Introduction Git:

Git is a distributed revision control and source code management system.Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.s a distributed revision control system it is aimed at speed,data integrity, and support for distributed, non-linear workflows.

Introduction of GitLab:

GitLab do almost everything that GitHub does.
GitLab Community Edition is free and open sourced.That in itself gives it a huge boost when compared to GitHub Enterprise Edition.
With GitLab community edition you might be comfortable until the end of time or you might start with the enterprise edition right away.

Basic Difference of GitHub and GitLab :

Features	**GitLab**	**GitHub**
Pricing	**Unlimited public and private repositories / unlimited public and private collaborators**	**Free for public repositories / Paid plans for private repositories**
Code review features	**yes**	**yes**
Bug & issue tracking	**yes**	**yes**
Private branch	**yes**	**Yes (with paid plans)**
Build system	**yes**	**Yes (with 3rd party service)**
Self-hosting	**Yes**	**Yes (with enterprise plan)**
detailed pricing	**Free: gitlab.com / Free: GitLab Community Edition / $39 / Year: GitLab Enterprise**	**Free: public projects / $7/month: Personal plan / $25/month: organization plan / $2.500/year: Enterprise**

Advantages :

1. Free and open source.
2. Fast and small.
3. No need of powerful hardware.

4. Easier branching.

Installation of Git :

Debian/Ubuntu

Installation steps:

- sudo add-apt-repository ppa:git-core/ppa
- sudo apt-get update
- sudo apt-get install git

Note : For Ubuntu, this PPA provides the latest stable upstream Git version.

Centos6/7:

Installation steps:

- yum install git

GitHub for Windows:

Please check the below link for windows installation.
https://windows.github.com

Note : Before Starting the setup you need to create git account Click here to create git account

SETUP :

Configuring user information used across all local repositories

command : **git config --global user.name "[firstname lastname]"**
set a name that is identifiable for credit when review version history

command : git config --global user.email “[valid-email]”
set an email address that will be associated with each history marker

SETUP & INIT :

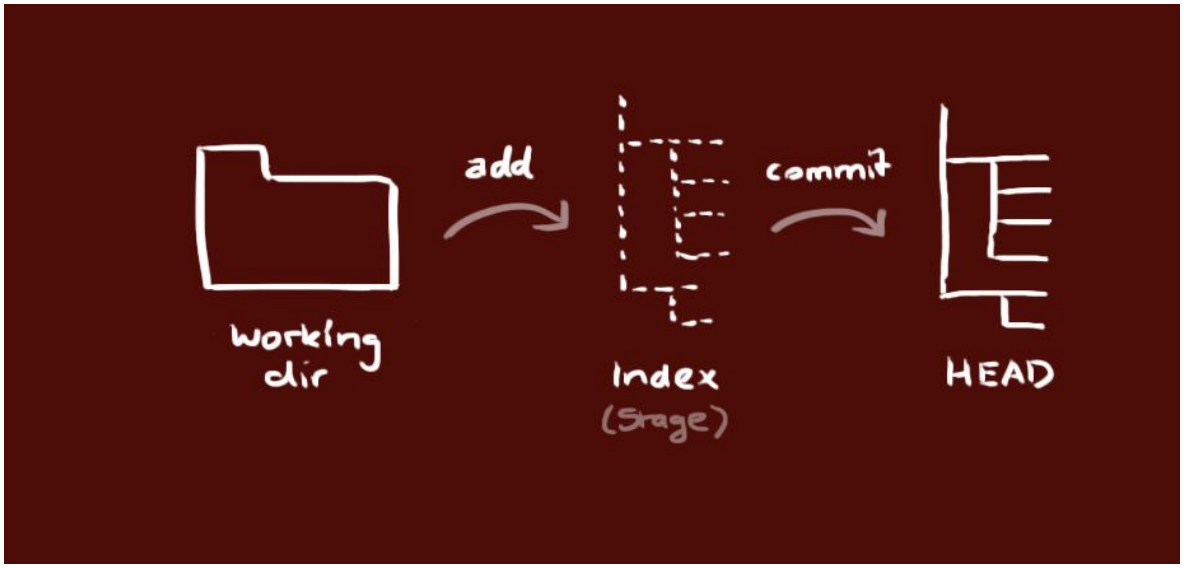
Configuring user information, initializing and cloning repositories.

