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.

What is Jenkins?

Jenkins is an open source automation tool written in Java programming language that allows continuous integration.

10 Sec

Hello Java Program for Beginners

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.

Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.

With the help of Jenkins, organizations can speed up the software development process through automation. Jenkins adds development life-cycle processes of all kinds, including build, document, test, package, stage, deploy static analysis and much more.

Jenkins achieves CI (Continuous Integration) with the help of plugins. Plugins is used to allow the integration of various DevOps stages. If you want to integrate a particular tool, you have to install the plugins for that tool. For example: Maven 2 Project, Git, HTML Publisher, Amazon EC2, etc.

**For example:** If any organization is developing a project, then **Jenkins** will continuously test your project builds and show you the errors in early stages of your development.

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

**Work Flow:**



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?

Continuous Integration *(CI)* is a development practice in which the developers are needs 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.

Continuous Integration with Jenkins

Let's consider a scenario where the complete source code of the application was built and then deployed on test server for testing. It sounds like a perfect way to *develop software*, but this process has many problems.

* Developer teams have to wait till the complete software is developed for the test results.
* There is a high prospect that the test results might show multiple bugs. It was tough for developers to locate those bugs because they have to check the entire source code of the application.
* It slows the software delivery process.
* Continuous feedback pertaining to things like architectural or coding issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.
* The whole process was manual which increases the threat of frequent failure.

It is obvious from the above stated problems that not only the software delivery process became slow but the quality of software also went down. This leads to customer dissatisfaction.

So to overcome such problem there was a need for a system to exist where developers can continuously trigger a build and test for every change made in the source code.

This is what Continuous Integration (CI) is all about. Jenkins is the most mature Continuous Integration tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.

Let's see a generic flow diagram of Continuous Integration with Jenkins:



**Let's see how Jenkins works**. The above diagram is representing the following functions:

* 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.

Advantages and Disadvantages of using Jenkins

**Advantages 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.

**Disadvantages of Jenkins**

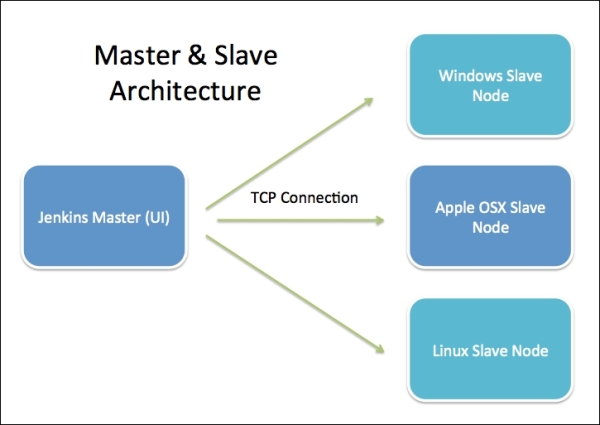
* Its interface is out dated and not user friendly compared to current user interface trends.
* Not easy to maintain it because it runs on a server and requires some skills as server administrator to monitor its activity.
* CI regularly breaks due to some small setting changes. CI will be paused and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

Jenkins Architecture

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

Jenkins architecture has two components:

* Jenkins Master/Server
* Jenkins Slave/Node/Build Server



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. We can add more nodes using IP address, user name and password using the ssh, jnlp or webstart methods.

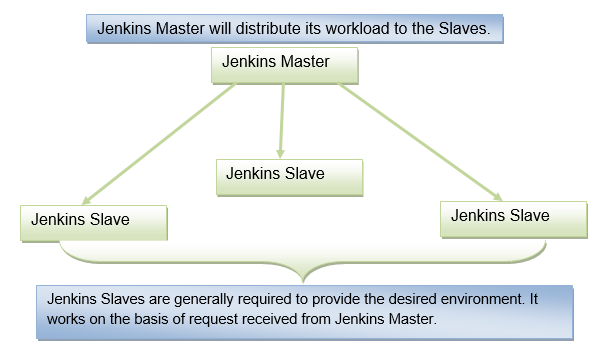
The server's job or master's job is to handle:

* 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.

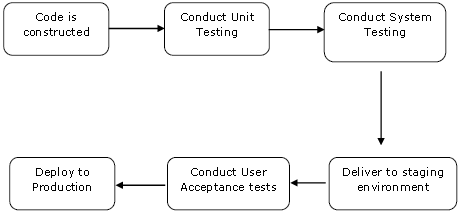
As we know Jenkins is developed using Java is platform independent thus Jenkins Master/Servers and Slave/nodes can be configured in any servers including Linux, Windows, and Mac.



The above diagram is self explanatory. It consists of a Jenkins Master which is managing three Jenkins Slaves.

Jenkins - Continuous Deployment

Jenkins is used in providing good support for continuous deployment and delivery. The flow of a software development till the deployment is shown below:



The main part of continuous deployment to make sure that the above entire process is automated. Jenkins provides various plugins for all these things. One of them is "**Deploy to container**" plugin, which was seen in earlier sections.

Jenkins provides various plugins which are used to give a graphical representation of the continuous deployment process.

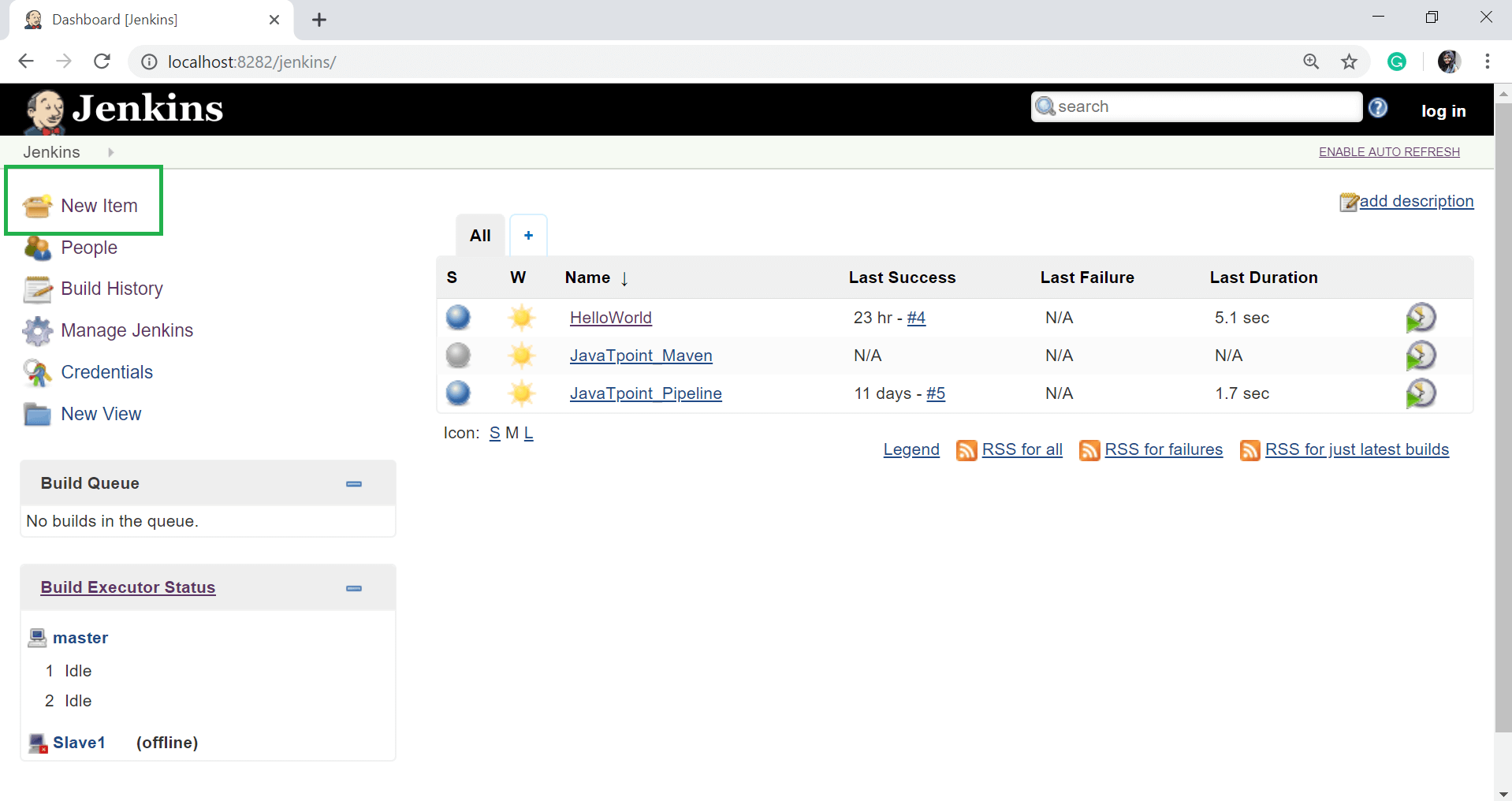
To understand that, let's first create another project in Jenkins so that we can see how it works and which emulates the QA stage:

