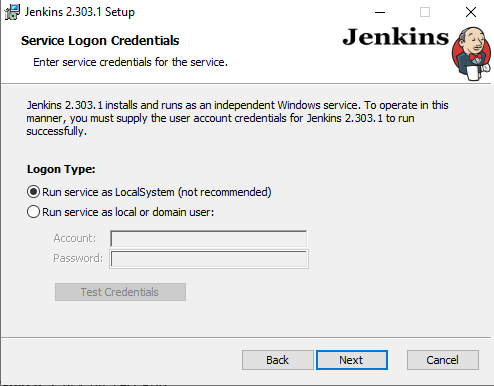
**Step 3:** Click “Next” to start the installation.



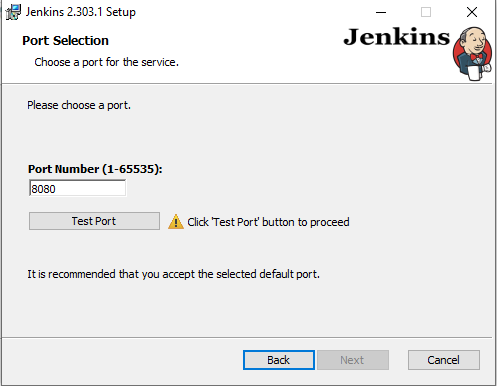
**Step 4:** Click the “Change…” button if you want to install Jenkins in another folder. In my case, I will keep the default option and click on the “Next” button.

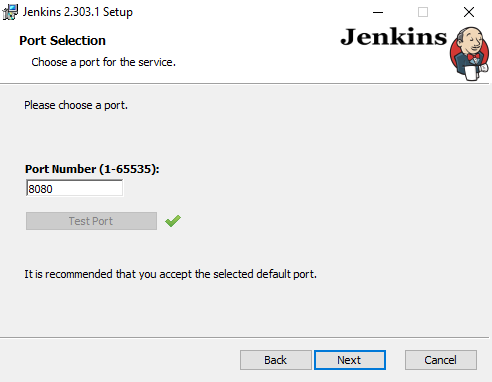


**Step 5:** Select Run service as LocalSystem

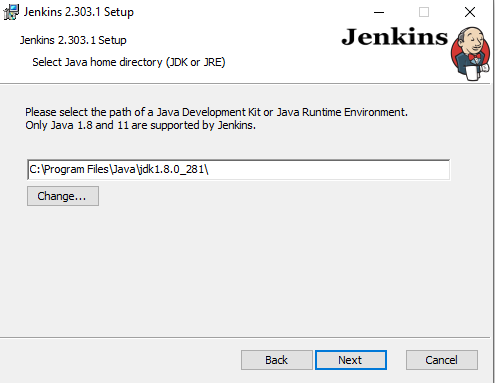


**Step 6:** Click on Test Port

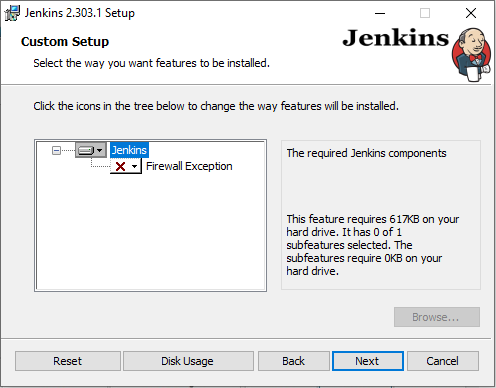




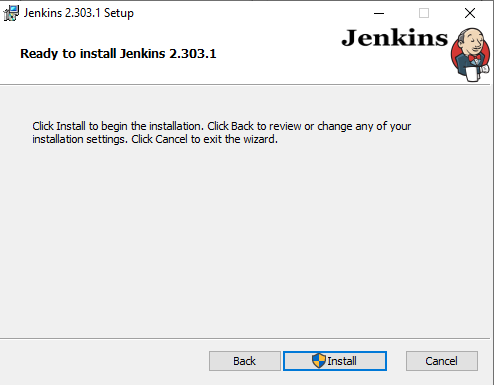
**Step 7:** Give the path to your jdk



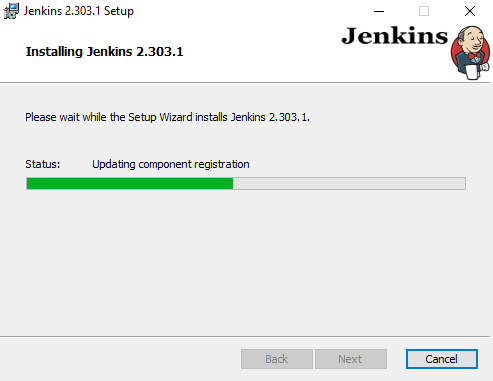
**Step 8:** Keep it as default and click on Next



