**[Jenkins](https://www.geeksforgeeks.org/jenkins-and-git-integration-using-ssh-key/)**

It is a tool that is used for automation, and it is an open-source server that allows all the developers to build, test and deploy software. It works or runs on java as it is written in java. By using Jenkins we can make a continuous integration of projects(jobs) or end-to-endpoint automation.

Let us do discuss the necessity of this tool before going ahead to the procedural part for installation. Nowadays, humans are becoming lazy day by day so even having digital screens and just one click button in front of us then also need some automation. Here, I’m referring to that part ofautomation where we need not have to look upon a process(here called a job) for completion and after it doing another job. For that, we have Jenkins with us.

**Installation of Jenkins in windows:**

1. Before installing Jenkins you should have already installed java on your device as above said it works on java, so it’s important to install and for that, you should install version jdk8. You can find the [download](https://www.oracle.com/in/java/technologies/javase/javase-jdk8-downloads.html) it from the link provided. Make sure while downloading JDK8, choose a favorable option based on your OS (like windows X 64 bit), and then install the JDK8.
   * After installing java check it by using command prompt by java -version and this will show whether you have it or not.
2. Now, go to the official website of Jenkins i.e. www.jenkins.io. Here you will have two options one is an LTS release and one is a weekly release. Download Jenkins for windows from theLTS section(recommended).
3. After download, open that zip(compressed) file, and by right click and choose the all extract option extract the file. We are doing extraction to unzip the compressed file.
4. After extracting, you can see a new folder of Jenkins. Open it and install Jenkins. After installing, it will automatically take you to the browser where localhost::8080 is running. 8080 is the by default port of Jenkins. And if the browser automatically doesn’t open then simply search localhost::8080.

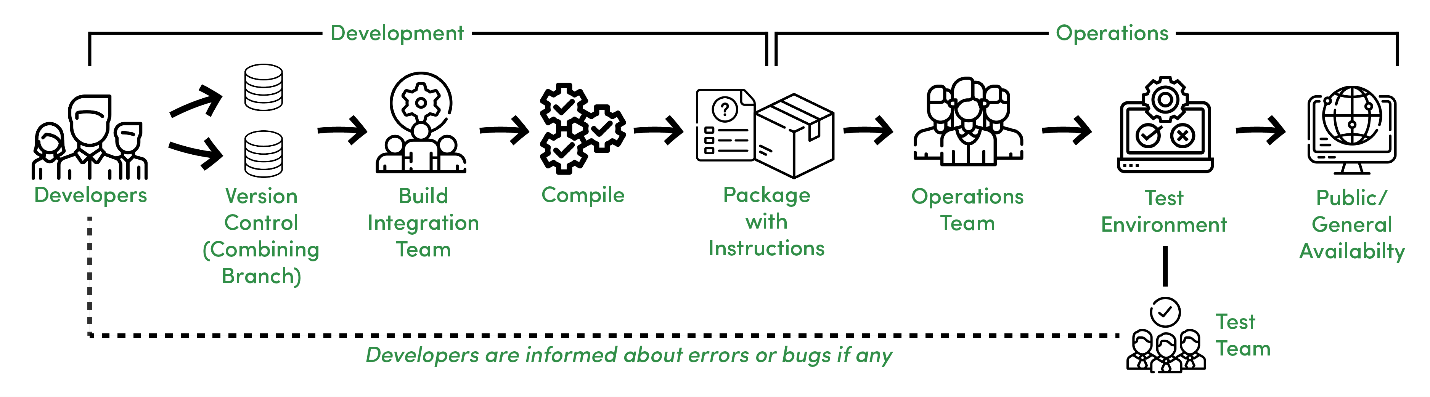
After this, you will see the dashboard or interface of Jenkins. If you are installing it the first time then it will show you some plugins which are recommended to install. And then, you are ready to use Jenkins.