16.7M

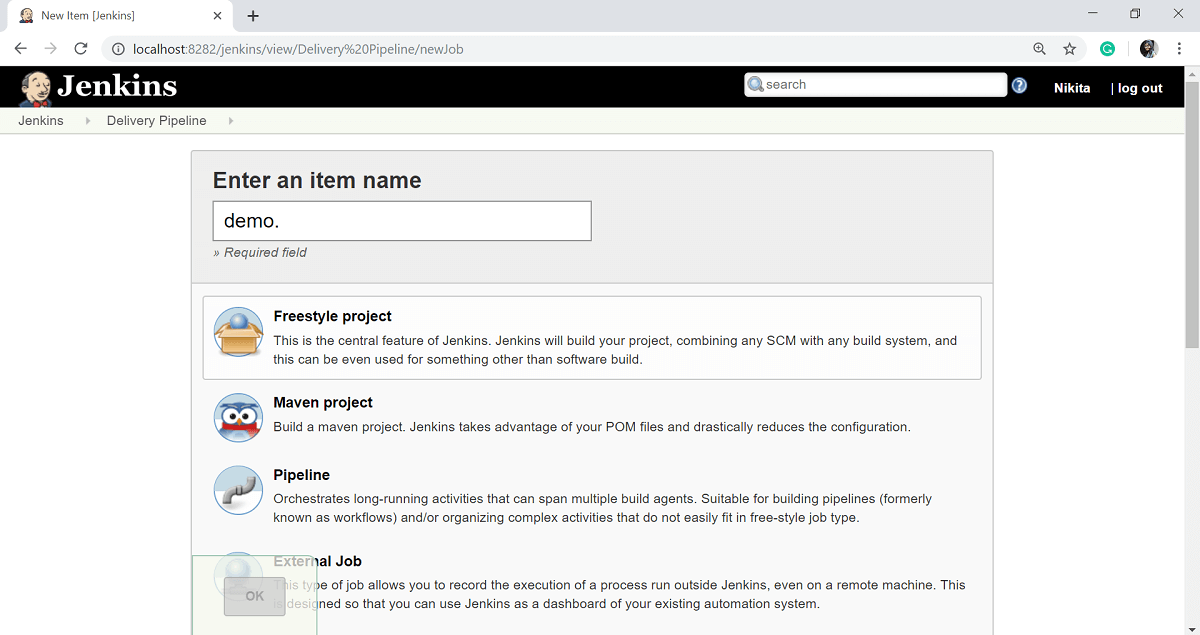
367

C++ vs Java

**Step 1:** Go to the Jenkins Dashboard and select **New Item**.

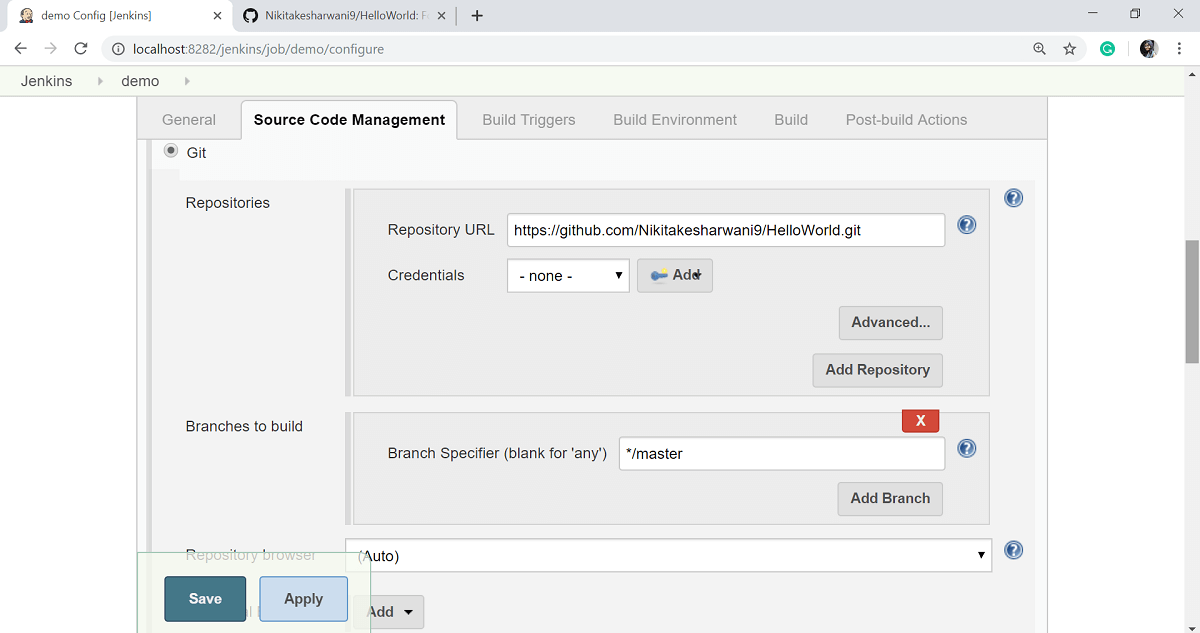


**Step 2:** Give the Item name and choose **Freestyle project** option. Here I have given the item name "**demo**". Click on OK button.

