**JENKINS**

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on.

It is a free source that can handle any kind of build or continuous integration.

You can integrate Jenkins with a number of testing and deployment technologies.

**How Jenkins works**

Jenkins runs as a server on a variety of platforms including Windows, MacOS, Unix variants and especially, [Linux](https://www.techtarget.com/searchdatacenter/definition/Linux-operating-system).

It requires a Java 8 VM and above and can be run on the Oracle JRE or [OpenJDK](https://www.theserverside.com/definition/OpenJDK).

Usually, Jenkins runs as a Java servlet within a Jetty application server.

It can be run on other Java application servers such as Apache Tomcat.

More recently, Jenkins has been adapted to run in a [Docker container](https://searchitoperations.techtarget.com/definition/Docker). There are read-only Jenkins images available in the Docker Hub online repository.

To operate Jenkins, pipelines are created.

A pipeline is a series of steps the Jenkins server will take to perform the required tasks of the CI/CD process.

These are stored in a plain text Jenkinsfile. The Jenkinsfile uses a curly bracket syntax that looks similar to [JSON](https://theserverside.techtarget.com/definition/JSON-Javascript-Object-Notation).

Steps in the pipeline are declared as commands with parameters and encapsulated in curly brackets.

The Jenkins server then reads the Jenkinsfile and executes its commands, pushing the code down the pipeline from committed source code to production runtime.

A Jenkinsfile can be created through a GUI or by writing code directly.

**JENKINS INSTALLATION**

1. Download the latest Jenkins package for Windows (currently it is version 2.).

2. Once it is downloaded, it will open a wizard. Click “Next” to start the installation.

3. Click the “Change…” button if you want to install Jenkins in another folder. In this example, I will keep the default option and click on the “Next” button.

4. Enter the service logon credentials. It is recommended to select the second option *“Run service as local or domain user”* as it is safer. To run Jenkins service using this option, you have to enter the domain username and password. Click on the “Test Credentials” button to test your domain credentials and it will enable the “Next” button. Click on the “Next” button.

5. Enter the port on which Jenkins will be running. Click on the “Test Port” button to validate if it is free

6. Select the Java home directory and click on the “Next” button.

7. Select other services that need to be installed with Jenkins and click on the “Next” button

8. Click the “Install” button to start the installation process.

9. The installation will proceed.

10. When done, click on “Finish” to complete the installation process.

11. 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.

12. To unlock Jenkins, copy the password from the file intialAdminPassword. This file should be found under the Jenkins installation path (set at step 3 in Jenkins installation). If a custom path was entered, then you should check that location.

Copy the content of the initialAdminPassword file and paste it in the Administrator password field. Then, click the Continue button.

13. You can install either the suggested plugins or selected plugins you choose. To keep it simple, we will install the suggested plugins.

14. Wait until the plugins are completely installed.

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

16. Click on Save and Finish to complete the Jenkins installation.

17. Now, click on Start using Jenkins to start Jenkins.

**Plugins**

A plugin is an enhancement to the Jenkins system.

They help extend Jenkins capabilities and integrated Jenkins with other software.

Plugins can be downloaded from the online Jenkins Plugin repository and loaded using the Jenkins Web UI or CLI.

Plugins help to integrate other developer tools into the Jenkins environment, add new user interface elements to the Jenkins Web UI, help with administration of Jenkins, and enhance Jenkins for build and source code management.

One of the more common uses of plugins is to provide integration points for CI/CD sources and destinations.

These include software version control systems (SVCs) such as Git and Atlassian BitBucket, container runtime systems -- especially Docker, virtual machine [hypervisors](https://searchservervirtualization.techtarget.com/definition/hypervisor) such as VMware vSphere, public cloud instances including Google Cloud Platform and Amazon AWS, and private cloud systems such as OpenStack.

There are also plugins that assist in communicating with operating systems over FTP, CIFS, and SSH.

A plugin is written in Java.

Plugins use their own set of Java Annotations and design patterns that define how the plugin is instantiated, extension points, the function of the plugin and the UI representation in the Jenkins Web UI.

Plugin development also makes use of Maven deployment to Jenkins.