**Key Points to Remember:**

* Since we know in open source we use git and GitHub and to do automation we have to make interact Jenkins with GitHub so you need plugins to do so. And the plugin is nothing other than a software program that adds some important features to the existing one. Why **plugins**? Jenkins knows how to clone from GitHub and integrate, but plugins give it a mind to understand various commands of git.
* Whenever a job is downloading a file/folder so that Jenkins has to provide a space and that space is known as workspace. Normally, we use Jenkins for job chaining.
* **Job chaining**– Connecting the jobs with each other that is whenever the earlier job will run fine then Jenkins will go to the next job. This is also known asa **pipeline**. For example- Suppose you have made 3 jobs in Jenkins and you want them to run when the earlier one completes. So, when Job1 has been run already then Job2 will run and then Job3. So to do this, copy earlier job in copy tab and from triggers choose ‘build after other projects’ option. Here, Job2 is upstream with respect to Job3 and downstream with respect to Job1.
* Install pipeline plugins to have a better view because by this we can visualize for upstream and downstream. By default, we can run only two jobs parallel in Jenkins. But we can also run more jobs parallel, and it depends on your RAM of OS.

Developers follow a process/model known as a [**Software Development Life Cycle (SDLC**)](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/) for developing Software Projects. A traditional[SDLC](https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/) has various stages of development.

1. **Planning –**Documenting and determining project structure and schedule.
2. **Analysis –**Analyzing project requirements and collecting resources.
3. **Design –**Designing software model with architecture and Interface designing.
4. **Implementation –**Developing an actual product by considering design and requirements.
5. **Testing –**Testing the built, fixing errors & bugs, and refactoring the code.
6. **Deploying and Maintenance –**The software is deployed and monitored for further enhancements.



*Traditional Software Development Process*

**Features –**

An overview of a Traditional [SDLC](https://www.geeksforgeeks.org/basic-frame-structure-of-sdlc/) shown in the above image works in the following manner.

* Different Development teams maintain different code repositories.
* Build and Integration team integrates the code.
* Integrated code is compiled as a package.
* A package with instructions is sent to the Operations team.
* The Operations team pushes the package into a test environment, the package is tested by the Testing team.
* The Test team informs the Development team about any possible issues else gives a green signal for Production Release.

However, it is not valuable in this fastly adapting world to practice the above process and models like the Waterfall Model wherein we have to wait till the end to receive a functional product, generally delivered at the end of a cycle.

**CI/CD Definition :**

That’s where CI/CD comes to the rescue. It is more of a culture and practice than tools, but what is CI/CD?

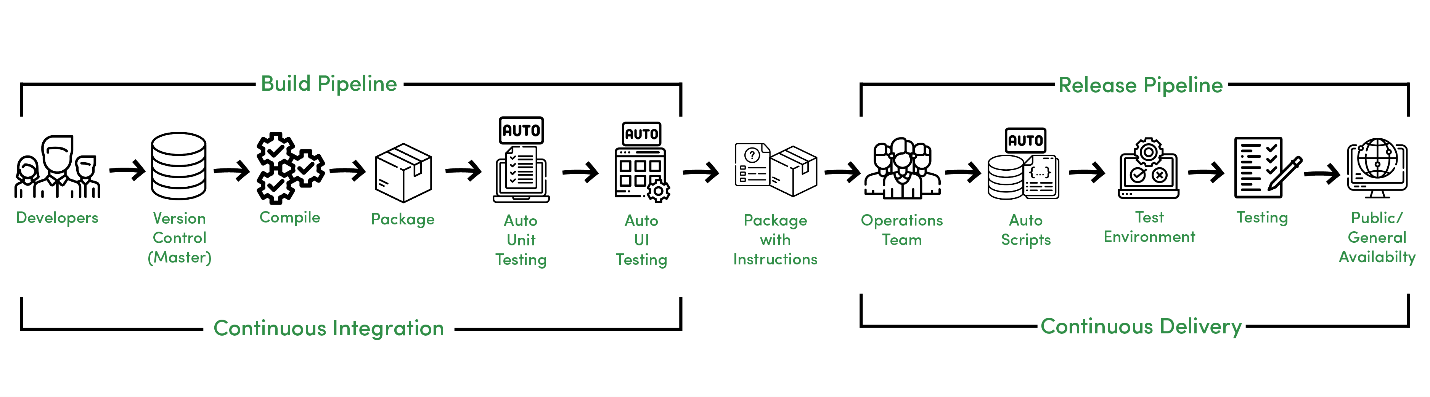
***“CI/CD stands for Continuous Integration and Continuous Delivery (or Continuous Deployment), it is about how an Integrated code on a shared repository is used to release software to production multiple times a day with the help of automation.”***

