Git features

- 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

- Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development. Linus Torvalds created Git in 2005 for the development of the linux kernel.
- developers used to submit their codes to the central server without having copies of their own
- Any changes made to the source code were unknown to the other developers
- There was no communication between any of the developers

Features of Git

- Tracks history
- Free and open source
- Supports non-linear development
- Creates backups
- Scalable
- Supports collaboration
- Branching is easier
- Distributed development
- The Git workflow is divided into three states:
- Working directory - Modify files in your working directory
- Staging area (Index) - Stage the files and add snapshots of them to your staging area
- Git directory (Repository) - Perform a commit that stores the snapshots permanently to your Git directory. Checkout any existing version, make changes, stage them and commit.

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.

Commands in Git

- Create Repositories
  git Init
- Make Changes
  add
  commit
  status
- Parallel Development
  branch
  merge
  rebase
- Sync Repositories
  push
  pull
  add origin

- Centralized Version Control System
- Uses a central server to store all the files
- Every operation is performed directly on the repository
- All the versions of the file are stored on the Central VCS server
- In case the central server crashes, the entire data of the project will be lost. Hence, distributed VCS was introduced.
- Distributed Version Control System
- Every programmer has a copy of all the versions of the code on their local systems
- Distributed VCS moves from the client-server approach of central VCS to a peer-to-peer approach
- They can update their local repositories with new data from the central server and changes are reflected in the principal repository
- Git is one such distributed VCS tool

Git Tools

- To explore the robust functionality of Git, we need some tools. Git comes with some of its tools like Git Bash, Git GUI to provide the interface between machine and user. It supports inbuilt as well as third-party tools.
- Git comes with built-in GUI tools like **git bash**, **git-gui** , and **gitk** for committing and browsing. It also supports several third-party tools for users looking for platform-specific experience.

- Git GUI
- Git GUI is a powerful alternative to Git BASH. It offers a graphical version of the Git command line function, as well as comprehensive visual diff tools. We can access it by simply right click on a folder or location in windows explorer. Also, we can access it through the command line by typing below command.
- $ git gui
- Git facilitates with some built-in GUI tools for committing (git- gui) and browsing (gitk), but there are many third-party tools for users looking for platform-specific experience.
- Gitk
- gitk is a graphical history viewer tool. It's a robust GUI shell over **git log** and **git grep**. This tool is used to find something that happened in the past or visualize your project's history.
- Gitk can invoke from the command-line. Just change directory into a Git repository, and type:
- $ gitk [git log options]
- This command invokes the gitk graphical interface and displays the project history. The Gitk interface looks like this:
- Gitk supports several command-line options, most of which are passed through to the underlying git log action.

Git Third-Party Tools

- Git Third-Party Tools
- Many third-party tools are available in the market to enhance the functionality of Git and provide an improved user interface. These tools are available for distinct platforms like Windows, Mac, Linux, Android, iOS.
- A list of popular third party Git tools are as follows:

Git Third-Party Tools List

Tools	Platforms					Price	License Type
	Windows	Mac	Linux	Android	iOS		
Source Tree	Yes	Yes	No	No	No	Free	Proprietary
GitHub Desktop	Yes	Yes	No	No	No	Free	MIT
Tortoise Git	Yes	No	No	No	No	Free	GNU GPL

Git Extensions	Yes	Yes	Yes	No	No	Free	GNU GPL
GitKraken	Yes	Yes	Yes	No	No	Free/$29/$49	Proprietary
SmartGit	Yes	Yes	Yes	No	No	$79/user /free for non-commercial use	Proprietary
Tower	Yes	Yes	No	No	No	$79/user (30 days free trial)	Proprietary
Git Up	No	Yes	No	No	No	Free	GNU GPL
GitEye	Yes	Yes	Yes	No	No	Free	Proprietary

gitg	Yes	No	Yes	No	No	Free	GNUGPL
Git2Go	No	No	No	No	Yes	Free with in-app purchases	Proprietary
GitDrive	No	No	No	No	Yes	Free with in-app purchases	Proprietary
GitFinder	No	Yes	No	No	No	$24.95	Proprietary
SnailGit	No	Yes	No	No	No	&9.99/Lite version	Proprietary
Pocket Git	No	No	No	Yes	No	1.99€	Proprietary
Sublime Merge	Yes	Yes	Yes	No	No	$99/user, $75 annual business sub, free eval	Proprietary