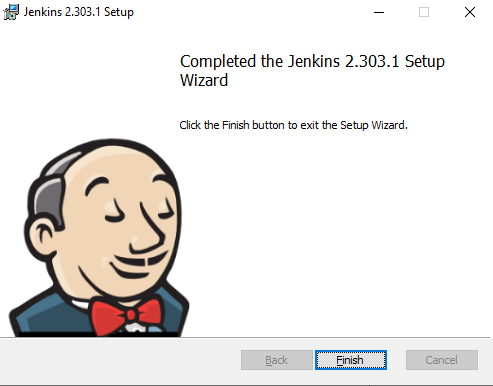
**Step 9:** Click the “Install” button to start the installation process.



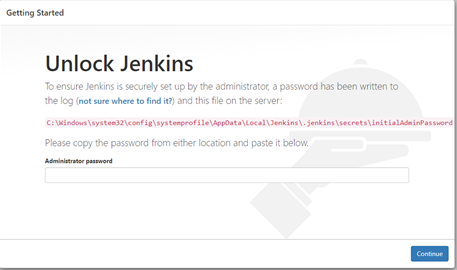
**Step 10:** The installation will proceed.



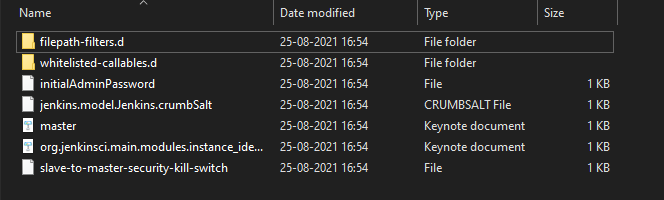
**Step 11:** When done, click the “Finish” button to complete the installation process.

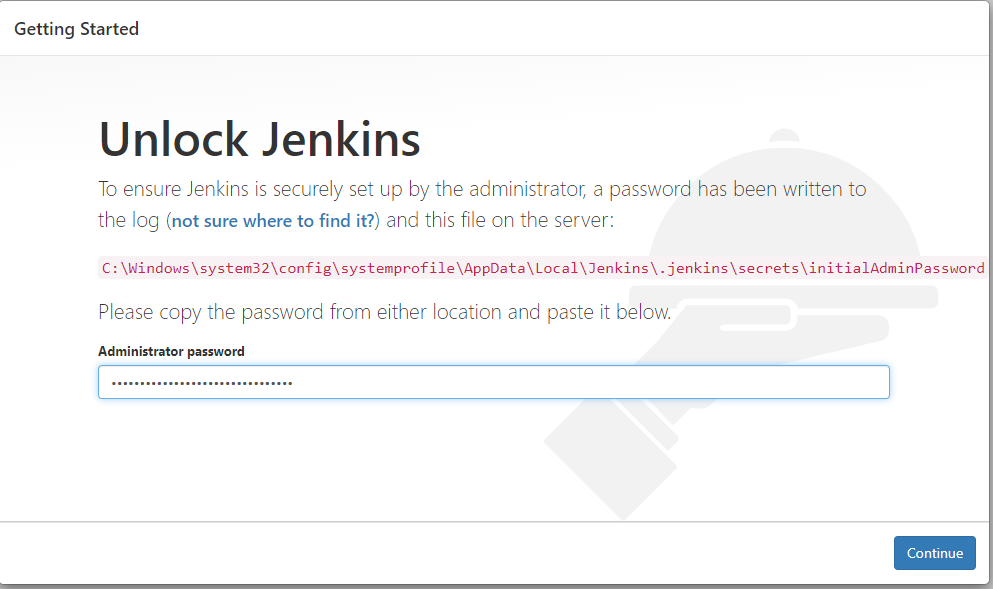


**2:** You will automatically be redirected to a local Jenkins page, or you can paste the URL [http://localhost:8080](http://localhost:8080/) in a browser.