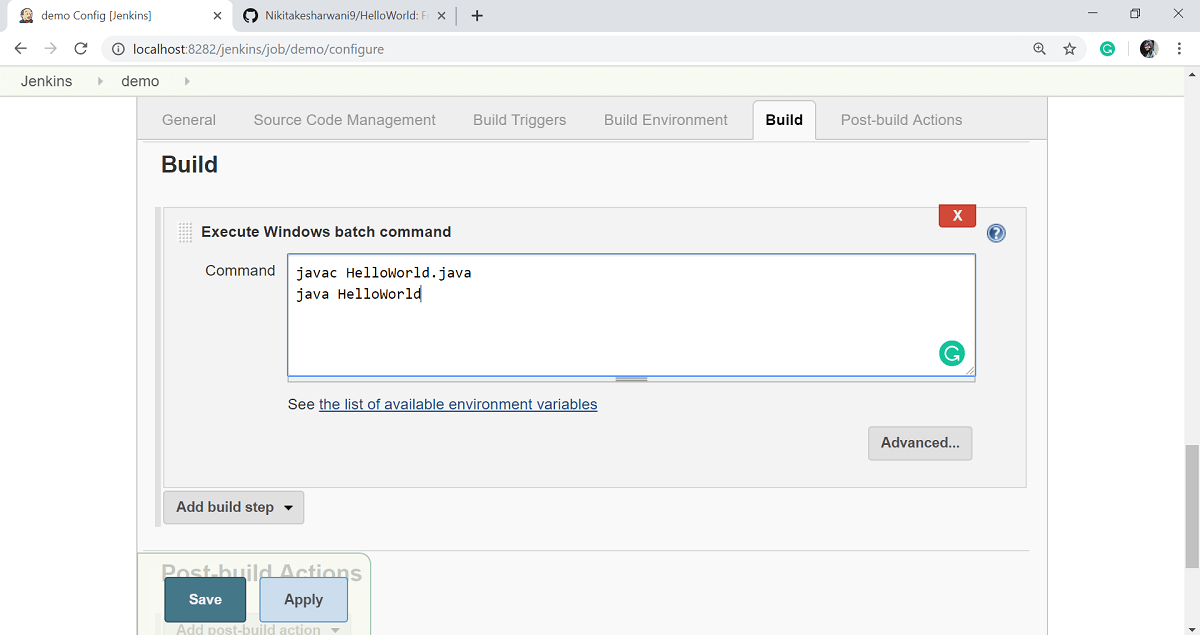


**Step 3:** In this example, we are keeping it simple and just using to print HelloWorld.

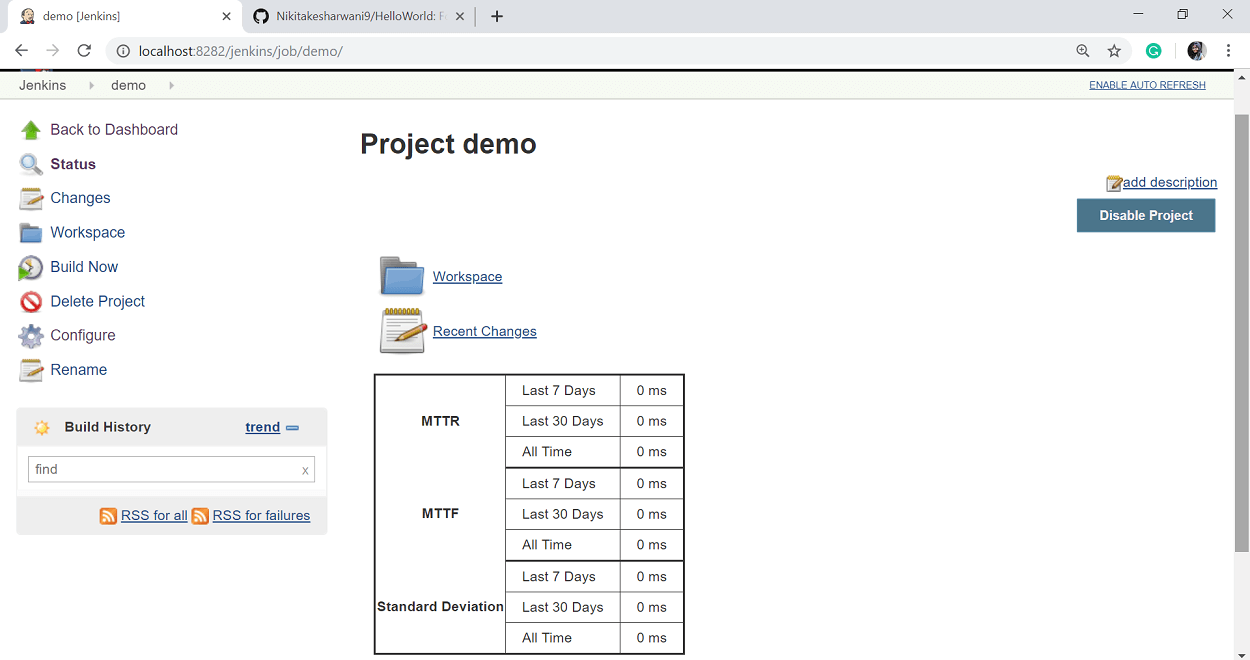
Select the Git option and enter the GitHub repository of your **HelloWorld** program in the **Repository URL** section.



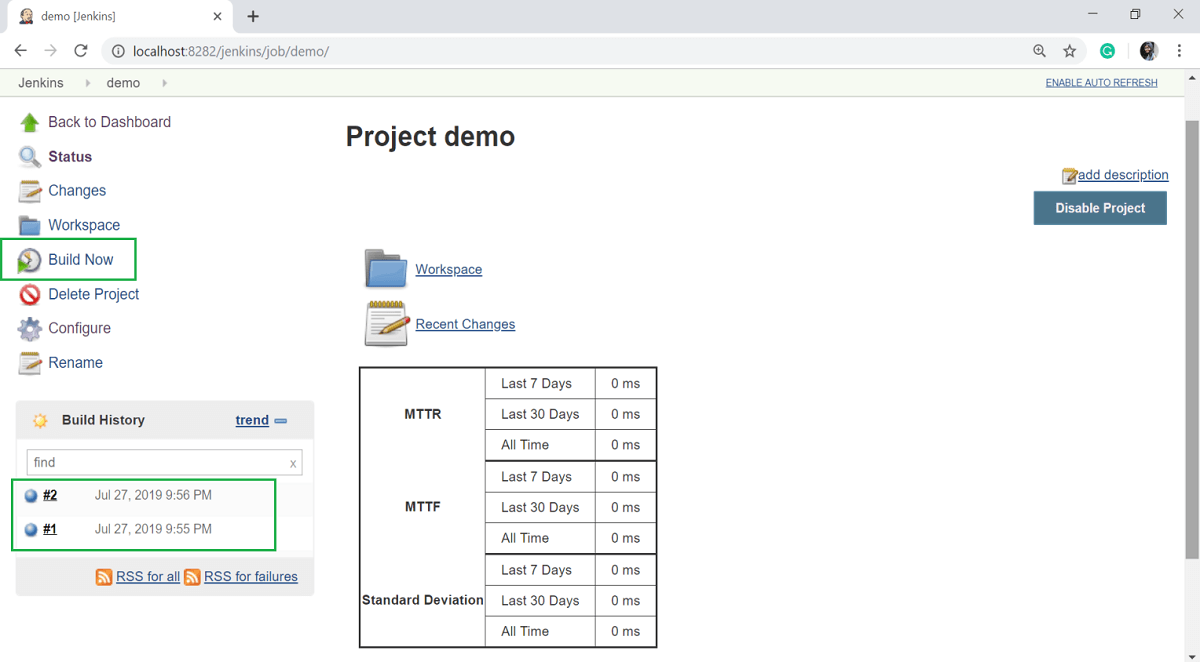
**Step 4:** Select the **Execute Windows batch command** option from the **add build step** button and give the command to run your java program.



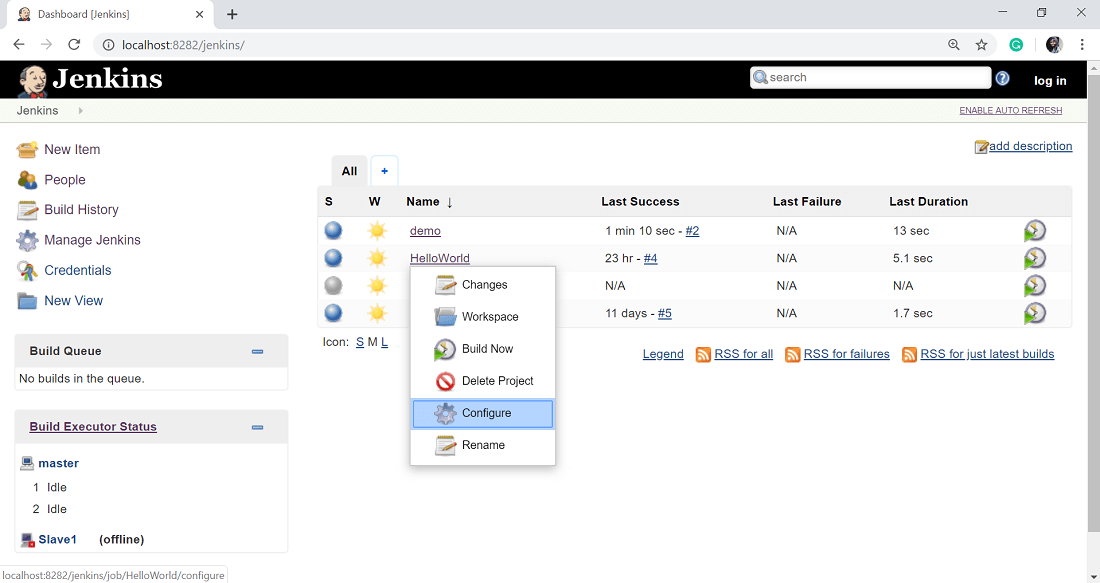
**Step 5:** Click on **Apply** then **Save** button.



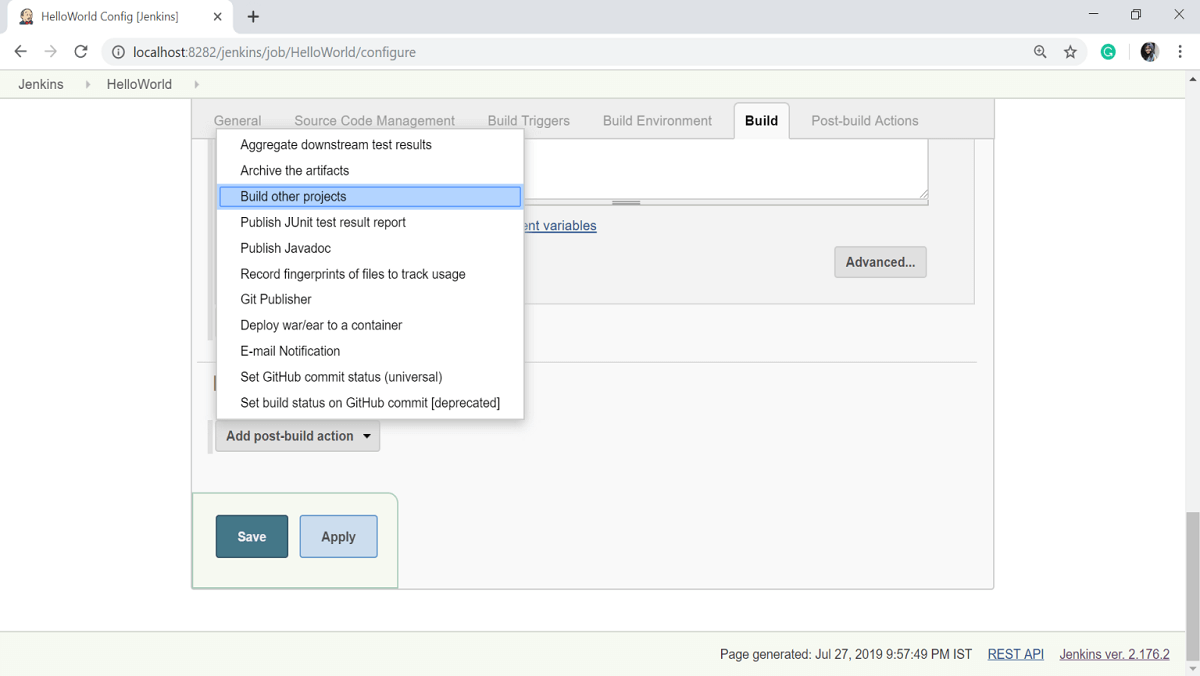
So our project demo is now created. You can check a build to see if the build is successfully created or not. To check a build, click on the **Build Now** option.



