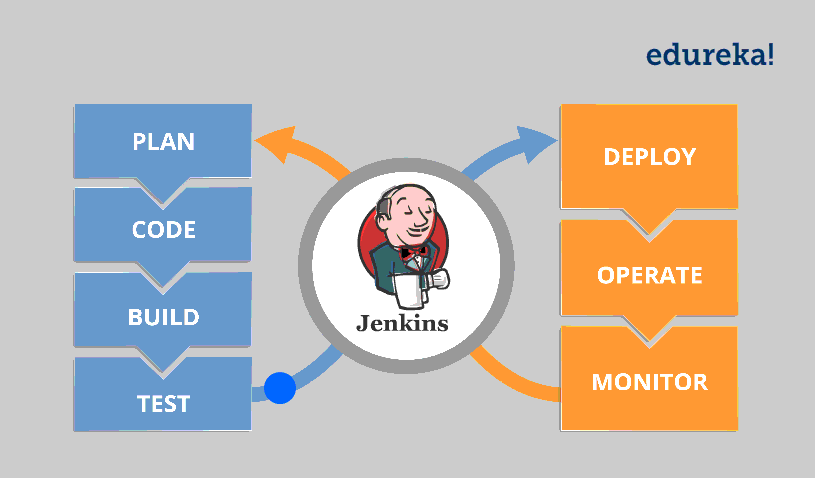
Jenkins Tutorial | Continuous Integration Using Jenkins | Edureka

[Saurabh](https://www.edureka.co/blog/author/saurabhedureka-co/) Nov 7, 2016 2,202 [**2**](https://www.edureka.co/blog/jenkins-tutorial/#disqus_thread)

**Jenkins Tutorial**

I hope you have read my previous blog on [***What is Jenkins***](http://www.edureka.co/blog/what-is-jenkins/?utm_source=blog&utm_medium=content-link&utm_campaign=jenkins-tutorial). In this Jenkins Tutorial blog, I will focus on Jenkins architecture and Jenkins build pipeline along with that I will show you how to create a build in Jenkins.

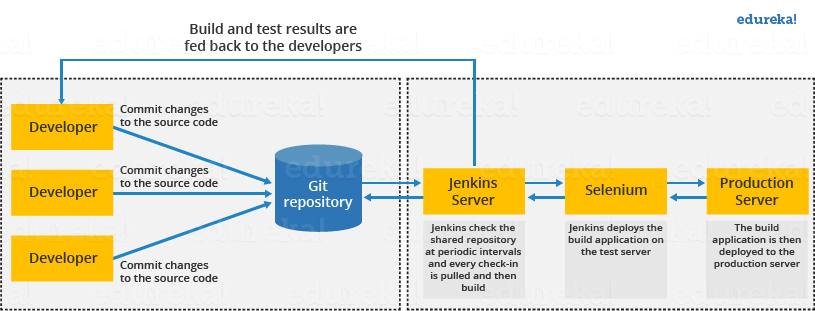
Before we proceed with Jenkins Tutorial, the key takeaways from the previous blog are:

* Jenkins is used to integrate all DevOps stages with the help of plugins.
* Commonly used Jenkins plugins are Git, Amazon EC2, Maven 2 project, HTML publisher etc.
* Jenkins has well over 1000 plugins and 147,000 active installations along with over 1 million users around the world.
* With Continuous Integration every change made in the source code is built. It performs other functions as well, that depends on the tool used for Continuous Integration.
* Nokia shifted from Nightly build to Continuous Integration.
* Process before Continuous Integration had many flaws. As a result, not only the software delivery was slow but the quality of software was also not up to the mark. Developers also had a tough time in locating and fixing bugs.
* Continuous Integration with Jenkins overcame these shortcomings by continuously triggering a build and test for every change made in the source code.

Now is the correct time to understand Jenkins architecture.

**Jenkins Architecture**

Let us revise the standalone Jenkins architecture that I have explained to you in the [***previous blog***](http://www.edureka.co/blog/what-is-jenkins/?utm_source=blog&utm_medium=content-link&utm_campaign=jenkins-tutorial), below diagram depicts the same.



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 was introduced.