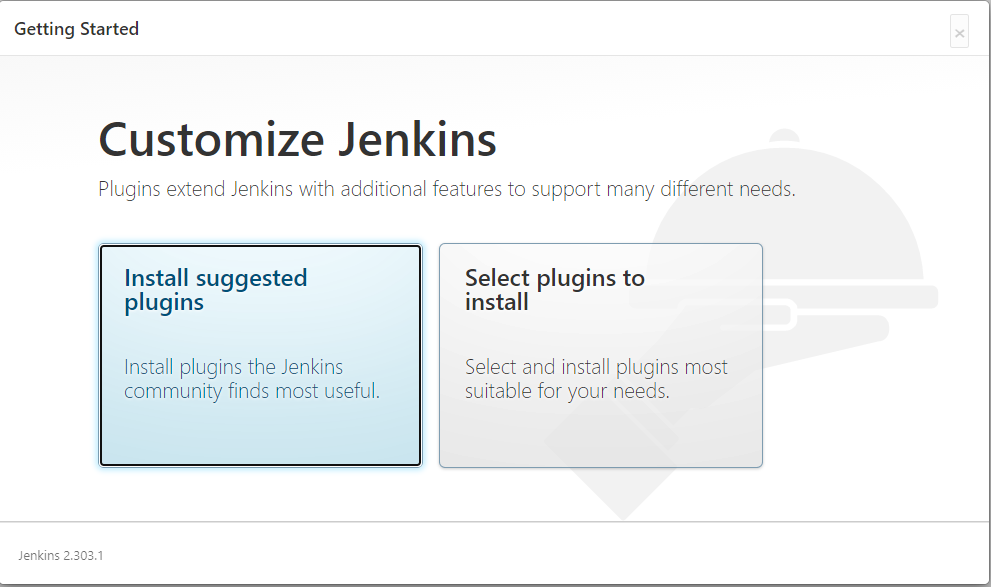


**Step 13:** To unlock Jenkins, copy the password from the file at C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword and paste it in the Administrator password field. Then, click the “Continue” button.

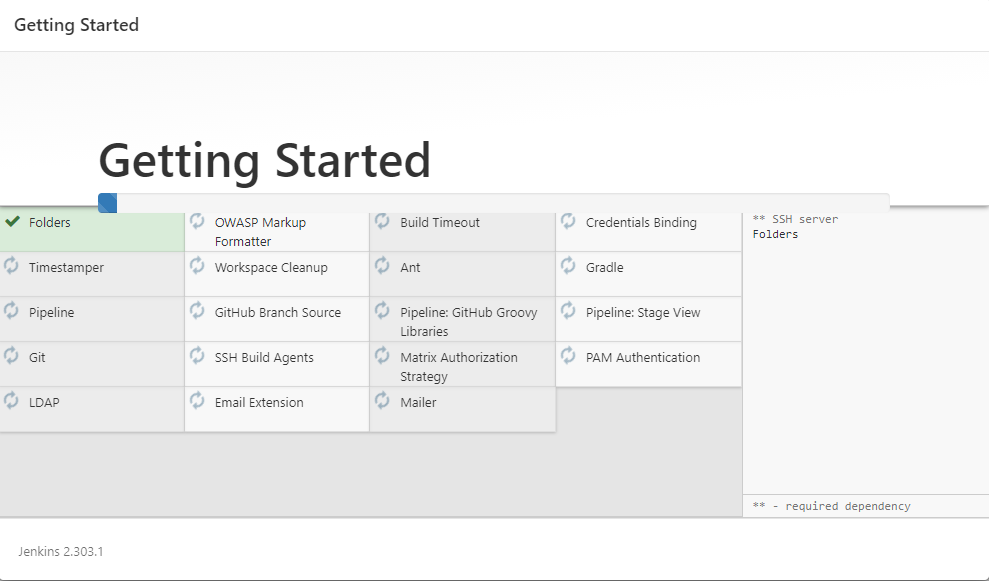




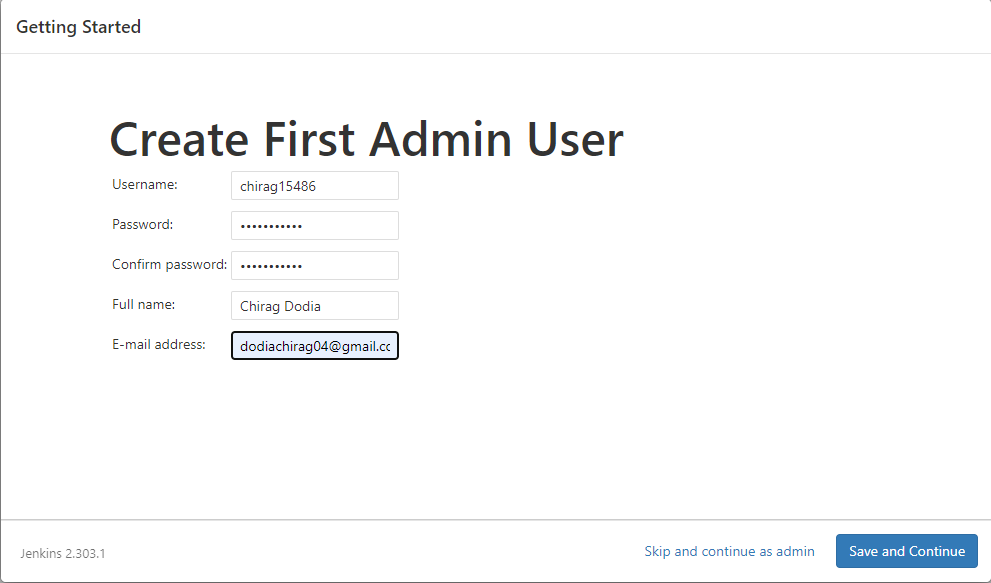
**Step 14:** You can install either the suggested plugins or selected plugins you choose. To keep it simple, we will install the suggested plugins.