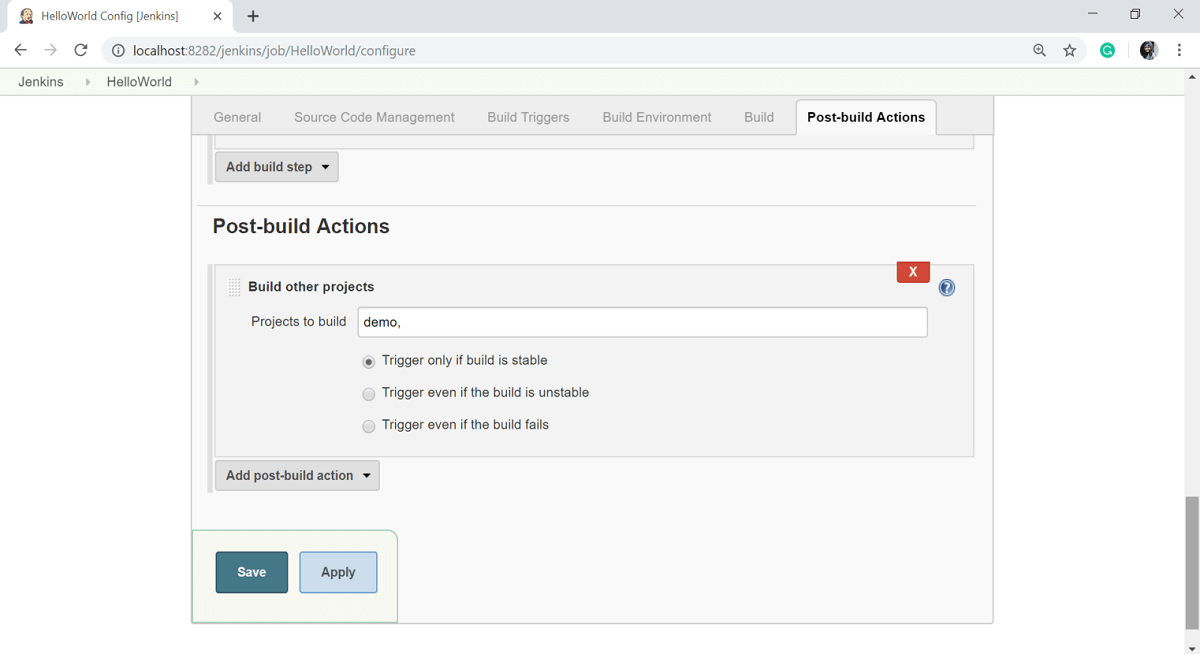
**Step 6:** Now, go to your previously created **Helloworld** project and click on the **Configure** option.



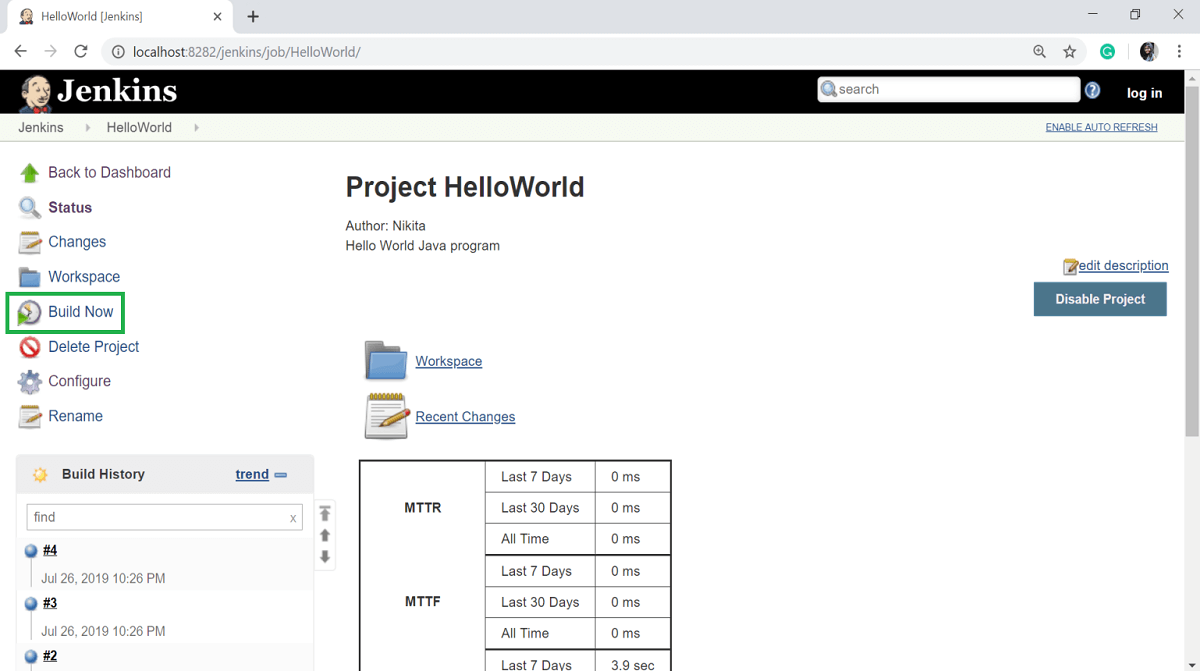
**Step 7:** In the Project configuration, select the **Add post-build** action and choose **Build other projects** option.



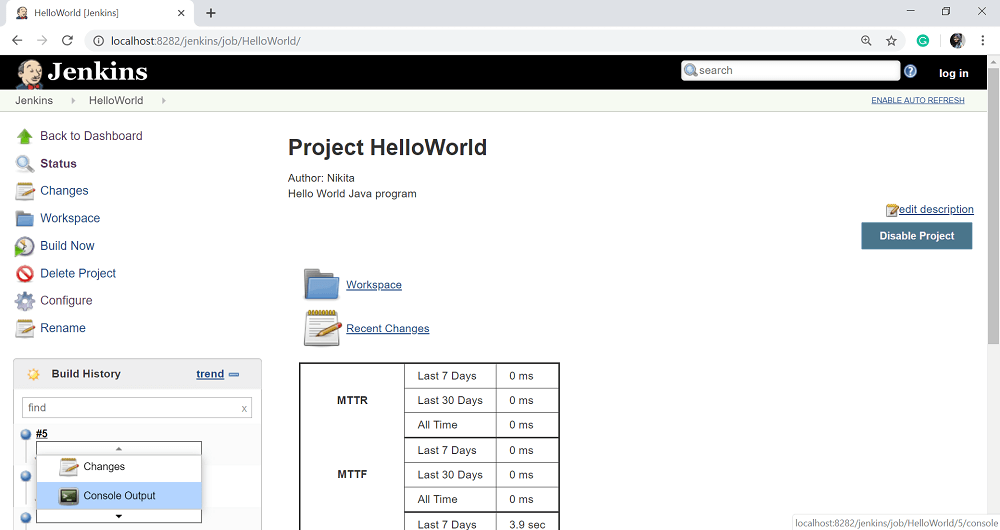
**Step 8:** In the **Projects to build** option, enter the "demo" as the project name to build. You can leave the other option as the default. Click on **Apply** then the **Save** button.