**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:

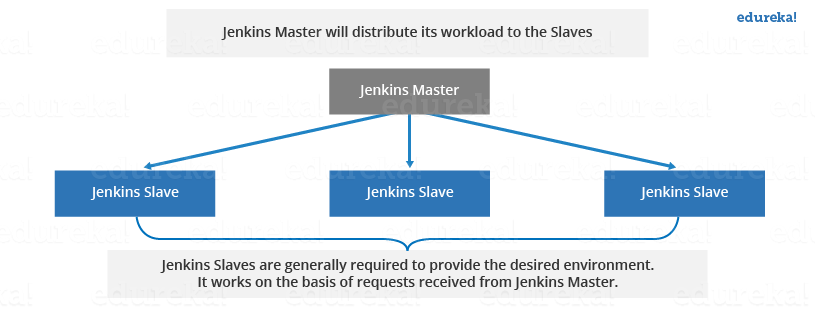
* Scheduling build jobs.
* Dispatching builds to the slaves for the actual execution.
* Monitor the slaves (possibly taking them online and offline as required).
* Recording and presenting the build results.
* A Master instance of Jenkins can also execute build jobs directly.

**Jenkins Slave**

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

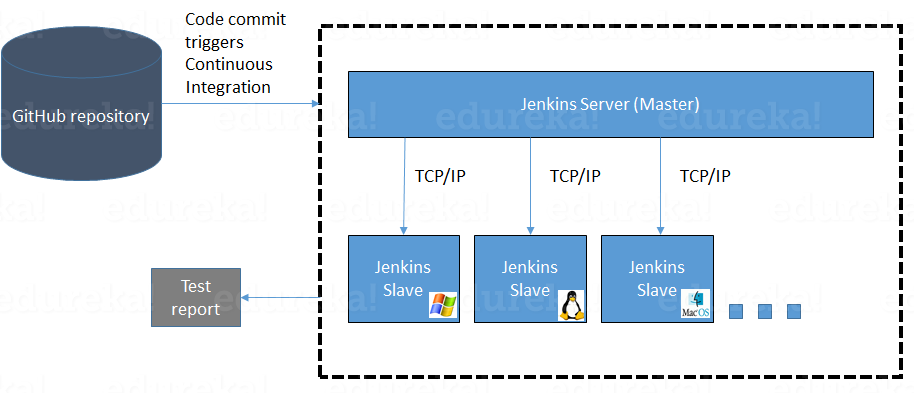
* It hears requests from the Jenkins Master instance.
* Slaves can run on a variety of operating systems.
* The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.
* You can configure a project to always run on a particular Slave machine, or a particular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



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

The diagram below represents the same:



The following functions are performed in the above image:

* 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.

[Learn Jenkins in our DevOps Course](https://goo.gl/raowqw?utm_source=blog&utm_medium=button&utm_campaign=jenkins-tutorial-learn-jenkins-in-our-devops-course)

**Jenkins Build Pipeline**

It is used to know which task Jenkins is currently executing. Often several different changes are made by several developers at once, so it is useful to know which change is getting tested or which change is sitting in the queue or which build is broken. This is where pipeline comes into picture. The Jenkins Pipeline gives you an overview of where tests are up to. In build pipeline the build as a whole is broken down into sections, such as the unit test, acceptance test, packaging, reporting and deployment phases. The pipeline phases can be executed in series or parallel, and if one phase is successful, it automatically moves on to the next phase (hence the relevance of the name “pipeline”).The below image shows how a multiple build Pipeline looks like.



Hope you have understood the theoretical concepts. Now, let’s have some fun with hands-on.

I will create a new job in Jenkins, it is a **Freestyle Project**. However, there are 3 more options available. Let us look at the types of build jobs available in Jenkins.

**Freestyle Project:**

Freestyle build jobs are general-purpose build jobs, which provides maximum flexibility. The freestyle build job is the most flexible and configurable option, and can be used for any type of project. It is relatively straightforward to set up, and many of the options we configure here also appear in other build jobs.

**Multiconfiguration Job:**

The “multiconfiguration project” (also referred to as a “matrix project”) allows you run the same build job on different environments. It is used for testing an application in different environments, with different databases, or even on different build machines.

**Monitor an External Job:**

The “Monitor an external job” build job lets you keep an eye on non-interactive processes, such as cron jobs.

**Maven Project:**

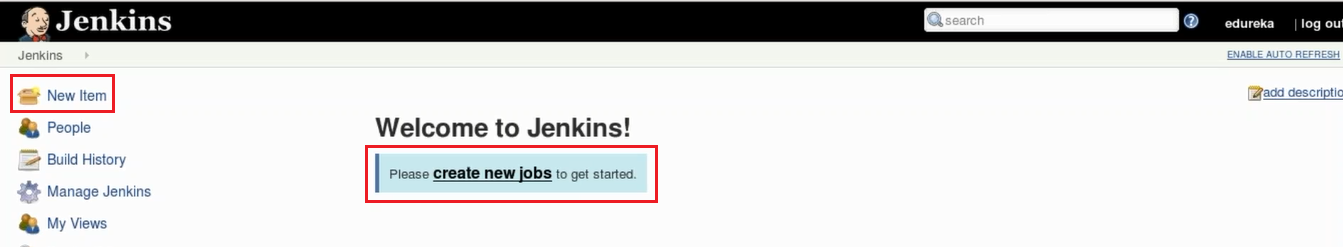
The “maven2/3 project” is a build job specially adapted to Maven projects. Jenkins understands Maven pom files and project structures, and can use the information gleaned from the pom file to reduce the work you need to do to set up your project.