**5:** Wait until the plugins are completely installed.

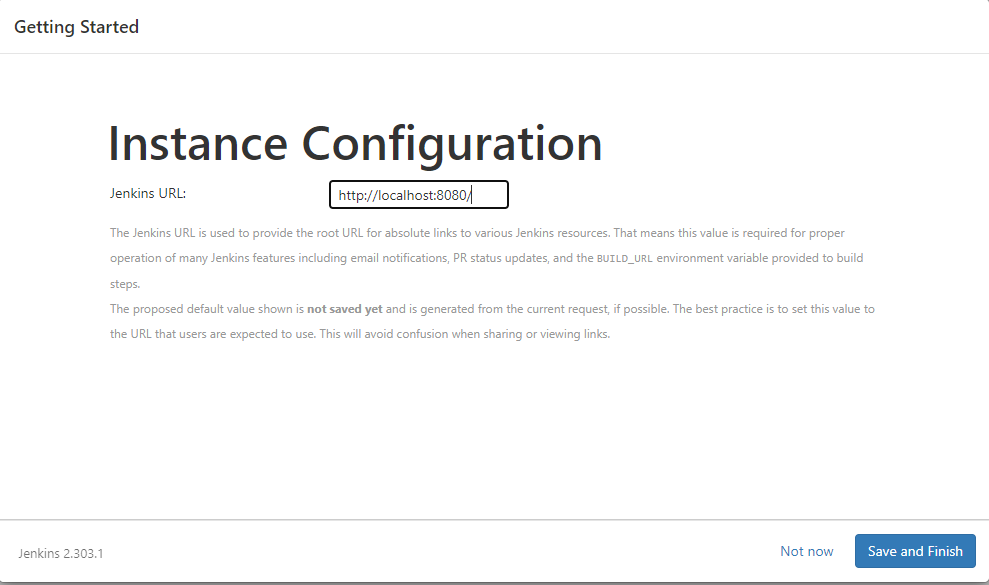


**Step 15:** The next thing that you should do is create an Admin user for Jenkins. Then, enter your details and click “Save and Continue”.