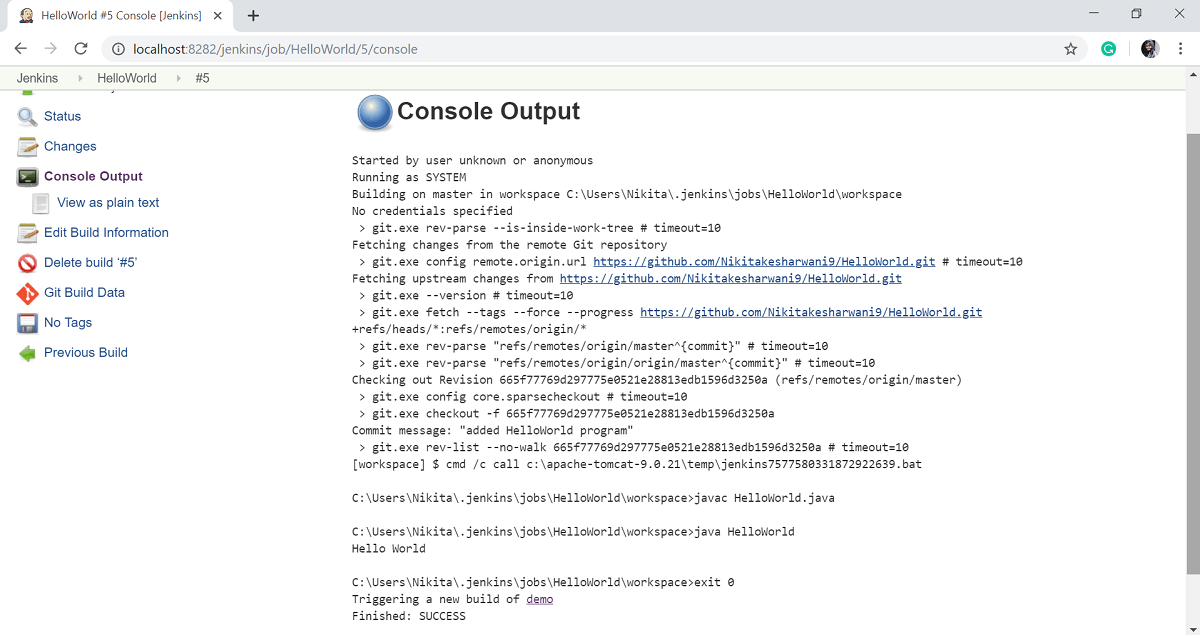
**Step 9:** Now, build the HelloWorld project. To do that, click on the **Build Now** option.



**Step 10:** Click on the latest build and select the **Console Output** option.

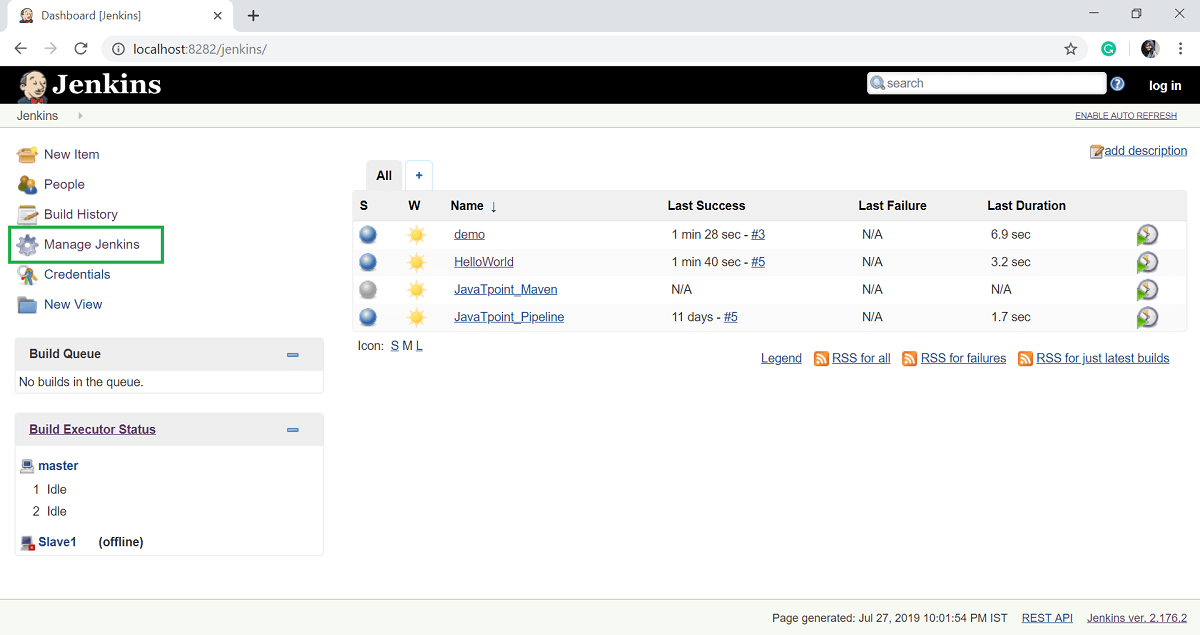


Now, if you see the Console output, you will also see that after the HelloWorld project is successfully built, the build of the demo project will also happen.

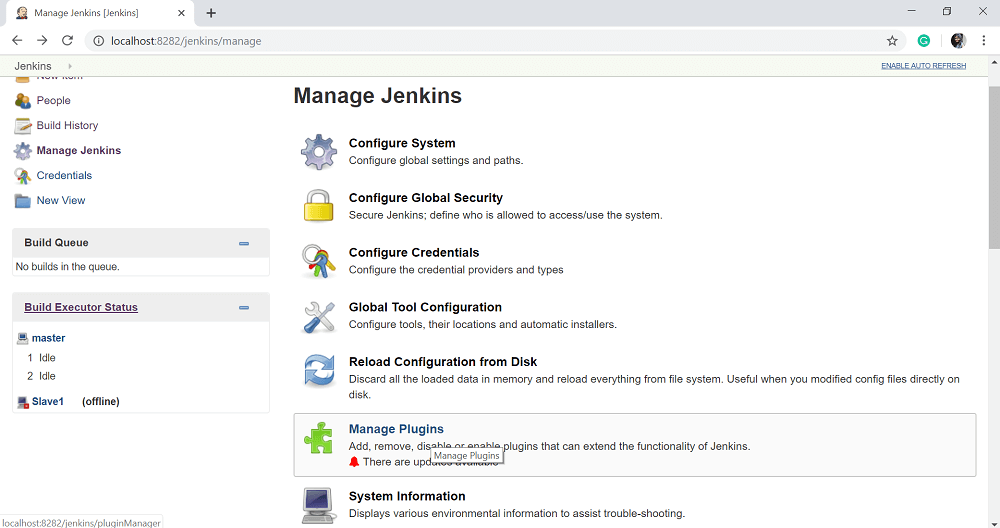


Delivery Pipeline Plugin

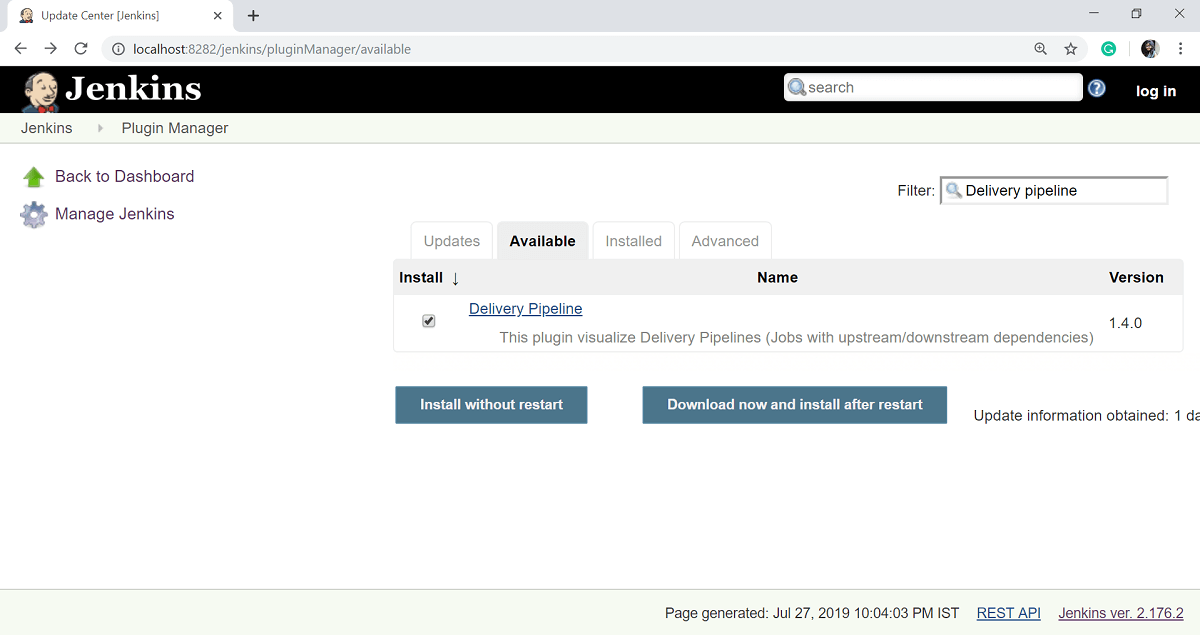
**Step 11:** Let's now install the **Delivery pipeline plugin**. Select **Manage Jenkins** option.