**Continuous Integration :**

Continuous Integration is about how developers integrate code using a shared repository multiple times a day with the help of automation.

**Continuous Delivery :**

Continuous Delivery is about automatically releasing software to the test or production environment.



*Continuous Integration and Continuous Delivery Process*

**Features –**

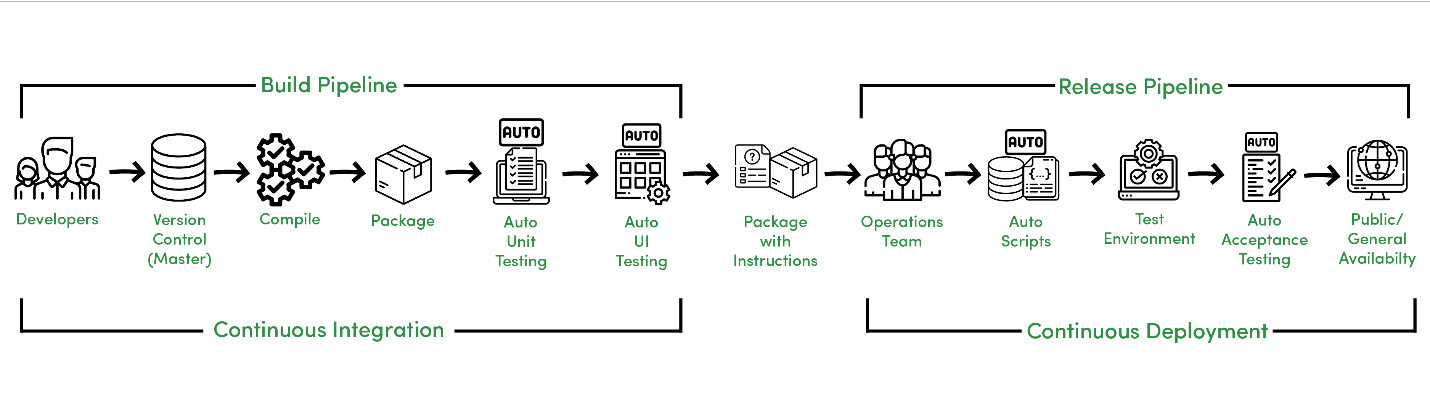
The above image shows Continuous Integration and Continuous Delivery process that differ from traditional SDLC due to the following reasons,

* Developers manage code on a shared repository.
* After compilation, Automated Unit and UI testing are performed.
* The Operations Team needs to take care of Automated scripts that further go to the test environment.
* Testing is performed and after approval, the software is sent to production.

Here Continuous Integration can be referred to as Build Pipeline and Continuous Delivery can be referred to as Release Pipeline.

**Continuous Deployment :**

Continuous Deployment is about releasing software to production automatically without human intervention.



*Continuous Integration and Continuous Deployment Process*

Continuous Deployment is an advancement and it reduces manual work of the Operations and Testing team to deploy the software to production as it is automatically deployed to production after Automated Acceptance Testing.

**Features –**

CI/CD helps for rapid and reliable software development in the following manner.

* Automated and quick code integration.
* Error detection and response are fast due to the short cycle and iterations.
* Less prone to human errors.
* Faster than following manual instructions due to automated scripting.
* Deployment can be done several times a day.

Moreover, CI/CD in [DevOps](https://www.geeksforgeeks.org/lifecycle-of-devops/) is about unifying process and automation, it helps to act as a bridge between Development and Operations team to make the process more reliable and adaptive.