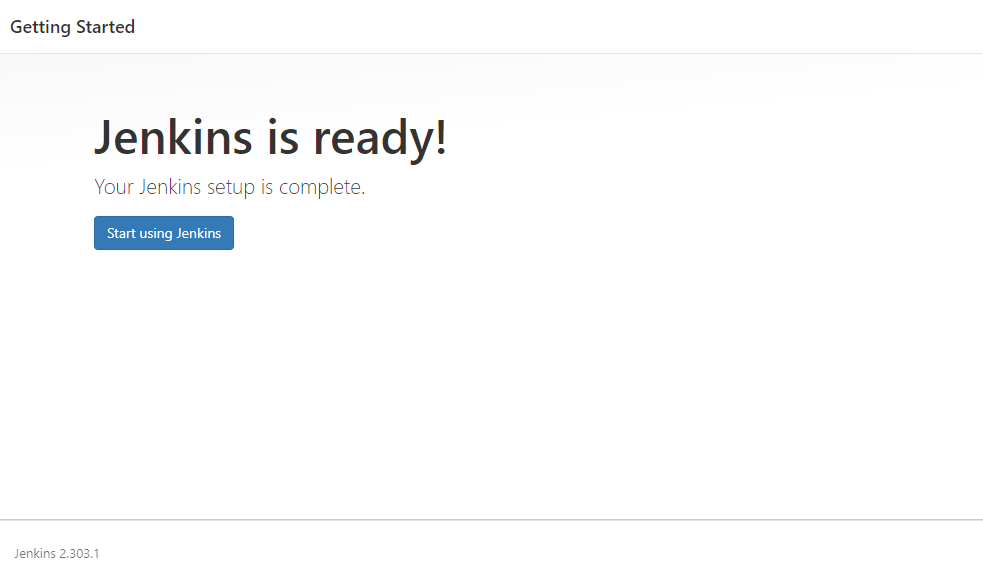


**Step 16:**

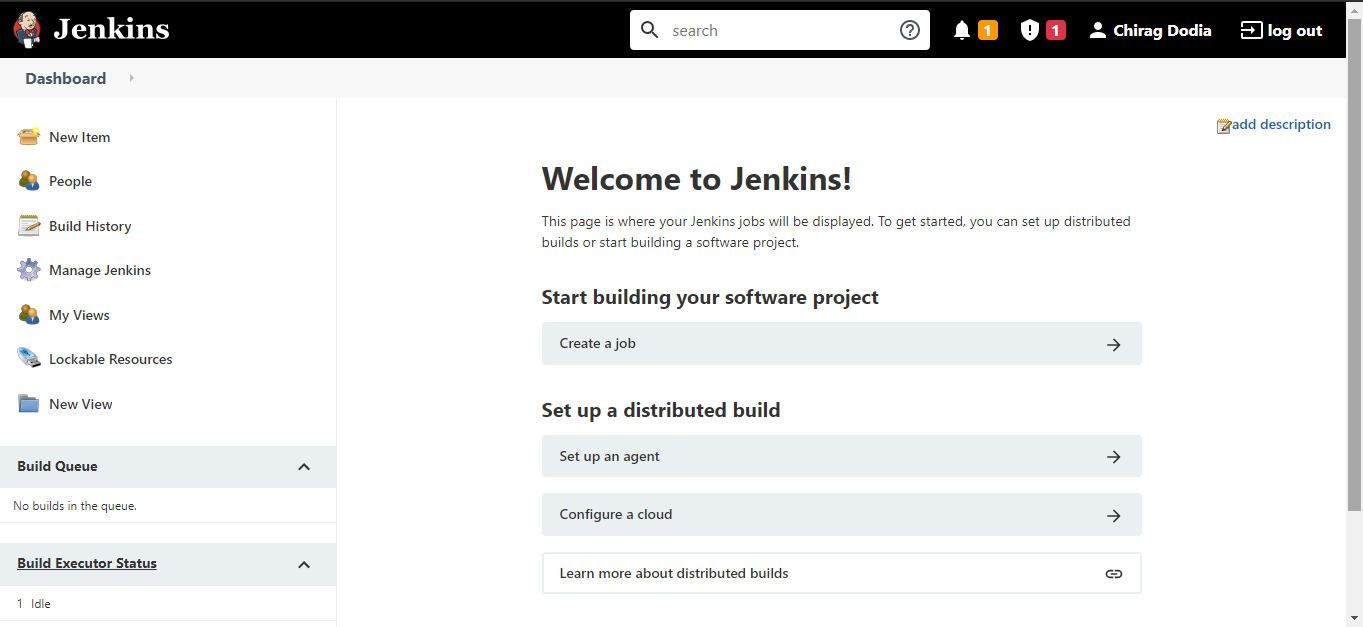
**7:** Click “Save and Finish” to complete the Jenkins installation.