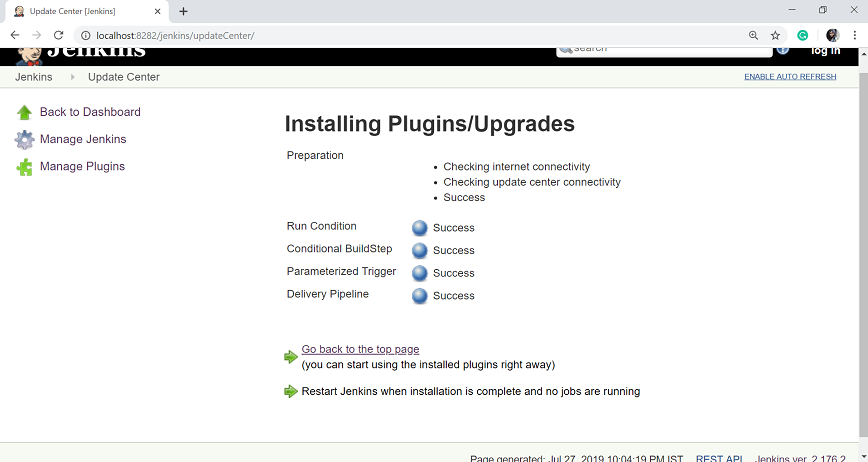
**Step 12:** Select the **Manage Plugins** option.



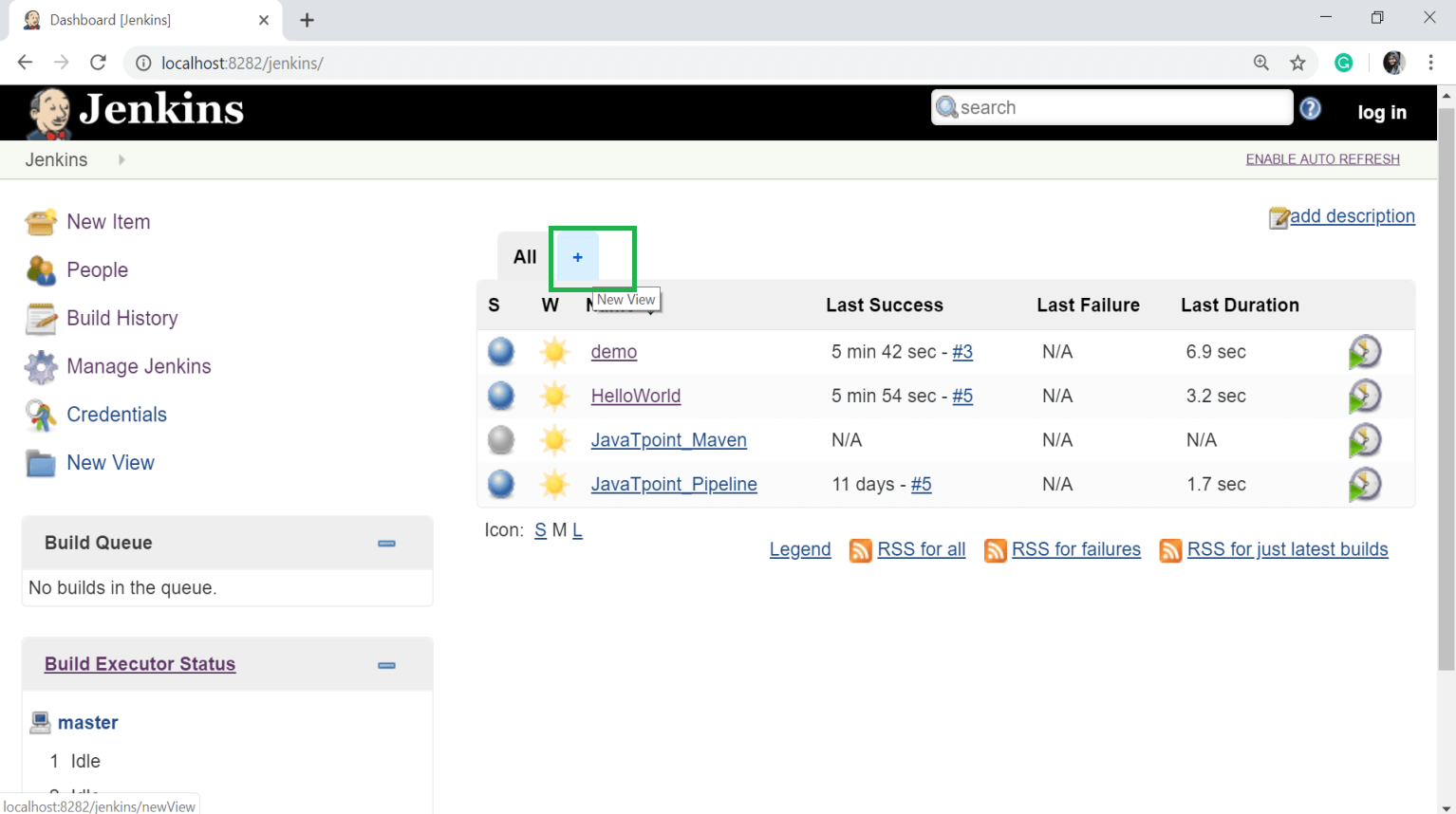
**Step 13:** In the **Available** tab, search for "**Delivery Pipeline**" plugin in the filter option. Select **Delivery Pipeline** plugin and click on **Install without Restart**.



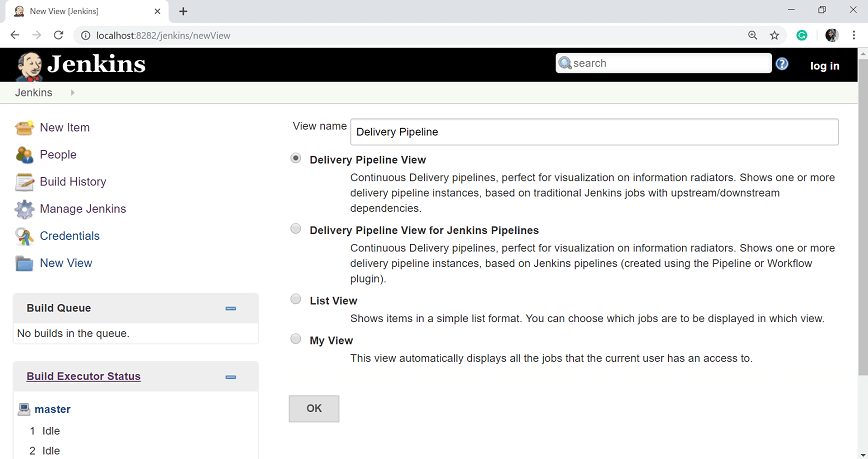
After successful installation of plugins, click on **Go back to the top page** link.



**Step 14:** To see the Delivery Pipeline in action, click on + symbol in the tab next to the **All** tab on the Jenkins Dashboard screen.