command : git init
initialize an existing directory as a Git repository.

command: git clone [url]
retrieve an entire repository from a hosted location via URL

Git Workflow :

your local repository consists of three "trees" maintained by git. the first one is your **Working Directory** which holds the actual files. the second one is the **Index** which acts as a staging area and finally the **HEAD** which points to the last commit you've made.



The HEAD last commit snapshot, next parent

The HEAD in Git is the pointer to the current branch reference, which is in turn a pointer to the last commit you made or the last commit that was checked out into your working directory. That also means it will be the parent of the next commit you do. It's generally simplest to think of it as HEAD is the snapshot of your last commit.

The Index next proposed commit snapshot

The Index is your proposed next commit. Git populates it with a list of all the file contents that were last checked out into your working directory and what they looked like when they were originally checked out. It's not technically a tree structure, it's a flattened manifest, but for our purposes it's close enough. When you run git commit, that command only looks at your Index by default, not at anything in your working directory. So, it's simplest to think of it as the Index is the snapshot of your next commit.

Git add & commit:

You can propose changes (add it to the Index) using:

Command : git add <filename> or git add *

This is the first step in the basic git workflow. To actually commit these changes use.

Command : git commit -m "Commit message"

Now the file is committed to the HEAD, but not in your remote repository yet.

pushing changes :

Your changes are now in the HEAD of your local working copy. To send those changes to your remote repository, execute.

Command : git push origin master

Change master to whatever branch you want to push your changes to.

If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

Command : git remote add origin <server>

Now you are able to push your changes to the selected remote server.

Major GIT commands:

1. git config :
   Sets configuration values for your user name, email, gpg key, preferred diff algorithm,

file formats and more. Example: git config --global user.name "My Name" git config --global user.email "user@domain.com" cat ~/.gitconfig [user] name = My Name email = user@domain.com

2. git init :
   Initializes a git repository – creates the initial '.git' directory in a new or in an existing project. Example: cd /home/user/my_new_git_folder/ git init
3. git clone:
   Makes a Git repository copy from a remote source. Also adds the original location as a remote so you can fetch from it again and push to it if you have permissions. Example: git clone git@github.com:user/test.git
4. git add:
   Adds files changes in your working directory to your index. Example: git add .
5. git rm:
   Removes files from your index and your working directory so they will not be tracked. Example: git rm filename.
6. git commit:
   Takes all of the changes written in the index, creates a new commit object pointing to it and sets the branch to point to that new commit. Examples: git commit -m 'committing added changes' git commit -a -m 'committing all changes, equals to git add and git commit'
7. git status:
   Shows you the status of files in the index versus the working directory. It will list out files that are untracked (only in your working directory), modified (tracked but not yet updated in your index), and staged (added to your index and ready for committing). Example: git status # On branch master # # Initial commit # # Untracked files: # (use "git add <file>..." to include in what will be committed) # # README nothing added to commit but untracked files present (use "git add" to track)
8. git branch:
   Lists existing branches, including remote branches if '-a' is provided. Creates a new branch if a branch name is provided. Example: git branch -a * master remotes/origin/master
9. git checkout:
   Checks out a different branch – switches branches by updating the index, working tree, and HEAD to reflect the chosen branch. Example: git checkout new branch.
10. git merge:
    Merges one or more branches into your current branch and automatically creates a new commit if there are no conflicts. Example: git merge new branch version.
11. git fetch:
    Fetches all the objects from the remote repository that are not present in the local one. Example: git fetch origin

12. git pull:

    Fetches the files from the remote repository and merges it with your local one. This command is equal to the git fetch and the git merge sequence. Example: git pull origin

13. git push:

    Pushes all the modified local objects to the remote repository and advances its branches. Example: git push origin master