Here is a video on Jenkins tutorial for better understanding of Jenkins. Check out this Jenkins tutorial video.

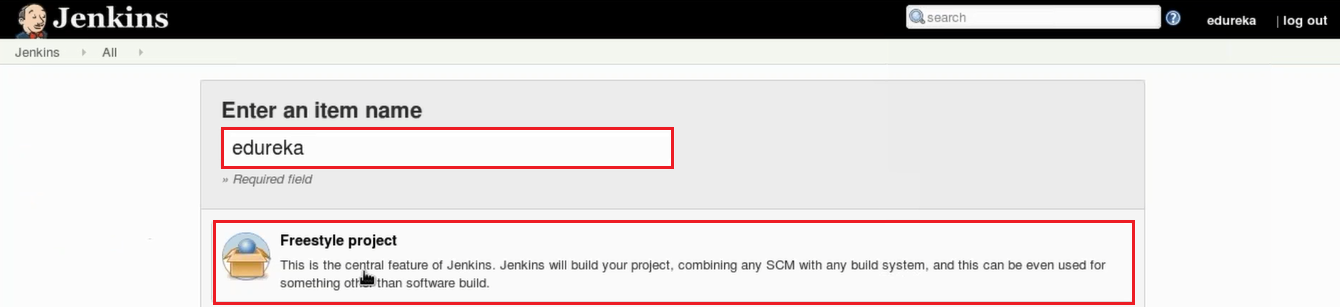
**Getting Started With Jenkins | Jenkins and DevOps tutorial | Jenkins for Beginners | Edureka**

**Creating a Build Using Jenkins**

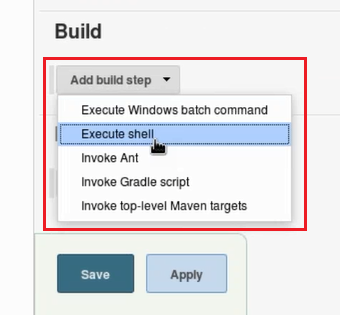
**Step 1:** From the Jenkins interface home, select **New Item.**