**Step 17:** Now, click “Start using Jenkins” to start Jenkins.



**9:** Finally, here is the default Jenkins page



**Steps to create job in Jenkins:**

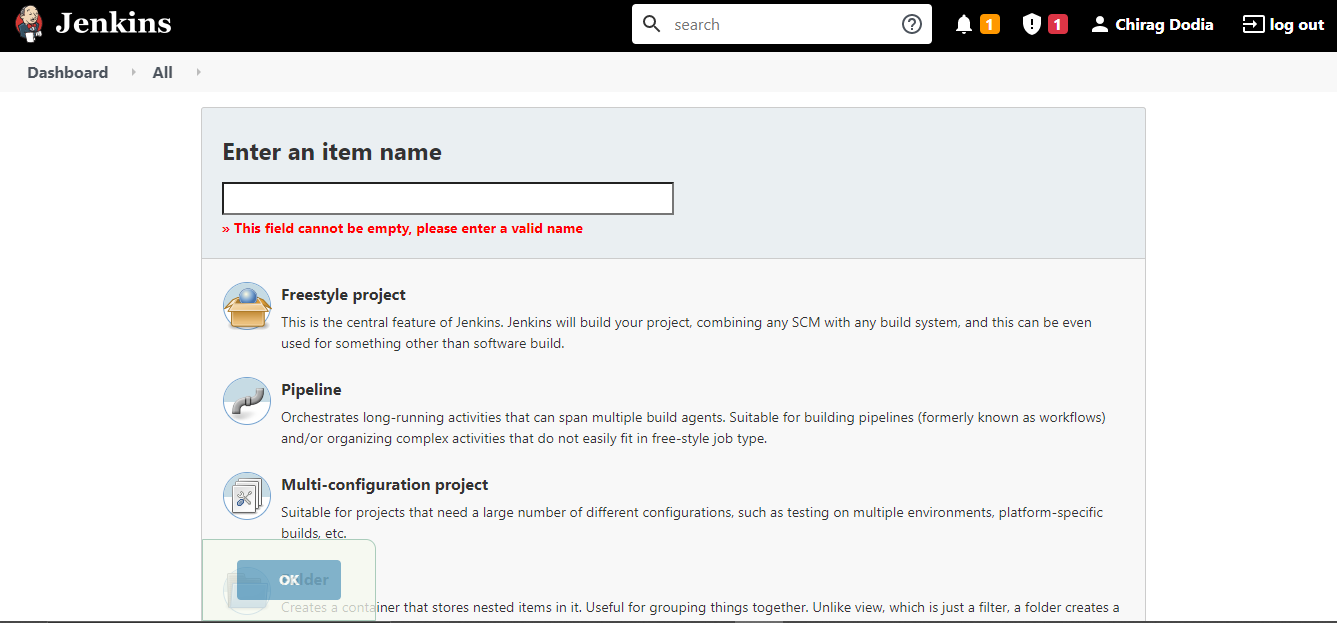
**Step 1:** Click on “New Item” at the top left-hand side of your dashboard.

A screenshot of a computer

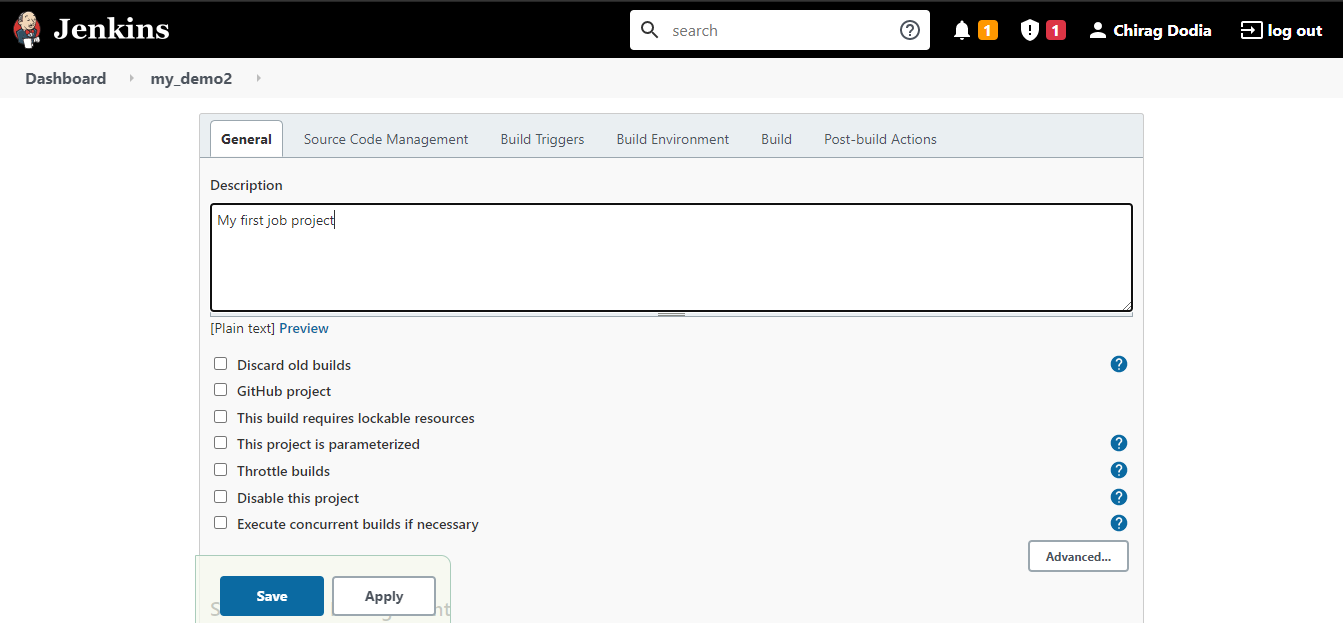
Description automatically generated

**Step 2:** In the next screen,

1. Enter the name of the item you want to create. We shall use the “Hello world” for this demo.
2. Select Freestyle project
3. Click Okay