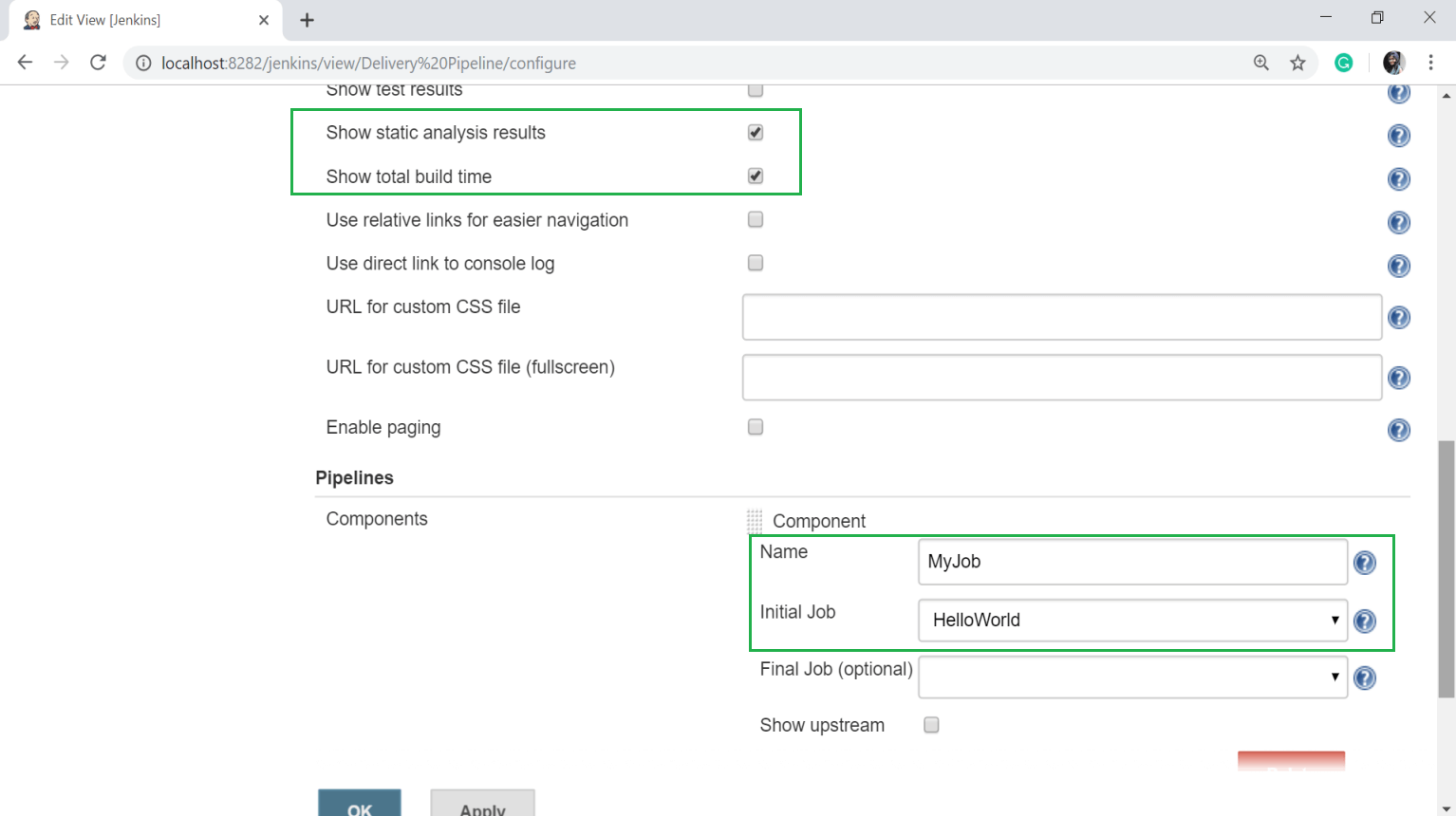


**Step 15:** Give the **View name** and select **Delivery Pipeline View**. Click on OK button.

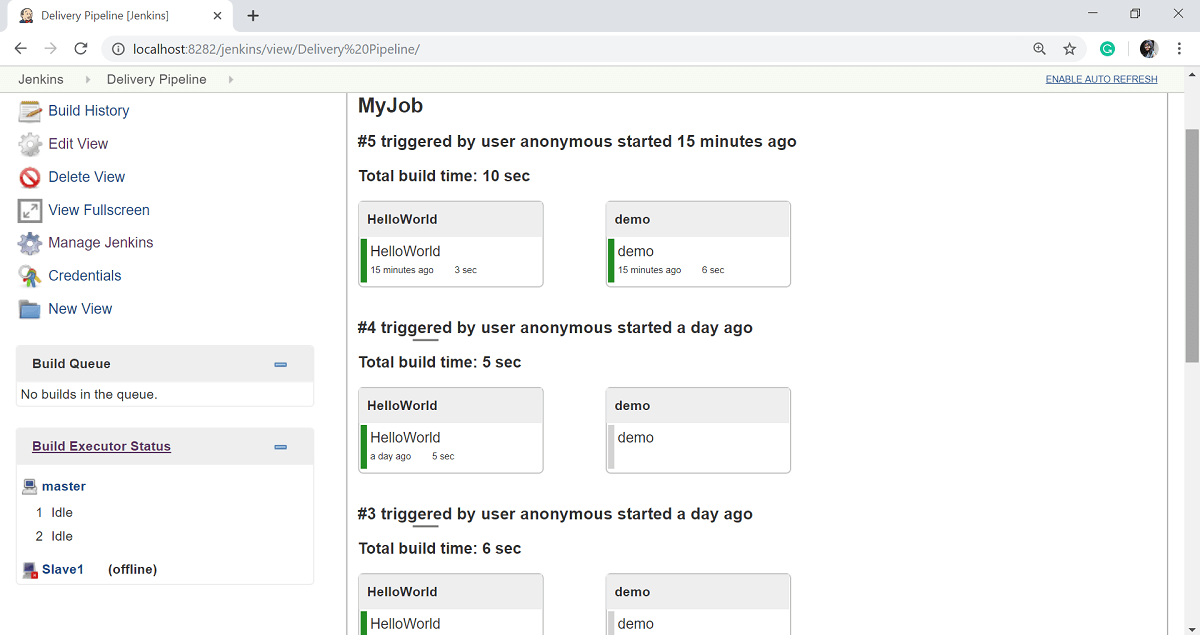


**Step 16:** In the next page, leave the default options. Scroll down and change the following settings:

* Make sure "Show static analysis results" option is checked.
* Make sure the option "Show total build time" is checked.
* In the Pipelines section for the Initial job enter the Helloworld project as the first job which should build.
* Give any name for the Pipeline
* Click the Apply and OK button.



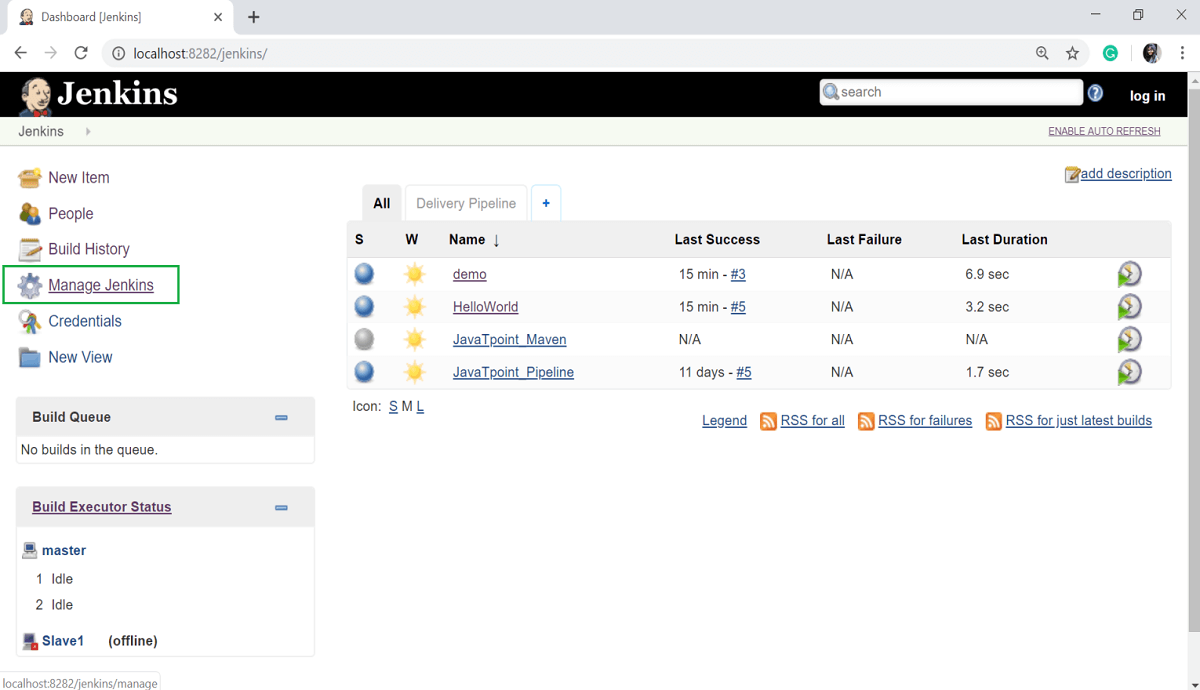
Now you will see a view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.