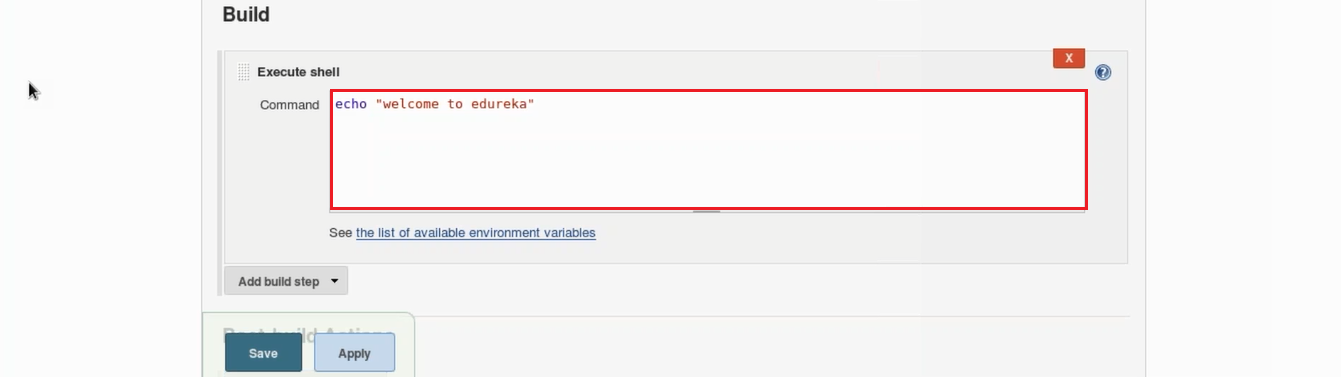


**Step 2:** Enter a name and select **Freestyle project**.

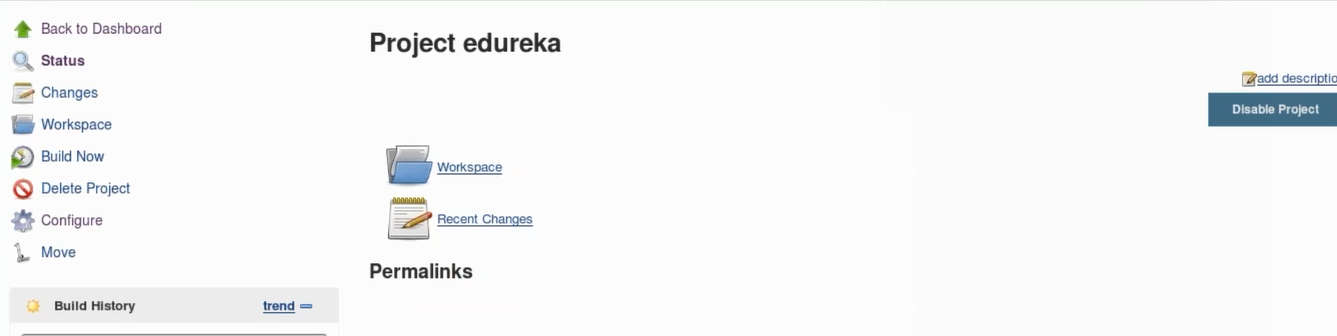


**Step 3:** This next page is where you specify the job configuration. As you’ll quickly observe, there are a number of settings available when you create a new project. On this configuration page, you also have the option to **Add build step** to perform extra actions like running scripts. I will execute a shell script.

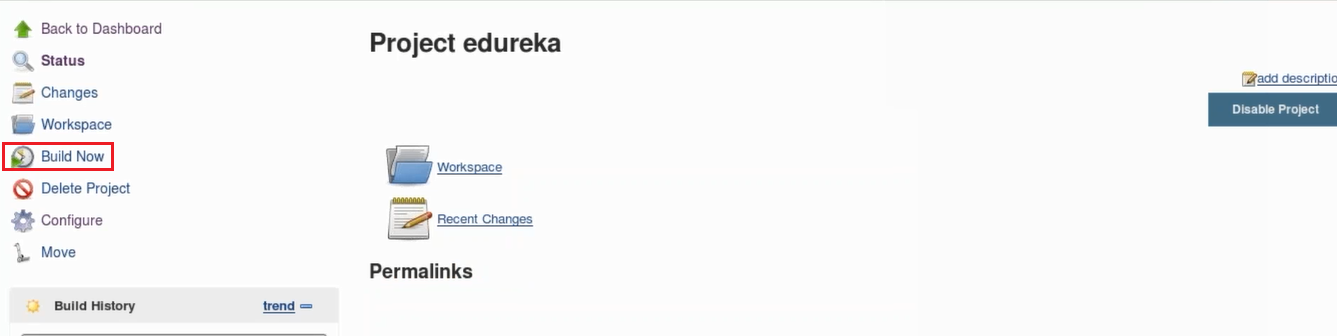
This will provide you with a text box in which you can add whatever commands you need. You can use scripts to run various tasks like server maintenance, version control, reading system settings, etc. I will use this section to run a simple script.



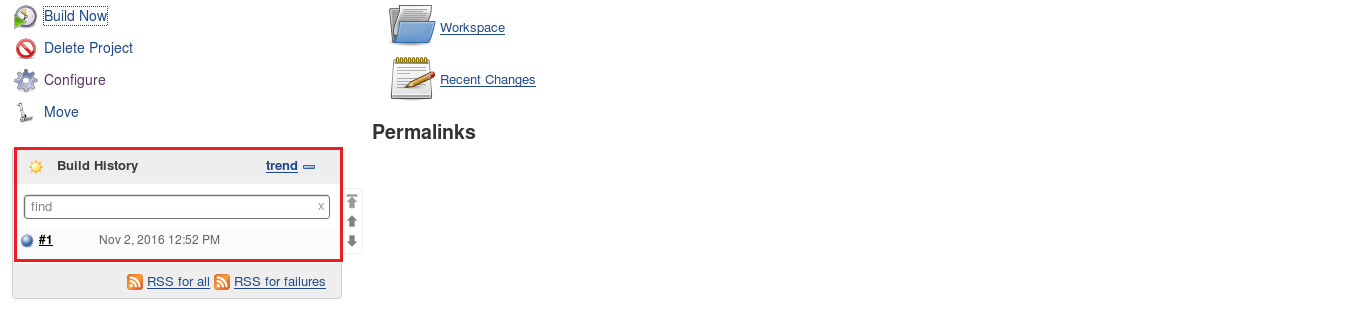
**Step 4:** Save the project, and you’ll be taken to a project overview page. Here you can see information about the project, including its built history.