**Step 3:** Enter the details of the project you want to test.

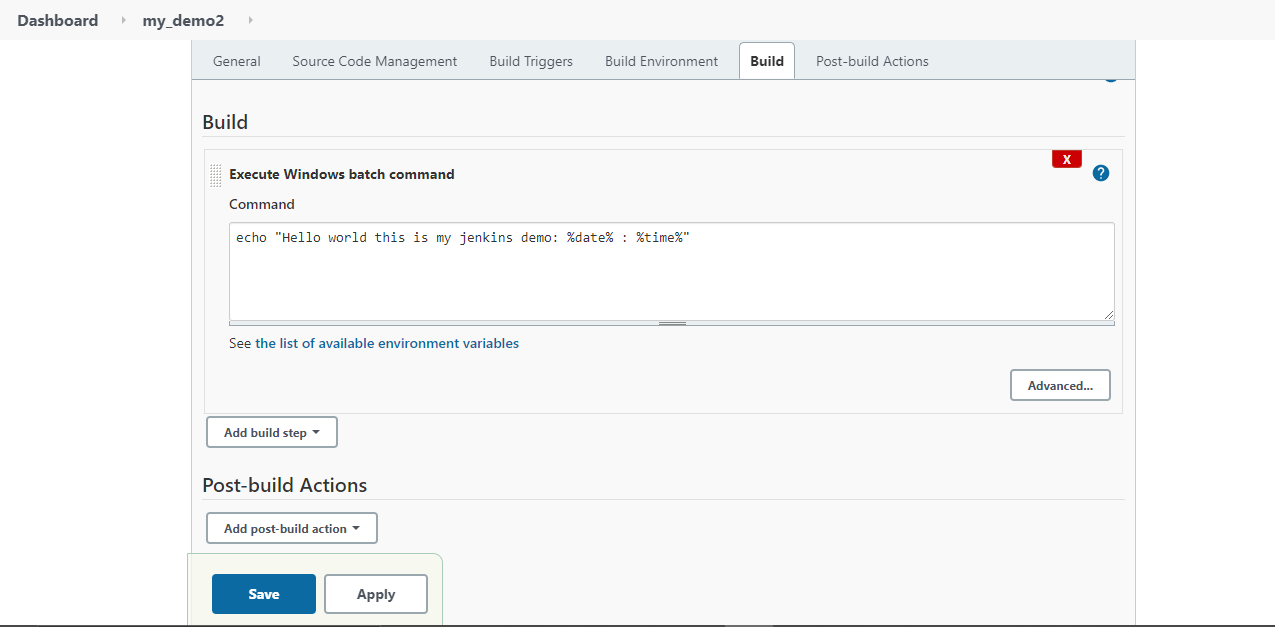


**Step 4:** Now that you have provided all the details, it’s time to build the code. Tweak the settings under the build section to build the code at the time you want. You can even schedule the build to happen periodically, at set times.

Under build,

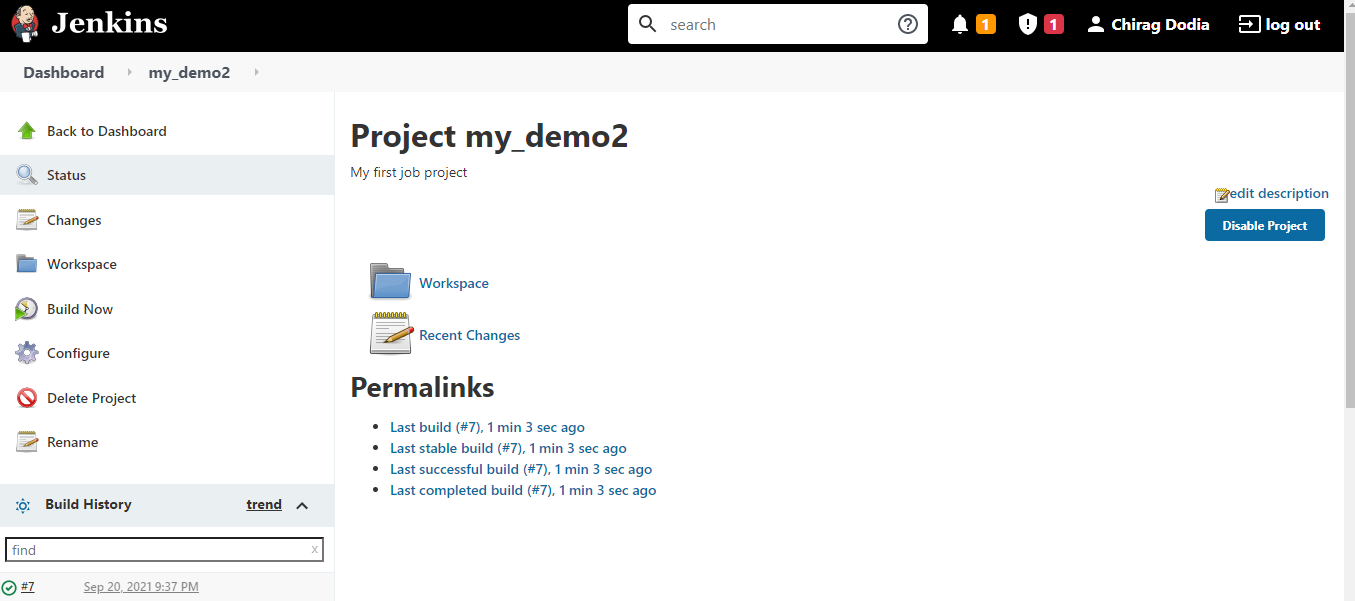
1. Click on “Add build step”
2. Click on “Execute Windows batch command” and add the commands you want to execute during the build process.