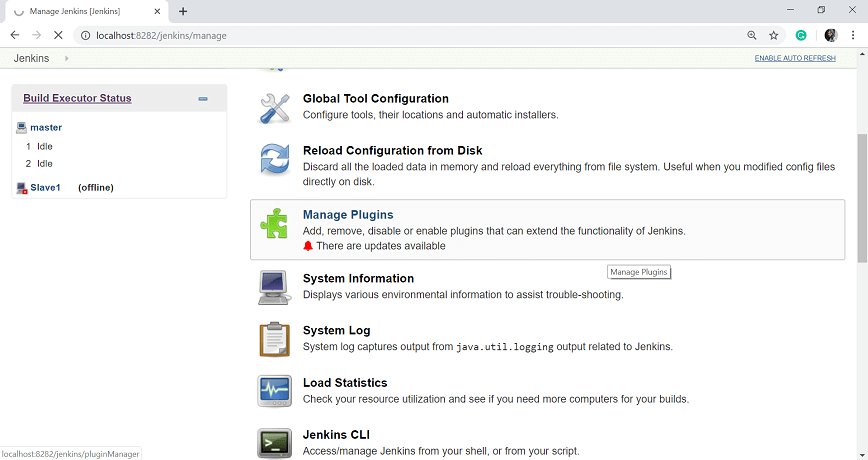
Build Pipeline Plugin

Another important plugin of Jenkins is the "**Build pipeline**" plugin. Let's take a look at this plugin:

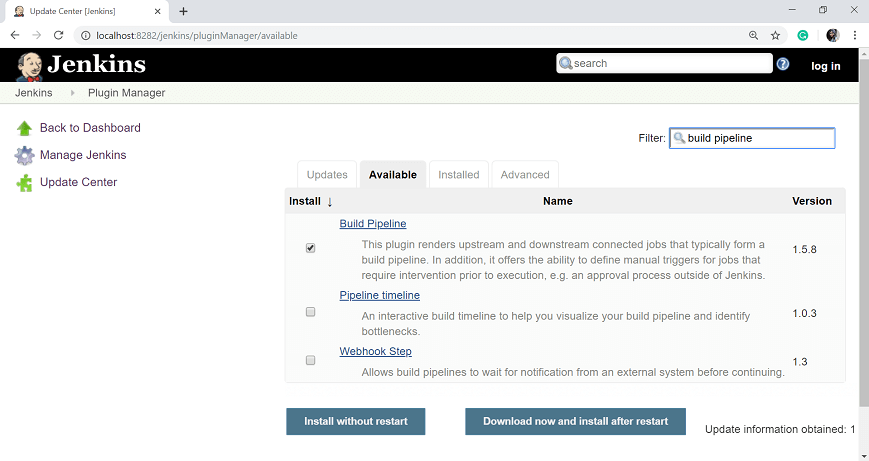
**Step 1:** On the Jenkins Dashboard, select **Manage Jenkins**.



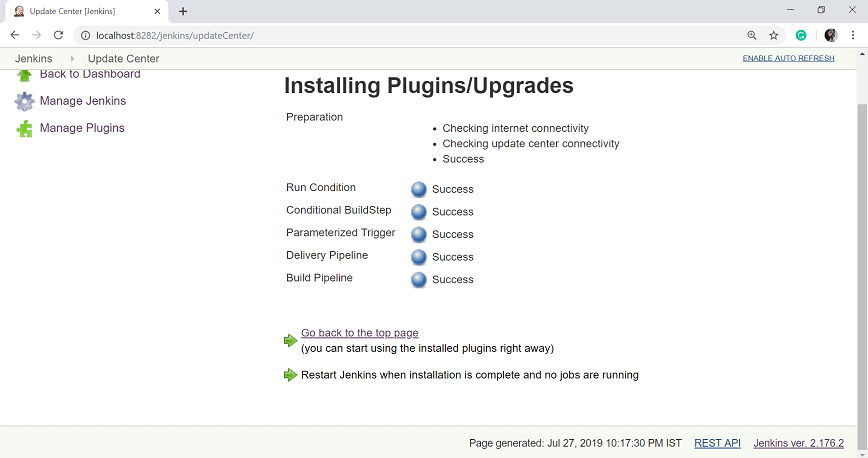
**Step 2:** Select the **Manage Plugins** option.



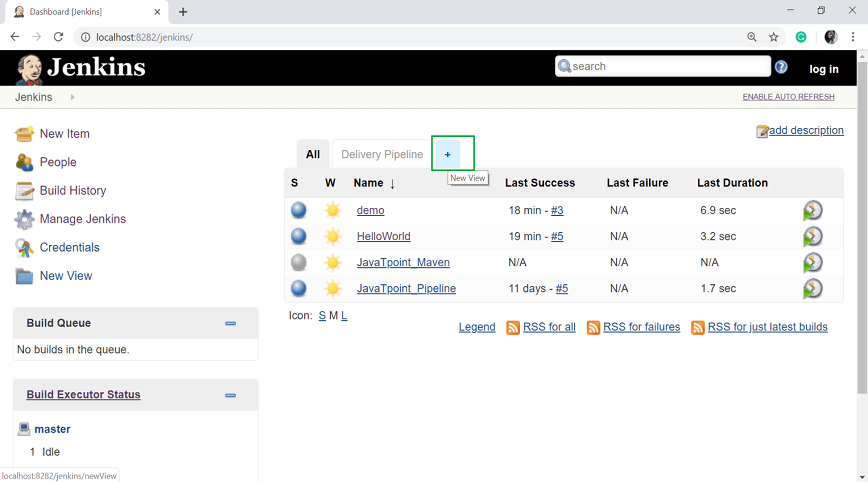
**Step 3:** In the Available tab, filter for Build pipeline and select **Build pipeline** plugin, then click on **Install without Restart** button.



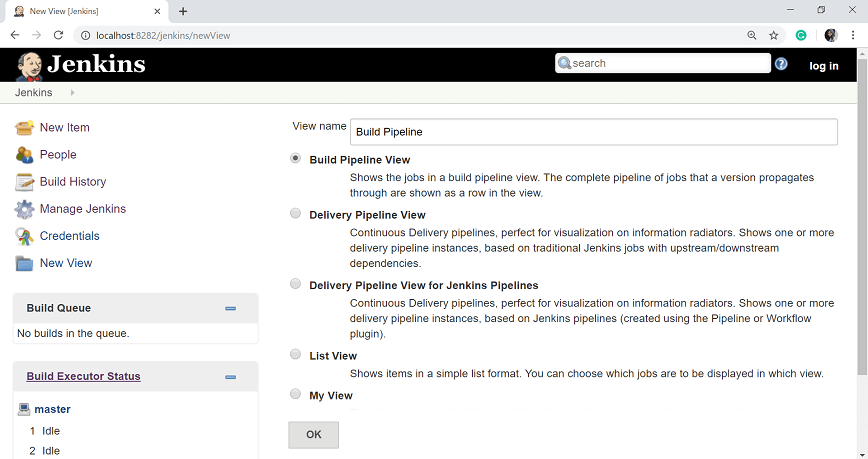
**Step 4:** Once the installation is completed successfully, click on **Go back to the top page** link.



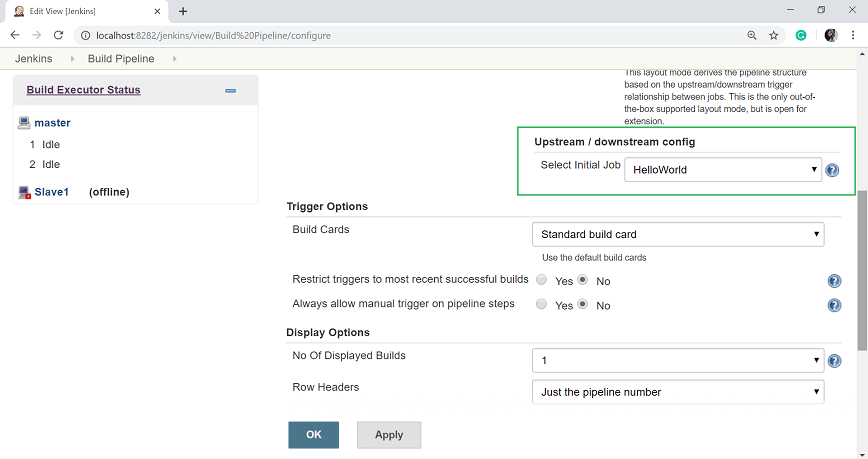
**Step 5:** To see the Build pipeline in action, click on the + symbol in the Tab next to the **All** tab in the Jenkins Dashboard.



**Step 6:** In the View name option, enter any name and choose the Build **Pipeline View** Option.



**Step 7:** Leave all the default option and scroll down. In the **Upstream/downstream config** section enter the name of the HelloWorld project for the select initial job option. Then click on the **OK** button.



Now you can see a view of the entire delivery pipeline, and you will be able to see the status of each project in the whole pipeline.