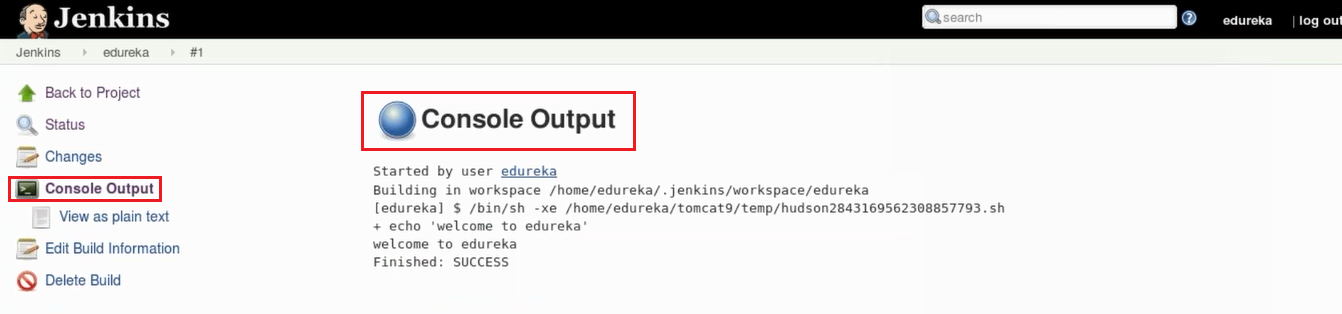
**Step 5:** Click **Build Now** on the left-hand side to start the build.



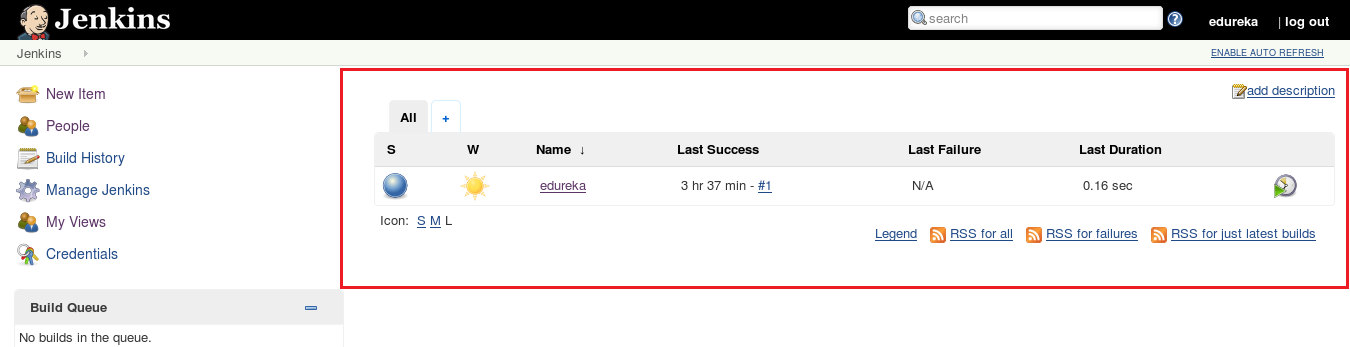
**Step 6:** To see more information, click on that build in the build history area, whereupon you’ll be taken to a page with an overview of the build information.



**Step 7:** The **Console Output** link on this page is especially useful for examining the results of the job in detail.



**Step 8:** If you go back to Jenkins home, you’ll see an overview of all projects and their information, including status.



Status of the build is indicated in two ways, by a weather icon and by a colored ball. The weather icon is particularly helpful as it shows you a record of multiple builds in one image.

As you can see in the above image, the sun represents that all of my builds were successful. The color of the ball gives us the status of that particular build, in the above image the color of the ball is blue which means that this particular build was successful.

In this Jenkins Tutorial, I have just given an introductory example. In my next blog, I will show you how to pull and build code from the GitHub repository using Jenkins.

[Learn DevOps From Experts](https://goo.gl/AEO5nM?utm_source=blog&utm_medium=button&utm_campaign=jenkins-tutorial-learn-devops-from-experts)

*If you found this* ***Jenkins Tutorial****relevant, check out the* [***DevOps training***](http://www.edureka.co/devops/?utm_source=blog&utm_medium=content-link&utm_campaign=jenkins-tutorial) *by Edureka, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe. The Edureka DevOps Certification Training course helps learners gain expertise in various DevOps processes and tools such as Puppet, Jenkins, Nagios and GIT for automating multiple steps in SDLC.*

*Got a question for us? Please mention it in the comments section and we will get back to you.*