GIT

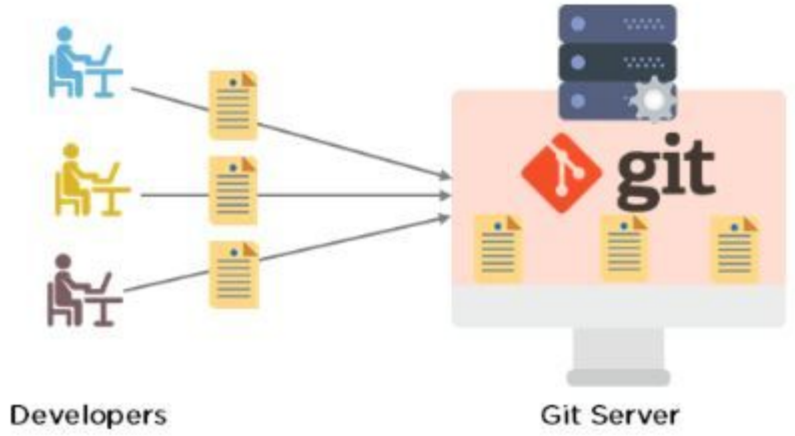
What is GIT?

Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development.

- Git is used to tracking changes in the source code
- The distributed version control tool is used for source code management
- It allows multiple developers to work together
- It supports non-linear development through its thousands of parallel branches

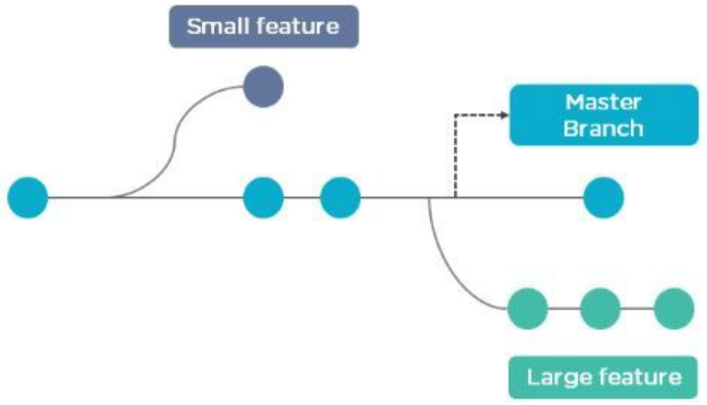
Features of GIT

- Tracks history
- Free and open source
- Supports non-linear development
- Creates backups
- Scalable
- Supports collaboration
- Branching is easier
- Distributed development



Branch in GIT

Branch in Git is used to keep your changes until they are ready. You can do your work on a branch while the main branch (master) remains stable. After you are done with your work, you can merge it with the main office.The above diagram shows there is a master branch. There are two separate branches called “small feature” and “large feature.” Once you are finished working with the two separate branches, you can merge them and create a master branch.