**Continuous Integration-**

**Continuous Integration** is a software development method where team members integrate their work at least once a day.

In this method, every integration is checked by an automated build to detect errors.

**Continuous Delivery-**

**Continuous Delivery** is a software engineering method in which a team develops software products in a short cycle.

It ensures that software can be easily released at any time.

The main aim of continuous delivery is to build, test, and release software with good speed and frequency.

It helps you to reduce the cost, time, and risk of delivering changes by allowing for frequent updates in production.

## How to Integrate Jenkins with GitHub

**Step 1)**Create a new job in Jenkins, open the Jenkins dashboard with your Jenkins URL. For example, http://localhost:8080/

Click on **create new jobs**

**Step 2)**Enter the item name, select job type and click **OK**. We shall create a Freestyle project as an example.

**Step 3)**once you click **OK,**the page will be redirected to its project form. Here you will need to enter the project information

**Step 4)**you will see a **Git**option under **Source Code Management**if your Git plugin has been installed in Jenkins

NOTE: If the **Git**option does not appear, try re-installing the plugins, followed by a restart and a re-login into your Jenkins dashboard.

You will now be able to see the **Git**option as mentioned above.

**Step 5)**Enter the Git repository URL to pull the code from GitHub.

**Step 6)**you might get an error message the first time you enter the repository URL.

This happens if you do not have Git installed in your local machine.

To install Git in your local machine, go to <https://git-scm.com/downloads>

Download the appropriate Git file for your Operating System, in this case, Windows, and install it onto your local machine running Jenkins.

**Step 7)**you can execute Git repositories in your Jenkins once Git has been installed on your machine.

To check if it has been successfully installed onto your system, open your **command prompt,** type “Git” and press enter. You should see different options come up for Git:

This means that Git has been installed in your system.

Note: If you have GIT already installed in your system, just add git.exe path in Global Tool Configuration.

**Step 8)**Once you have everything in place, try adding the Git URL into Jenkins. You will not see any error messages for Jenkins Git integration

Git is now properly configured on your system.

**Jenkins Artifactory Plug-in**

The popular Jenkins Artifactory Plugin brings [Artifactory's Build Integration](https://www.jfrog.com/confluence/display/JFROG/Build+Integration) support to Jenkins.

This integration allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them.

The plugin includes a vast collection of features, including a rich pipeline API library and [release management for Maven and Gradle builds](https://www.jfrog.com/confluence/display/JFROG/Jenkins+Artifactory+Plugin+-+Release+Management) with Staging and Promotion.

The plugin currently require version **2.159** or above of Jenkins.

## Integration with JFrog Pipelines

### General

[JFrog Pipelines](https://www.jfrog.com/confluence/display/JFROG/JFrog+Pipelines) integration with Jenkins is supported since version 1.6 of JFrog Piplines and version 3.7.0 of the Jenkins Artifactory Plugin.

This integration allows triggering a Jenkins job from JFrog Pipelines.

The Jenkins job is triggered using JFrog Pipeline's native Jenkins step.

When the Jenkins job finishes, it reports the status to back to JFrog Piplines.

The integration supports:

1. Passing build parameters to the Jenkins job.
2. Sending data from Jenkins back to JFrog Pipelines.

### **Setting Up the Integration**

* Open the JFrog Piplines UI.
* Under ***Integrations***, click on the ***Add an Integration*** button.
* Choose ***Jenkins Server*** integration type and fill out all of the required fields.
* Click on the ***Generate*** button to generate a token. This token is used by Jenkins to authenticate with JFrog Pipelines.
* Copy the Callback URL and Token, and save them in **Jenkins** | **Manage** | **Configure System** | **JFrog Pipelines server**

**What is Jenkins Pipeline?**

**Jenkins Pipeline** is a combination of plugins that supports integration and implementation of continuous delivery pipelines.

It has an extensible automation server to create simple and complex delivery pipelines as code via pipeline DSL.

A Pipeline is a group of events interlinked with each other in a sequence.

# **Jenkins Security**

The Jenkins project takes security seriously. We make every possible effort to ensure users can adequately secure their automation infrastructure. To that end, we work with Jenkins core and plugin developers, as well as security researchers, to fix security vulnerabilities in Jenkins in a timely manner, and to improve the security of Jenkins in general.