What is 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.

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.

ADVERTISING

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**

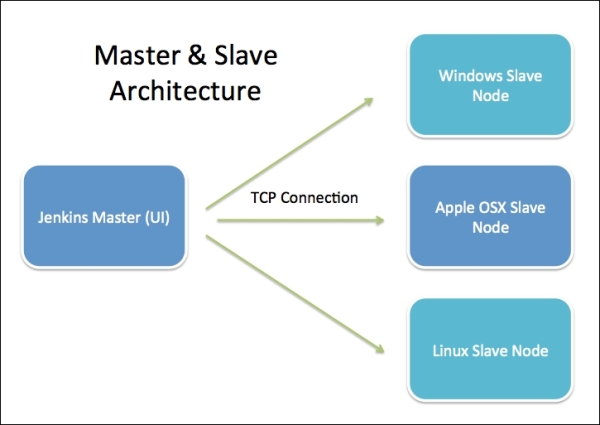
* Its interface is out dated and not user friendly compared to current user interface trends.
* Not easy to maintain it because it runs on a server and requires some skills as server administrator to monitor its activity.
* CI regularly breaks due to some small setting changes. CI will be paused and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

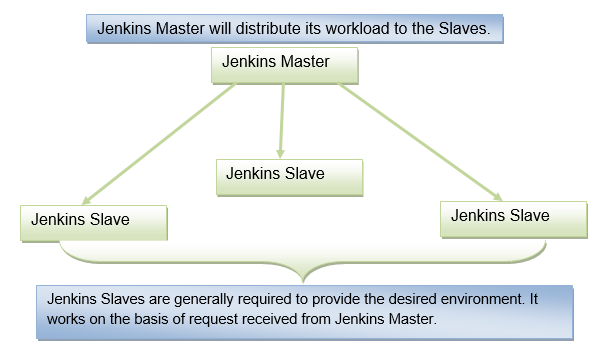
The server's job or master's job is to handle:

* 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.

## What is Git?

Git is a widely used modern version control system for tracking changes in computer files. The term version control system suggests a system that records all the changes made to a file or set of data, so a specific version can be considered whenever needed. This feature makes the process of collaboration so feasible with all team members, making it considerably more comfortable to work over a big project.

Git makes it possible for several people involved in the project to work together and track each other's progress over time. In software development, the tool helps in Source Code Management. Git favors not only programmers but also non-technical users by keeping track of their project files.

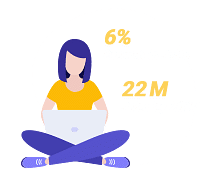
While [working on Git,](https://www.simplilearn.com/tutorials/git-tutorial/git-tutorial-for-beginner) we actively use two repositories.

* Local repository: The local repository is present on our computer and consists of all the files and folders. This Repository is used to make changes locally, review history, and commit when offline.
* Remote repository: The remote repository refers to the server repository that may be present anywhere. This repository is used by all the team members to exchange the changes made.

Both repositories have their own set of commands. There are separate Git Commands that work on different types of repositories.

#### **Get All Your Questions Answered**

PRINCE2® Foundation & Practitioner Certification[EXPLORE COURSE](https://www.simplilearn.com/project-management/prince2-foundation-and-practitioner-certification-training?source=GhPreviewCTABanner)



## Git Commands: Working With Local Repositories

### git init

* The command git init is used to create an empty Git repository.
* After the git init command is used, a .git folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.

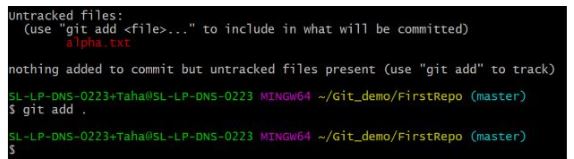
|  |
| --- |
| git init |



### git add

* Add command is used after checking the status of the files, to add those files to the staging area.
* Before running the commit command, "git add" is used to add any new or modified files.

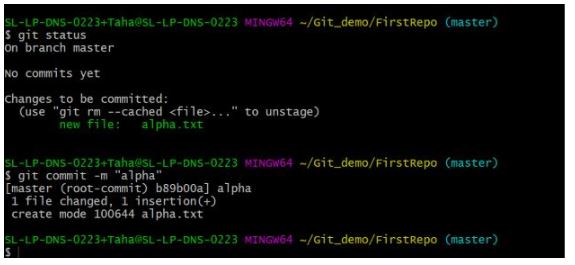
|  |
| --- |
| git add . |



### git commit

* The commit command makes sure that the changes are saved to the local repository.
* The command "git commit –m <message>" allows you to describe everyone and help them understand what has happened.