How to install GIT

- ➔ **Linux :**

  https://git-scm.com/download/linux

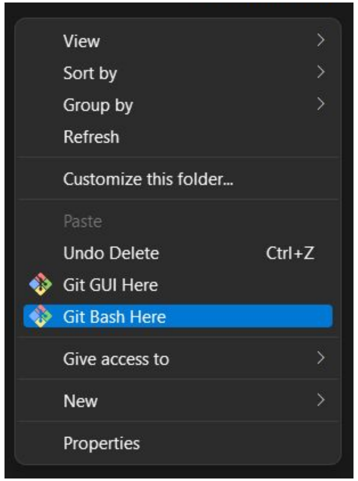
- ➔ **Mac :**

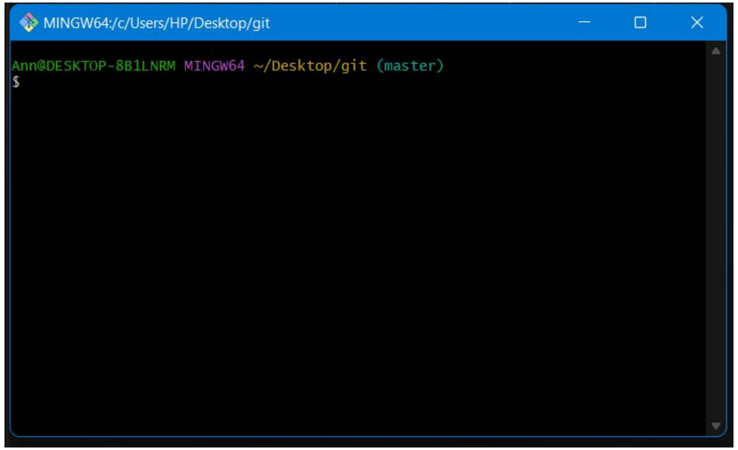
  https://git-scm.com/download/mac

- ➔ **Windows :**

  https://git-scm.com/download/win

➔ Create an empty folder → right-click




MINGW64:/c/Users/HP/Desktop/git

```
Ann@DESKTOP-8B1LNRM MINGW64 ~/Desktop/git (master)
$
```

Basic Commands in GIT

➔ **git init**

Initializes a new Git repository. If you want to place a project under revision control, this is the first command you need to learn.

➔ **git add**

Moves changes from the working directory to the staging area. This gives you the opportunity to prepare a snapshot before committing it to the official history.

➔ **git branch**

This command is your general-purpose branch administration tool. It lets you create isolated development environments within a single repository.

- ## git commit

  Takes the staged snapshot and commits it to the project history. Combined with git add, this defines the basic workflow for all Git users.

- ## git commit -amend

  Passing the --amend flag to git commit lets you amend the most recent commit. This is very useful when you forget to stage a file or omit important information from the commit message

- ## git push

  Pushing is the opposite of fetching (with a few caveats). It lets you move a local branch to another repository, which serves as a convenient way to publish contributions. This is like svn commit, but it sends a series of commits instead of a single changeset.

- ## git pull

  Pulling is the automated version of git fetch. It downloads a branch from a remote repository, then immediately merges it into the current branch. This is the Git equivalent of svn update.

➔ git checkout

In addition to checking out old commits and old file revisions, git checkout is also the means to navigate existing branches. Combined with the basic Git commands, it’s a way to work on a particular line of development.

➔ git clean

Removes untracked files from the working directory. This is the logical counterpart to git reset, which (typically) only operates on tracked files.

➔ git clone

Creates a copy of an existing Git repository. Cloning is the most common way for developers to obtain a working copy of a central repository.

➔ git config

A convenient way to set configuration options for your Git installation. You’ll typically only need to use this immediately after installing Git on a new development machine.

➔ git fetch

Fetching downloads a branch from another repository, along with all of its associated commits and files. But, it doesn't try to integrate anything into your local repository. This gives you a chance to inspect changes before merging them with your project.

➔ git log

Lets you explore the previous revisions of a project. It provides several formatting options for displaying committed snapshots.

➔ git merge

A powerful way to integrate changes from divergent branches. After forking the project history with git branch, git merge lets you put it back together again.

➔ git rebase

Rebasing lets you move branches around, which helps you avoid unnecessary merge commits. The resulting linear history is often much easier to understand and explore.

- ➔ **git rebase -i**

The -i flag is used to begin an interactive rebasing session. This provides all the benefits of a normal rebase, but gives you the opportunity to add, edit, or delete commits along the way.

- ➔ **git reflog**

Git keeps track of updates to the tip of branches using a mechanism called reflog. This allows you to go back to changesets even though they are not referenced by any branch or tag.

- ➔ **git remote**

A convenient tool for administering remote connections. Instead of passing the full URL to the fetch, pull, and push commands, it lets you use a more meaningful shortcut.

- ➔ **git reset**

Undoes changes to files in the working directory. Resetting lets you clean up or completely remove changes that have not been pushed to a public repository.

➔ **git revert**

Undoes a committed snapshot. When you discover a faulty commit, reverting is a safe and easy way to completely remove it from the code base.

➔ **git status**

Undoes a committed snapshot. When you discover a faulty commit, reverting is a safe and easy way to completely remove it from the code base.