Git

Git is a distributed version control system primarily used for managing source code during software development.

Features of git

* **Open Source**  
  Git is an **open-source tool**. It is released under the **GPL** (General Public License) license.
* **Scalable**  
  Git is **scalable**, which means when the number of users increases, the Git can easily handle such situations
* **Distributed**  
  One of Git's great features is that it is **distributed**. Distributed means that instead of switching the project to another machine, we can create a "clone" of the entire repository. Also, instead of just having one central repository that you send changes to, every user has their own repository that contains the entire commit history of the project. We do not need to connect to the remote repository; the change is just stored on our local repository. If necessary, we can push these changes to a remote repository.
* **Security**  
  Git is secure. It uses the **SHA1 (Secure Hash Function)** to name and identify objects within its repository. Files and commits are checked and retrieved by its checksum at the time of checkout. It stores its history in such a way that the ID of particular commits depends upon the complete development history leading up to that commit. Once it is published, one cannot make changes to its old version.
* **Speed**  
  Git is very **fast**, so it can complete all the tasks in a while. Most of the git operations are done on the local repository, so it provides a **huge speed**. Also, a centralized version control system continually communicates with a server somewhere.  
  Performance tests conducted by Mozilla showed that it was **extremely fast compared to other VCSs**. Fetching version history from a locally stored repository is much faster than fetching it from the remote server. The **core part of Git**is **written in C**, which **ignores** runtime overheads associated with other high-level languages.  
  Git was developed to work on the Linux kernel; therefore, it is **capable** enough to **handle large** **repositories** effectively. From the beginning, **speed** and **performance** have been Git's primary goals.
* **Supports non-linear development**  
  Git supports **seamless branching and merging**, which helps in visualizing and navigating a non-linear development. A branch in Git represents a single commit. We can construct the full branch structure with the help of its parental commit.
* **Branching and Merging**  
  **Branching and merging** are the **great feature**s of Git, which makes it different from the other SCM tools. Git allows the **creation of multiple branches** without affecting each other. We can perform tasks like **creation**, **deletion**, and **merging** on branches, and these tasks take a few seconds only. Below are some features that can be achieved by branching:
  + We can **create a separate branch** for a new module of the project, commit and delete it whenever we want.
  + We can have a **production branch**, which always has what goes into production and can be merged for testing in the test branch.
  + We can create a **demo branch** for the experiment and check if it is working. We can also remove it if needed.
  + The core benefit of branching is if we want to push something to a remote repository, we do not have to push all of our branches. We can select a few of our branches, or all of them together.
* **Data Assurance**  
  The Git data model ensures the **cryptographic integrity** of every unit of our project. It provides a **unique commit ID** to every commit through a **SHA algorithm**. We can **retrieve** and **update** the commit by commit ID. Most of the centralized version control systems do not provide such integrity by default.
* **Staging Area**  
  The **Staging area** is also a **unique functionality** of Git. It can be considered as a **preview of our next commit**, moreover, an **intermediate area** where commits can be formatted and reviewed before completion. When you make a commit, Git takes changes that are in the staging area and make them as a new commit. We are allowed to add and remove changes from the staging area. The staging area can be considered as a place where Git stores the changes.  
  Although, Git doesn't have a dedicated staging directory where it can store some objects representing file changes (blobs). Instead of this, it uses a file called index.

Another feature of Git that makes it apart from other SCM tools is that **it is possible to quickly stage some of our files and commit them without committing other modified files in our working directory.**

* **Maintain the clean history**  
  Git facilitates with Git Rebase; It is one of the most helpful features of Git. It fetches the latest commits from the master branch and puts our code on top of that. Thus, it maintains a clean history of the project.

Git vs Svn

| **GIT** | **SVN** |
| --- | --- |
| Git is open source distributed vice control system developed by Linus Torvalds in 2005. It emphasis on speed and data integrity | Apache Subversion is an open source software version and revision control system under Apache license. |
| Git has a Distributed Model. | SVN has a Centralized Model. |
| In git every user has their own copy of code on their local like their own branch. | In SVN there is central repository has working copy that also make changes and committed in central repository. |
| In git we do not required any Network to perform git operation. | In SVN we required Network for runs the SVN operation. |
| Git is more difficult to learn. It has more concepts and commands. | SVN is much easier to learn as compared to git. |
| Git deals with large number of files like binary files that change quickly that why it become slow. | SVN control the large number of binary files easily. |
| In git we create only .git directory. | In SVN we create .svn directory in each folder. |
| It does not have good UI as compared to SVN. | SVN has simple and better user interface . |
| Features of GIT:   * Distributed System. * Branching. * Compatibility. * Non-linear Development. * Lightweight. * Open source. | Features of SVN:   * Directories are versioned * Copying, deleting, and renaming. * Free-form versioned metadata . * Atomic commits. * Branching and tagging. * Merge tracking. * File locking. |

Git config

git config --global user.name “Aakash”

git config –global user.email [aa@gmail.com](mailto:aa@gmail.com)

**git config** is a command-line tool that allows you to set, get, and manage Git configuration variables. Git configuration variables control aspects of how Git behaves and operates, such as user information, default behaviors, and preferences.

Git init

Git init is a commant it initializes a git repository in the directory.

. When you run **git init** in a folder, it creates a new subdirectory named **.git** inside that folder. This **.git** directory contains all the necessary files and subdirectories that Git needs to manage the repository, including metadata, object database, configuration settings, and more.

Git status

**git status** is a command in Git that provides information about the current state of the working directory and the staging area. When you run **git status**, Git displays a summary of changes since the last commit, including files that have been modified, staged, or are untracked.

Git branch

Git branch name

Git branch –d name

Git branch –m name newname

Git merge

Fast merge…

Git stash

Git diff