**Step 5:** When you have entered all the data, Save the project.

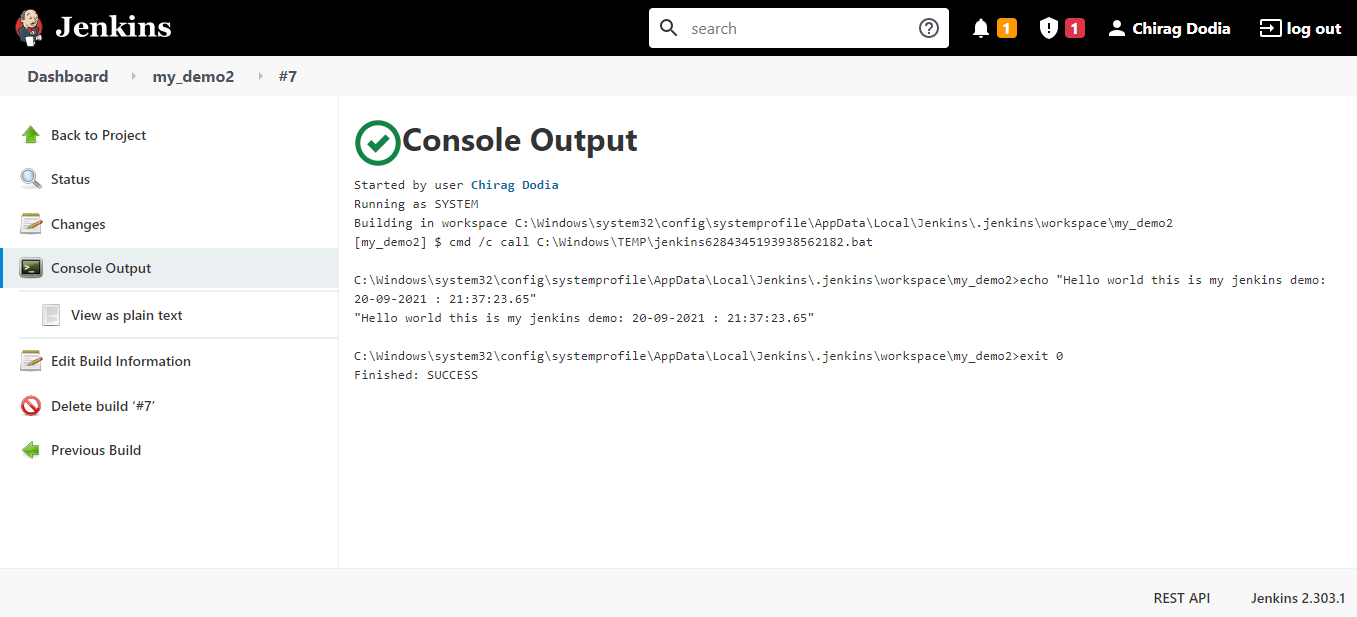


**Step 6:** Now, in the main screen, Click the Build Now button on the left-hand side to build the source code.

**Step 7:** After clicking on Build now, you can see the status of the build you run under Build History.



**Step 8:** 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 success message, provided you have followed the setup properly as shown in the below Jenkins create new job example.



**Conclusion:** In this experiment, we understood the Continuous Integration, installed and configured Jenkins with Maven/Ant/Gradle to setup a build Job.