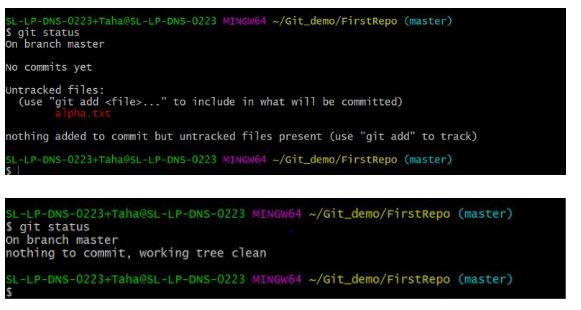
|  |
| --- |
| git commit -m “commit message” |



### git status

* The git status command tells the current state of the repository.
* The command provides the current working branch. If the files are in the staging area, but not committed, it will be shown by the git status. Also, if there are no changes, it will show the message no changes to commit, working directory clean.

|  |
| --- |
| git status |



#### **Get Certified in DevOps with Caltech CTME**

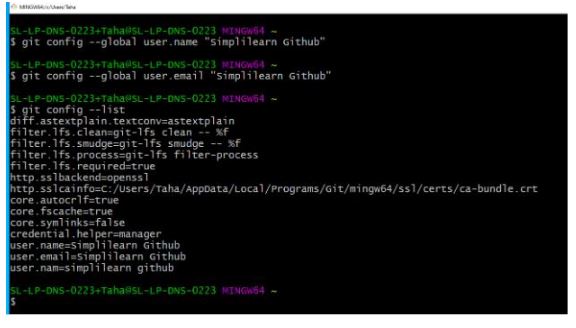
Free Webinar | Watch the Webcast[EXPLORE NOW!](https://www.simplilearn.com/career-information-session-get-certified-in-devops-with-caltech-ctme-webinar?utm_source=webinarpromo&utm_medium=FRS-banner&utm_campaign=July)



### git config

* The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.
* When git config is used with --global flag, it writes the settings to all repositories on the computer.

|  |
| --- |
| git config --global user.name “any user name”  git config --global user.email <email id> |



### git branch

* The git branch command is used to determine what branch the local repository is on.
* The command enables adding and deleting a branch.

|  |
| --- |
| # Create a new branch   git branch <branch\_name> |
| # List all remote or local branches   git branch -a |
| # Delete a branch   git branch -d <branch\_name> |

### git checkout

* The git checkout command is used to switch branches, whenever the work is to be started on a different branch.
* The command works on three separate entities: files, commits, and branches.

|  |
| --- |
| # Checkout an existing branch   git checkout <branch\_name> |
| # Checkout and create a new branch with that name   git checkout -b <new\_branch> |

### git merge

* The [git merge](https://www.simplilearn.com/tutorials/git-tutorial/merge-conflicts-in-git" \t "_blank" \o "git merge) command is used to integrate the branches together. The command combines the changes from one branch to another branch.
* It is used to merge the changes in the staging branch to the stable branch.

|  |
| --- |
| git merge <branch\_name> |

#### **Earn the Most Coveted DevOps Certification!**

DevOps Engineer Masters Program[EXPLORE PROGRAM](https://www.simplilearn.com/devops-engineer-masters-program-certification-training?source=GhPreviewCTABanner)



However, these are popular and basic git commands used by developers.

## Git Commands: Working With Remote Repositories

### git remote

* The git remote command is used to create, view, and delete connections to other repositories.
* The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

|  |
| --- |
| git remote add origin <address> |



### git clone

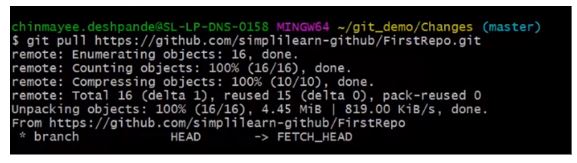
* The git clone command is used to create a local working copy of an existing remote repository.
* The command downloads the remote repository to the computer. It is equivalent to the Git init command when working with a remote repository.

|  |
| --- |
| git clone <remote\_URL> |

### git pull

* The [git pull command](https://www.simplilearn.com/tutorials/git-tutorial/git-pull-request" \t "_blank" \o "git pull command) is used to fetch and merge changes from the remote repository to the local repository.
* The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

|  |
| --- |
| git pull <branch\_name> <remote URL> |



### git push

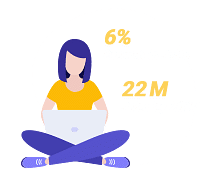
* The command [git push](https://www.simplilearn.com/tutorials/git-tutorial/git-push-command" \t "_blank" \o "git push) is used to transfer the commits or pushing the content from the local repository to the remote repository.
* The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

|  |
| --- |
| git push -u origin master |



#### **Get All Your Questions Answered**

PRINCE2® Foundation & Practitioner Certification[EXPLORE COURSE](https://www.simplilearn.com/project-management/prince2-foundation-and-practitioner-certification-training?source=GhPreviewCTABanner)



## Some Advanced Git Commands

### git stash

* The git stash command takes your modified tracked files and saves it on a pile of incomplete changes that you can reapply at any time. To go back to work, you can use the stash pop.
* The git stash command will help a developer switch branches to work on something else without committing to incomplete work.

|  |
| --- |
| # Store current work with untracked files   git stash -u |
| # Bring stashed work back to the working directory   git stash pop |

### git log

* The git log command shows the order of the commit history for a repository.
* The command helps in understanding the state of the current branch by showing the commits that lead to this state.

|  |
| --- |
| git log |



## Conclusion

We hope this Git commands tutorial has helped you understand various useful commands in Git. You have learned the basics of Git and the different commands that are used, and also saw different basic Git commands and advanced too. according to the repository they are used in, followed by some advanced commands. We understood what each of the mentioned commands does and also came across the syntax of each command.